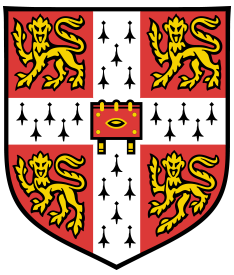
Statistical Models for Noise-Robust Speech Recognition

ROGIER CHRISTIAAN VAN DALEN

QUEENS' COLLEGE



Department of Engineering

University of Cambridge

This dissertation is submitted for the degree of Doctor of Philosophy

October 2011

Statistical Models for Noise-Robust Speech Recognition

Rogier Christiaan van Dalen

Abstract

A standard way of improving the robustness of speech recognition systems to noise is model compensation. This replaces a speech recogniser's distributions over clean speech by ones over noise-corrupted speech. For each clean speech component, model compensation techniques usually approximate the corrupted speech distribution with a diagonal-covariance Gaussian distribution. This thesis looks into improving on this approximation in two ways: firstly, by estimating full-covariance Gaussian distributions; secondly, by approximating corrupted-speech likelihoods without any parameterised distribution.

The first part of this work is about compensating for within-component feature correlations under noise. For this, the covariance matrices of the computed Gaussians should be full instead of diagonal. The estimation of off-diagonal covariance elements turns out to be sensitive to approximations. A popular approximation is the one that state-of-the-art compensation schemes, like VTS compensation, use for *dynamic coefficients*: the *continuous-time approximation*. Standard speech recognisers contain both per-time slice, static, coefficients, and dynamic coefficients, which represent signal changes over time, and are normally computed from a window of static coefficients. To remove the need for the continuous-time approximation, this thesis introduces a new technique. It first compensates a distribution over the window of statics, and then applies the same linear projection that extracts dynamic coefficients. It introduces a number of methods that address the correlation changes that occur in noise within this framework. The next problem is decoding speed with full covariances. This thesis reanalyses the previously-introduced *predictive linear transformations*, and shows how they can model feature correlations at low and tunable computational cost.

The second part of this work removes the Gaussian assumption completely. It introduces a sampling method that, given speech and noise distributions and a mismatch function, in the limit calculates the corrupted speech likelihood exactly. For this, it transforms the integral in the likelihood expression, and then applies sequential importance resampling. Though it is too slow to use for recognition, it enables a more fine-grained assessment of compensation techniques, based on the KL divergence to the ideal compensation for one component. The KL divergence proves to predict the word error rate well. This technique also makes it possible to evaluate the impact of approximations that standard compensation schemes make.

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.

In particular, two parts should be mentioned. Sections 6.2 and 6.3 of this thesis extend work on predictive linear transformations originally submitted for an M.Phil. thesis (van Dalen 2007). The work on component-independent predictive transformations, section 6.4, was joint work with Federico Flego (and published as van Dalen *et al.* 2009).

Most of the work in this thesis has been published, as technical reports (van Dalen and Gales 2009*a*; 2010*b*), in conference proceedings (van Dalen and Gales 2008; 2009*b*; van Dalen *et al.* 2009; van Dalen and Gales 2010*a*), and as a journal article (van Dalen and Gales 2011).

The length of this thesis including appendices, bibliography, footnotes, tables and equations is 63 532 words. It contains 39 figures and 15 tables.

Acknowledgements

Firstly, thanks must go to the people who have enabled me to do a Ph.D. at Cambridge University. Toshiba Research Europe Ltd. funded my Ph.D. completely, and after specifying the subject area, environment-aware speech recognition, did not constrain me at all. I have much appreciated that.

The other enabler is Dr Mark Gales, my supervisor. I could not have wished for a supervisor more attentive to the big picture and the details, more enthusiastic, or happier to let me work at my own schedule. I am grateful that he has always set aside an hour a week for me, and has always made time to address my questions. Above all, I appreciate his dedication to helping me get as much out of my Ph.D. as possible.

Hank Liao can be called my predecessor, and I inherited much of his code and experimental set-ups. Thanks go to Federico Flego not only for providing infinitely patient help with my experiments, but also for collaborating on exciting work (as specified in the Declaration).

Milica Gašić and Matt Shannon provided invaluable comments to a draft version of this thesis. Thanks also go to past and present occupants of my office, and other members of the speech group, in particular, Chris Longworth, Zoi Roupakia, Anton Ragni, Milica Gašić, Catherine Breslin, and three Matts — Matt Gibson, Matt Shannon, and Matt Seigel — for scintillating discussions, whether it be speech recognition, machine learning, or subjects less directly related to our research. All have enriched my time here.

Finally, to my parents and family: dank jullie wel voor alle aanmoediging en steun door de jaren heen. Zonder jullie hulp was ik hier nimmer terechtgekomen, noch had ik het zo weten af te maken!

Contents

Contents v

1 Introduction 7

I Background 11

2 Speech recognition 13
2.1 Feature extraction . . . 13
2.1.1 Mel-frequency cepstral coefficients . . . 14
2.1.2 Dynamic coefficients . . . 16
2.2 Hidden Markov models . . . 18
2.2.1 State sequences . . . 20
2.2.1.1 Language modelling . . . 21
2.2.1.2 Latent discrete sequence . . . 22
2.3 Training . . . 28
2.3.1 Empirical distributions . . . 28
2.3.2 Maximum-likelihood estimation . . . 29
2.3.2.1 Expectation–maximisation . . . 31
2.3.3 Baum–Welch . . . 33
2.4 Decoding . . . 36
2.5 Summary . . . 38

3 Adaptation 39
3.1 Unsupervised adaptation . . . 39
3.1.1 Adaptive training . . . 43
3.2 Linear adaptation . . . 44
3.2.1 Constrained transformation . . . 45
3.2.2 Covariance adaptation . . . 47
3.3 Covariance modelling . . . 49
3.3.1 Structured precision matrices . . . 50
3.3.2 Maximum likelihood projection schemes . . . 53
3.4 Summary . . . 55

4 Noise-robustness **57**
4.1 Methods for noise-robustness . . . 58
4.2 Noise-corrupted speech . . . 59
4.2.1 Log-spectral mismatch function . . . 60
4.2.1.1 Properties of the phase factor . . . 65
4.2.2 Cepstral mismatch function . . . 69
4.2.3 Mismatch function for dynamic coefficients . . . 70
4.3 The corrupted speech distribution . . . 70
4.3.1 Sampling from the corrupted speech distribution . . . 74
4.4 Model compensation . . . 75
4.4.1 Data-driven parallel model combination . . . 77
4.4.2 Vector Taylor series compensation . . . 82
4.4.3 Joint uncertainty decoding . . . 87
4.4.3.1 Estimating the joint distribution . . . 90
4.4.4 Single-pass retraining . . . 92
4.4.4.1 Assessing the quality of model compensation . . . 94
4.5 Observation-dependent methods . . . 95
4.5.1 The Algonquin algorithm . . . 95
4.5.2 Piecewise linear approximation . . . 101
4.6 Model-based feature enhancement . . . 103
4.6.1 Propagating uncertainty . . . 105
4.6.2 Algonquin . . . 106
4.7 Noise model estimation . . . 107
4.7.1 Linearisation . . . 109
4.7.2 Numerical approximation . . . 111
4.8 Summary . . . 112

II Contributions **113**

5 Compensating correlations **115**
5.1 Correlations under noise . . . 116
5.2 Compensating dynamics with extended feature vectors . . . 118
5.2.1 The extended Gaussian . . . 120
5.2.2 Validity of optimising an extended distribution . . . 121
5.3 Compensating extended distributions . . . 124
5.3.1 Extended DPMC . . . 125
5.3.2 Extended IDPMC . . . 128
5.3.3 Extended VTS . . . 130
5.3.3.1 Relationship with VTS . . . 132
5.3.3.2 Computational cost . . . 134
5.3.4 Algonquin with extended feature vectors . . . 135
5.4 Extended joint uncertainty decoding . . . 138
5.5 Extended statistics . . . 141

5.5.1 Clean speech statistics . . . 141
5.5.2 Noise model estimation . . . 143
5.5.2.1 Zeros in the noise variance estimate . . . 146
5.5.2.2 Estimating an extended noise model . . . 147
5.5.3 Phase factor distribution . . . 149
5.6 Summary . . . 149

6 Predictive transformations **151**
6.1 Approximating the predicted distribution . . . 152
6.1.1 Per component . . . 154
6.1.2 Per sub-phone state . . . 156
6.2 Predictive linear transformations . . . 160
6.2.1 Predictive CMLLR . . . 162
6.2.2 Predictive covariance MLLR . . . 163
6.2.3 Predictive semi-tied covariance matrices . . . 164
6.3 Correlation modelling for noise . . . 166
6.3.1 Predictive CMLLR . . . 169
6.3.2 Predictive covariance MLLR . . . 170
6.3.3 Predictive semi-tied covariance matrices . . . 171
6.3.4 Computational complexity . . . 172
6.3.5 Predictive HLDA . . . 173
6.4 Front-end PCMLLR . . . 175
6.4.1 Observation-trained transformation . . . 177
6.4.2 Posterior-weighted transformations . . . 179
6.5 Summary . . . 180

7 Asymptotically exact likelihoods **181**
7.1 Likelihood evaluation . . . 182
7.2 Importance sampling over the speech and noise . . . 185
7.3 Importance sampling in a transformed space . . . 189
7.3.1 Single-dimensional . . . 190
7.3.1.1 The shape of the integrand . . . 194
7.3.1.2 Importance sampling from the integrand . . . 199
7.3.1.3 Related method . . . 202
7.3.2 Multi-dimensional . . . 202
7.3.2.1 Per-dimension sampling . . . 205
7.3.2.2 Postponed factorisation . . . 207
7.3.2.3 Quasi-conditional factorisation . . . 208
7.3.2.4 Applying sequential importance resampling . . . 210
7.4 Approximate cross-entropy . . . 212
7.5 Summary . . . 214

8 Experiments **215**
8.1 Correlation modelling . . . 216

8.1.1 Resource Management . . . 217
8.1.1.1 Compensation quality . . . 219
8.1.1.2 Extended noise reconstruction . . . 222
8.1.1.3 Per-component compensation . . . 223
8.1.1.4 Per-base class compensation . . . 225
8.1.2 AURORA 2 . . . 228
8.1.2.1 Extended VTS . . . 228
8.1.2.2 Front-end PCMLLR . . . 229
8.1.3 Toshiba in-car database . . . 232
8.2 The effect of approximations . . . 233
8.2.1 Set-up . . . 234
8.2.2 Compensation methods . . . 238
8.2.3 Diagonal-covariance compensation . . . 242
8.2.4 Influence of the phase factor . . . 244

9 Conclusion 247
9.1 Modelling correlations . . . 248
9.2 Asymptotically exact likelihoods . . . 249
9.3 Future work . . . 250

Appendices 253

A Standard techniques 255
A.1 Known equalities . . . 255
A.1.1 Transforming variables of probability distributions . . . 255
A.1.2 Matrix identities . . . 255
A.1.3 Multi-variate Gaussian factorisation . . . 256
A.2 Kullback-Leibler divergence . . . 258
A.2.1 KL divergence between Gaussians . . . 260
A.2.2 Between mixtures . . . 261
A.3 Expectation–maximisation . . . 264
A.3.1 The expectation step: the hidden variables . . . 265
A.3.2 The maximisation step: the model parameters . . . 266
A.3.3 Convergence . . . 267
A.4 Monte Carlo . . . 268
A.4.1 Plain Monte Carlo . . . 269
A.4.2 Importance sampling . . . 269
A.4.3 Sequential importance sampling . . . 272
A.4.4 Resampling . . . 275
A.4.5 Sampling from the target distribution . . . 277

B Derivation of linear transformations 279
B.1 CMLLR . . . 279
B.1.1 Adaptive . . . 280

B.1.2 Predictive . . . 281
B.2 Covariance MLLR . . . 282
B.2.1 Adaptive . . . 283
B.2.2 Predictive . . . 284
B.3 Semi-tied covariance matrices . . . 284
B.3.1 From data . . . 285
B.3.2 Predictive . . . 286

C Mismatch function **289**
C.1 Jacobians . . . 289
C.2 Mismatch function for other spectral powers . . . 290

D Derivation of model-space Algonquin **293**

E The likelihood for piecewise linear approximation **297**
E.1 Single-dimensional . . . 297
E.2 Multi-dimensional . . . 299

F The likelihood for transformed-space sampling **303**
F.1 Transforming the single-dimensional integral . . . 304
F.2 Transforming the multi-dimensional integral . . . 306
F.3 Postponed factorisation of the integrand . . . 310
F.4 Terms of the proposal distribution . . . 313

Bibliography **315**

Notation

Operators	
$\arg\max_x \phi(x)$	The value of x that maximises $\phi(x)$
$*$	Convolution
$\lvert\cdot\rvert$	Absolute value

Matrixes and vectors	
A	Matrix
$a_i = [A]_i$	Row i of A
$a_{ij} = [A]_{ij}$	Element (i, j) of A
b	Vector
b_i	Element i of b
I	Identity matrix
0	Vector/matrix with all entries 0
1	Vector/matrix with all entries 1
$\lvert\cdot\rvert$	Determinant
$\lVert\cdot\rVert$	Norm
$\mathrm{Tr}(\cdot)$	Trace
A^{-1}	Inverse
$A^\top$	Transpose
$A^{-\top}$	Inverse and transpose
$\mathrm{diag}(\cdot)$	Matrix diagonalisation
$\mathbf{exp}(\cdot)$, $\mathbf{log}(\cdot)$, $\circ$	Element-wise exponentiation, logarithm, multiplication

Distributions	
p	Real distribution
q	Approximate distribution
$\tilde{p}$	Empirical distribution
$\mathcal{E}\{a\}$	Expected value of a
$\mathcal{E}_p\{a\}$	Expected value of a under p
$\mathrm{Var}\{a\}$	Variance of a
$\mathcal{KL}(p\lVert q)$	Kullback-Leibler divergence to p from q
$\mathcal{H}(p\lVert q)$	Cross-entropy of p and q

$\mathcal{H}(p)$	Entropy of p
$\mathbb{1}(\cdot)$	Indicator function (Kronecker delta): evaluates to 1 if the argument is true and to 0 otherwise
$\delta_a(\mathbf{x})$	Dirac delta: distribution with non-zero density only at $\mathbf{x} = \mathbf{a}$, where the density is infinite, but $\int \delta_a(\mathbf{x}) d\mathbf{x} = 1$
$\mathbf{a} \sim \mathcal{N}(\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a)$	$\mathbf{a}$ is Gaussian-distributed with mean $\boldsymbol{\mu}_a$ and covariance $\boldsymbol{\Sigma}_a$
$\mathcal{N}(\mathbf{a}; \boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a)$	Gaussian density evaluated at $\mathbf{a}$
$u \sim \text{Unif}[a, b]$	u is uniformly distributed between a and b

Speech recognition signals

$\mathcal{X}$	Training data
$\mathcal{Y}$	Test data
$\mathcal{U}$	Hidden variables
$\mathbf{y}$	Observation feature vector
$\mathbf{x}$	Clean speech
$\mathbf{n}$	Additive noise
$\mathbf{h}$	Convolutional noise
$\boldsymbol{\alpha}$	Phase factor
β	Power of spectrum
$.^s$	Static feature vector
$.^\Delta, .^{\Delta^2}$	Vectors with first- and second- order dynamic features
$.^e$	Extended feature vector
$.^{\log}$	Log-mel-spectral feature vector
$\cdot[k]$	Spectral-domain signal
$\cdot[t]$	Time domain signal
$\mathbf{f}(\cdot)$	Mismatch function
$\mathbf{J}_x, \mathbf{J}_n$	Jacobian of mismatch function with respect to the speech and noise

Speech recogniser parameters

m	Speech recogniser component
$\boldsymbol{\mu}^{(m)}$	Mean of component m
$\boldsymbol{\Sigma}^{(m)}$	Covariance matrix of component m
$\boldsymbol{\Lambda}^{(m)}$	Precision matrix of component m
W	Word sequence
ϵ	Empty symbol
θ	Sub-phone state
$m \in \Omega^{(\theta)}$	Component in mixture for θ
$\gamma_t^{(m)}$	Occupancy for component m at time t
$\pi_m^{(\theta)}$	Weight of component m in the mixture for θ
$\mathcal{M}_n$	Speech Model

$\mathcal{A}$	Transformation
$\mathbf{A}$	Linear transformation
$\mathbf{b}$	Bias
$\mathbf{H}$	Linear transformation expressed on the model parameters
$\mathbf{g}$	Bias expressed on the model parameters
$\mathbf{D}$	Projection to static and dynamic coefficients
$\mathcal{L}$	Log-likelihood
$\mathcal{F}$	Lower or upper bound

Indices

t	Time
w	Window size
r	Front-end component
r	Base class
$m \in \Omega^{(r)}$	Component associated with base class r
k	Iteration
d	Feature vector length
s	Static feature vector length
k	Fourier frequency
i	Filter bin index

Monte Carlo

γ	Integrand
π	Normalised target distribution
Z	Normalising constant
ρ	Proposal distribution
$\mathbf{u}$	Sample
l	Sample index
w	Sample weight

Chapter 1

Introduction

Automatic speech recognition is employed in many places. The most popular operating system for personal computers, Microsoft Windows, has shipped with a dictation system for years. Google's Android operating system for phones now supports using speech to run commands and input text. Mobile phones in particular are likely to be used in noisy places. This tends to highlight that speech recognisers are more sensitive to noise than humans: performance deteriorates quickly under noise that hardly affects human speech recognition (Lippmann 1997).

There are two approaches to make speech recognisers more robust to noise. One is to reconstruct the clean speech before it enters the speech recogniser. The other is to compensate the speech recogniser so it expects noise-corrupted rather than clean speech.

The first approach is called *feature enhancement*. Since it aims to reconstruct the clean speech feature vectors and then passes them to the unchanged speech recogniser, it is usually fast. However, propagating just a point estimate of the clean speech discards the information on the uncertainty of the estimate, which is important. For example, loud noise can mask quiet speech completely. Though this makes the clean speech estimate meaningless, the speech recogniser treats it as the real clean speech, which causes recognition errors.

The other approach, *model compensation*, is the focus of this thesis. It has ob-

tained better performance than feature enhancement, especially under low signal-to-noise ratios. Model compensation finds a distribution over the noise-corrupted speech. It computes this from a trained speech recogniser, which constitutes a model of the training data, and an estimated model of the noise. Most model compensation methods replace each clean speech Gaussian with a corrupted speech Gaussian with a diagonal covariance matrix. This is an approximation. If the speech and noise distributions were correct and the replacement distribution were the exact one, then according to the Bayes decision rule the speech recogniser would yield the optimal hypothesis.

This work will therefore look into modelling the corrupted speech more precisely than with a diagonal-covariance Gaussian. It will look into two different aspects. First, diagonalising covariance matrices neglects the changes in correlations between feature dimensions under noisy conditions. However, in reality, feature correlations do change. For example, in the limit as the noise masks the speech, the correlations of the input data become equal to those of the noise. The first part of this thesis will therefore estimate full-covariance Gaussians. The second part derives from the observation that given a standard form for the relationship between the corrupted speech and the speech and noise, the corrupted speech is not Gaussian even if the speech and noise are. Rather than using a parameterised distribution, it will approximate the corrupted speech likelihood directly, with a sampling method. The following outlines the contributions of this thesis. Publications that have arisen from the Ph.D. work are indicated with "(published as van Dalen and Gales 2009*b*)".

The first part will find compensation that models correlation changes under noise. The obvious approach, which is to forgo the diagonalisation of estimated corrupted speech covariances, will encounter two problems. The first problem is that common approximations to estimate parameters for *dynamic coefficients* cause off-diagonal elements of the covariance matrices to be misestimated. These dynamics indicate the change in the signal from time slice to time slice. They are computed from a window of per-time slice, *static*, coefficients. The state-of-the-art *VTS* compensation scheme

uses the *continuous-time approximation*, which simplifies computation by assuming the dynamic coefficients to be time derivatives. Chapter 5 will propose using distributions over vectors of all static features in a window, called **extended** feature vectors. It proposes **extended DPMC** (published as van Dalen and Gales 2008) and **extended VTS** (published as van Dalen and Gales 2009*b*;*a*; 2011). These compensation schemes, derived from the DPMC and VTS compensation schemes, compute the effect of the noise on each time instance separately and then perform the mapping to statics and dynamics. The more precise modelling enables speech recognisers to use full covariance matrices to model correlation changes.

Having estimated full covariance matrices, decoding with full covariances is slow. Chapter 6 will therefore propose a general mechanism to approximate one speech recogniser parameterisation with another one. This effectively trains the latter on predicted statistics of the former. Chapter 6.2 will re-analyse **predictive transformations** (first introduced in van Dalen 2007; Gales and van Dalen 2007), as minimising the Kullback-Leibler divergence between the two models. This is used to convert full-covariance compensation into a tandem of a linear transformation and a diagonal bias on the covariance matrices. Since the covariance bias is diagonal, once compensation has finished, decoding will be as fast as normal. When combined with a scheme that applies compensation to clusters of Gaussians at once, *joint uncertainty decoding*, compensation is also fast. The choice of the number of clusters provides a trade-off between compensation accuracy and decoding speed. The combination of a compensation scheme with extended feature vectors, joint uncertainty decoding, and predictive linear transformations yields practical schemes for noise-robustness.

The second part of this thesis is more theoretical. The corrupted speech distribution is not Gaussian, even though most model compensation methods assume it is. There has been no research on how good model compensation could be with speech and noise models that modern recognisers use. Chapter 7 will therefore investigate how well speech recognisers would cope with noise if they used the exact corrupted speech distribution. It turns out that this is impossible: there is no closed form

for the corrupted speech distribution. However, a useful approach is to re-formulate the problem. In practice, no closed form for the distribution is necessary: a speech recogniser only needs likelihoods for the observation vectors in the data. These likelihoods can be approximated with a cascade of transformations and a Monte Carlo method, sequential importance sampling. As the size of the sample cloud increases, the approximation converges to the real likelihood. Section 7 will discuss this **transformed-space sampling** (published as van Dalen and Gales 2010*a*;*b*) in detail.

In coming close to the real likelihood, transformed-space sampling becomes so slow that implementing a speech recogniser with it is infeasible. It is, however, possible to make a more fine-grained assessment of speech recogniser compensation, with a metric based on the KL divergence (section 7.4). In the limit, the new sampling method will effectively give the point where the KL divergence is zero, which is otherwise not known. This calibration will make it possible to determine how far well-known compensation methods are from the ideal. This work will examine how well the KL divergence predicts speech recogniser word error rate. It will compare different compensation schemes, and examine the effect of common approximations. This includes assuming the corrupted speech distribution Gaussian and diagonalising its covariance matrix, and approximations to the mismatch function. This illustrates that the new method is an important new research tool.

Part I

Background

Chapter 2

Speech recognition

Speech recognition is the conversion of speech into text. "Text" here means a sequence of written words. This chapter will give an overview of speech recognition.

A speech recogniser first converts the incoming audio into a sequence of feature vectors that each represent a fixed-duration time slice. Section 2.1 will discuss the standard type of features that this thesis will use: MFCCs. The feature vectors serve as observations to a generative probabilistic model, the topic of section 2.2, that relates them to sequences of words. Section 2.3 will then explain how the acoustic part of the model, the focus of this thesis, is trained. The process of finding the best word sequence from the observation sequence, *decoding*, will be the topic of section 2.4.

2.1 Feature extraction

The initial digital representation of an audio signal is a series of amplitude samples expressing the pressure waveform at, say, 8 or 16 kHz. Speech recognisers apply a number of transformations to translate the stream of samples into a sequence of low-dimensional feature vectors. The standard type of feature vector for a time instance contains mel-frequency cepstral coefficients (MFCCs) extracted from one time slice (section 2.1.1). Appended to these "static" coefficients are dynamic coefficients that represent the changes in the static features (section 2.1.2).

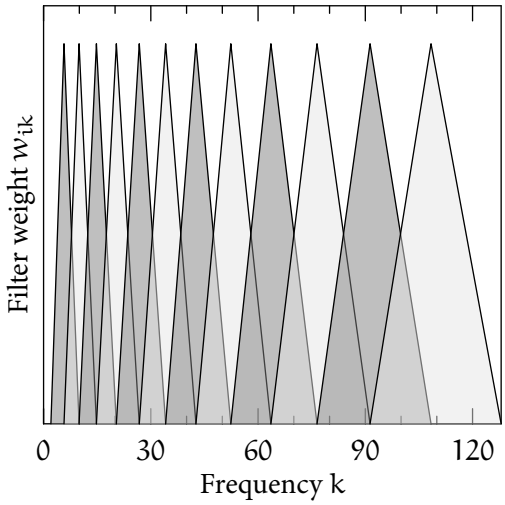


Figure 2.1 *Mel-spaced filter bins. Alternate filters are coloured differently.*

2.1.1 *Mel-frequency cepstral coefficients*

The objective of feature extraction is to represent the audio signal at discrete intervals by a small number of features that are useful for speech recognition. The procedure that produces mel-frequency cepstral coefficients (Davis and Mermelstein 1980) first finds coefficients that indicate the energy in human-hearing-inspired frequency bands. It then converts these into (usually 24) features that have decreasing relation on the shape of the mouth and increasing relation with voice qualities, and retains only the first (usually 13). The following goes into more detail.

The spacing of the feature vectors, usually 10 ms, is chosen so that for one time slice the spectrum can be assumed to be stationary. A Hamming window is applied to a short time slice (of usually 25 ms). A Fourier transformation of the signal waveform x[t] results in the spectrum X[k] representing the audio in that time slice. The phase information of the spectrum is then discarded by taking the magnitude, or a power, of the spectrum. The spectrum has a resolution higher than necessary for representing the shape of spectrum for speech recognition. Triangular filters (usually 24) on a mel-scale, which imitates the varying resolution of the human ear, are therefore applied. Figure 2.1 contains a depiction of filters on the mel-scale. Lower-indexed bins

are narrower, and span fewer frequencies, than higher-indexed bins. Describing filter bin i with filter weights w_{ik} for frequencies k, the mel-filtered spectrum with is

$$\tilde{X}_i = \sum_k w_{ik} |X[k]|^\beta \,. \tag{2.1}$$

Usual values for the power β are 1 (indicating the magnitude spectrum) or 2 (the power spectrum). Filter bank coefficients $\tilde{X}_i$ are called mel-spectral coefficients.

The next step is motivated by a source-filter model. The source models the vocal cords, and the filter models the mouth. Since in speech recognition, voice quality is mainly irrelevant but the shape of the mouth is of interest, attributes influenced by the source should be disregarded and attributes influenced by the filter retained. In the time domain, the filter convolves the source signal; in the spectral domain, this becomes multiplication. Then, the logarithm of each the filter bank coefficients is found:

$$\boldsymbol{x}^{\log} = \mathbf{log}\left(\begin{bmatrix} \tilde{X}_1 \\ \vdots \\ \tilde{X}_I \end{bmatrix}\right), \tag{2.2}$$

with $\mathbf{log}(\cdot)$ denoting the element-wise logarithm. The resulting vector $\boldsymbol{x}^{\log}$ represents the log-spectrum.

In this domain, the source and the filter are additive. It is assumed that the filter determines the overall shape of the elements of $\boldsymbol{x}^{\log}$. To separate it from the source, frequency analysis can be performed again, this time on $\boldsymbol{x}^{\log}$. This uses the discrete cosine transform (DCT), which is a matrix $\mathbf{C}_{[I]}$, the elements of which are defined as

$$c_{ij} = \sqrt{\tfrac{2}{I}} \cos\left(\frac{(2j-1)(i-1)\pi}{2I}\right). \tag{2.3}$$

By taking the discrete cosine transform of $\boldsymbol{x}^{\log}$, an I-dimensional vector of mel-frequency cepstral coefficients $\boldsymbol{x}^{s}_{[I]}$ is found:

$$\boldsymbol{x}^{s}_{[I]} = \mathbf{C}_{[I]} \boldsymbol{x}^{\log}. \tag{2.4}$$

The following step is to discard the higher coefficients in the feature vector. One way of viewing the effect of this is by converting back to log-spectral features. (In section 4.2

this will actually be done to express the effect of noise.) Back in the log-spectral domain, higher-frequency changes from coefficient to coefficient, which are assumed to relate to the source more than to the filter, have been removed. Discarding the last part of the MFCC feature vector therefore has the effect of smoothing the spectrum.

Truncating $\mathbf{x}^{s}_{[I]}$ to the first s elements,

$$\mathbf{x}^{s}_{[s]} = \mathbf{C}_{[s]}\mathbf{x}^{\log}, \tag{2.5}$$

where $\mathbf{C}_{[s]}$ is the first s rows of $\mathbf{C}$, and $s \leq I$. For notational convenience, (2.5) will be written as

$$\mathbf{x}^{s} = \mathbf{C}\mathbf{x}^{\log}. \tag{2.6}$$

These features form the "cepstrum". To represent the sequence of operations leading to them in noun compound summary format, the features are called "mel-frequency cepstral coefficients" (MFCCs).

MFCCs are popular features in speech recognition. The first coefficient represents a scaled average of the coefficients in the log-spectral feature vector in (2.2). This coefficient is often replaced by the normalised total energy in the filter bank. However, like most work on noise-robust speech recognition, this thesis will use feature vectors with only MFCCs.

2.1.2 *Dynamic coefficients*

It will be discussed in section 2.2 that the feature vectors will be modelled with a *hidden Markov model* (HMM). Hidden Markov models assume that given the sequence of *states* that generate the feature vector, consecutive observations are independent. This implies that consecutive feature vectors generated by a single sub-phone are independent and identically distributed. Thus, no information about changes over time is encoded, other than through switching sub-phone states. To alleviate this problem, consecutive feature vectors can be related by appending extra coefficients that approximate time derivatives (Furui 1986). Usually, "delta" and "delta-delta" coefficients are

included, representing feature velocity and acceleration, respectively. These dynamic coefficients lead to large improvements in recognition accuracy and are standard in speech recognition systems.

As an approximation to time derivatives, linear regression over a window of consecutive frames is used. For exposition, assume a window ± 1 around the static feature vector $\mathbf{x}_t^s$. The dynamic coefficients $\mathbf{x}_t^\Delta$ are then found by

$$\mathbf{x}_t^\Delta = \begin{bmatrix} -\frac{1}{2}\mathbf{I} & \mathbf{0} & \frac{1}{2}\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{t-1}^s \\ \mathbf{x}_t^s \\ \mathbf{x}_{t+1}^s \end{bmatrix} = \mathbf{D}^\Delta \mathbf{x}_t^e, \tag{2.7a}$$

where $\mathbf{x}_t^e$ is an *extended* feature vector, which is a concatenation of the static coefficients in a window, and $\mathbf{D}^\Delta$ is the projection from $\mathbf{x}_t^e$ to $\mathbf{x}_t^\Delta$. Then, a feature vector with statics and first-order dynamics ("deltas") is

$$\mathbf{x}_t = \begin{bmatrix} \mathbf{x}_t^s \\ \mathbf{x}_t^\Delta \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} \\ -\frac{1}{2}\mathbf{I} & \mathbf{0} & \frac{1}{2}\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{t-1}^s \\ \mathbf{x}_t^s \\ \mathbf{x}_{t+1}^s \end{bmatrix} = \begin{bmatrix} \mathbf{D}^s \\ \mathbf{D}^\Delta \end{bmatrix} \mathbf{x}_t^e = \mathbf{D}\mathbf{x}_t^e, \tag{2.7b}$$

where $\mathbf{D}^s$ projects $\mathbf{x}_t^e$ to $\mathbf{x}_t^s$, and $\mathbf{D}$ projects $\mathbf{x}_t^e$ to the final feature vector $\mathbf{x}_t$. With the window of one time instance left and one right, this uses *simple differences* between static feature vectors.

In general, first-order coefficients $\mathbf{x}_t^\Delta$ are found from consecutive static coefficients $\mathbf{x}_{t-w}^s, \ldots, \mathbf{x}_{t+w}^s$ by

$$\mathbf{x}_t^\Delta = \frac{\sum_{i=1}^{w} i\left(\mathbf{x}_{t+i}^s - \mathbf{x}_{t-i}^s\right)}{2\sum_{i=1}^{w} i^2}. \tag{2.8}$$

Second-order coefficients $\mathbf{x}_t^{\Delta^2}$ are found analogously from $\mathbf{x}_{t-v}^\Delta, \ldots, \mathbf{x}_{t+v}^\Delta$, and similar for higher-order coefficients. The extended feature vector $\mathbf{x}_t^e$ then contains the statics in a window $\pm(w+v)$. The transformation from $\mathbf{x}_t^e$ to feature vector $\mathbf{x}_t$ with these higher-order terms remains linear, so that $\mathbf{D}$ in (2.7b) is straightforwardly generalised. The feature vector with static and first- and second-order dynamic coefficients

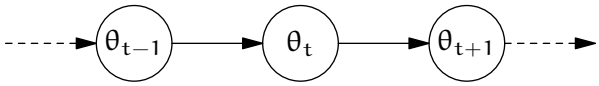


Figure 2.2 *Graphical model of a Markov model. Circles represent random variables.*

("deltas" and "delta-deltas") is then computed with

$$\boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{x}_t^s \\ \boldsymbol{x}_t^\Delta \\ \boldsymbol{x}_t^{\Delta^2} \end{bmatrix} = \mathbf{D}\boldsymbol{x}_t^e, \qquad \boldsymbol{x}_t^e = \begin{bmatrix} \boldsymbol{x}_{t-w-v}^s \\ \vdots \\ \boldsymbol{x}_{t+w+v}^s \end{bmatrix}. \tag{2.9}$$

For exposition, however, this work assumes (2.7b).

2.2 Hidden Markov models

To describe the relation between word sequences and feature vector sequences, speech recognisers use a generative model. This is a joint distribution of word sequence W and feature vector sequence $\mathbf{X}$, $p(W, \mathbf{X})$. In such a model, the words generate feature vectors, called observations. A problem is that a word has a variable duration, and therefore the number of observation vectors it generates is also variable. To solve this, a latent discrete sequence of fixed-interval units is introduced. The observation at time t is assumed generated by the *state* at time t, written θ_t. Denoting the sequence of states with Θ,

$$p(\mathbf{X}, W) = \sum_{\Theta} p(\mathbf{X}|\Theta)\, P(\Theta, W)\,. \tag{2.10}$$

The distribution $P(\Theta|W)$ performs a mapping from a sequence of words to a sequence of states. To make training (section 2.3) and recognition (section 2.4) feasible, the Markov property is assumed: the state at time t only depends on its predecessor, not on further history. Figure 2.2 has a simple graphical model of this, with θ_t a random variable that represents the active state at time t. Section 2.2.1.2 will give more detail on how the transition probabilities are determined.

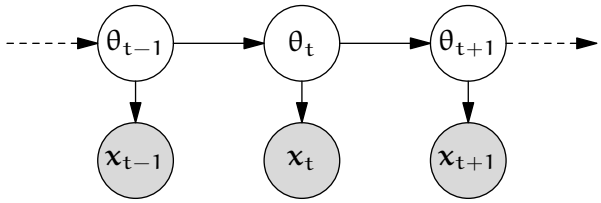


Figure 2.3 *Graphical model of a hidden Markov model. The shaded circles represent observed variables.*

To include the acoustic information in this model, the feature vectors, representing the acoustics at time t, are attached. Given the state sequence, the feature vectors are assumed independent. Because when training or decoding the feature vectors generated by the states are observed, but not the states themselves, the resulting model is called a *hidden Markov model* (HMM). Figure 2.3 shows the graphical model. Two important properties of the HMM are the Markov property, and the conditional independence assumption: the observation at time t only depends on the state at time t.

The arrow from state θ_t to observation $\mathbf{x}_t$ in figure 2.3 represents the state output distribution. This distribution, $q^{(\theta)}(\mathbf{x})$, is usually a mixture of Gaussians:

$$q^{(\theta)}(\mathbf{x}) = \sum_m \pi_m^{(\theta)} q^{(m)}(\mathbf{x}), \qquad \sum_m \pi_m^{(\theta)} = 1, \tag{2.11}$$

where $\pi_m^{(\theta)}$ is the mixture weight for mixture θ and component m, and $q^{(m)}$ the component's Gaussian distribution. This can also be expressed as a graphical model: see figure 2.4 on the following page. The component that generates the observation is indicated by a random variable m_t. Figures 2.3 and 2.4 represent different ways of looking at the same speech recogniser, but showing the components m_t explicitly as in figure 2.4 is useful for training, and making them implicit in the distribution of $\mathbf{x}_t$ for decoding.

Components are usually multi-variate Gaussians, which are parameterised with mean $\boldsymbol{\mu}_x^{(m)}$ and covariance $\boldsymbol{\Sigma}_x^{(m)}$:

$$\begin{aligned} q^{(m)}(\mathbf{x}) &= \mathcal{N}\big(\mathbf{x};\, \boldsymbol{\mu}_x^{(m)},\, \boldsymbol{\Sigma}_x^{(m)}\big) \\ &= \big|2\pi\boldsymbol{\Sigma}_x^{(m)}\big|^{-\frac{1}{2}} \exp\Big(-\tfrac{1}{2}\big(\mathbf{x}-\boldsymbol{\mu}_x^{(m)}\big)^{\mathsf{T}} \boldsymbol{\Sigma}_x^{(m)^{-1}} \big(\mathbf{x}-\boldsymbol{\mu}_x^{(m)}\big)\Big). \end{aligned} \tag{2.12}$$

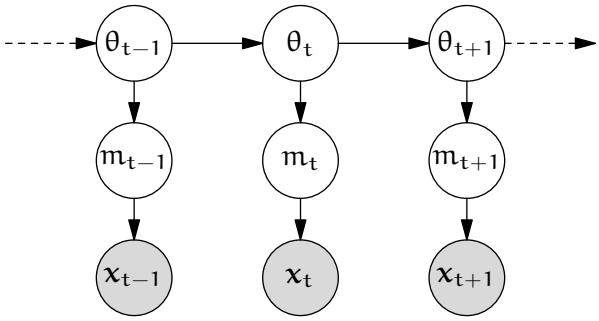


Figure 2.4 *Graphical model of a hidden Markov model with mixture models as output distributions.*

Sometimes it is more useful to express the Gaussian in terms of the inverse covariance. This *precision matrix* will be written $\boldsymbol{\Lambda}_x^{(m)} = \boldsymbol{\Sigma}_x^{(m)^{-1}}$.

The covariance is often constrained to be diagonal, so that one Gaussian requires less data to estimate. This does not model within-component correlations that are there. To model some correlations while reducing the amount of training data required, structure can be introduced in covariances. This will be the topic of section 3.3.

2.2.1 *State sequences*

To use HMMs in a speech recogniser, it must define probabilities for state sequences and relate them to words. That is, it must define $P(\Theta, W)$ in (2.10). It is convenient to write this distribution in terms of the prior distribution $P(W)$ and the likelihood of the state sequence $P(\Theta)\, W$:

$$P(\Theta, W) = P(W)\, P(\Theta|W)\,. \tag{2.13}$$

The distribution over word sequences $P(W)$ is called the *language model*.[1] The distribution $P(\Theta|W)$ performs a mapping from a sequence of variable-length words to a sequence of fixed-length states. It is possible, if not very insightful, to describe this mapping with a graphical model (Murphy 2002; Wiggers *et al.* 2010). A more insightful method is to use *weighted finite state transducers*.

[1] Whereas in linguistics speech is considered the only real form of language and spelling a confusing artefact, in computational linguistics "language" refers to written text.

Thus, section 2.2.1.1 will describe the language model in terms of a probabilistic model, $P(W)$ and section 2.2.1.2, will describe $P(\Theta|W)$ as a composition of weighted finite state transducers.

2.2.1.1 *Language modelling*

In the generative model in (2.10), the probability of each possible word sequence must be defined. For tasks where the inputs are constrained, this is often straightforward. For example, if a ten-digit phone number is expected, the probability of any such digit sequence can be set to the same value, and any other word sequence can receive a zero probability. For some tasks it is possible to set up a limited grammar. However, if free-form input is expected, no word sequences are impossible, though some may be improbable. Higher probabilities may be assigned to grammatical sentences (or fragments) compared to ungrammatical utterances, and semantically likely sequences compared to unlikely ones. This is often done by training a statistical model on data.

A language model can be seen as a probability distribution over word sequences. If sentences are considered independently, the probability of a sentence $W = w_1, \ldots, w_L$ is[2]

$$\begin{aligned} P(W) &= P(w_1)\, P(w_2|w_1)\, P(w_3|w_1, w_2) \cdots P(w_L|w_1, \ldots, w_{L-1}) \\ &= \prod_{i=1}^{L} P(w_i|w_1, \ldots, w_{i-1})\,. \end{aligned} \tag{2.14}$$

This factors the probability of a word sequence into probabilities of each word conditional on the word history, which can be trained from data. The usual strategy is to apply maximum-likelihood estimation, which in this case sets probabilities of words for a given word history proportional to its count in the training data. However, though there may be enough data to do this for zero-length histories, the last word depends on all previous words, which most likely occur only once in the training data and never in the test data. This is an example of over-training: the trained model would assign a

[2] Boundary effects are ignored for simplicity.

zero probability to most sentences. The word history is therefore usually constrained to $N - 1$ words:

$$P(w_i|w_1, \ldots, w_{i-1}) \simeq P(w_i|w_{i-N+1}, \ldots, w_{i-1}) \,. \tag{2.15}$$

This type of model is called an N-gram model, and a typical value for N is 3.

For $N \geq 2$, usually not all word tuples have been seen in training data, so it is often necessary to recursively *back off* to data from a shorter word history, or to interpolate between N-gram models for different values of N. A state-of-the-art back-off scheme is *modified Kneser-Ney* (Chen and Goodman 1998). An alternative method to guard against over-training is to apply Bayesian methods, where a prior over parameters keeps them from assigning all probability mass to seen events. An interesting approach derives from a *hierarchical Pitman–Yor process* (Teh 2006).

2.2.1.2 *Latent discrete sequence*

To map words onto fixed-time units, words are converted to sequences of fixed-time discrete states. The state space can be described in various ways, but an insightful one is as a network of states. A formalism that produces these networks (and can be fast when embedded in a speech recogniser) is that of *weighted finite state transducers*. Mohri *et al.* (2008) gives a good overview of how this works. The following will briefly describe how to construct a state network for speech recognition.

A finite-state automaton has discrete states that it switches between at each time step. Which transitions are possible is specified explicitly. Finite state transducers add to each transition an input symbol and an output symbol. They are therefore able to convert one type of symbol sequence into another. It is allowable for a transition to either not input any symbol, or not to output any. An empty input or output is indicated with "ϵ". This is useful if input and output sequences have different lengths, as when converting from words to states. Weighted finite state transducers, finally, add weights to transitions. For the sake of clarity, the weights will here stand for probabilities that are multiplied at each transition. This section will use weighted finite state transducers to convert word sequences into state sequences.

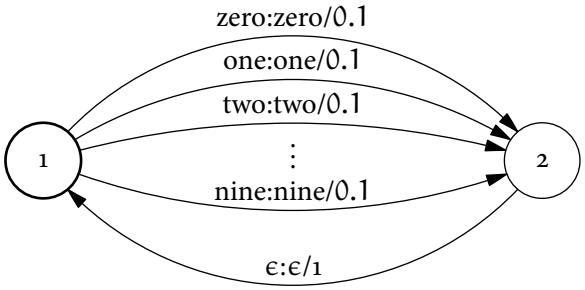
Transducer	Result	Weight
Input sequence	three two	
Language model	three two	0.01
Pronunciation lexicon	θ r i: t u:	0.01
Sub-phones	θ1 θ2 θ2 θ3 r1 r2 r3 r3 r3 i:1 i:1 i:2 i:3 t1 t2 t2 t2 t3 t3 u:1 u:1 u:2 u:2 u:2 u:3 u:3	$5.8 \cdot 10^{-20}$

Table 2.1 *Conversions that the weighted finite state transducers in figure 2.5 on the following page, applied after one another, may apply, starting from an input sequence.*

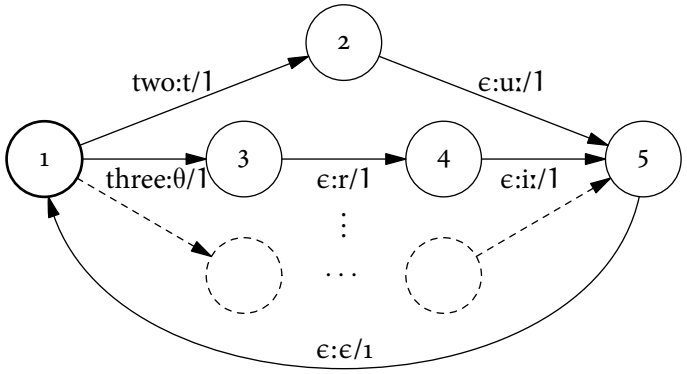
Figure 2.5 on the next page illustrates component weighted finite state transducers that when composed can convert a sequence of words to a sequence of discrete equally-spaced states. The transducers in the chain translate between a number of alphabets. The output alphabet of the one must be the input alphabet of the other, et cetera. The arbitrarily numbered circles represent states. The bold circles are start states, and the double-bordered ones end states, which can, but do not necessarily, end the sequence. The arrows represent transitions. Their labels consist of an input symbol, a colon (:), an output symbol, a slash (/), and a transition weight (here, a probability).

Figure 2.5a contains a simple language model that can take digit sequences. Its input and output symbols at each transition are equal. Table 2.1 shows the effect of applying this transducer to a sample sequence (first and second rows): it merely computes a weight. This weight stands for the probability of the word sequence. It has a weight of 0.1 for each word, and a transition from state 2 to 1, without consuming or producing any symbols, to allow repetition. It is straightforward to use the same ingredients to generate a representation of a fixed grammar, or of a probabilistic language model trained on text data.

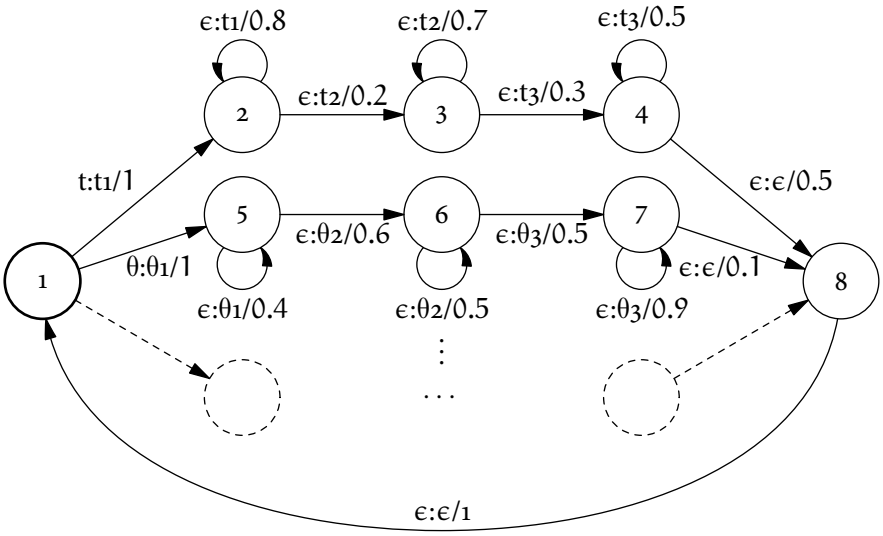
For a small vocabulary, it is possible to train the acoustics of every word from audio separately. In the digit sequence example, this could be an option, if the data contains enough examples of all words. However, in general words are mapped to a sequence consisting of symbols from a smaller alphabet with a pronunciation dictionary. These sub-word units are supposed to represent the pronunciation of the words.



(a) Simple language model for a digit sequence.



(b) Simple pronunciation lexicon: mapping words to phones.



(c) Mapping phones to sub-phones.

Figure 2.5 *Component weighted finite state transducers for building a speech recognition network.*

However, standard linguistic units are usually considered to be at the wrong level of detail. There are *phonemes*, which are supposed to encode an underlying representation. Thus, the second *c* in "electri*c*" and in "electri*c*ity" could be encoded with the same symbol (see, e.g., Harris 1994), even though they are always pronounced differently. On the other hand, *phonetic* transcriptions encode differences due to accents and sheer chance. Thus, it changes the transcription whether the last sound in a realisation of the word "hands" sounds like a *z* or a *s*, or changes halfway through (see, e.g., Collins and Mees 1999). It also encodes allophonic differences, like in realisations of the *l* in the words "*cl*ear", "voice*l*ess', and "effu*l*gent". Speech recognisers' acoustic models are powerful enough to deal with part of the pronunciation variability, and it is also possible to encode sound context (see below). A level of transcription in between phonemic and phonetic is therefore usually chosen for the sub-word units. These units are referred to with the linguistically neutral term *phone*.

The third row of table 2.1 shows what the mapping from a word sequence into a sequence of phones produces. Figure 2.5b shows a weighted finite state transducer that performs the mapping. Because most words consist of more than one phone, the transducer needs to generate output symbols without consuming any input; this is indicated on the transitions by "ϵ" for the input symbol. It does not make a difference in theory which transition on a deterministic path carries the non-empty input. For performance reasons, practical speech recognisers will apply operations on the transducer (for more details, see Mohri *et al.* 2008) to move the word symbol further back. Since there is no pronunciation variation that needs encoding, the transducer here is deterministic, with weights 1 throughout. It is, however, possible to include alternative phone sequences for one word with the appropriate probabilities.

Movement of articulators is a continuous process. To represent the resulting change in acoustics during the realisation of a phone, with discrete units, phones are split into sub-phones. There must be a balance between time resolution of the acoustics and the amount of training data available for each sub-phone. The canonical number of sub-phones per phone is therefore three.

Figure 2.5c shows part of a weighted finite state transducer that converts phones (e.g. "θ") into sub-phones (e.g. "θ2"). Since one phone generates more than one sub-phone, many transitions take no input symbol, which is indicated with "ϵ". Some states have self-transitions, which produce the same sub-phone label every time they are chosen. This makes the transducer non-deterministic and allows the sub-phone sequence to have varying lengths. Duration modelling is not very sophisticated: computational constraints practically dictate a geometric distribution for the sub-phone duration on one path through the network. The weight on a self-transition is the parameter of this geometric distribution.

The bottom row of table 2.1 contains an example sub-phone sequence derived from the phone sequence. Each of the sub-phones produces one time slice, of which section 2.2 will discuss the properties.

There are a number of additional steps in producing a real-world speech recogniser. One is to introduce context-sensitive phones. This divides phones up depending on previous and following phones, which is straightforward to implement as a finite state transducer. To combat the resulting explosion of the number of parameters, it is usual to map phone or sub-phone models that share properties and acoustics into equivalence classes. A decision tree is built that for each split picks from a list the phonetically-inspired question that best separates the data. For example, an *l* like the one in "effu*l*gent" may be separated from other allophones of *l* if the question "Is the next phone a consonant?" appears in the decision tree. Once the mapping into equivalence classes has been found, a finite state transducer straightforwardly performs the conversion.

Another trick, used for decoding, is to apply *acoustic deweighting*. The acoustic model's probabilities have too great a dynamic range, so that they swamp out the language model's probabilities. The usual work-around is to take the acoustic model's transition weights to a power smaller than 1 before multiplying them with the language model's.

The discussion has so far considered separate transducers that take an input se-

quence and produce an output sequence. However, when decoding, many paths must be considered at the same time, through the whole cascade of transducers at once, and it is the inverse transducers that are necessary. A transducer is inverted by exchanging input and output symbols on all transitions. Following many paths in a cascade of transducers is less easy.

Conceptually, the transducers are composed into one big transducer that consumes a word sequence and non-deterministically produces a sub-phone sequence. As long as the empty symbol ϵ is not considered, composing two transducers straightforwardly yields a new transducer. Its state space is the product space of the two transducers'. For transducers with ϵ-transitions, performing composition is less straightforward, but possible (Mohri *et al.* 2008). It is also often beneficial to *determinise* and *minimise* the transducers so that, for example, words in the pronunciation lexicon share states as much as possible. These operations are generic, but it is also possible to use algorithms for specific network types (e.g. Dobrišek *et al.* 2010). It is possible to expand the whole network for a system with a large vocabulary and language model. However, this often requires much memory. Alternatively, though it is non-trivial, the composition operation can be performed on the fly, by introducing *filters* (Oonishi *et al.* 2009; Allauzen *et al.* 2009), some of which are necessary for correct operation, and some increase decoding speed. Transducer composition is an associative operation. This allows some of the composition operations to be performed off-line, and the resulting network to be stored, and the rest to be done on the fly.

It is possible to express training a speech recogniser and decoding with it as operations on weighted finite state transducers. This requires setting up a linear transducer, with states that represent times, and transitions that convert all sub-phones into the feature vector found between these times, with the correct probabilities. However, this is not the most enlightening way of looking at this. The following section will use the active sub-phone at any given time as a random variable, probabilities of sequences of which are governed by the fully composed weighted finite state transducer.

2.3 Training

Speech recognisers, like most statistical models in machine learning, are trained on data. The criterion used to optimise the parameters is usually the likelihood. This is sometimes followed by discriminative training. However, since in this work the noise model must be estimated on unlabelled data, maximum-likelihood estimation will be used. This is only consistent if the speech model is generatively trained, so this thesis will only use maximum-likelihood estimation for the speech model.

The objective is usually to find the model parameters that maximise the likelihood of the labelled training data. Section 2.3.2 will discuss maximum-likelihood estimation and its instantiation for models with hidden parameters, expectation–maximisation. Section 2.3.3 will discuss how expectation–maximisation is applied to speech recognisers.

Maximum-likelihood estimation, which normally uses training data, can be extended in two ways that will be important for this work. First, it is possible to adapt the model parameters to unlabelled audio that is to be recognised rather than labelled data. This will be the topic of chapter 3. Second, chapter 6 will introduce *predictive* methods, which train parameters not from data, but on predicted distributions.

The generalisation of methods that implement maximum-likelihood estimation to training distributions requires an unusual presentation. The training data will be written as a distribution of samples, an *empirical distribution*, which will be the topic of section 2.3.1.

2.3.1 Empirical distributions

It is often useful to approximate distributions by a number of samples. In this work, these approximations will be called *empirical distributions*. If p is the real distribution over $\mathbf{u}$, then its empirical approximation $\tilde{p}$ is defined by L samples $\mathbf{u}^{(l)}$:

$$p \simeq \tilde{p} = \frac{1}{L} \sum_{l} \delta_{\mathbf{u}^{(l)}}, \tag{2.16}$$

where $\delta_{\mathbf{u}^{(l)}}$ indicates a Dirac delta at $\mathbf{u}^{(l)}$. Sometimes, the samples will be weighted. This work will use empirical distributions for two purposes.

The first is in the well-known Monte Carlo family of algorithms. As models in machine learning get more complicated, it quickly becomes infeasible to perform exact inference. Sometimes distributions can be approximated parametrically, with a choice of forms and types of approximation (e.g. Minka 2005). Monte Carlo algorithms, on the other hand, replace a parameterised distribution by an empirical distribution acquired by sampling from it. Appendix A.4 discusses Monte Carlo methods that approximate integrals. However, even for message passing in general graphical models, distributions can be represented with samples (Dauwels *et al.* 2006). For example, Gibbs sampling (Geman and Geman 1984) can then be seen as loopy belief propagation (Frey and MacKay 1997) with single-sample messages.

The main use for empirical distributions in the next sections, however, is to represent training data. Data points (in this work: audio recordings of speech utterances) can be interpreted as samples from a stochastic process. This stochastic process, speech production, is hard to approximate and arguably impossible to model exactly. The next section will interpret maximum-likelihood estimation using the empirical distribution representing the training data, where every utterance is a data point.

2.3.2 *Maximum-likelihood estimation*

Training speech recogniser parameters, whether on thousands of hours of data with transcriptions, or a few parameters for adaptation on a few seconds, usually applies *maximum-likelihood estimation* or an approximation. This sets the model parameters to maximise the likelihood of the training data. The distribution that the model represents will be written $q_{\mathcal{X}}$, and one training data point $\mathcal{X}$. For mathematical convenience, the maximisation of the likelihood is usually rephrased as a maximisation of the log-likelihood, which will be written $\mathcal{L}(\cdot)$. The log-likelihood of data point $\mathcal{X}$

according to $q_{\mathcal{X}}$ is

$$\mathcal{L}(\mathcal{X}, q_{\mathcal{X}}) \triangleq \log q_{\mathcal{X}}(\mathcal{X}). \tag{2.17}$$

The likelihood of a set of independent and identically distributed data points $\{\mathcal{X}^{(l)}\}$ is the product of the likelihoods of the points, so the log-likelihood of that set is the sum of the individual log-likelihoods:

$$\log \prod_{l} q_{\mathcal{X}}\left(\mathcal{X}^{(l)}\right) = \sum_{l} \log q_{\mathcal{X}}\left(\mathcal{X}^{(l)}\right). \tag{2.18}$$

In chapter 6, methods trained on data with maximum-likelihood estimation will be generalised to train on distributions. It is therefore useful at this stage to write the training data as a distribution. This empirical distribution $\tilde{p}(\mathcal{X})$ has a Dirac delta at each point in the training set, as in (2.16). The log-likelihood of the training data can then be written

$$\mathcal{L}(\tilde{p}, q_{\mathcal{X}}) \triangleq \int \tilde{p}(\mathcal{X}) \log q_{\mathcal{X}}(\mathcal{X}) \mathrm{d}\mathcal{X}. \tag{2.19}$$

Maximum-likelihood estimation then finds

$$\hat{q}_{\mathcal{X}} = \arg\max_{q_{\mathcal{X}}} \mathcal{L}(\tilde{p}, q_{\mathcal{X}}). \tag{2.20}$$

Maximum-likelihood estimation invites over-fitting of the training data, which endangers generalisation. To guard against this, Bayesian approaches are possible, which factor in a prior over the parameters. However, because of the temporal structure of speech recognisers, using a distribution over parameters is not feasible and must be approximated (Watanabe *et al.* 2004). Instead, speech recognition therefore uses techniques that control the amount of data that parameters are trained on.

Many learning problems in statistical pattern processing have unobserved, hidden, variables. Finding the parameters of the distributions over both the hidden and the observed parameters that maximise the likelihood is often intractable. An iterative algorithm that approximates the maximum-likelihood solution is *expectation–maximisation* (EM) (Dempster *et al.* 1977).

2.3.2.1 *Expectation–maximisation*

Expectation–maximisation increases the log-likelihood $\mathcal{L}$ of the observations by optimising a lower bound $\mathcal{F}$ of the likelihood. This section will introduce the algorithm by writing it in terms of the empirical distribution. Appendix A.3 gives a derivation and proof of convergence. The two stages of the expectation–maximisation algorithm are the expectation stage and the maximisation stage. The expectation stage optimises the lower bound, making it equal to the log-likelihood. The maximisation stage optimises the model parameters.

The statistical model whose parameters are trained will be denoted with $q_{\mathcal{U}\mathcal{X}}(\mathcal{U}, \mathcal{X})$, which is a distribution over the hidden variable $\mathcal{U}$ and observed variables $\mathcal{X}$. Marginalising out over the hidden variables gives the distribution over the observed variables:

$$q_{\mathcal{X}}(\mathcal{X}) = \int q_{\mathcal{U}\mathcal{X}}(\mathcal{U}, \mathcal{X}) \, \mathrm{d}\mathcal{U} \tag{2.21a}$$

The log-likelihood for one data point is then

$$\mathcal{L}(\mathcal{X}, q_{\mathcal{U}\mathcal{X}}) \triangleq \log \int q_{\mathcal{U}\mathcal{X}}(\mathcal{U}, \mathcal{X}) \, \mathrm{d}\mathcal{U}. \tag{2.21b}$$

The lower bound that expectation–maximisation maximises is defined for a single data point $\mathcal{X}$ as

$$\mathcal{F}(\mathcal{X}, \rho, q_{\mathcal{U}\mathcal{X}}) \triangleq \int \rho(\mathcal{U}|\mathcal{X}) \log \frac{q_{\mathcal{U}\mathcal{X}}(\mathcal{U}, \mathcal{X})}{\rho(\mathcal{U}|\mathcal{X})} \, \mathrm{d}\mathcal{U}. \tag{2.21c}$$

Compared to $\mathcal{L}$, its lower bound $\mathcal{F}$ explicitly takes an extra parameter, ρ, which is the distribution over the hidden variables $\mathcal{U}$ for each data point $\mathcal{X}$.

Expectation–maximisation is an iterative algorithm. An initial parameter setting $q_{\mathcal{U}\mathcal{X}}^{(k-1)}(\mathcal{U}, \mathcal{X})$ must be given. The expectation stage of expectation–maximisation optimises the distribution over the hidden parameters ρ. Appendix A.3.1 shows that the optimal setting for ρ makes the lower bound equal to the log-likelihood. ρ is then equal to the posterior distribution of the hidden variables given the old parameter

setting:

$$\rho^{(k)}(\mathcal{U}|\mathcal{X}) := q^{(k-1)}_{\mathcal{U}|\mathcal{X}}(\mathcal{U}|\mathcal{X}) = \frac{q^{(k-1)}_{\mathcal{U}\mathcal{X}}(\mathcal{U},\mathcal{X})}{q^{(k-1)}_{\mathcal{X}}(\mathcal{X})}. \tag{2.22}$$

The maximisation step now sets the model parameters $q_{\mathcal{U}\mathcal{X}}$ to maximise the lower bound evaluated on the whole training data, written as an empirical distribution as in (2.19):

$$q^{(k)}_{\mathcal{U}\mathcal{X}} := \arg\max_{q_{\mathcal{U}\mathcal{X}}} \int \tilde{p}(\mathcal{X}) \int \rho^{(k)}(\mathcal{U}|\mathcal{X}) \log q_{\mathcal{U}\mathcal{X}}(\mathcal{U},\mathcal{X}) \mathrm{d}\mathcal{U}\mathrm{d}\mathcal{X}. \tag{2.23}$$

It is often possible to perform this maximisation analytically. If not, *generalised* EM may be used, which merely requires a new value of $q^{(k)}_{\mathcal{U}\mathcal{X}}$ that improves the lower bound. Appendix A.3 proves that in both cases the likelihood increases at least as much as the lower bound. The full EM iteration therefore causes the likelihood to converge to a maximum.

Another way of looking at the optimisation in the maximisation step is as minimising the KL divergence to the inferred distribution over the complete data (the observed variables as well as the hidden data). This distribution p combines the empirical distribution and the approximation to the distribution of the hidden variables:

$$p(\mathcal{U},\mathcal{X}) = \tilde{p}(\mathcal{X})\rho^{(k)}(\mathcal{U}|\mathcal{X})\,. \tag{2.24}$$

Minimising the KL divergence of the model $q_{\mathcal{U}\mathcal{X}}$ to p can be written as

$$\begin{aligned}\arg\min_{q_{\mathcal{U}\mathcal{X}}} \mathcal{KL}(p\|q_{\mathcal{U}\mathcal{X}}) &= \arg\min_{q_{\mathcal{U}\mathcal{X}}} \int\int \tilde{p}(\mathcal{X})\rho^{(k)}(\mathcal{U}|\mathcal{X}) \log \frac{\tilde{p}(\mathcal{X})\rho^{(k)}(\mathcal{U}|\mathcal{X})}{q_{\mathcal{U}\mathcal{X}}(\mathcal{U},\mathcal{X})} \mathrm{d}\mathcal{U}\mathrm{d}\mathcal{X} \\ &= \arg\max_{q_{\mathcal{U}\mathcal{X}}} \int \tilde{p}(\mathcal{X}) \int \rho^{(k)}(\mathcal{U}|\mathcal{X}) \log q_{\mathcal{U}\mathcal{X}}(\mathcal{U},\mathcal{X}) \mathrm{d}\mathcal{U}\mathrm{d}\mathcal{X},\end{aligned} \tag{2.25}$$

which is exactly the expression in (2.23).

Many generative statistical models, including standard speech recognisers, consist of a distribution over the hidden variables, and one over the observed variables given the hidden ones. $q_{\mathcal{U}\mathcal{X}}$ then factorises as

$$q_{\mathcal{U}\mathcal{X}}(\mathcal{U},\mathcal{X}) = q_{\mathcal{U}}(\mathcal{U})\, q_{\mathcal{X}|\mathcal{U}}(\mathcal{X}|\mathcal{U})\,. \tag{2.26}$$

The logarithm of this in the maximisation in (2.23) then becomes a sum, so that the two distributions can be optimised separately:

$$\log q_{\mathcal{U}\mathcal{X}}(\mathcal{U},\mathcal{X}) = \log q_{\mathcal{U}}(\mathcal{U}) + \log q_{\mathcal{X}|\mathcal{U}}(\mathcal{X}|\mathcal{U})\,; \tag{2.27a}$$

$$q_{\mathcal{U}}^{(k+1)} := \arg\max_{q_{\mathcal{U}}} \int \tilde{p}(\mathcal{X}) \int \rho(\mathcal{U}|\mathcal{X}) \log q_{\mathcal{U}}(\mathcal{U})\, d\mathcal{U}\, d\mathcal{X}; \tag{2.27b}$$

$$q_{\mathcal{X}|\mathcal{U}}^{(k+1)} := \arg\max_{q_{\mathcal{X}|\mathcal{U}}} \int \tilde{p}(\mathcal{X}) \int \rho(\mathcal{U}|\mathcal{X}) \log q_{\mathcal{X}|\mathcal{U}}(\mathcal{X}|\mathcal{U})\, d\mathcal{U}\, d\mathcal{X}. \tag{2.27c}$$

All the distributions that this thesis will apply expectation–maximisation to will have this form.

2.3.3 *Baum–Welch*

The instantiation of expectation–maximisation for HMMs is also called Baum–Welch training because it was introduced (Baum *et al.* 1970) before its generalisation. The discussion of expectation–maximisation in section 2.3.2.1 denoted the set of hidden variables with $\mathcal{U}$ and the observed variables $\mathcal{X}$. Applying this to speech recognition, the hidden variables are the sub-phone state and the component at every time instance: $\mathcal{U} = \{\theta_t, m_t\}$. The observed variables $\mathcal{X}$ consist of the feature vectors for one utterance $\{\mathbf{x}_t\}_{1\ldots T_{\mathcal{X}}}$, and, for training, transcriptions W of the audio.[3] Assuming the transcriptions are given at the word level, the weighted finite state transducer for the language model is replaced by a simple word sequence for each utterance. This constrains the state space, so that it is feasible to keep the distribution of the hidden variables for one utterance in memory. The empirical distribution $\tilde{p}(\mathcal{X})$ has Dirac deltas at the utterances in the training data with their transcriptions.

The expectation step of EM finds a distribution $\rho(\mathcal{U}|\mathcal{X})$ over the hidden variables for an utterance. The most convenient form for ρ will drop out of the derivation below. For speech recognisers, the model factorises as a distribution over the hidden variables and a distribution over the observed variables given the hidden ones. Once the distribution ρ has been found, the two factors are optimised separately as in (2.27). How

[3] For decoding, there will be no transcriptions.

to train $q_{\mathcal{U}}$, the state transitions and the mixture weights, is well-known (e.g. Bilmes 1998) and will not be discussed here. How to train $q_{\mathcal{X}|\mathcal{U}}$, though also well-known, will become important for estimating adaptation transformations in section 3 and later, so it will be discussed here in detail.

The form that $q_{\mathcal{X}|\mathcal{U}}$ takes for speech recognition is the product of the likelihood for component Gaussians for every time t. Let $q^{(m)}$ represent the distribution of component m, of which the parameters are to be trained. The likelihood of one data point $\mathcal{X}$, of length $T_{\mathcal{X}}$, given a setting for the hidden variables $\mathcal{U}$ is

$$q_{\mathcal{X}|\mathcal{U}}(\mathcal{X}|\mathcal{U}) = \prod_{t=1}^{T_{\mathcal{X}}} \sum_{m} 1(m_t = m)\, q^{(m)}(\mathbf{x}_t), \tag{2.28}$$

where $1(\cdot)$ is the indicator function, or Kronecker delta, which is 1 when its argument is true and 0 otherwise. Here, $\sum_m 1(m_t = m)$ merely selects the correct component. Therefore, the log-likelihood given $\mathcal{U}$ is

$$\log q_{\mathcal{X}|\mathcal{U}}(\mathcal{X}|\mathcal{U}) = \sum_{t=1}^{T_{\mathcal{X}}} \sum_{m} 1(m_t = m) \log q^{(m)}(\mathbf{x}_t). \tag{2.29}$$

The inner integral in (2.27c), the expected log-likelihood under the distribution over the hidden variables ρ, then is

$$\begin{aligned}\int \rho^{(k)}(\mathcal{U}|\mathcal{X}) \log q^{(k)}_{\mathcal{X}|\mathcal{U}}(\mathcal{X}|\mathcal{U})\, d\mathcal{U} &= \int \rho^{(k)}(\mathcal{U}|\mathcal{X}) \sum_{t=1}^{T_{\mathcal{X}}} \sum_{m} 1(m_t = m) \log q^{(m)(k)}(\mathbf{x}_t)\, d\mathcal{U} \\ &= \sum_{m} \sum_{t=1}^{T_{\mathcal{X}}} \left[\int \rho^{(k)}(\mathcal{U}|\mathcal{X})\, 1(m_t = m)\, d\mathcal{U} \right] \log q^{(m)(k)}(\mathbf{x}_t). \end{aligned} \tag{2.30}$$

As section 2.3.2.1 has discussed, the expectation step of expectation–maximisation sets ρ to the posterior of the hidden variables using the old model parameters. The value of the integral, in square brackets in the last expression, can therefore be seen as the posterior marginal probability of component m at time t. For training $q^{(m)}$, it is all that is necessary to know of the distribution of the hidden parameters. It will be written $\gamma_t^{(m)}$ with

$$\gamma_t^{(m)} \triangleq \int \rho(\mathcal{U}|\mathcal{X})\, 1(m_t = m)\, d\mathcal{U}, \tag{2.31a}$$

and the summed component occupancy over the whole training data

$$\gamma^{(m)} \triangleq \int \tilde{p}(\mathcal{X}) \sum_{t=1}^{T_{\mathcal{X}}} \gamma_t^{(m)} \mathrm{d}\mathcal{X}. \tag{2.31b}$$

Finding the component–time posterior $\gamma_t^{(m)}$ uses the *forward–backward* algorithm (Baum *et al.* 1970). This is an instantiation of the belief propagation algorithm (Pearl 1988), which finds the posterior distribution of random variables in a graphical model by message-passing between adjacent variables. In HMMs, the forward probability is the distribution of m_{t-1} given observations $\mathbf{x}_1 \dots \mathbf{x}_{t-1}$. The distribution of m_t given observations $\mathbf{x}_1 \dots \mathbf{x}_t$ can be computed from the forward message and the observed $\mathbf{x}_t$. The backward probability is the distribution of m_{t+1} given $\mathbf{x}_{t+2} \dots \mathbf{x}_T$. Together with the observed $\mathbf{x}_{t+1}$ this yields the distribution of m_t given $\mathbf{x}_{t+1} \dots \mathbf{x}_T$. Multiplying the forward and backward probability for time t yields the distribution of m_t given $\mathbf{x}_1 \dots \mathbf{x}_T$, which is the component–time posterior $\gamma_t^{(m)}$. Since forward and backward probabilities are computed recursively from opposite ends of the sequence, either the forward or the backward probabilities are required in the reverse order from the one in which they are computed. For a state space of size Θ, the natural implementation of the forward-backward algorithm therefore uses $\mathcal{O}(T \cdot \Theta)$ space and $\mathcal{O}(T \cdot \Theta)$ time. To deal with long sequences, it is also possible to cache the probabilities only at intervals and reduce the space requirement to $\mathcal{O}(\Theta \log T)$ at the cost of requiring $\mathcal{O}(\Theta \cdot T \cdot \log T)$ time (Murphy 2002). However, in practice longer utterances also contain more words and thus more states. To deal with this, pruning is used: forward and backward probabilities below a threshold are set to zero.

Having computed $\gamma_t^{(m)}$, the maximisation step instantiates (2.27c), rewritten using (2.30) and (2.31a):

$$q_{\mathcal{X}|\mathcal{U}} := \arg\max_{q_{\mathcal{X}|\mathcal{U}}} \int \tilde{p}(\mathcal{X}) \sum_m \sum_{t=1}^{T_{\mathcal{X}}} \gamma_t^{(m)} \log q^{(m)}(\mathbf{x}_t) \mathrm{d}\mathcal{X}. \tag{2.32}$$

This maximisation is used for training the output distributions' parameters. When training all speech recogniser parameters, the expectation and maximisation steps are applied iteratively. Chapter 3 and section 4.7 will discuss adaptation within this same

framework. There is usually enough training data to train all parameters of Gaussians directly.

The parameters of each Gaussian can be estimated separately. The instantiation of (2.27c) for training speech recognition parameters sets the parameters of each component to maximise its expected log-likelihood under the distribution of m_t:

$$q^{(m)} := \arg\max_{q^{(m)}} \int \tilde{p}(\mathcal{X}) \sum_{t=1}^{T_{\mathcal{X}}} \gamma_t^{(m)} \log q^{(m)}(\mathbf{x}_t) \mathrm{d}\mathcal{X}. \tag{2.33}$$

Taking the derivative of the integral to be maximised, the distribution $q^{(m)} \sim \mathcal{N}\left(\boldsymbol{\mu}^{(m)}, \boldsymbol{\Sigma}^{(m)}\right)$ is maximised when

$$\boldsymbol{\mu}^{(m)} = \frac{1}{\gamma^{(m)}} \int \tilde{p}(\mathcal{X}) \sum_{t=1}^{T_{\mathcal{X}}} \gamma_t^{(m)} \mathbf{x}_t \mathrm{d}\mathcal{X}; \tag{2.34a}$$

$$\boldsymbol{\Sigma}^{(m)} = \left(\frac{1}{\gamma^{(m)}} \int \tilde{p}(\mathcal{X}) \sum_{t=1}^{T_{\mathcal{X}}} \gamma_t^{(m)} \mathbf{x}_t \mathbf{x}_t^{\mathsf{T}} \mathrm{d}\mathcal{X} \right) - \boldsymbol{\mu}^{(m)} \boldsymbol{\mu}^{(m)\mathsf{T}}. \tag{2.34b}$$

2.4 Decoding

The purpose of a speech recogniser is to convert audio into text. With the audio to be recognised represented by feature vector sequence $\mathbf{X}$ and the word sequence denoted with W, Bayes' rule relates finding the most likely word sequence $\hat{W}$ to the generative model in (2.10):

$$\begin{aligned} \hat{W} = \arg\max_W P(W|\mathbf{X}) &= \arg\max_W \frac{P(W,\mathbf{X})}{p(\mathbf{X})} \\ &= \arg\max_W P(W,\mathbf{X}) = \arg\max_W P(W)\, p(\mathbf{X}|W)\,. \end{aligned} \tag{2.35a}$$

Since $1/p(\mathbf{X})$ does not depend on W, when decoding it is a constant factor in the maximand and can be ignored. Section 2.2 has discussed the form of the generative model $P(W,\mathbf{X})$. To find the most likely word sequence, the sub-phone state sequence Θ should be marginalised out, as in (2.10):

$$\hat{W} = \arg\max_W P(W) \sum_\Theta p(\mathbf{X}|\Theta)\, P(\Theta|W)\,. \tag{2.35b}$$

However, this marginalisation turns out to be computationally infeasible. Therefore, the sum in (2.35b) is replaced by a max operator. Rather than finding the best word sequence, speech recognisers therefore find the word sequence corresponding to the best sub-phone state sequence.

$$\hat{W} \simeq \arg\max_{W} P(W) \max_{\Theta} p(\mathbf{X}|\Theta)\, P(\Theta|W)\,. \tag{2.35c}$$

Note that if $p(\mathbf{X}|\Theta)$ is off by a factor, this does not influence the maximisation. This property is often useful in speech recogniser adaptation.

This sequence can be computed with the Viterbi algorithm (Viterbi 1982), which is a dynamic programming algorithm. The following describes it briefly. The property of the network it needs is the Markov property discussed in section 2.2: the variables at time t depend only on the variables at time t, and not on anything before that. This means that if the best possible path that ends in sub-phone θ at time t goes through θ' at time $t-1$, it contains the best path ending in θ at time $t-1$. Finding the best paths to all states at one time therefore only requires the best paths to all states at the previous time. The task of finding the best path to a final state at the final time therefore becomes a recursion backwards through time.

An approximation that increases decoding speed is *pruning*. This removes unlikely states from the set of paths at every time step. It defines a *pruning beam*, the difference in log-likelihood between the most likely state and the least likely state to be allowed through. Pruning does introduce search errors, so setting the pruning beam gives a trade-off between speed and accuracy.

To assess speech recogniser performance, the *word error rate* (WER) is often used. This metric gives the distance from the reference transcription. It is the lowest number of deletions, insertions, and substitutions required to transform the reference transcription into the result of the speech recognition, as a fraction of the number of words in the transcription.

2.5 Summary

This chapter has described the structure of a speech recogniser, and how to use it. Section 2.1 has discussed how the audio is converted into feature vectors that form the observations to a probabilistic model. The influence of the noise on feature vectors extracted from noisy data will be derived from this (in section 4.2.1). Section 2.2 has discussed the structure of the generative model. How this model is trained with expectation–maximisation was the topic of section 2.3. Similar methods will be applied for adaptation and noise model estimation (chapter 3 and section 4.7). However, in the maximisation step the parameters will then be constrained so that they can be robustly estimated on limited amounts of data. Section 2.4 has discussed decoding, which will be used for the experiments (chapter 8).

Chapter 3

Adaptation

Speech recognisers are often employed in different environments to the one they were trained on. There may be, for example, differences in speaker, speaking style, accent, microphone, and, the topic of this thesis, background noise. This mismatch could be resolved by retraining the recogniser in the new environment. Re-training the model on data that is to be recognised is called *adaptation*. However, usually too little data is available to robustly train all parameters, and it is unlabelled. To deal with this, the model parameters are usually constrained. Section 3.1 introduces the concept of adaptation and general strategies. Section 3.2 discusses training linear transformations of speech recogniser parameters. Linear transformations for covariance modelling while training are mathematically similar and will therefore be the topic of section 3.3.

3.1 Unsupervised adaptation

This thesis will denote an utterance from the training environment with $\mathcal{X}$, with observations $\mathbf{x}_t$. An utterance to be recognised, which is from a different environment, will be written $\mathcal{Y}$ with observations $\mathbf{y}_t$. The adaptation methods that this chapter will introduce are general and can adapt a speech recogniser to many types of difference between environments. In chapter 4 about methods for noise-robustness, $\mathcal{X}$ will explicitly be assumed noise-free, clean, data, and $\mathcal{Y}$ noise-corrupted.

If sufficient training data and the correct transcriptions were available, the mismatch between the environment the recogniser was trained in and the environment it is used in could be resolved by retraining the recogniser in the new environment. One approach would be to apply maximum a posteriori (MAP) training to the speech recognition parameters (Gauvain and Lee 1994). There is no conjugate prior density for an HMM with mixtures of Gaussians, but if mixture weights and component parameters are assumed independent, maximum a posteriori estimates for them can be found. The main problem with this is that each Gaussian's parameters are re-estimated separately, so that to have an effect, sufficient data must be observed for each Gaussian. MAP adaptation of speech recognition parameters is therefore ill-suited to scenarios with limited adaptation data.

An alternative is to constrain the parameters to a subspace, by only training a transformation of the speech recognition parameters that itself has fewer parameters than the speech recogniser. Ideally, decoding with adaptation would jointly optimise the word sequence and speech recogniser transformation that maximises, for example, the likelihood. If $\mathcal{L}$ is the function that is to be optimised with respect to word sequence W and speech recogniser transformation $\mathcal{A}$, and $\mathcal{Y}$ is the adaptation data, then the joint optimisation can be written as

$$\left(\hat{W}, \hat{\mathcal{A}}\right) := \arg\max_{W,\mathcal{A}} \mathcal{L}(\mathcal{Y}, W, \mathcal{A}). \tag{3.1}$$

It is possible to approximate this by estimating $\hat{\mathcal{A}}$ for a number of hypotheses (Matsui and Furui 1998; Yu and Gales 2007). However, this is slow. The normal approach therefore uses coordinate ascent and interleaves optimising word sequence W and optimising speech recogniser transformation $\hat{\mathcal{A}}$. As an approximation to optimising the word sequence, decoding as discussed in section 2.4 is applied. Optimising the speech recogniser transformation normally uses expectation–maximisation or generalised expectation–maximisation. By controlling the number of parameters, the need for Bayesian schemes is avoided. One form of speech recogniser transformation is an affine transformation of parameters of output distributions $\mathrm{q}^{(\mathrm{m})}$, which section 3.2 will

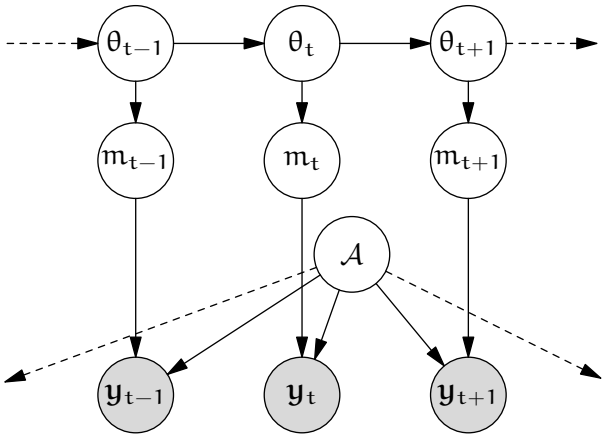


Figure 3.1 *Directed graphical model of a speech hidden Markov model with $\mathcal{A}$ transforming the parameters of the component-conditional distribution. The $\mathbf{y}_t$ are observations from a different environment than the $\mathbf{x}_t$ in figure 2.4 on page 20.*

discuss. Methods specifically for noise-robustness, which will be the topic of chapter 4, can also be seen as adaptation if a noise model is estimated.

Figure 3.1 shows a graphical model of the speech HMM with a transformation, where the observations from the training environment $\mathbf{x}_t$ in figure 2.4 have been replaced by those from the recognition environment $\mathbf{y}_t$. $\mathbf{y}_t$ depends not only on m_t, but also on $\mathcal{A}$. The component output distribution $q^{(m)}(\mathbf{x}_t)$ is replaced with $q^{(m)}(\mathbf{y}_t|\mathcal{A})$. $q^{(m)}(\mathbf{y}_t|\mathcal{A})$ can have various forms, some of which section 3.2 will discuss, but all can be seen as transforming the parameters of the component output distribution. Note that the transformation does not affect m_t, nor the state transitions. For decoding, the algorithm from section 2.4 still applies, except that the output distributions are replaced by their transformed versions.

Figure 3.2 on the following page gives a flow diagram for unsupervised estimation of a transformation. First, the recogniser uses an initial transformation (often the identity transformation) to find a transcription hypothesis W that probably has many errors. To estimate the transformation, the distribution over the component sequence is found (the expectation step of expectation–maximisation). In the maximisation step, this can then be used to decrease the mismatch between the component distri-

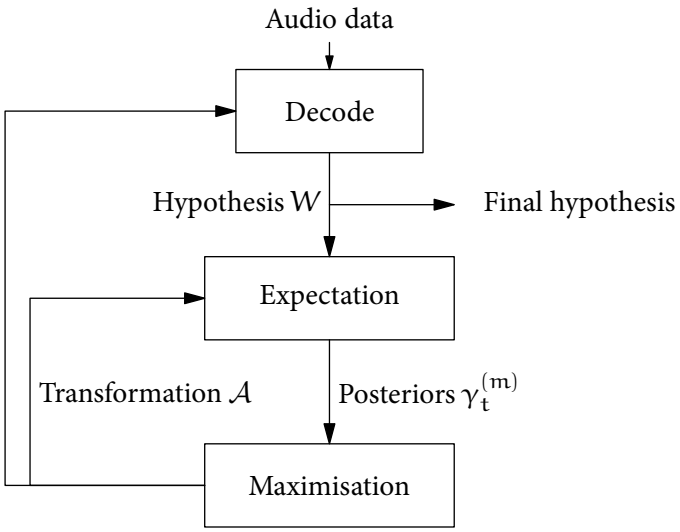


Figure 3.2 *Unsupervised adaptation.*

bution and the actual observations. (This will be discussed in greater detail below.) In the short loop in figure 3.2 the expectation step immediately follows. Running round this loop implements expectation–maximisation or generalised expectation–maximisation. The new transformation $\mathcal{A}$ is therefore guaranteed not to decrease the likelihood in the joint maximisation in (3.1). It is also possible to replace the hypothesis with a new one by running the decoder with the latest estimate of the transformation (the long loop). Since decoding only finds the state sequence, not the word sequence, with the highest likelihood (see section 2.4), the new hypothesis is not guaranteed to yield a better likelihood. If it does, then it is a step towards the joint maximisation in (3.1). After a small number of iterations, this process can stop and yield the final hypothesis.

The expectation step finds the distribution of the hidden variables. As when training a recogniser, this distribution is represented by component–time posteriors $\gamma_t^{(m)}$. The maximisation step finds the best transformation, in a process similar to training output distributions, but using the hypothesis on adaptation utterances $\mathcal{Y}$ rather than training utterances $\mathcal{X}$. The expression is very similar to (2.32), but rather than directly estimating the output distribution's parameters, the transformation $\mathcal{A}$ is estimated by

maximising

$$\mathcal{A}^{(k)} := \arg\max_{\mathcal{A}} \int \tilde{p}(\mathcal{Y}) \sum_{m} \sum_{t=1}^{T_{\mathcal{Y}}} \gamma_t^{(m)} \log q^{(m)}(\mathbf{y}_t|\mathcal{A}) d\mathcal{Y}. \tag{3.2}$$

The empirical distribution $\tilde{p}(\mathcal{Y})$ here is assumed to represent utterances from a homogeneous part of the training data.

Unsupervised estimation often works well even if the initial hypothesis contains many errors. The key to this is controlling the number of parameters that are trained on a given amount of data. Gaussian components are usually grouped in clusters that share one transformation. The grouping is normally hierarchical, in a regression class tree (Leggetter 1995; Gales 1996; Haeb-Umbach 2001). When performing adaptation, the tree is pruned so that the resulting leaf nodes have enough data for robust estimation of the transformation. The leaf nodes of an unpruned tree result in a component clustering into *base classes* $1 \ldots R$. For the methods that this thesis will introduce (in chapter 6), the amount of adaptation data will be irrelevant, so base classes will feature most prominently. Estimating the transformations is completely separate per class, so to keep notation from being cluttered, the notation in sections 3.2 and 3.3 will assume one class.

3.1.1 Adaptive training

The discussion of adaptation has so far assumed that the speech recogniser model is trained on homogeneous and noise-free data, and that that the test data is different from that. In reality, the training data often has different speakers, and sometimes is even explicitly *multi-environment* (for example, different noise data may be artificially added to the audio), to try and capture the different environments the recogniser might be employed in.

It is possible to use the graphical model with transformations, in figure 3.1, when training as well. For every set of homogeneous utterances (for example, utterances from one speaker, or from one noise environment, or just for one utterance) a transformation is trained to maximise its likelihood. The speech recogniser parameters are

then re-estimated to maximise the average likelihood over all speakers. The speech recogniser and the transformation are optimised iteratively. This creates a *canonical* speech recogniser model, which, unlike a normally trained one, does not represent the training data without the transformation. It is a conceptually pleasing property that it unifies the model for training and testing. However, it creates a chicken-and-egg problem: the speech recogniser does not represent the data without a transformation and there is no well-defined initial setting for the transformation without a hypothesis. There is therefore no clear starting point for the interleaved estimation of transformation and hypothesis. This can be solved by using a conventionally-trained recogniser initially, and only then using the adaptively-trained one with an appropriate transformation, or using a heuristically determined initial transformation.

Some schemes that apply adaptive training introduce an extra stochastic variable that represents the identity of the speaker or cluster of speakers. This includes speaker adaptive training (Anastasakos *et al.* 1996) and cluster adaptive training (Gales 2000). Some more recent adaptive training schemes (Liao and Gales 2007; Kalinli *et al.* 2010; Flego and Gales 2009; Kim and Gales 2010) have found improvements from using transformations on speech recognisers with a standard form of speech recogniser adaptation.

3.2 Linear adaptation

Linear adaptation methods are instantiations of adaptation as discussed in section 3.1. A graphical model was given in figure 3.1 on page 41: the distribution of observations $\mathbf{y}_t$ depend on the component generating it, m_t, and the transformation $\mathcal{A}$. Transformation $\mathcal{A}$ changes the parameters of Gaussian component m, which models the training data:

$$p^{(m)}(\mathbf{x}) = \mathcal{N}\left(\mathbf{x};\, \boldsymbol{\mu}_x^{(m)},\, \boldsymbol{\Sigma}_x^{(m)}\right). \tag{3.3a}$$

The parameters of transformed distribution $q^{(m)}(\mathbf{y}|\mathcal{A})$ represents the found data with

$$q^{(m)}(\mathbf{y}|\mathcal{A}) = \mathcal{N}\big(\mathbf{y};\, \boldsymbol{\mu}_y^{(m)},\, \boldsymbol{\Sigma}_y^{(m)}\big). \tag{3.3b}$$

The general form of transformation that will be considered is an affine transformation $\{\mathbf{H}, \mathbf{g}\}$ to the mean vector, and a linear transformation $\mathbf{H}'$ to the covariance matrix:

$$\boldsymbol{\mu}_y^{(m)} := \mathbf{H}\boldsymbol{\mu}_x^{(m)} - \mathbf{g}; \tag{3.3c}$$

$$\boldsymbol{\Sigma}_y^{(m)} := \mathbf{H}'\boldsymbol{\Sigma}_x^{(m)}\mathbf{H}'^{\mathsf{T}}. \tag{3.3d}$$

The first adaptation method that applied an explicitly ML-estimated affine transform was maximum-likelihood linear regression (MLLR) (Leggetter and Woodland 1995), which only adapts the parameters of the mean. Since the main interest in adaptation transformations for this thesis is in modelling correlations for noise-robustness, the following sections will focus on two different forms. The first, CMLLR (Gales 1998*a*) constrains the mean and covariance transformations to be the same. The second only transforms the covariance (Neumeyer *et al.* 1995; Gales and Woodland 1996). Per class, both have in the order of d^2 parameters, which means that about 1000 frames are required to train them robustly.

3.2.1 *Constrained transformation*

Constrained MLLR (CMLLR) constrains the linear transform applied to the mean and covariance to be equal. Thus, the likelihood for component m becomes

$$q^{(m)}(\mathbf{y}|\mathcal{A}) = \mathcal{N}\big(\mathbf{y};\, \mathbf{H}\boldsymbol{\mu}_x^{(m)} - \mathbf{g},\, \mathbf{H}\boldsymbol{\Sigma}_x^{(m)}\mathbf{H}^{\mathsf{T}}\big). \tag{3.4a}$$

$\mathbf{H}$ can be diagonal (Digalakis *et al.* 1995) or full (Gales 1998*a*). The latter shape is of most interest for this work since it can model some feature correlations. One of its useful properties is that this can alternatively be written as a transformation of the observations: $\mathcal{A} = \{\mathbf{A}, \mathbf{b}\}$ where $\mathbf{A} = \mathbf{H}^{-1}$ and $\mathbf{b} = -\mathbf{H}^{-1}\mathbf{g}$ (Gales 1998*a*):

$$q^{(m)}(\mathbf{y}|\mathcal{A}) = |\mathbf{A}| \cdot \mathcal{N}\big(\mathbf{A}\mathbf{y} + \mathbf{b};\, \boldsymbol{\mu}_x^{(m)},\, \boldsymbol{\Sigma}_x^{(m)}\big). \tag{3.4b}$$

This means that each observation vector is transformed before being passed to the Gaussian components. In practice, components are usually clustered into classes based on their distance to each other, with a different transformation for each class (see section 3.1). In effect, CMLLR then performs a piecewise linear transformation of the observations. In terms of the implementation, models can calculate the observation likelihood on the appropriately transformed feature vector. Transforming $\mathbf{y}$ to R parallel feature vectors $\mathbf{A}^{(r)}\mathbf{y} + \mathbf{b}^{(r)}$ can be computationally cheaper compared to transforming the parameters of each Gaussian. For a diagonal transformation matrix, this depends on the number of feature vectors to be transformed and the number of components: transforming one feature vector has the same complexity as transforming one component. If the transformation matrix is full, transforming one feature vector costs $\mathcal{O}(d^2)$ time, whereas transforming one covariance matrix costs $\mathcal{O}(d^3)$ time. Additionally, transforming the features means that the original diagonal covariance matrices can be used, so that no extra memory is required to store the models and the likelihood computation is not slowed down much.

The interest here is in a method that works for diagonal covariance matrices, so that decoding is cheap. This allows for the row-wise optimisation algorithm in Gales (1998*a*). Estimating transformations for full covariance matrices, with a generalisation of the row-wise algorithm (Sim and Gales 2005) or gradient optimisation (Ghoshal *et al.* 2010), will not be discussed. Covariance matrix entries are denoted with $\sigma^{(m)}_{x,ii}$.

The maximisation step implements (3.2) with the likelihood calculation in (3.4b). A derivation of the optimisation is in section B.1.1. What is interesting here is the statistics that this optimisation requires. (Section 6.2.1 will discuss how to train the same form of transformation from predicted statistics. The only difference will be the form of the statistics.) The required statistics from the adaptation data are γ, $\mathbf{k}^{(i)}$, and $\mathbf{G}^{(i)}$ with (from (B.5))

$$\gamma \triangleq \int \tilde{p}(\mathcal{Y}) \sum_{m} \sum_{t=1}^{T_{\mathcal{Y}}} \gamma_t^{(m)} d\mathcal{Y}; \tag{3.5a}$$

$$\mathbf{k}^{(i)} \triangleq \int \tilde{p}(\mathcal{Y}) \sum_m \frac{\mu_{x,i}^{(m)}}{\sigma_{x,ii}^{(m)}} \sum_{t=1}^{T_\mathcal{Y}} \gamma_t^{(m)} \left[\mathbf{y}_t^\mathsf{T} \; 1 \right] \mathrm{d}\mathcal{Y}; \tag{3.5b}$$

$$\mathbf{G}^{(i)} \triangleq \int \tilde{p}(\mathcal{Y}) \sum_m \frac{1}{\sigma_{x,ii}^{(m)}} \sum_{t=1}^{T_\mathcal{Y}} \gamma_t^{(m)} \begin{bmatrix} \mathbf{y}_t \mathbf{y}_t^\mathsf{T} & \mathbf{y}_t \\ \mathbf{y}_t^\mathsf{T} & 1 \end{bmatrix} \mathrm{d}\mathcal{Y}. \tag{3.5c}$$

The optimisation algorithm then maximises the likelihood with respect to $\boldsymbol{A}$ per row. It iterates over each row a number of times. The likelihood is therefore guaranteed not to decrease, which makes the overall process an instantiation of generalised expectation–maximisation. If the transformation matrix $\boldsymbol{A}$ is constrained to a block-diagonal shape, the likelihood expression factorises into likelihoods for these blocks of coefficients. They can therefore be optimised separately. In the extreme case, $\boldsymbol{A}$ is constrained to be diagonal, and the optimisation is separate for each dimension. The optimisation procedure then yields the global maximum immediately, so that iterating is not necessary, and the process is an instantiation of expectation–maximisation. The final transformation is equivalent to the one described in Digalakis *et al.* (1995), which applies it in model space, though with an iterative method.

For the full-transformation case, the computational complexity of this algorithm is dominated by the cost of calculating the cofactors and the inverse of $\mathbf{G}^{(i)}$, which is necessary for each row. The latter costs $\mathcal{O}(d^3)$ per matrix (with d the dimension of the feature vector). A naive implementation of the former costs $\mathcal{O}(d^3)$ per matrix per iteration, but using the Sherman-Morrison matrix inversion lemma this can be reduced to $\mathcal{O}(d^2)$ (Gales and van Dalen 2007). Thus, for R transforms and L iterations, the cost of estimating the transforms is $\mathcal{O}(RLd^3 + Rd^4)$.

3.2.2 *Covariance adaptation*

Covariance MLLR (Neumeyer *et al.* 1995; Gales and Woodland 1996; Gales 1998*a*) updates only the covariances of the component Gaussian. It was originally proposed to be used in combination with mean MLLR. The likelihood of transformed Gaussian

component m becomes

$$q^{(m)}(\mathbf{y}|\mathcal{A}) = \mathcal{N}\big(\mathbf{y};\, \boldsymbol{\mu}_x^{(m)},\, \mathbf{H}\boldsymbol{\Sigma}_x^{(m)}\mathbf{H}^\mathsf{T}\big). \tag{3.6a}$$

As in the constrained case, this is better expressed with the inverse transformation $\mathbf{A} = \mathbf{H}^{-1}$, so that

$$q^{(m)}(\mathbf{y}|\mathcal{A}) = |\mathbf{A}| \cdot \mathcal{N}\big(\mathbf{A}\mathbf{y};\, \mathbf{A}\boldsymbol{\mu}_x^{(m)},\, \boldsymbol{\Sigma}_x^{(m)}\big). \tag{3.6b}$$

It may not be immediately obvious why this is a better formulation than (3.6a). Normally, the covariance matrix $\boldsymbol{\Sigma}_x^{(m)}$ is diagonal. Storing the updated covariance $\mathbf{H}\boldsymbol{\Sigma}_x^{(m)}\mathbf{H}^\mathsf{T}$ would therefore require extra storage, whereas storing $\mathbf{A}\boldsymbol{\mu}_x^{(m)}$ does not. An even more important reason in this work, which will use a variant of covariance MLLR to speed up decoding, is that computing the likelihood is faster with the form in (3.6b) than with the form in (3.6a), again because of the diagonal covariance matrix.

The derivation is in appendix B.2.1. Just like for CMLLR, section 6.2.2 will compute the same form of transformation but from predicted statistics. The form of the statistics are therefore of most interest here. The statistics from the adaptation data are γ and $\mathbf{G}^{(i)}$ with

$$\gamma \triangleq \int \tilde{p}(\mathcal{Y}) \sum_m \sum_{t=1}^{T_\mathcal{Y}} \gamma_t^{(m)} \mathrm{d}\mathcal{Y}; \tag{3.7a}$$

$$\mathbf{G}^{(i)} \triangleq \int \tilde{p}(\mathcal{Y}) \sum_m \frac{1}{\sigma_{x,ii}^{(m)}} \sum_{t=1}^{T_\mathcal{Y}} \gamma_t^{(m)} \big(\mathbf{y}_t - \boldsymbol{\mu}_x^{(m)}\big)\big(\mathbf{y}_t - \boldsymbol{\mu}_x^{(m)}\big)^\mathsf{T} \mathrm{d}\mathcal{Y}. \tag{3.7b}$$

γ here is the same as for CMLLR (in (3.5a)); $\mathbf{G}^{(i)}$ is similar to part of (3.5c) but uses $\mathbf{y}_t - \boldsymbol{\mu}_x^{(m)}$ instead of $\mathbf{y}_t$.

Similarly to CMLLR, the optimisation is row-wise, and this again implements generalised expectation–maximisation. Just as for CMLLR, calculating the inverse of $\mathbf{G}^{(i)}$ and finding the cofactors are necessary for each row update and form the main computational cost. Therefore, estimating transforms for R classes in L iterations is $\mathcal{O}\big(RLd^3 + Rd^4\big)$, with d the size of the feature vector. Covariance MLLR also needs to transform all model means, which takes $\mathcal{O}\big(Md^2\big)$.

3.3 Covariance modelling

The second type of transformation in this chapter aims to model the training data correlations better rather than to resolve a mismatch between training and test data. These transformations are normally trained during speech recognition training. Computational cost and data sparsity are therefore not as problematic as in adaptation, and component parameters can also be updated. This section will denote the observations from training data with $\mathbf{x}$. In terms of the mathematics, however, there is no significant difference with adaptation transformations. Training more sophisticated correlation models again uses expectation–maximisation. In chapter 6.2, they will be applied to speed up decoding with correlation compensation for noise-robustness in much the same way as transformations for adaptation will be.

Traditionally, speech recognisers make the assumption that the coefficients of one feature vector are independent. In that case, Gaussian distributions can have diagonal covariance matrices, robust estimates for which need less data than full ones. Also, it is expensive to compute the likelihood of a full-covariance Gaussian. However, real data does show within-component correlations. There are various techniques to improve modelling of correlations while for robustness restricting the number of extra parameters to be trained. It is possible to derive a structured form of covariance modelling from factor analysis (Gopinath *et al.* 1998; Saul and Rahim 2000):

$$\boldsymbol{\Sigma}^{(m)} = \mathbf{A}_{[p]}^{\mathsf{T}} \mathbf{A}_{[p]} + \boldsymbol{\Sigma}_{\mathsf{diag}}^{(m)}. \tag{3.8}$$

The *loading matrix* $\mathbf{A}_{[p]}$ can be specific to the component, but for p small that yields little extra modelling power, whereas as p becomes equal to the number of features d, as many parameters are introduced as with full covariances. Alternatively, a generalisation of this (Rosti and Gales 2004) ties $\mathbf{A}_{[p]}$ over all components in a base class. This reduces the number of parameters to be trained compared to full covariances. For small p the Sherman–Morrisson–Woodbury formula can make the likelihood computation efficient. However, to attain good modelling, the loading matrices are normally tied across many components and p is large, which makes decoding as slow

as with full covariance matrices.

The following sections will discuss two different types of correlation modelling that require fewer parameters to be estimated and allow faster decoding than full covariance matrices. The first form models *precision matrices*, inverse covariance matrices (section 3.3.1). The second form to be discussed (section 3.3.2) are projection schemes, which choose dimensions that discriminate best and reduce the dimensionality of the data.

3.3.1 Structured precision matrices

It is possible to model the precision matrices, the inverse covariance matrices, directly as a weighted sum of basis matrices (Olsen and Gopinath 2004; Axelrod *et al.* 2002; Vanhoucke and Sankar 2004; Sim and Gales 2004):

$$\boldsymbol{\Sigma}^{(m)^{-1}} = \sum_i \pi_i^{(m)} \mathbf{B}_i, \tag{3.9}$$

where $\mathbf{B}_i$ is a basis matrix for modelling the precision matrices. The advantage of modelling the precision matrices rather than covariance matrices is decoding speed. The resulting likelihood calculation is

$$q^{(m)}(\mathbf{x}) \propto \exp\left(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}^{(m)})^T \boldsymbol{\Sigma}^{(m)^{-1}} (\mathbf{x}-\boldsymbol{\mu}^{(m)})\right) \tag{3.10}$$

$$= \exp\left(\sum_i \pi_i^{(m)} \left(-\tfrac{1}{2}\mathbf{x}^T \mathbf{B}_i \mathbf{x} + \boldsymbol{\mu}^{(m)^T} \mathbf{B}_i \mathbf{x} - \tfrac{1}{2}\boldsymbol{\mu}^{(m)^T} \mathbf{B}_i \boldsymbol{\mu}^{(m)}\right)\right). \tag{3.11}$$

Since $\mathbf{x}^T \mathbf{B}_i \mathbf{x}$ and $\mathbf{B}_i \mathbf{x}$ do not depend on the component, it is possible to cache them and share the result between components.

A more restricted, but effective, form of precision matrix modelling is semi-tied covariance matrices (Gales 1999). Each basis matrix is of rank 1, and components in one base class share as many basis matrices as there are feature dimensions. It can therefore be written in a different form, where components have diagonal covariance matrices that share one rotation matrix per base class. The algorithm finds a transformation that results in a feature space in which a diagonal covariance matrix is a

more valid assumption than in the original feature space. The covariance matrix in the transformed space will be denoted with $\widetilde{\boldsymbol{\Sigma}}^{(m)}_{x,\mathrm{diag}}$. The expression for the likelihood is

$$q^{(m)}(\boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{x};\, \boldsymbol{\mu}^{(m)}_x,\, \mathbf{H}\widetilde{\boldsymbol{\Sigma}}^{(m)}_{x,\mathrm{diag}}\mathbf{H}^{\mathsf{T}}\big). \tag{3.12a}$$

Just like for covariance MLLR, in (3.6a), the effective covariance is a component-specific diagonal covariance in a space specified by a transformation $\mathbf{H}$. The transformation is tied over all components in a base class. (Again, the dependence on the base class is not written since the optimisation is separate for each base class.) Unlike for covariance MLLR, data sparsity is not a problem, because all training data is used, so that component-dependent covariance $\widetilde{\boldsymbol{\Sigma}}^{(m)}_{x,\mathrm{diag}}$ is estimated as well as the transformation. To improve decoding speed, it is again useful to describe the covariance transformation by its inverse, $\boldsymbol{A} = \mathbf{H}^{-1}$, so that the likelihood is expressed

$$q^{(m)}(\boldsymbol{x}) = |\boldsymbol{A}| \cdot \mathcal{N}\big(\boldsymbol{A}\boldsymbol{x};\, \boldsymbol{A}\boldsymbol{\mu}^{(m)}_x,\, \widetilde{\boldsymbol{\Sigma}}^{(m)}_{x,\mathrm{diag}}\big). \tag{3.12b}$$

The estimation of $\boldsymbol{A}$ and $\widetilde{\boldsymbol{\Sigma}}^{(m)}_{x,\mathrm{diag}}$ is iterative. First, $\boldsymbol{A}$ is estimated, with the same form of statistics and procedure as covariance MLLR. Then, $\widetilde{\boldsymbol{\Sigma}}^{(m)}_{x,\mathrm{diag}}$ is straightforwardly set to the maximum-likelihood estimate. As for CMLLR and covariance MLLR, the interest here is in the statistics that estimation requires. A derivation is in section B.3.1. Statistics that do not change when component covariances are updated are γ, again, and the sample covariance for component m, $\boldsymbol{W}^{(m)}$:

$$\gamma \triangleq \int \tilde{p}(\mathcal{X}) \sum_m \sum_{t=1}^{T_{\mathcal{X}}} \gamma^{(m)}_t \mathrm{d}\mathcal{X}; \tag{3.13a}$$

$$\boldsymbol{W}^{(m)} \triangleq \frac{1}{\gamma^{(m)}} \int \tilde{p}(\mathcal{X}) \sum_{t=1}^{T_{\mathcal{X}}} \gamma^{(m)}_t \big(\boldsymbol{x}_t - \boldsymbol{\mu}^{(m)}_x\big)\big(\boldsymbol{x}_t - \boldsymbol{\mu}^{(m)}_x\big)^{\mathsf{T}} \mathrm{d}\mathcal{X}. \tag{3.13b}$$

The statistics $\mathbf{G}^{(i)}$ are of the same form as for covariance MLLR, in (3.7b). They depend on $\widetilde{\sigma}^{(m)}_{x,ii}$, diagonal element i of $\widetilde{\boldsymbol{\Sigma}}^{(m)}_{x,\mathrm{diag}}$, which for semi-tied covariance matrices is updated in every iteration.

$$\mathbf{G}^{(i)} \triangleq \sum_m \frac{\gamma^{(m)}}{\widetilde{\sigma}^{(m)}_{x,ii}} \boldsymbol{W}^{(m)}. \tag{3.13c}$$

function ESTIMATE-SEMI-TIED-COVARIANCE-MATRICES($\{\boldsymbol{W}^{(m)}, \gamma^{(m)}\}, \gamma$)
  for all components m **do**
    Initialise $\widetilde{\boldsymbol{\Sigma}}^{(m)}_{x,\text{diag}} \leftarrow \text{diag}\left(\boldsymbol{W}^{(m)}\right)$
  Initialise $\boldsymbol{A} \leftarrow \mathbf{I}$
  repeat
    $\mathbf{G}^{(i)} \leftarrow \sum_m \frac{1}{\widetilde{\sigma}^{(m)}_{x,ii}} \gamma^{(m)} \boldsymbol{W}^{(m)}$
    $\boldsymbol{A} \leftarrow$ ESTIMATE-COVARIANCE-MLLR($\gamma, \mathbf{G}^{(i)}$)
    for all components m **do**
      $\widetilde{\boldsymbol{\Sigma}}^{(m)}_{x,\text{diag}} \leftarrow \text{diag}\left(\boldsymbol{A}\boldsymbol{W}^{(m)}\boldsymbol{A}^{\mathsf{T}}\right)$
  until convergence
  return $\left\{\widetilde{\boldsymbol{\Sigma}}^{(m)}_{x,\text{diag}}\right\}, \boldsymbol{A}$

Algorithm 1 *The maximisation step of expectation–maximisation for estimating semi-tied covariance matrices.*

The estimation procedure is given in algorithm 1. $\widetilde{\boldsymbol{\Sigma}}^{(m)}_{x,\text{diag}}$ is initialised to the diagonalised original covariance, $\text{diag}\left(\boldsymbol{W}^{(m)}\right)$, and $\boldsymbol{A}$ to the identity matrix $\mathbf{I}$. In the first step, the transformation $\boldsymbol{A}$ is updated in the same way as is done for covariance MLLR transforms, described in section 3.2.2. However, the current estimate for the covariance, which changes every iteration, is used. The statistics must therefore be re-computed for every iteration. In the second step, $\widetilde{\boldsymbol{\Sigma}}^{(m)}_{x,\text{diag}}$ is set to the maximum-likelihood diagonal covariance in the feature space given by $\boldsymbol{A}$. This process is repeated until convergence. Both steps are guaranteed not to decrease the likelihood. This is therefore a generalised expectation–maximisation algorithm.

Because decoding with semi-tied covariance uses diagonal-covariance Gaussians, it is almost as fast as decoding with plain diagonal-covariance Gaussians. Adjusting the number of base classes that transformations are computed for allows a trade-off between the number of parameters and the accuracy of the covariance model.

Just like for CMLLR and covariance MLLR, the computational complexity of this algorithm is dominated by the cost of calculating the cofactors and the inverse of $\mathbf{G}^{(i)}$. The former costs $\mathcal{O}\left(d^2\right)$ (with d the dimension of the feature vector) per dimension per iteration. The latter costs $\mathcal{O}\left(d^3\right)$ per dimension. Thus, for R transforms, K outer

loop iterations, and L inner loop (of estimating $\mathbf{A}$) iterations, the cost of estimating the transforms is $\mathcal{O}\left(\mathrm{RKL}\mathrm{d}^3 + \mathrm{RK}\mathrm{d}^4\right)$.

3.3.2 *Maximum likelihood projection schemes*

A projection of feature vectors onto a different overall feature space, as opposed to a component-specific one, can also improve speech recogniser performance. This notion motivates the final step of computing MFCCs (in (2.6)), which aims to decorrelate the features with a discrete cosine transform, and then reduces the dimensionality. The same goes for deriving dynamic features from a window of static features (in (2.7b)). These two projection schemes have an intuitive motivation. This section, however, discusses data-driven approaches to decorrelation and dimensionality reduction that can be applied in combination with or instead of the projections for DCT and dynamic coefficients.

Linear discriminant analysis (LDA) (Fukunaga 1972) is a standard linear projection scheme that transforms the feature vectors to maximise between-class distance and minimise with-class correlation. For speech recognisers, the classes are usually Gaussian components. The transformation is supposed to make the assumption that the components have diagonal covariance matrices more reasonable. An alternative projection scheme that does not assume that the component covariances are equal, but does not optimise the feature space to be diagonal, is heteroscedastic discriminant analysis (Saon *et al.* 2000).

Heteroscedastic linear discriminant analysis (HLDA) (Kumar 1997) is a method that finds the best projection as well as a transformation that improves the diagonal covariance approximation. It finds a projection $\mathbf{A}_{[\mathrm{p}]}$ that projects the d-dimensional original feature space to p-dimensional subspace of useful features. The parameters for the $(\mathrm{d} - \mathrm{p})$-dimensional *nuisance* subspace are tied over all components in the

base class. Thus

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{[p]} \\ \mathbf{A}_{[d-p]} \end{bmatrix}, \tag{3.14}$$

so that the base class-specific transformed feature vector is

$$\hat{\mathbf{x}} = \mathbf{A}\mathbf{x} = \begin{bmatrix} \mathbf{A}_{[p]}\mathbf{x} \\ \mathbf{A}_{[d-p]}\mathbf{x} \end{bmatrix}. \tag{3.15}$$

The new parameters for component m become

$$\hat{\boldsymbol{\mu}}^{(m)} = \begin{bmatrix} \hat{\boldsymbol{\mu}}^{(m)}_{[p]} \\ \hat{\boldsymbol{\mu}}_{[d-p]} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{[p]}\boldsymbol{\mu}^{(m)} \\ \mathbf{A}_{[d-p]}\boldsymbol{\mu} \end{bmatrix}; \tag{3.16}$$

$$\hat{\boldsymbol{\Sigma}}^{(m)} = \begin{bmatrix} \hat{\boldsymbol{\Sigma}}^{(m)}_{[p]} & 0 \\ 0 & \hat{\boldsymbol{\Sigma}}_{[d-p]} \end{bmatrix}, \tag{3.17}$$

where $\boldsymbol{\mu}$ is the global mean, and

$$\hat{\boldsymbol{\Sigma}}^{(m)}_{[p]} = \operatorname{diag}\left(\mathbf{A}_{[p]}\mathbf{W}^{(m)}\mathbf{A}^{\mathsf{T}}_{[p]}\right); \tag{3.18}$$

$$\hat{\boldsymbol{\Sigma}}_{[d-p]} = \operatorname{diag}\left(\mathbf{A}_{[d-p]}\boldsymbol{\Sigma}\mathbf{A}^{\mathsf{T}}_{[d-p]}\right). \tag{3.19}$$

where $\mathbf{W}^{(m)}$ is the actual covariance within components, and $\boldsymbol{\Sigma}$ is the global covariance. The transformation is found with maximum-likelihood estimation. Details of the process are in Kumar (1997).

Because the nuisance dimensions have been tied over all components in the base class, the component likelihood computation can be split up into a global Gaussian and a component-specific one:

$$\begin{aligned} q^{(m)}(\mathbf{x}) &= |\mathbf{A}|\mathcal{N}\left(\mathbf{x};\, \hat{\boldsymbol{\mu}}^{(m)},\, \hat{\boldsymbol{\Sigma}}^{(m)}\right) \\ &= |\mathbf{A}|\mathcal{N}\left(\mathbf{A}_{[d-p]}\mathbf{x};\, \hat{\boldsymbol{\mu}}_{[d-p]},\, \hat{\boldsymbol{\Sigma}}_{[d-p]}\right)\mathcal{N}\left(\mathbf{A}_{[p]}\mathbf{x};\, \hat{\boldsymbol{\mu}}^{(m)}_{[p]},\, \hat{\boldsymbol{\Sigma}}^{(m)}_{[p]}\right), \end{aligned} \tag{3.20}$$

which reduces the computational complexity because for one observation the first Gaussian is constant. Since a constant factor does not influence decoding (see section 2.4), the determinant and the first Gaussian do not normally have to be computed.

An useful property of HLDA is that it finds a model for the complete feature space, not just for the useful dimensions. This allows for a generalisation, multiple heteroscedastic linear discriminant analysis (MHLDA) (Gales 2002), which finds a separate HLDA-like transformation for each base class. (The determinant and the first Gaussian in (3.20) then do affect decoding, so they do have to be computed.) At the same time, it is also an generalisation of semi-tied covariance matrices that reduces the feature dimensionality.

3.4 Summary

This chapter has discussed the general mechanism of speech recogniser adaptation. Section 3.1 has described how the usual approach, given unlabelled adaptation data, iterates between decoding and estimating a transformation of the recogniser. Methods for noise-robustness that the next chapter will discuss can be cast in the same framework when the noise is estimated (section 4.7). Section 3.2, however, has discussed adaptation without a model of the environment, but with linear transformations. They can therefore adapt to many types of mismatch, and by placing the transformations in the right places in the likelihood equation, they are essentially as fast to decode with as without. A similar scheme, semi-tied covariance matrices, discussed in section 3.3, can be used while training to model covariances, and similarly hardly reduces decoding speed. Section 6 will train both types of linear transformations from predicted distributions (for example, over noise-corrupted speech) rather than adaptation data.

Chapter 4

Noise-robustness

This chapter will discuss methods that make speech recognisers robust to noise. Section 4.1 will present a number of strategies for noise-robustness. Compared with the generic adaptation methods discussed in chapter 3, they need less adaptation data. This is possible since they make stronger assumptions about the mismatch between training and test environments. The assumptions are the model of the noise and how it influences the incoming feature vectors. They will be the topic of section 4.2. Section 4.3 will then describe the resulting corrupted speech distribution. It has no closed form, so it needs to be approximated. Section 4.4 will discuss specific methods of *model compensation*, which replace the recogniser's clean speech distributions with distributions over the corrupted speech. Rather than using precomputed parameterised distributions, it is possible to approximate the noise-corrupted speech likelihoods only as the observations come in. This will be the topic of section 4.5. Section 4.6 will discuss an alternative model-based scheme that reconstructs the clean speech before passing it to the recogniser. Section 4.7 will describe methods to estimated the noise model parameters can be estimated in an adaptation framework as described in section 3.1.

4.1 Methods for noise-robustness

There are two categories of approaches for making speech recognisers robust to noise. Feature enhancement aims to reconstruct the clean speech before it enters the speech recogniser, and is therefore relatively fast. Model compensation, on the other hand, aims to perform joint inference over the clean speech and the noise, and is slower but yields better accuracy. If input from multiple microphones is available, then it is sometimes possible to reconstruct the signal from a source at a specific location. However, this work will consider the general case where only the input from one microphone is available.

Feature enhancement can work on any representation of the audio signal that is available in a speech recogniser. For example, the spectrum is sometimes used. *Spectral subtraction* (Boll 1979) requires the spectrum of the noise to be given, and assumes the noise is stationary. Alternatively, it is possible to find a minimum mean square error estimate of the speech and the noise (Ephraim 1990). This requires probabilistic models of the speech and the noise. It is possible to formulate these in the spectral domain, and assume all spectral coefficients independent. Though this assumption is true for Gaussian white noise, this type of noise is not normally found outside of research papers. For speech, the assumption is particularly unhelpful.

Therefore, approaches to feature enhancement that work on the log-mel-spectrum or the cepstrum have over the past two decades become successful. Though it makes formulating the interaction of speech and noise harder, it makes the models of speech and noise that assume independence between dimensions more accurate. To find the minimum mean square error estimate of the clean speech, its distribution is normally assumed independent and identically distributed, often as a mixture of Gaussians. A joint distribution of the clean and the corrupted speech then needs to be derived. This normally applies the same methods that model compensation does. Section 4.6 will discuss this.

However, this work will focus on model compensation, which replaces a speech recogniser's distributions over clean speech by ones over noise-corrupted speech. Con-

ceptually, a speech recogniser is a classifier that takes an observation sequence as input and classifies it as belonging to one of a set of word sequences. The Bayes decision rule for classification says that the best choice for labelling observations is the one with the highest probability. The probability is factorised into the prior probability of the label, and the likelihood, the probability of the observations given the label. If the prior and the likelihood are the true ones, then the Bayes decision rule produces the optimal classification. Speech recognisers essentially implement this rule.

Assuming that the speech and noise distributions are the true ones, decoding with the exact distributions for the corrupted speech would therefore yield the best recogniser performance. The objective of feature enhancement, reconstructing the clean speech, may be useful where the clean speech is required, for example, as a preprocessing step before passing the signal to humans. However, as a preprocessing step for speech recognition, it gives no guarantees about optimality under any assumptions. This thesis will therefore aim to find accurate corrupted speech distributions for model compensation.

4.2 Noise-corrupted speech

Noise can be described as the change to the clean speech signal before the speech recogniser receives it. It is possible to identify a number of different types of noise.

The most obvious is that background noise may be added to the signal. This will be called "additive noise" and denoted by $\mathbf{n}$. It can be represented by its time-domain signal. It will be assumed independent of the speech. Second, due to the properties of the microphone and other elements of the channel, some frequencies may be amplified and others reduced. This can be represented by convolution in the time domain. The convolutional noise will be written $\mathbf{h}$ and will be assumed constant and independent of the additive noise and the speech. The environment model with additive and convolutional noise has been standard for model-based noise-robustness since it was introduced (Acero 1990). These two types of noise will be handled explicitly in this

work.

Other forms of noise need to be handled separately, with approaches that may well be combinable with the approaches from this thesis. One type of noise is non-linear distortion, for example, gain distortion or clipping. Another is reverberation, resulting from the characteristics of the room that the microphone is in. Noise also influences how people speak. To help the listener decode the message, people alter their speaking style in noisy conditions. This is called the Lombard effect (Junqua and Anglade 1990) and it has turned out to be hard to model.

As section 2.1 has discussed, speech recognisers preprocess the time-domain signal to produce feature vectors at fixed intervals. It is the influence of the noise on these feature vectors that is of interest for making speech recognisers robust to noise. The function representing the observation vector that results from vectors for the speech, the additive noise, and the convolutional noise is called the *mismatch function*. The following sections will find an expression for this influence in the log-spectral domain, the cepstral domain, and for dynamic features. The noise will be modelled in the same domain as the speech. Just like in chapter 3, the observations from the training data will be written $\mathbf{x}$ and the ones from the recognition data $\mathbf{y}$. In the context of noise-robustness, $\mathbf{x}$ is the noise-free, clean, speech, and $\mathbf{y}$ the noise-corrupted speech.

The observation distribution, the distribution that results from combining distributions of the speech and the noise through the mismatch function, is what most methods for noise-robustness in this thesis aim to model. Section 4.3 will find the exact expression.

4.2.1 Log-spectral mismatch function

The relationship between the corrupted speech, the clean speech and the noise is central to noise-robust speech recognition. The term *mismatch function* (Gales 1995) or *interaction function* (Kristjansson 2002) is often used for the function that takes the speech and noise signals and returns the corrupted speech signal. This section will derive a mismatch function in terms of speech recogniser feature vectors. Many speech

recognisers use feature vectors in the cepstral domain, which was described in section 2.1.1. Cepstral features are related to log-spectral features by the discrete cosine transform (DCT), which is a linear transformation. The reason the cepstral domain is often preferred is that the DCT goes a long way to decorrelating the features within a feature vector. For the purpose of modelling the interaction between speech, noise, and observations, however, the log-spectral domain has an advantage: the interaction is per dimension. Chapter 7 will need the log-spectral domain; the speech recogniser experiments in chapter 8 will use the cepstral domain. The following will therefore initially derive the relation for log-spectral features. The conversion to cepstral feature vectors is then found by converting to and from log-spectral features, in section 4.2.2.

The derivation will rewrite the mismatch function by going through the steps for feature extraction described in section 2.1 in parallel for different feature vectors. It will assume the power spectrum ($\beta = 2$), as is often done; appendix C.2 generalises it to other factors. It follows Deng *et al.* (2004); Leutnant and Haeb-Umbach (2009*a*;*b*).

In the time domain, the relationship between the corrupted speech $y[t]$, the clean speech $x[t]$, the additive noise $n[t]$, and the convolutional noise $h[t]$ is simply (Acero 1990)

$$y[t] = h[t] * x[t] + n[t], \tag{4.1}$$

which in the frequency domain (after applying a Fourier transformation) becomes a relation between complex numbers:

$$Y[k] = H[k]X[k] + N[k]. \tag{4.2}$$

To find the power spectrum, the absolute value of this complex value is squared:

$$\begin{aligned} |Y[k]|^2 &= |H[k]X[k] + N[k]|^2 \\ &= |H[k]|^2|X[k]|^2 + |N[k]|^2 + 2|H[k]X[k]N[k]| \cos\theta_k, \end{aligned} \tag{4.3}$$

where θ_k is the angle in the complex plane between $H[k]X[k]$ and $N[k]$. This relates to the phase difference at frequency k between the clean speech and the noise. Since

there is no process in speech production that synchronises the phase to background noise, this angle is uniformly distributed, so that the expected value of the cosine of the angle is

$$\mathcal{E}\{\cos\theta_k\} = 0. \tag{4.4}$$

To extract coefficients for speech recognition, the next step is to reduce the number of coefficients, by applying I filter bins to the power-spectral coefficients. There are usually 24 triangular bins. As in (2.1), let w_{ik} specify the contribution of the kth frequency to the ith bin. The mel-filtered power spectrum is then given by coefficients $\bar{Y}_i$:

$$\begin{aligned}\bar{Y}_i &= \sum_k w_{ik}|Y[k]|^2 \\ &= \sum_k w_{ik}\Big(|H[k]|^2|X[k]|^2 + |N[k]|^2 + 2|H[k]X[k]||N[k]|\cos\theta_k\Big)\,.\end{aligned} \tag{4.5}$$

Because this is a weighted average and the expectation of the term with $\cos\theta_k$ is 0, as shown in (4.4), the cross-term of the speech and the noise is often dropped. Though retaining this term complicates the derivation, here (4.5) will be used as is.

This value of the mel-filtered power spectrum for the corrupted speech can be defined in terms of values of the clean speech, the additive noise, and the convolutional noise in the same domain:

$$\bar{X}_i = \sum_k w_{ik}|X[k]|^2\,; \quad \bar{N}_i = \sum_k w_{ik}|N[k]|^2\,; \quad \bar{H}_i = \sum_k w_{ik}|H[k]|^2\,. \tag{4.6}$$

The rewrite requires an approximation and the introduction of a random variable. First, the convolutional noise $H[k]$ is assumed equal for all k in one bin, so that for any frequency k in bin i,

$$\bar{H}_i \simeq |H[k]|^2\,. \tag{4.7}$$

Then, replacing the terms in the right-hand side of (4.5) yields

$$\bar{Y}_i = \sum_k w_{ik}|Y[k]|^2 = \bar{H}_i\bar{X}_i + \bar{N}_i + 2\alpha_i\sqrt{\bar{H}_i\bar{X}_i\bar{N}_i}, \tag{4.8a}$$

where a new random variable α_i indicating the *phase factor* is defined as

$$\alpha_i \triangleq \frac{\sum_k w_{ik}|H[k]||X[k]||N[k]|\cos\theta_k}{\sqrt{\bar{X}_i\bar{N}_i}}. \tag{4.8b}$$

The next subsections will find properties of α_i that are independent of the spectra of the sources: first, that α_i is constrained to $[-1,+1]$, and then that its distribution is approximately Gaussian.

The mel-power-spectral coefficients are usually converted to their logarithms so that $y_i^{\log} = \log(\bar{Y}_i)$, $x_i^{\log} = \log(\bar{X}_i)$, $n_i^{\log} = \log(\bar{N}_i)$, and $h_i^{\log} = \log(\bar{H}_i)$. The mismatch expression in the log-spectral domain trivially becomes

$$\exp\left(y_i^{\log}\right) = \exp\left(x_i^{\log} + h_i^{\log}\right) + \exp\left(n_i^{\log}\right) + 2\alpha_i \exp\left(\tfrac{1}{2}\left(x_i^{\log} + h_i^{\log} + n_i^{\log}\right)\right). \tag{4.9}$$

This relationship is per coefficient i of each feature vector. Section 4.2.2 will express the mismatch function in terms of mel-cepstral feature vectors, which are linearly-transformed log-spectral vectors. It is therefore useful to write the mismatch between log-spectral vectors. The relationship between the speech vector $\mathbf{x}^{\log}$, the additive noise $\mathbf{n}^{\log}$, the convolutional noise $\mathbf{h}^{\log}$, and the observation $\mathbf{y}^{\log}$ in the log-spectral domain is

$$\mathbf{exp}\left(\mathbf{y}^{\log}\right) = \mathbf{exp}\left(\mathbf{x}^{\log} + \mathbf{h}^{\log}\right) + \mathbf{exp}\left(\mathbf{n}^{\log}\right) + 2\boldsymbol{\alpha} \circ \mathbf{exp}\left(\tfrac{1}{2}\left(\mathbf{x}^{\log} + \mathbf{h}^{\log} + \mathbf{n}^{\log}\right)\right), \tag{4.10a}$$

where $\mathbf{exp}(\cdot)$ and $\circ$ denote element-wise exponentiation and multiplication respectively. To express the observation as a function of the speech and noise, using $\mathbf{log}(\cdot)$ for the element-wise logarithm,

$$\begin{aligned}\mathbf{y}^{\log} &= \mathbf{log}\Big(\mathbf{exp}\left(\mathbf{x}^{\log} + \mathbf{h}^{\log}\right) + \mathbf{exp}\left(\mathbf{n}^{\log}\right) + 2\boldsymbol{\alpha} \circ \mathbf{exp}\left(\tfrac{1}{2}\left(\mathbf{x}^{\log} + \mathbf{h}^{\log} + \mathbf{n}^{\log}\right)\right)\Big) \\ &\triangleq \mathbf{f}(\mathbf{x}^{\log}, \mathbf{n}^{\log}, \mathbf{h}^{\log}, \boldsymbol{\alpha}).\end{aligned} \tag{4.10b}$$

The symmetry between the clean speech and the additive noise will be important in this work. However, other work often uses a different form. (4.10b) is often rewritten to bring out the effect of the noise on the clean speech, as in (4.10c), or on the channel-filtered speech, as in (4.10d):

$$\begin{aligned}\mathbf{y}^{\log} &= \mathbf{x}^{\log} + \mathbf{log}\Big(\mathbf{exp}\big(\mathbf{h}^{\log}\big) + \mathbf{exp}\big(\mathbf{n}^{\log} - \mathbf{x}^{\log}\big) \\ &\qquad + 2\boldsymbol{\alpha} \circ \mathbf{exp}\big(\tfrac{1}{2}\big(\mathbf{h}^{\log} + \mathbf{n}^{\log} - \mathbf{x}^{\log}\big)\big)\Big) \qquad (4.10c) \\ &= \mathbf{x}^{\log} + \mathbf{h}^{\log} + \mathbf{log}\Big(1 + \mathbf{exp}\big(\mathbf{n}^{\log} - \mathbf{x}^{\log} - \mathbf{h}^{\log}\big) \\ &\qquad + 2\boldsymbol{\alpha} \circ \mathbf{exp}\big(\tfrac{1}{2}\big(\mathbf{n}^{\log} - \mathbf{x}^{\log} - \mathbf{h}^{\log}\big)\big)\Big). \qquad (4.10d)\end{aligned}$$

In all cases, $\boldsymbol{\alpha}$ encapsulates the phase difference between the two signals that are added (channel-filtered speech and noise) in one mel-bin. The phase information is discarded in the conversion to the log-spectral domain, so that with speech and noise models in that domain, the phase factor $\boldsymbol{\alpha}$ is a random vector, the distribution of which will be discussed below.

Before going into the properties of the phase factor term, it is worth identifying a split in the literature on the model for the phase factor. Traditionally (Gales 1995; Moreno 1996; Acero *et al.* 2000), the phase factor term has been ignored. Some papers (Deng *et al.* 2004; Leutnant and Haeb-Umbach 2009*a*) have gone into mathematical depth to get as close as possible to the real mismatch. For example, they show the elements of $\boldsymbol{\alpha}$ to be between -1 and $+1$.

Other papers have been more pragmatic. After all, in practice, model compensation for noise robustness is a form of adaptation to the data. The parameters that in theory make up the noise model are usually estimated from the data, with the aim to maximise the likelihood of the adapted model (section 4.7 will discuss this in more detail). The difference between traditional linear transformation methods and methods for model compensation is therefore the space to which the adapted model are constrained. In both cases, it could be argued that the best choice for this space is the one that yields the lowest word error rate, which is not necessarily the mathematically correct one. The mismatch function is one element that determines this space.

In this vein, it is possible to show that the optimal value for the phase factor is the mathematically inconsistent $\alpha_i = 2.5$ for the AURORA 2 corpus (Li *et al.* 2009). What has really happened is that the mismatch function has been tuned. The possibility of tuning the mismatch function to assume the features use a specific power of the spectrum (e.g. the magnitude or power spectrum) had been noted before (Gales 1995). Setting the phase factor to 2.5 has a very similar effect to assuming the power of the spectrum used to be 0.75 (Gales and Flego 2010). This parameter setting improves performance for AURORA 2, but not for other corpora (Gales and Flego 2010). This illustrates that adjusting arbitrary parameters in the mismatch function, in effect adjusting the space in which the optimal adapted model is sought, can have an impact on word error rate in some cases.

However, the following will analyse properties of the phase factor distribution mathematically.

4.2.1.1 Properties of the phase factor

Two important observations about α_i, which was defined in (4.8b), can be made.

First, it is possible to determine the range of α_i (Deng *et al.* 2004). Since a cosine is constrained to $[-1, 1]$, from (4.8b) the following inequality holds:

$$|\alpha_i| \leq \frac{\sum_k w_{ik}|X[k]||N[k]|}{\sqrt{\bar{X}_i \bar{N}_i}}. \tag{4.11}$$

It is possible to write the fraction in (4.11) as a normalised inner product of two vectors. The vectors are $\widetilde{X}_i$ and $\widetilde{N}_i$, with entries

$$\widetilde{X}_{ik} = \sqrt{w_{ik}}|X[k]|\,; \tag{4.12a}$$

$$\widetilde{N}_{ik} = \sqrt{w_{ik}}|N[k]|\,. \tag{4.12b}$$

Then, $\sqrt{\bar{X}_i}$ in (4.11) can be written as the norm of $\widetilde{X}_i$, so that

$$\frac{\sum_k w_{ik}|X[k]||N[k]|}{\sqrt{\bar{X}_i \bar{N}_i}} = \frac{\sum_k \sqrt{w_{ik}}|X[k]|\,\sqrt{w_{ik}}|N[k]|}{\|\widetilde{X}\|\|\widetilde{N}\|} = \frac{\widetilde{X}_i^{\mathsf{T}}\widetilde{N}_i}{\|\widetilde{X}\|\|\widetilde{N}\|}, \tag{4.13}$$

which is a normalised inner product of two vectors with non-negative entries, which is always in $[0, 1]$. The inequality in (4.11) then shows that α_i is constrained to $[-1, 1]$:

$$|\alpha_i| \leq \frac{\widetilde{X}_i^T \widetilde{N}_i}{\|\widetilde{X}\| \|\widetilde{N}\|} \leq 1. \tag{4.14}$$

Second, an approximation can decouple the distribution of α_i from the values of $|X[k]|$ and $|N[k]|$ (Leutnant and Haeb-Umbach 2009*a*). Removing magnitude-spectral terms from the equation is useful because they are not usually modelled individually in speech recognisers. The assumption is that for one frequency bin i, all $|X[k]|$ have the same value, and similar for all $|N[k]|$. Since the bins overlap, this is known to be exactly true only if $|X[k]|$ is equal for all k. However, especially for the narrowest bins i, the lower ones (see section 2.1.1), it may be a reasonable approximation. By dividing both sides of the fraction in (4.8b) by $|X[k]|$ and $|N[k]|$, assuming that they are equal for all k, α_i is approximated as

$$\begin{aligned} \alpha_i &= \frac{\sum_k w_{ik} |X[k]||N[k]| \cos\theta_k}{\sqrt{\sum_k w_{ik} |X[k]|^2 \sum_k w_{ik} |N[k]|^2}} \\ &\simeq \frac{\sum_k w_{ik} \cos\theta_k}{\sqrt{\left(\sum_k w_{ik}\right)\left(\sum_k w_{ik}\right)}} = \frac{\sum_k w_{ik} \cos\theta_k}{\sum_k w_{ik}}, \end{aligned} \tag{4.15}$$

α_i can thus be approximated as a weighted average of cosines over independently distributed uniform variables θ_k. The distribution that this model produces is close to the empirical distribution on various combinations of noises and signal-to-noise ratios on the AURORA 2 corpus (Leutnant and Haeb-Umbach 2009*a*).

The distribution of the phase factor The distribution of α_i is most easily viewed by sampling from it. Drawing a sample is straightforward if three assumptions are used. Since there is no process in speech production that synchronises the speech phase with the noise phase at a specific frequency, the phase θ_k is uniformly distributed and independent of noise and speech. Also, the phase is assumed independently distributed for different frequencies k. Thirdly, the distribution of α_i is approximated as in (4.15), removing the influence of particular value of the speech and noise signals.

procedure DRAW-α_i-SAMPLE(i)
  for frequency k for which $w_{ik} \neq 0$ **do**
    sample $\theta_k \sim \text{Unif}[-\pi, +\pi]$;
    compute $v_k = \cos\theta_k$.
  Compute the sample $\alpha_i = \frac{\sum_k w_{ik} v_k}{\sum_k w_{ik}}$.

Algorithm 2 *Drawing a sample from* $p(\alpha_i)$.

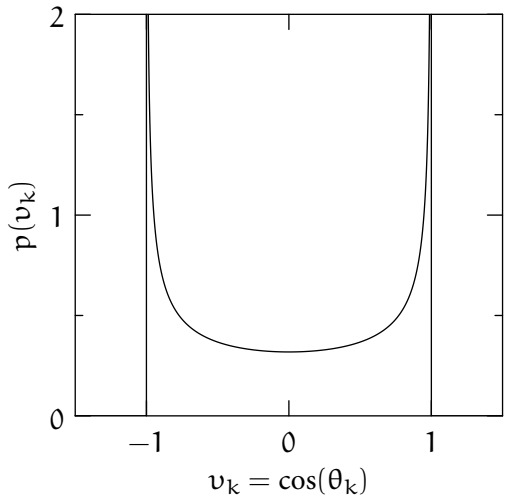


Figure 4.1 *The distribution of* $v_k = \cos\theta_k$ *for one frequency* k *(after Leutnant and Haeb-Umbach 2009*b*).*

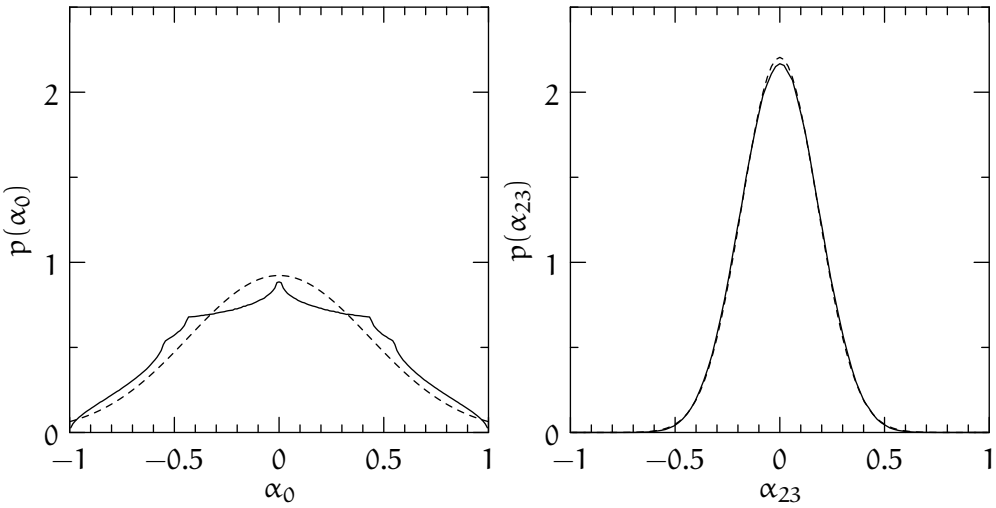
Algorithm 2 shows how to sample from α_i. Samples for θ_k are drawn independently for all frequencies in one bin. This yields samples for $\cos\theta_k$, which will be called v_k. A sample for $p(\alpha_i)$ can be drawn by taking the weighted average over these.

It is also possible to find a parametric distribution of $v_k = \cos\theta_k$. It can be shown to be (Leutnant and Haeb-Umbach 2009*a*;*b*)

$$p(v_k) = \begin{cases} \frac{1}{\pi\sqrt{1-v_k^2}}, & |v_k^2| \leq 1, \\ 0, & \text{otherwise.} \end{cases} \tag{4.16}$$

This distribution is pictured in figure 4.1.

The distribution of α_i is a weighted average of distributions over the v_ks in the bin. As the number of frequencies goes up, the central limit theorem means that the dis-



(a) Bin 0 is the narrowest, so that α_0 has the least Gaussian-like distribution.

(b) Bin 23 is the widest, so that α_{23} has the most Gaussian-like distribution.

Figure 4.2 *The distribution of α_i for different mel-filter channels i (—), and their Gaussian approximations (- -).*

tribution of α_i becomes closer to a Gaussian (Deng *et al.* 2004). For lower-frequency bins, the number of frequencies that is summed over is smaller, so the distribution of α_i is expected to be further away from a Gaussian. This effect can be seen in figure 4.2, which shows the distributions for two values of i. These distributions were found by sampling many times from α_i using algorithm 2 on the preceding page. The dashed lines show Gaussian approximations. For α_0, the Gaussian is least appropriate, but still a reasonable approximation.

The covariance of the Gaussian can be set to the second moment of the real distribution. It can be shown that, again assuming that all spectral coefficients in one filter bin are equal, that (Leutnant and Haeb-Umbach 2009*a*)

$$\sigma^2_{\alpha,i} \triangleq \mathcal{E}\{\alpha_i^2\} = \frac{\sum_k w_{ik}^2}{2\left(\sum_k w_{ik}\right)^2}. \tag{4.17}$$

This gives values very close to the actual variance of α_i on various subsets of AURORA 2 (Leutnant and Haeb-Umbach 2009*a*). This work will therefore approximate the dis-

tribution of α_i as a truncated Gaussian with

$$p(\alpha_i) \propto \begin{cases} \mathcal{N}(\alpha_i;\, 0,\, \sigma^2_{\alpha,i}) & \alpha_i \in [-1, +1]; \\ 0 & \text{otherwise.} \end{cases} \tag{4.18}$$

Evaluating this density at any point requires the normalisation constant $1/\int_{-1}^{+1} \mathcal{N}(\alpha;\, 0,\, \sigma^2)\mathrm{d}\alpha$, which could be approximated with an approximation to the Gaussian's cumulative distribution function. However, it is straightforward to draw samples from this distribution, by sampling from the Gaussian and rejecting any samples not in $[-1, +1]$.

4.2.2 *Cepstral mismatch function*

The recognition experiments in this thesis will use cepstral (MFCC) features. To convert the log-spectral mismatch function in (4.10b) to the cepstral domain, the feature vectors must be converted to and from log-spectral feature vectors. This uses the DCT matrix $\mathbf{C}$ (analogously to (2.6)):

$$\mathbf{y}^s = \mathbf{C}\mathbf{y}^{\log}. \tag{4.19a}$$

This converts a log-spectral feature vector into a cepstral-domain one. As discussed in section 2.1, the DCT matrix is normally truncated, so that the cepstral feature vector $\mathbf{y}^s$ is shorter than the log-spectral one $\mathbf{y}^{\log}$. Converting from $\mathbf{y}^s$ to $\mathbf{y}^{\log}$ therefore incurs smoothing of the coefficients: high-frequency changes from coefficient to coefficient disappear. The pseudo-inverse of the truncated DCT matrix is performed by the truncated transpose of the matrix, which will be written $\mathbf{C}^{-1}$. The reconstructions of the speech and the additive and convolutional noise are then

$$\mathbf{x}^{\log} \simeq \mathbf{C}^{-1}\mathbf{x}^s; \qquad \mathbf{n}^{\log} \simeq \mathbf{C}^{-1}\mathbf{n}^s; \qquad \mathbf{h}^{\log} \simeq \mathbf{C}^{-1}\mathbf{h}^s. \tag{4.19b}$$

Substituting (4.10b) and then (4.19b) into (4.19a) yields the cepstral-domain mismatch function:

$$\begin{aligned} \mathbf{y}^s &\simeq \mathbf{C}\mathbf{log}\Big(\mathbf{exp}\big(\mathbf{C}^{-1}(\mathbf{x}^s + \mathbf{h}^s)\big) + \mathbf{exp}\big(\mathbf{C}^{-1}\mathbf{n}^s\big) + 2\boldsymbol{\alpha} \circ \mathbf{exp}\big(\tfrac{1}{2}\mathbf{C}^{-1}\big(\mathbf{x}^s + \mathbf{h}^s + \mathbf{n}^s\big)\big)\Big) \\ &\triangleq \mathbf{f}(\mathbf{x}^s, \mathbf{n}^s, \mathbf{h}^s, \boldsymbol{\alpha}^s). \end{aligned} \tag{4.20}$$

4.2.3 *Mismatch function for dynamic coefficients*

As section 2.1.2 has discussed, speech recogniser feature vectors normally contain static as well as dynamic features. Dynamics features represent the change over time of the static features. A mismatch function is required for dynamic features to compensate model parameters. The dynamics features $\mathbf{y}_t^\Delta$ are a linear combination of static features in a window. As in section 2.1.2, the window will be assumed ± 1 for exposition. The dynamic coefficients of the corrupted speech are

$$\mathbf{y}_t^\Delta = \mathbf{D}^\Delta \begin{bmatrix} \mathbf{y}_{t-1}^s \\ \mathbf{y}_t^s \\ \mathbf{y}_{t+1}^s \end{bmatrix} = \mathbf{D}^\Delta \begin{bmatrix} \mathbf{f}(\mathbf{x}_{t-1}^s, \mathbf{n}_{t-1}^s, \mathbf{h}_{t-1}^s, \boldsymbol{\alpha}_{t-1},) \\ \mathbf{f}(\mathbf{x}_t^s, \mathbf{n}_t^s, \mathbf{h}_t^s, \boldsymbol{\alpha}_t,) \\ \mathbf{f}(\mathbf{x}_{t+1}^s, \mathbf{n}_{t+1}^s, \mathbf{h}_{t+1}^s, \boldsymbol{\alpha}_{t+1},) \end{bmatrix}, \tag{4.21}$$

where $\mathbf{D}^\Delta$ projects an extended feature vector to dynamic features. The specific instance of this form where the dynamics are computed as the difference between the statics at time $t + w$ and $t - w$ ("simple differences") was used in Gales (1995) for noise-robustness.

However, usually an approximation is used: the *continuous-time approximation* (Gopinath *et al.* 1995). The approximation assumes that the dynamic coefficients are actual derivatives with respect to time:

$$\mathbf{y}_t^\Delta \simeq \frac{\partial \mathbf{f}(\mathbf{x}_t^s, \mathbf{n}_t^s, \mathbf{h}_t^s, \boldsymbol{\alpha}_t,)}{\partial t}. \tag{4.22}$$

Section 4.4.2 will discuss how this approximation is used for speech recogniser compensation with the state of the art vector Taylor series approximation.

4.3 The corrupted speech distribution

Section 4.1 has argued that if the models of the clean speech and the noise, and the mismatch function are the correct ones, decoding with the true corrupted-speech distribution would yield the optimal recogniser performance. Later sections will discuss specific methods for model compensation, and model-based feature enhancement. This section discusses how the speech and noise models can be combined to find a

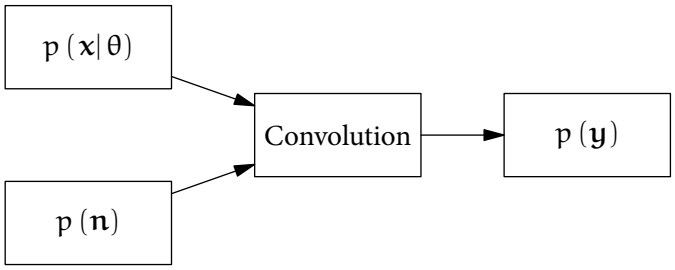


Figure 4.3 *Model combination: a schematic view.*

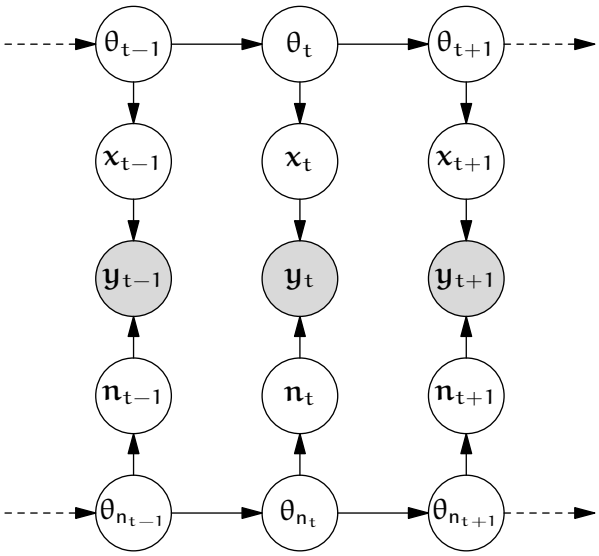


Figure 4.4 *Model combination: a speech HMM with states θ_t and a noise HMM with states θ_{n_t} generate feature vectors that combine to form observations $\mathbf{y}_t$.*

model for the noise-corrupted speech. This process is called *model combination*, and is pictured in figure 4.3. This section will assume the speech to be modelled by an HMM. It would be possible to model the noise with a hidden Markov model that is independent of the speech. This allows for as much structure for the noise as the speech model does. Figure 4.4 depicts a graphical model that combines a speech HMM with a noise HMM. As in section 2.2, the speech states θ_t generate feature vectors $\mathbf{x}_t$. In this model, the noise states θ_{n_t} similarly generate feature vectors $\mathbf{n}_t$. They combine to form observation vector $\mathbf{y}_t$.

Using this model directly results in a two-dimensional model containing a state

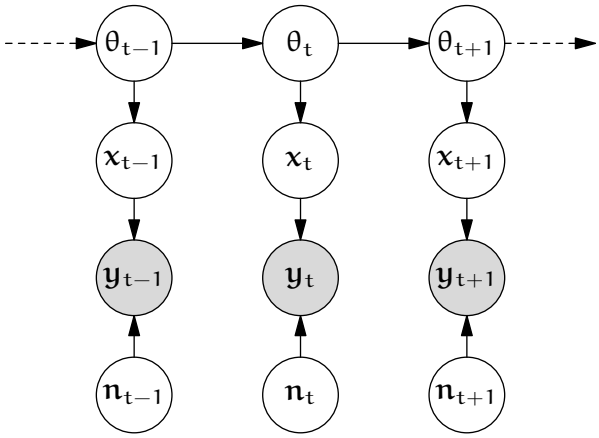


Figure 4.5 *Model combination with a simplified noise model compared to figure 4.4: the noise feature vectors are independent and identically distributed.*

for every pair of the clean speech and noise state (θ_t, θ_{n_t}). For recognition, a "three-dimensional Viterbi decoder" (Varga and Moore 1990) can be used (the third dimension is time). A problem is that the number of states in the resulting model explodes to the product of the number of states in the clean speech and noise models. Increasing the number of states is undesirable since it slows down decoding. Another problem is that unlike the speech model, the noise model is usually not known in advance and usually has to be estimated from test data. (Section 4.7 will discuss this in greater detail.) A noise model with fewer parameters can be estimated robustly on less data.

It is therefore standard to model the noise as independent and identically distributed. Figure 4.5 depicts that: the noise at each time instance is independent. The number of states in the HMM that results from combining this model for the speech and the noise is the same as of the original, clean speech, HMM. To keep the number of parameters low, the prior over the noise feature vectors $\mathbf{n}_t$ is usually restricted to be one Gaussian for the additive noise, and the convolutional noise is assumed fixed:

$$\mathbf{n} \sim \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n); \qquad \mathbf{h} = \boldsymbol{\mu}_h. \tag{4.23}$$

This work will therefore use the noise model $\mathcal{M}_n = \{\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n, \boldsymbol{\mu}_h\}$. Section 4.7 will discuss the structure of its parameters and how to estimate them.

Per time frame, the noise is assumed independent and identically distributed and

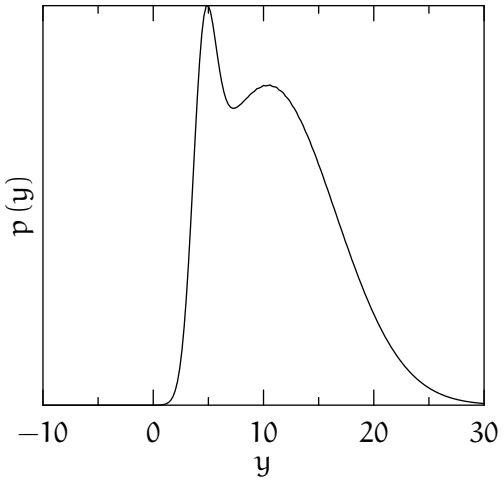


Figure 4.6 *The corrupted speech distribution for speech* $x \sim \mathcal{N}(10.5, 36)$ *noise* $n \sim \mathcal{N}(4, 1)$*, and phase factor* $\alpha = 0$*.*

the speech is independent and identically distributed given sub-phone θ. The phase factor is assumed independent of the noise and speech, and independent and identically distributed per time frame, as in (4.15). None of those variables are directly observed. For decoding, it is therefore possible to marginalise them out and directly describe the distribution of the observation vector given the sub-phone. This can be performed in either the log-spectral domain (with mismatch function $\mathbf{f}$ as in (4.10b)) or the cepstral domain (with mismatch function $\mathbf{f}$ as in (4.20)). Given vectors for the speech, noise, and the phase factor, the observation vector is fully determined by the mismatch function. Denoting vectors in the appropriate domain with $\mathbf{y}, \mathbf{x}, \mathbf{n}, \mathbf{h}, \boldsymbol{\alpha}$, the distribution of the observation vector is

$$
\begin{aligned}
p^{(\theta)}(\mathbf{y}) &= p(\mathbf{y}|\theta) \\
&= \int p(\mathbf{y}|\mathbf{x})\, p(\mathbf{x}|\theta)\, d\mathbf{x} && \text{(4.24a)} \\
&= \int\int\int p(\mathbf{y}|\mathbf{x}, \mathbf{n}, \mathbf{h})\, p(\mathbf{h})\, d\mathbf{h}\, p(\mathbf{n})\, d\mathbf{n}\, p(\mathbf{x}|\theta)\, d\mathbf{x} && \text{(4.24b)} \\
&= \int\int\int\int \delta_{\mathbf{f}(\mathbf{x},\mathbf{n},\mathbf{h},\boldsymbol{\alpha})}(\mathbf{y})\, p(\boldsymbol{\alpha})\, d\boldsymbol{\alpha}\, p(\mathbf{h})\, d\mathbf{h}\, p(\mathbf{n})\, d\mathbf{n}\, p(\mathbf{x}|\theta)\, d\mathbf{x} && \text{(4.24c)} \\
&= \int\int\int\int \delta_{\mathbf{f}(\mathbf{x},\mathbf{n},\mathbf{h},\boldsymbol{\alpha})}(\mathbf{y})\, p(\mathbf{x}, \mathbf{n}, \boldsymbol{\alpha}, \mathbf{h}|\theta)\, d\boldsymbol{\alpha} d\mathbf{h} d\mathbf{n} d\mathbf{x}. && \text{(4.24d)}
\end{aligned}
$$

where $\delta_{\mathbf{f}(\mathbf{x},\mathbf{n},\mathbf{h},\boldsymbol{\alpha})}(\mathbf{y})$ is the Dirac delta at $\mathbf{f}(\mathbf{x}, \mathbf{n}, \mathbf{h}, \boldsymbol{\alpha})$. This expression is exact. (It is

still valid if $\boldsymbol{\alpha}$ is fixed: then $p(\boldsymbol{\alpha})$ is a Dirac delta at the fixed value.) However, since the mismatch function $\mathbf{f}$ is non-linear, for non-trivial distributions for $\mathbf{x}$, $\mathbf{n}$, $\mathbf{h}$, $\boldsymbol{\alpha}$, like Gaussians, the expression does not have a closed form. Figure 4.6 shows a one-dimensional example of the corrupted speech distribution for Gaussian speech and noise. The topic of much of this thesis will discuss how to best approximate the distribution.

4.3.1 Sampling from the corrupted speech distribution

Expressing the corrupted speech distribution parametrically is normally not possible. However, if distributions for $\mathbf{x}$, $\mathbf{n}$, $\mathbf{h}$, and $\boldsymbol{\alpha}$ are available, then it is straightforward to draw samples $\mathbf{y}^{(l)}$ from the distribution. This applies Monte Carlo to the expression in (4.24d). The joint distribution $p(\mathbf{x}, \mathbf{n}, \mathbf{h}, \boldsymbol{\alpha}|\theta)$ is replaced by an empirical version by sampling each variable from its prior:

$$\mathbf{x}^{(l)} \sim p(\mathbf{x}|\theta)\,; \qquad \mathbf{n}^{(l)} \sim p(\mathbf{n})\,; \qquad \mathbf{h}^{(l)} \sim p(\mathbf{h})\,; \qquad \boldsymbol{\alpha}^{(l)} \sim p(\boldsymbol{\alpha})\,. \tag{4.25a}$$

The empirical distribution over these then becomes

$$\tilde{p}(\mathbf{x}, \mathbf{n}, \mathbf{h}, \boldsymbol{\alpha}|\theta) = \frac{1}{L} \sum_{l} \delta_{\mathbf{x}^{(l)}, \mathbf{n}^{(l)}, \mathbf{h}^{(l)}, \boldsymbol{\alpha}^{(l)}}((\mathbf{x}, \mathbf{n}, \mathbf{h}, \boldsymbol{\alpha}))\,. \tag{4.25b}$$

By substituting this in in (4.24d), the empirical distribution over $\mathbf{y}$ becomes

$$\tilde{p}(\mathbf{y}|\theta) = \frac{1}{L} \sum_{l} \delta_{\mathbf{f}(\mathbf{x}^{(l)}, \mathbf{n}^{(l)}, \mathbf{h}^{(l)}, \boldsymbol{\alpha}^{(l)})}(\mathbf{y})\,, \tag{4.26a}$$

where through the Dirac delta in (4.24d) the observation samples are defined by the mismatch function applied to the samples of $\mathbf{x}$, $\mathbf{n}$, $\mathbf{h}$, $\boldsymbol{\alpha}$:

$$\mathbf{y}^{(l)} = \mathbf{f}\big(\mathbf{x}^{(l)}, \mathbf{n}^{(l)}, \mathbf{h}^{(l)}, \boldsymbol{\alpha}^{(l)}\big). \tag{4.26b}$$

Sampling from the corrupted speech distribution will be used in this work to train parametric distributions (DPMC and IDPMC) in section 4.4.1, and to examine how well approximated distributions match the actual distribution in section 7.4.

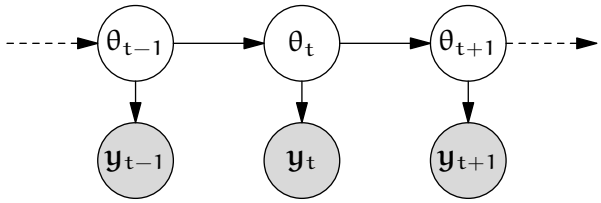


Figure 4.7 *Model compensation: the speech and noise in the graphical model in figure 4.5 have been marginalised out.*

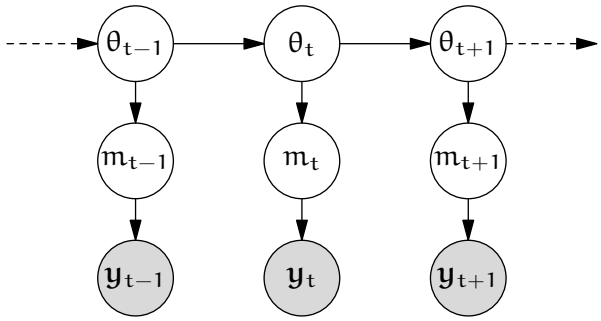


Figure 4.8 *The conventional implementation of model compensation: each component is compensated separately.*

4.4 Model compensation

Integrating out the speech and noise, in (4.24), leads to the graphical model in figure 4.7. It is a simple HMM. This has the useful property that the structure is the same as that of a normal speech recogniser for clean speech (in figure 2.3 on page 19), with the state output distribution now modelling the corrupted speech. In an implementation, this means that if the corrupted speech can be approximated with the same form of model as clean speech uses, the speech parameters in the original speech recogniser can be replaced by estimated corrupted speech parameters. This is called *model compensation*.

For computational reasons the compensation is normally performed per Gaussian mixture component rather than per sub-phone state. Figure 4.8 show a graphical model for this, which has the same structure as the model in figure 2.4 on page 20.

Model compensation normally approximates (4.24) with a Gaussian:

$$q^{(m)}(\mathbf{y}) = \mathcal{N}\left(\mathbf{y}; \boldsymbol{\mu}_y^{(m)}, \boldsymbol{\Sigma}_y^{(m)}\right). \tag{4.27}$$

This Gaussian then replaces the clean speech Gaussian in the original speech recogniser (in (2.12)). This Gaussian, $\mathcal{N}\left(\boldsymbol{\mu}_x^{(m)}, \boldsymbol{\Sigma}_x^{(m)}\right)$, also gives the clean speech statistics.

As section 4.3 has shown, given standard speech and noise distributions, the corrupted speech distribution has no closed form. Model compensation methods therefore minimise the KL divergence between the predicted distribution $p^{(m)}$ in (4.24) and the Gaussian $q^{(m)}$:[1]

$$\begin{aligned} q^{(m)} &:= \arg\min_{q^{(m)}} \mathcal{KL}\left(p^{(m)} \middle\| q^{(m)}\right) \\ &= \arg\min_{q^{(m)}} \int p^{(m)}(\mathbf{y}) \log q^{(m)}(\mathbf{y}) \mathrm{d}\mathbf{y}. \end{aligned} \tag{4.28}$$

The observation vector $\mathbf{y}$ contains static coefficients $\mathbf{y}^s$ and dynamic coefficients $\mathbf{y}^\Delta$. It is often harder to estimate compensation for dynamics. (Indeed, most of chapter 5 will be dedicated to that subject.) Not all compensation methods that the next sections will discuss also compensate dynamic parameters.

There is a range of model compensation schemes that produce Gaussians, including the log-normal approximation (Gales 1995), Jacobian compensation (Sagayama *et al.* 1997), the unscented transformation (Hu and Huo 2006; van Dalen and Gales 2009*b*), and a piecewise linear approximation to the mismatch function (Seltzer *et al.* 2010). Only the following schemes will be be discussed in this work. DPMC, which section 4.4.1 will discuss, approximates the predicted distribution $p^{(m)}$ by sampling, and trains the optimal Gaussian on that. A variant is iterative DPMC, which also uses samples, but trains mixtures of Gaussians rather than single Gaussians. The state-of-the-art scheme, which will be the topic of section 4.4.2, applies a vector Taylor series (VTS) approximation to the mismatch function, so that the resulting predicted distribution becomes Gaussian. Section 4.4.3 will describe a scheme that speeds up compensation by finding compensation per base class rather than per component. Finally,

[1] In chapter 6, this type of scheme will be interpreted as an instantiation of *predictive* methods, which approximate a predicted distribution.

section 4.4.4 will discuss single-pass retraining, which trains a speech recogniser on artificially corrupted speech to find the ideal compensation in some sense.

4.4.1 *Data-driven parallel model combination*

Data-driven parallel model combination (DPMC) (Gales 1995) approximates the distributions with samples and applies the correct mismatch function. A Gaussian assumption is made only when training the corrupted speech distribution on the samples. In the limit, it finds the optimal Gaussian distribution for the corrupted speech.

The original algorithm did not use phase factor $\boldsymbol{\alpha}$; however, the generalisation to include this term is straightforward. DPMC represents the predicted distribution $p^{(m)}$ by an empirical version $\tilde{p}^{(m)}$. Section 4.3.1 has discussed how to find this distribution by sampling. The empirical distribution has L delta spikes at positions $\mathbf{y}^{(l)}$ (see (4.26a)):

$$\tilde{p}^{(m)}(\mathbf{y}) = \frac{1}{L} \sum_{l} \delta_{\mathbf{y}^{(l)}}(\mathbf{y}) \,. \tag{4.29}$$

The parametric distribution for the corrupted speech that DPMC finds is chosen to minimise the KL divergence with the empirical distribution $\tilde{p}$, approximating (4.28) with

$$\begin{aligned} q^{(m)} &:= \arg\min_{q} \mathcal{KL}\big(\tilde{p}^{(m)} \big\| q\big) \\ &= \arg\max_{q} \int \tilde{p}^{(m)}(\mathbf{y}) \log q(\mathbf{y}) \mathrm{d}\mathbf{y} \\ &= \arg\max_{q} \sum_{l} \log q(\mathbf{y}^{(l)}). \end{aligned} \tag{4.30}$$

This is equivalent to finding the maximum-likelihood setting for $q^{(m)}$ from the samples. Standard DPMC finds a Gaussian distribution for the corrupted speech:

$$q^{(m)}(\mathbf{y}) = \mathcal{N}\Big(\mathbf{y};\, \boldsymbol{\mu}_y^{(m)},\, \boldsymbol{\Sigma}_y^{(m)}\Big) \,. \tag{4.31a}$$

The maximum-likelihood setting for its mean and covariance parameters are set to:

$$\boldsymbol{\mu}_y^{(m)} := \mathcal{E}_{\tilde{p}^{(m)}}\{\mathbf{y}\} = \frac{1}{L}\sum_{l=1}^{L}\mathbf{y}^{(l)}; \tag{4.31b}$$

$$\boldsymbol{\Sigma}_y^{(m)} := \mathcal{E}_{\tilde{p}}\{\mathbf{y}\mathbf{y}^{\mathsf{T}}\} - \boldsymbol{\mu}_y^{(m)}\boldsymbol{\mu}_y^{(m)^{\mathsf{T}}} = \left(\frac{1}{L}\sum_{l=1}^{L}\mathbf{y}^{(l)}\mathbf{y}^{(l)^{\mathsf{T}}}\right) - \boldsymbol{\mu}_y^{(m)}\boldsymbol{\mu}_y^{(m)^{\mathsf{T}}}, \tag{4.31c}$$

where $\mathcal{E}_{\tilde{p}}\{\cdot\}$ denotes the expectation under $\tilde{p}$.

This allows the static parameters to be compensated. In previous work on DPMC a method for compensating the dynamic parameters was proposed (Gales 1995). This approach is only applicable when simple differences (linear regression using a window of one time instance left and one right) are used. It uses the mismatch function for simple differences from section 4.2.3: by modelling the static coefficients from the previous time instance to the feature vector, $\mathbf{x}_{t-1}^{s}$, the dynamic coefficients for the noise-corrupted speech can be found using[2]

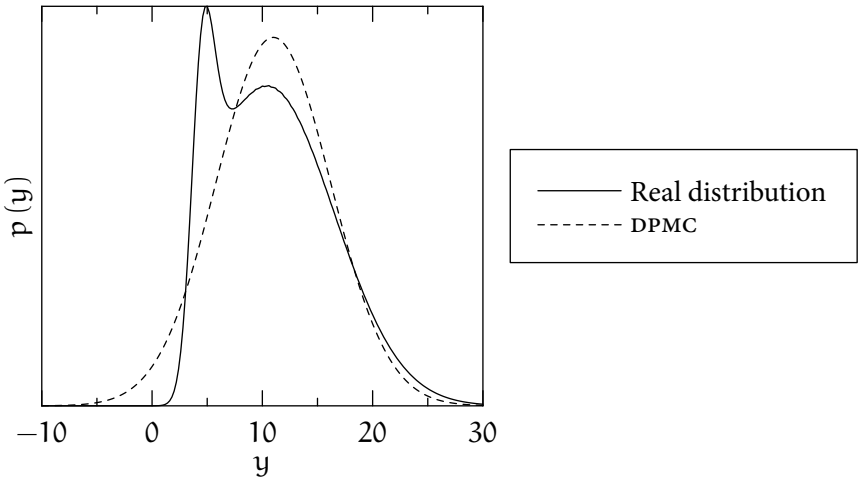
$$\mathbf{y}_t^{\Delta(k)} = \mathbf{f}\big(\mathbf{x}_t^{\Delta(l)} + \mathbf{x}_{t-1}^{s(l)}, \mathbf{n}_t^{\Delta(k)} + \mathbf{n}_{t-1}^{s(l)}, \mathbf{h}^{s}\big) - \mathbf{f}\big(\mathbf{x}_{t-1}^{s(l)}, \mathbf{n}_{t-1}^{s(l)}, \mathbf{h}^{s}\big). \tag{4.32}$$

However, this form of approximation cannot be used for the linear-regression-based dynamic parameters, which is the form in modern speech recognisers.

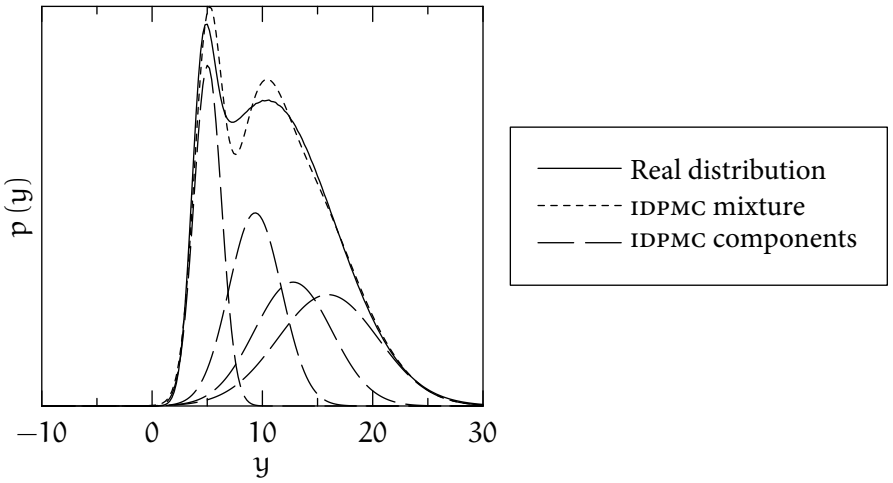
In the limit as the number of samples goes to infinity, DPMC yields the optimal Gaussian parameters given a mismatch function and distributions for the speech, noise, and phase factor. However, as a large number of samples are necessary to robustly train the noise-corrupted speech distributions, it is computationally expensive.

Figure 4.9a on the next page shows an example of the corrupted speech distribution and the DPMC approximation in one dimension. Even for the one-dimensional case, the corrupted speech can have a bimodal distribution that is impossible to model with one Gaussian, as the figure shows. Iterative DPMC (IDPMC) also finds a parametric distribution that is close to the empirical distribution, but the distribution is a mixture of Gaussians associated with a speech recogniser state rather than a single Gaussian. This allows it to model the multi-modal nature of the corrupted speech

[2] Normalisation of dynamic parameters is ignored for clarity of presentation.



(a) DPMC, with one component.



(b) IDPMC with four components.

Figure 4.9 *DPMC and IDPMC in one dimension. The corrupted speech distribution for speech* $x \sim \mathcal{N}(10.5, 36)$ *and noise* $n \sim \mathcal{N}(4, 1)$.

distribution. The word "iterative" in the name of the scheme refers to the iterations of expectation–maximisation necessary to train a mixture of Gaussians. To draw corrupted speech samples, the procedure in section 4.3.1 is applied again, but now the clean speech model is a state-conditional GMM. Analogously to (4.30), the approximation is[3]

$$\begin{aligned} q^{(\theta)} &:= \arg\min_{q^{(\theta)}} \mathcal{KL}\big(\tilde{p}^{(\theta)} \big\| q^{(\theta)}\big) \\ &= \arg\min_{q^{(\theta)}} \int \tilde{p}^{(\theta)}(\mathbf{y}) \log q^{(\theta)}(\mathbf{y}) \mathrm{d}\mathbf{y} \\ &= \arg\min_{q^{(\theta)}} \sum_{l} \log q^{(\theta)}(\mathbf{y}^{(l)}). \end{aligned} \tag{4.33}$$

The corrupted speech GMM is then trained on the samples, without reference to the clean models, and is not restricted to have the same number of components as the clean speech GMM. Let the sub-phone-conditional distribution be defined

$$q^{(\theta)}(\mathbf{y}) \triangleq \sum_{m \in \Omega^{(\theta)}} \pi_m q^{(m)}(\mathbf{y}), \tag{4.34}$$

where $\Omega^{(\theta)}$ is the set of components in the mixture of Gaussians for θ.

Training mixtures of Gaussians from samples is similar to expectation–maximisation. It works iteratively, as follows. At iteration k, first the hidden distribution, the posterior responsibilities of each component, is computed for each sample:

$$\rho^{(k)}(m, l) := \frac{\pi_m^{(k-1)} q^{(m)(k-1)}\big(\mathbf{y}^{(l)}\big)}{\sum_{m' \in \Omega^{(\theta)}} \pi_{m'}^{(k-1)} q^{(m')(k-1)}\big(\mathbf{y}^{(l)}\big)}. \tag{4.35a}$$

[3] In section 6.1.2 this will be analysed in terms of a predictive method, as an approximation of (6.9).

The new parameters for the mixture distribution are then trained with maximum likelihood using the distribution over the hidden variables. The component weights, means, and covariances for iteration k become

$$\pi_m^{(k)} := \frac{1}{L} \sum_l \rho^{(k)}(m, l); \tag{4.35b}$$

$$\boldsymbol{\mu}_y^{(m)(k)} := \frac{1}{\sum_l \rho^{(k)}(m, l)} \sum_l \rho^{(k)}(m, l) \mathbf{y}^{(l)}; \tag{4.35c}$$

$$\boldsymbol{\Sigma}_y^{(m)(k)} := \left(\frac{1}{\sum_l \rho^{(k)}(m, l)} \sum_l \rho^{(k)}(m, l) \mathbf{y}^{(l)} \mathbf{y}^{(l)\mathsf{T}} \right) - \boldsymbol{\mu}_y^{(m)(k)} \boldsymbol{\mu}_y^{(m)(k)\mathsf{T}}. \tag{4.35d}$$

Initialisation for training Gaussian mixture models is often a problem. However, in this case, DPMC provides a sensible initial setting with the number of components equal to the original state-conditional mixture. To increase the number of components, "mixing up" can be used, which progressively increases the number of components. The component with the largest mixture weight is split into two components with different offsets on the means, and a few iterations of expectation–maximisation training are run. This is repeated until the mixture has the desired number of components.

Figure 4.9 on page 79 shows how the approximation becomes more accurate when the number of Gaussians increases. In the limit as the number of Gaussians M goes to infinity, and the components are trained well, the mixture of Gaussians becomes equal to the real distribution.

However, this requires each component to be trained well, for which it needs sufficient samples. When increasing the number of components M, the number of total samples L must increase by at least the same factor. An iteration of expectation–maximisation takes $\mathcal{O}(ML)$ time, so in effect this is $\mathcal{O}(M^2)$. Additionally, the number of iterations of mixing up, which needs a number of iterations of expectation–maximisation at each step, increases linearly with M. In practice, then, the time complexity of IDPMC is at least $\mathcal{O}(M^3)$. This becomes impractical very quickly, especially since compared to figure 4.9, which shows only a one-dimensional example, many components

may be required to model the distribution well in a high-dimensional space.

4.4.2 *Vector Taylor series compensation*

Vector Taylor series (VTS) compensation (Moreno 1996; Acero *et al.* 2000; Deng *et al.* 2004) is a standard method that is faster than DPMC. Rather than approximating the noise-corrupted speech distribution directly, it applies a per-component vector Taylor series approximation to the mismatch function $\mathbf{f}$ in (4.20). The most important result of this is that, given Gaussians for the clean speech, the noise, and the phase factor, the predicted noise-corrupted speech also becomes Gaussian. VTS compensation does not aim to minimise any criterion, like the KL divergence.

The first-order vector Taylor series approximation to the mismatch function $\mathbf{f}$ with expansion point $(\mathbf{x}^s_0, \mathbf{n}^s_0, \mathbf{h}^s_0, \boldsymbol{\alpha}_0)$ is

$$\begin{aligned}\mathbf{f}^{(m)}_{\text{vts}}(\mathbf{x}^s, \mathbf{n}^s, \mathbf{h}^s, \boldsymbol{\alpha}) &= \mathbf{f}(\mathbf{x}^s_0, \mathbf{n}^s_0, \mathbf{h}^s_0, \boldsymbol{\alpha}_0) \\ &\quad + \mathbf{J}^{(m)}_x(\mathbf{x}^s - \mathbf{x}^s_0) + \mathbf{J}^{(m)}_n(\mathbf{n}^s - \mathbf{n}^s_0) + \mathbf{J}^{(m)}_h(\mathbf{h}^s - \mathbf{h}^s_0) + \mathbf{J}^{(m)}_\alpha(\boldsymbol{\alpha} - \boldsymbol{\alpha}_0), \quad (4.36a)\end{aligned}$$

where the Jacobians for the clean speech, additive noise, and phase factor are

$$\begin{aligned}\mathbf{J}^{(m)}_x &= \left.\frac{\partial \mathbf{y}^s}{\partial \mathbf{x}^s}\right|_{\mathbf{x}^s_0,\mathbf{n}^s_0,\mathbf{h}^s_0,\boldsymbol{\alpha}_0}; & \mathbf{J}^{(m)}_n &= \left.\frac{\partial \mathbf{y}^s}{\partial \mathbf{n}^s}\right|_{\mathbf{x}^s_0,\mathbf{n}^s_0,\mathbf{h}^s_0,\boldsymbol{\alpha}_0}; \\ \mathbf{J}^{(m)}_h &= \left.\frac{\partial \mathbf{y}^s}{\partial \mathbf{h}^s}\right|_{\mathbf{x}^s_0,\mathbf{n}^s_0,\mathbf{h}^s_0,\boldsymbol{\alpha}_0}; & \mathbf{J}^{(m)}_\alpha &= \left.\frac{\partial \mathbf{y}^s}{\partial \boldsymbol{\alpha}}\right|_{\mathbf{x}^s_0,\mathbf{n}^s_0,\mathbf{h}^s_0,\boldsymbol{\alpha}_0}. \quad (4.36b)\end{aligned}$$

Appendix C.1 gives expressions for these in terms of the expansion points. If the expansion point of the speech is much larger than that of the noise, the speech will dominate, so that the Jacobian for the speech $\mathbf{J}^{(m)}_x$ (in (C.3) and (C.5)) will tend to $\mathbf{I}$. The Jacobian for the noise $\mathbf{J}^{(m)}_n$ will then tend to $\mathbf{0}$. Conversely, under high noise conditions, $\mathbf{J}^{(m)}_x$ will tend to $\mathbf{0}$, and $\mathbf{J}^{(m)}_n$ will tend to $\mathbf{I}$.

An aspect that often goes unmentioned but is of importance is compensation of dynamic parameters. This usually uses the *continuous-time approximation* (Gopinath *et al.* 1995), discussed in section 4.2.3, which approximates the dynamic parameters. However, to model the influence of the phase factor $\boldsymbol{\alpha}$ on the dynamics, the dynamic

part of its covariance, $\boldsymbol{\Sigma}_{\alpha}^{\Delta}$ is required. Rather than finding it, previous work has assumed the phase factor 0 (Acero *et al.* 2000), or fixed to a different value, like 1 (Liao 2007) or 2.5 (Li *et al.* 2007). Assuming $\boldsymbol{\alpha}$ fixed causes $\boldsymbol{\alpha}^{\Delta}$ to be zero by definition. A phase factor distribution has previously only been used for feature enhancement (Deng *et al.* 2004), where no distribution over dynamics is required. The following will therefore assume that $\boldsymbol{\alpha} = 0$.

As discussed in section 2.1.2, when extracting feature vectors from audio, deltas and delta-deltas are computed from a window of static feature vectors. However, they aim to indicate time derivatives of the static coefficients. The continuous-time approximation (see section 4.2.3) assumes that they are in fact time derivatives:

$$\mathbf{y}_t^{\Delta} \simeq \left.\frac{d\mathbf{y}^s}{dt}\right|_t; \qquad \mathbf{x}_t^{\Delta} \simeq \left.\frac{d\mathbf{x}^s}{dt}\right|_t; \qquad \mathbf{n}_t^{\Delta} \simeq \left.\frac{d\mathbf{n}^s}{dt}\right|_t; \qquad \mathbf{h}_t^{\Delta} \simeq \left.\frac{d\mathbf{h}^s}{dt}\right|_t. \tag{4.37}$$

Combining this approximation and the VTS approximation in (4.36a), the dynamic coefficients become

$$\begin{aligned}\mathbf{y}_t^{\Delta} &\simeq \left.\frac{d\mathbf{y}^s}{d\mathbf{x}^s}\,\frac{d\mathbf{x}^s}{dt}\right|_t + \left.\frac{d\mathbf{y}^s}{d\mathbf{n}^s}\,\frac{d\mathbf{n}^s}{dt}\right|_t + \left.\frac{d\mathbf{y}^s}{d\mathbf{h}^s}\,\frac{d\mathbf{h}^s}{dt}\right|_t \\ &\simeq \mathbf{J}_x^{(m)}\mathbf{x}_t^{\Delta} + \mathbf{J}_n^{(m)}\mathbf{n}_t^{\Delta} + \mathbf{J}_h^{(m)}\mathbf{h}_t^{\Delta}.\end{aligned} \tag{4.38}$$

This uses the same Jacobians as the linearisation of the statics. The analogous expression for the delta-deltas yields

$$\mathbf{y}_t^{\Delta^2} \simeq \mathbf{J}_x^{(m)}\mathbf{x}_t^{\Delta^2} + \mathbf{J}_n^{(m)}\mathbf{n}_t^{\Delta^2} + \mathbf{J}_h^{(m)}\mathbf{h}_t^{\Delta^2}. \tag{4.39}$$

For clarity of notation, the following will only write first-order dynamics.

Having linearised the influence of both the static and the dynamic coefficients, the observation feature vector that results can be written as a mismatch function for static and dynamic coefficients: $\mathbf{f}_{vts}^{(m)}(\mathbf{x}, \mathbf{n}, \mathbf{h})$. It applies (4.36a) for the static coefficients and (4.38) for the dynamic coefficients. $\mathbf{f}_{vts}^{(m)}$ is a sum of linearly transformed independently Gaussian distributed variables. These are also Gaussian. For example, for the clean speech statics:

$$\mathbf{J}_x^{(m)}(\mathbf{x}^s - \mathbf{x}_0^s) \sim \mathcal{N}\left(\mathbf{J}_x^{(m)}(\boldsymbol{\mu}_x^{s(m)} - \mathbf{x}_0^s), \mathbf{J}_x^{(m)}\boldsymbol{\Sigma}_x^{s(m)}\mathbf{J}_x^{(m)^T}\right), \tag{4.40}$$

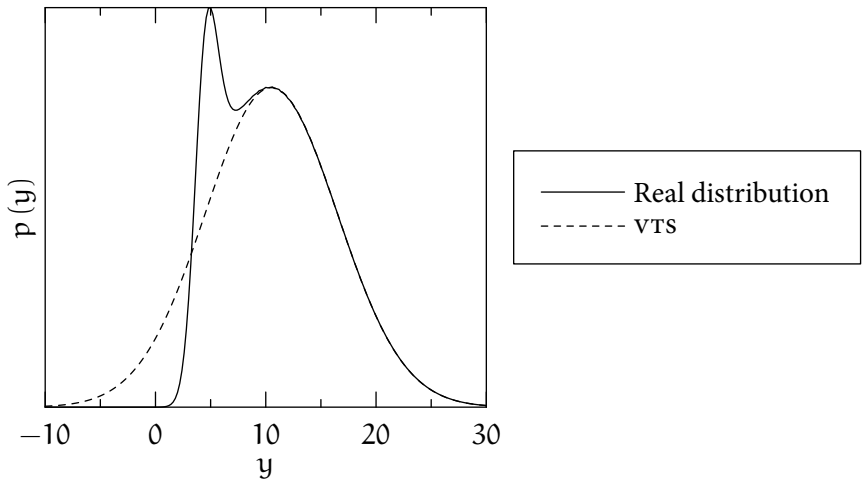


Figure 4.10 *VTS compensation in one dimension. The corrupted speech distribution for speech* $x \sim \mathcal{N}(10.5, 36)$ *and noise* $n \sim \mathcal{N}(4, 1)$.

and similar for the dynamics, and for the noise.

The linearised mismatch function $\mathbf{f}_{\text{vts}}^{(m)}$ replaces $\mathbf{f}$ in the delta function in (4.24c). The approximation for $\mathbf{y}$ then is the sum of the mismatch function at the expansion point and the two Gaussians (this assumes the convolutional noise $\mathbf{h}$ is fixed):

$$\begin{aligned} q^{(m)}(\mathbf{y}) &:= \int\int \delta_{\mathbf{f}_{\text{vts}}^{(m)}(\mathbf{x},\mathbf{n},\mu_h)}(\mathbf{y})\, \mathcal{N}(\mathbf{n};\, \mu_n,\, \Sigma_n)\, \mathrm{d}\mathbf{n}\, \mathcal{N}\big(\mathbf{x};\, \mu_x^{(m)},\, \Sigma_x^{(m)}\big)\, \mathrm{d}\mathbf{x} \\ &= \mathcal{N}\big(\mathbf{y};\, \mu_y,\, \Sigma_y\big). \end{aligned} \tag{4.41}$$

The parameters μ_y, Σ_y consist of parameters for the static and dynamic coefficients. Compensation for the static parameters applies (4.36a):

$$\mu_y^{s(m)} := \mathbf{f}(x_0^s, n_0^s, \mu_h, 0) + J_x^{(m)}(\mu_x^{s(m)} - x_0^s) + J_n^{(m)}(\mu_n^s - n_0^s); \tag{4.42a}$$

$$\Sigma_y^{s(m)} := J_x^{(m)} \Sigma_x^{s(m)} J_x^{(m)\mathsf{T}} + J_n^{(m)} \Sigma_n^s J_n^{(m)\mathsf{T}}, \tag{4.42b}$$

and compensation for dynamics applies (4.38):

$$\mu_y^{\Delta(m)} := J_x^{(m)} \mu_x^{\Delta(m)} + J_n^{(m)} \mu_n^{\Delta}; \tag{4.42c}$$

$$\Sigma_y^{\Delta(m)} := J_x^{(m)} \Sigma_x^{\Delta(m)} J_x^{(m)\mathsf{T}} + J_n^{(m)} \Sigma_n^{\Delta} J_n^{(m)\mathsf{T}}. \tag{4.42d}$$

Because in the mel-cepstral domain the Jacobians are non-diagonal, the covariance matrices $\Sigma_y^{s(m)}$ and $\Sigma_y^{\Delta(m)}$ are full even when the covariances of the clean speech and

the noise are assumed diagonal. The expansion points are usually set to the means of the distributions for the clean speech and the additive noise, so that the terms $(\boldsymbol{\mu}_x^{s(m)} - \mathbf{x}_0^s)$ and $(\boldsymbol{\mu}_n^s - \mathbf{n}_0^s)$ in (4.42a) vanish. The mean of the statics in (4.42a) then becomes

$$\boldsymbol{\mu}_y^{s(m)} := \mathbf{f}\big(\boldsymbol{\mu}_x^{s(m)}, \boldsymbol{\mu}_n^s, \boldsymbol{\mu}_h^s, \mathbf{0}\big). \tag{4.43}$$

Figure 4.10 on the preceding page illustrates a VTS approximation to the corrupted speech distribution. The approximation is reasonable, but not the same as the maximum likelihood Gaussian in figure 4.9a on page 79.

The parameters for the feature vector with statics and dynamics in (4.41) then are the concatenation of the parameters of the parts in (4.42):

$$\boldsymbol{\mu}_y^{(m)} := \begin{bmatrix} \boldsymbol{\mu}_y^{s(m)} \\ \boldsymbol{\mu}_y^{\Delta(m)} \end{bmatrix}; \qquad \boldsymbol{\Sigma}_y^{(m)} := \begin{bmatrix} \boldsymbol{\Sigma}_y^{s(m)} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_y^{\Delta(m)} \end{bmatrix}. \tag{4.44}$$

Since $\boldsymbol{\Sigma}_y^{s(m)}$ and $\boldsymbol{\Sigma}_y^{\Delta(m)}$ are full, the overall corrupted speech covariance $\boldsymbol{\Sigma}_y^{(m)}$ is block-diagonal. However, the block-diagonal structure is not normally applied for decoding, because of two problems. First, it is computationally expensive. Second, the continuous-time approximation for the dynamic parameters does not yield accurate block-diagonal compensation (section 8.1.1.1 will show this). Therefore, the standard form of VTS compensation is

$$q^{(m)} := \mathcal{N}\big(\mathbf{y};\, \boldsymbol{\mu}_y^{(m)},\, \operatorname{diag}\big(\boldsymbol{\Sigma}_y^{(m)}\big)\big), \tag{4.45}$$

where $\operatorname{diag}(\cdot)$ denotes matrix diagonalisation.

The most obviously useful effect of linearising the mismatch function is that the corrupted speech turns out Gaussian. There are also other advantages that arise from fixing the expansion points, so that the relationship between speech, noise, and corrupted speech becomes linear per component. The means of the noise model can be estimated with a fixed-point iteration (Moreno 1996) and the variance with a gradient-descent-based scheme (Liao 2007). Alternatively, since the first-order approximation makes the noise, speech, and corrupted speech jointly Gaussian, an EM approach (Kim *et al.* 1998; Frey *et al.* 2001*b*; Kristjansson *et al.* 2001) can be used. Section 4.7 will give

more details. Also, it is possible to use adaptive training with it (Liao and Gales 2007; Kalinli *et al.* 2009). These aspects make VTS compensation very useful in practice.

Compared to using a distribution over $\boldsymbol{\alpha}$, assuming it constant has two effects on the approximated distribution of the statics. One is that the term $\mathbf{J}_{\alpha}^{(m)}\boldsymbol{\Sigma}_{\alpha}\mathbf{J}_{\alpha}^{(m)\mathsf{T}}$ drops out from the covariance expression in (4.44). Since the entries of the phase factor covariance are small, this decreases the variances only slightly. Since $\boldsymbol{\Sigma}_{\alpha}$ is constant across components and $\mathbf{J}_{\alpha}^{(m)}$ changes only slightly between adjacent components, discrimination is hardly affected.

If $\boldsymbol{\alpha}$ is equal to its expected value, $\mathbf{0}$, then the mean of the compensated Gaussian does not change compared to when $\boldsymbol{\alpha}$ is assumed Gaussian. If $\boldsymbol{\alpha}$ is set to a higher value, the mean is overestimated. Also, Jacobians $\mathbf{J}_{x}^{(m)}$ and $\mathbf{J}_{n}^{(m)}$ move closer to $\frac{1}{2}\mathbf{I}$ (see appendix C.1 for the expressions).

In practice, $\boldsymbol{\alpha}$ is often assumed fixed but the noise model is estimated. This should subsume many of the effects that using a phase factor distribution would have had. This includes a wider compensated Gaussian and overestimation of the mode.

There are other ways of approximating the corrupted speech distribution with a Gaussian. One possibility is to approximate the mismatch function with a second-order vector Taylor series approximation (Stouten *et al.* 2005; Xu and Chin 2009*b*), and estimate the Gaussian to match the second moment of the resulting distribution. Another approximation that has recently attracted interest is the unscented transformation (Julier and Uhlmann 2004). This approximation draws samples like DPMC, but it the samples are chosen deterministically, and if the mismatch function were linear, then it would yield the exact distribution, just like VTS. It has been applied to feature enhancement (Shinohara and Akamine 2009), without compensation for dynamics, and model compensation (Li *et al.* 2010), with the continuous-time approximation. Yet another approach approximates the mismatch function with a piecewise linear approximation (Seltzer *et al.* 2010), the parameters of which are learned from data. Whatever the differences between how these methods deal with the mismatch function, they all approximate the corrupted speech distribution with a diagonal-covari-

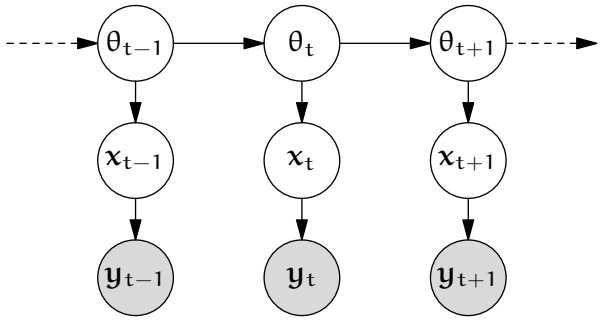


Figure 4.11 *Joint uncertainty decoding: the noise is subsumed in* $p(\mathbf{y}|\mathbf{x})$.

ance Gaussian. When applying maximum-likelihood estimation to estimate the noise model (see section 4.7), the differences between these compensation methods come down to slight variations in the parameterisation. For example, Li *et al.* (2010) finds that compensation with the VTS and the unscented transformation yields the same performance when parameters for both are correctly optimised. Rather than looking into all these variations, this thesis will look into full-covariance Gaussian (chapter 5) and non-Gaussian (chapter 7) distributions.

4.4.3 Joint uncertainty decoding

The model compensation schemes discussed so far incur considerable computational cost, since they compensate components individually. A way of overcoming this problem is to apply compensation at a different level. It is possible to describe the mismatch between the clean speech and the corrupted speech as a joint Gaussian of the speech and the observation. This is called *joint uncertainty decoding* (JUD) (Liao 2007). Figure 4.11 contains a graphical model for this. The influence of the noise is subsumed in the link between $\mathbf{x}$ and $\mathbf{y}$. The joint distribution is defined

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_x & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_y \end{bmatrix} \right). \tag{4.46}$$

If stereo data with parallel clean speech and corrupted speech is available, then this distribution can be trained directly (Neumeyer and Weintraub 1994; Moreno 1996).

However, this means the scheme cannot adapt to new noise environment. More general schemes estimate a noise model and apply a form of model compensation. Most of the parameters of this joint distribution can be found with a model compensation scheme that takes a clean speech Gaussian and produces a corrupted speech Gaussian, like VTS or DPMC. The cross-covariance $\boldsymbol{\Sigma}_{yx}$, however, needs an extension. Section 4.4.3.1 will discuss how to estimate a joint distribution with VTS and DPMC. The only approximation that this requires is the one that the original model compensation method applies. However, no additional approximation is necessary to find the corrupted-speech distribution for one speech recogniser component: it drops out as Gaussian. The following derivation will show that.

A useful property of a joint Gaussian is that the conditional distribution of one variable given the other one is also Gaussian. This is a known result, which is derived in appendix A.1.3. The distribution of the observation given the clean speech $p(\mathbf{y}|\mathbf{x})$ therefore becomes

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= \mathcal{N}\left(\mathbf{y};\, \boldsymbol{\mu}_y + \boldsymbol{\Sigma}_{yx}\boldsymbol{\Sigma}_x^{-1}(\mathbf{x}-\boldsymbol{\mu}_x),\, \boldsymbol{\Sigma}_y - \boldsymbol{\Sigma}_{yx}\boldsymbol{\Sigma}_x^{-1}\boldsymbol{\Sigma}_{xy}\right) \\ &= \mathcal{N}\left(\mathbf{y};\, \boldsymbol{\mu}_y + \mathbf{A}_{jud}^{-1}(\mathbf{x}-\boldsymbol{\mu}_x),\, \boldsymbol{\Sigma}_y - \mathbf{A}_{jud}^{-1}\boldsymbol{\Sigma}_{xy}\right) \\ &= |\boldsymbol{A}_{jud}|\mathcal{N}\left(\boldsymbol{A}_{jud}\mathbf{y};\, \boldsymbol{A}_{jud}\boldsymbol{\mu}_y + \mathbf{x} - \boldsymbol{\mu}_x,\, \boldsymbol{A}_{jud}\boldsymbol{\Sigma}_y\boldsymbol{A}_{jud}^{\mathsf{T}} - \boldsymbol{\Sigma}_x\right), \end{aligned} \tag{4.47a}$$

with

$$\mathbf{A}_{jud} = \boldsymbol{\Sigma}_x\boldsymbol{\Sigma}_{yx}^{-1}. \tag{4.47b}$$

For the component-conditional distribution of the corrupted speech, joint uncertainty decoding uses the expression for model compensation given in (4.24a). The environment model $p(\mathbf{y}|\mathbf{x})$ is replaced by the one in (4.47a). The component distribution for the corrupted speech that results from convolving this conditional with the component distribution of the clean speech is Gaussian. It can be written as a base-class-specific transformation of the clean speech parameters: an affine transformation of the observation and a bias on the covariance:

$$q^{(m)}(\mathbf{y}) \triangleq \int p(\mathbf{y}|\mathbf{x})\, p(\mathbf{x}|m)\, d\mathbf{x}$$

$$
\begin{aligned}
&= \int \mathcal{N}\Big(\mathbf{y};\, \boldsymbol{\mu}_y + \mathbf{A}_{\text{jud}}^{-1}(\mathbf{x} - \boldsymbol{\mu}_x),\, \boldsymbol{\Sigma}_y - \boldsymbol{\Sigma}_x\Big) \cdot \mathcal{N}\Big(\mathbf{x};\, \boldsymbol{\mu}_x^{(m)},\, \boldsymbol{\Sigma}_x^{(m)}\Big)\, \mathrm{d}\mathbf{x} \\
&= \int \big|\mathbf{A}_{\text{jud}}\big| \cdot \mathcal{N}\Big(\mathbf{A}_{\text{jud}}\mathbf{y};\, \mathbf{A}_{\text{jud}}\boldsymbol{\mu}_y + \mathbf{x} - \boldsymbol{\mu}_x,\, \mathbf{A}_{\text{jud}}\boldsymbol{\Sigma}_y\mathbf{A}_{\text{jud}}^{\mathsf{T}} - \boldsymbol{\Sigma}_x\Big) \\
&\qquad\qquad \cdot \mathcal{N}\Big(\mathbf{x};\, \boldsymbol{\mu}_x^{(m)},\, \boldsymbol{\Sigma}_x^{(m)}\Big)\, \mathrm{d}\mathbf{x} \\
&= \big|\mathbf{A}_{\text{jud}}\big| \cdot \mathcal{N}\Big(\mathbf{A}_{\text{jud}}\mathbf{y} + \boldsymbol{\mu}_x - \mathbf{A}_{\text{jud}}\boldsymbol{\mu}_y; \\
&\qquad\qquad\qquad \boldsymbol{\mu}_x^{(m)},\, \boldsymbol{\Sigma}_x^{(m)} + \mathbf{A}_{\text{jud}}\boldsymbol{\Sigma}_y\mathbf{A}_{\text{jud}}^{\mathsf{T}} - \boldsymbol{\Sigma}_x\Big) \\
&= \big|\mathbf{A}_{\text{jud}}\big| \cdot \mathcal{N}\Big(\mathbf{A}_{\text{jud}}\mathbf{y} + \mathbf{b}_{\text{jud}};\, \boldsymbol{\mu}_x^{(m)},\, \boldsymbol{\Sigma}_x^{(m)} + \boldsymbol{\Sigma}_{\text{bias}}\Big), \qquad (4.48a)
\end{aligned}
$$

with

$$\mathbf{A}_{\text{jud}} = \boldsymbol{\Sigma}_x\boldsymbol{\Sigma}_{yx}^{-1}; \qquad (4.48b)$$

$$\mathbf{b}_{\text{jud}} = \boldsymbol{\mu}_x - \mathbf{A}_{\text{jud}}\boldsymbol{\mu}_y; \qquad (4.48c)$$

$$\boldsymbol{\Sigma}_{\text{bias}} = \mathbf{A}_{\text{jud}}\boldsymbol{\Sigma}_y\mathbf{A}_{\text{jud}}^{\mathsf{T}} - \boldsymbol{\Sigma}_x. \qquad (4.48d)$$

If the joint distribution has full covariance, $\mathbf{A}_{\text{jud}}$ and $\boldsymbol{\Sigma}_{\text{bias}}$ are also full. However, if $\boldsymbol{\Sigma}_{\text{bias}}$ is full, the covariance matrices used for decoding become full as well, even if they were diagonal before being compensated. This means that decoding is slower. Conceivably, $\boldsymbol{\Sigma}_{\text{bias}}$ could simply be diagonalised, but this yields bad speech recogniser performance (Liao and Gales 2005). A solution is to find a joint distribution with diagonal covariances and cross-covariances, so that $\mathbf{A}_{\text{jud}}$ and $\boldsymbol{\Sigma}_{\text{bias}}$ drop out as diagonal. This is the approach taken in Liao and Gales (2006). Section 6.3 will discuss how to generate full $\mathbf{A}_{\text{jud}}$ and $\boldsymbol{\Sigma}_{\text{bias}}$ and convert them into a form that is faster in decoding.

Normally, joint uncertainty decoding associates every component m of the speech recogniser HMM with one base class r, each with a different Gaussian joint distribution. Note that a regression class tree is not necessary if the joint Gaussians are estimated with a compensation method, because the number of parameters in the noise model does not vary when the regression class tree is expanded. The speech recogniser components in one base class are close in acoustic space. Joint uncertainty decoding therefore can be viewed as partitioning the acoustic space and approximating the environment properties for each partition.

Since the model compensation scheme is applied to all components in one base class at once, it is faster than applying model compensation separately per component. Varying the number of base classes gives a trade-off between computational cost and accuracy. If every base class contains just one speech recogniser component, then the number of components is equal to the number of base classes, and the compensation is exactly equal to if the compensation scheme had been applied directly to the components.

4.4.3.1 Estimating the joint distribution

Any model compensation scheme that takes a clean speech Gaussian and produces a corrupted speech Gaussian, like VTS or DPMC, can be used to find most of the parameters of the joint distribution in (4.46). The parameters of the clean speech $\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x$ can be found from the clean training data. They can be derived from the distribution of all components in a base class, which is the obvious choice for joint uncertainty decoding. The parameters of the corrupted speech that a model compensation method finds give $\boldsymbol{\mu}_y$ and $\boldsymbol{\Sigma}_y$. That only leaves $\boldsymbol{\Sigma}_{yx} = \boldsymbol{\Sigma}_{xy}^T$ to be estimated.

With DPMC Finding the joint distribution with DPMC (Xu *et al.* 2006) extends the algorithm for model compensation with DPMC straightforwardly. The following only considers static parameters; section 5.4 will find a joint distribution over statics and dynamics.

Section 4.3.1 has discussed how to draw samples $\mathbf{y}^{(l)}$ from the noise-corrupted speech distribution. To train the joint distribution, sample pairs of both the clean speech and the corrupted speech are retained. The empirical distribution has L delta spikes at positions $(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})$, analogous to (4.29):

$$(\mathbf{x}^{(l)}, \mathbf{y}^{(l)}) \sim p^{(m)}(\mathbf{x}, \mathbf{y}); \qquad \tilde{p}(\mathbf{x}, \mathbf{y}) = \frac{1}{L} \sum_l \delta_{(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})}. \tag{4.49}$$

Just like in (4.31), the parameters are set to maximise the likelihood of the resulting distributions on the samples. However, for the joint distribution in (4.46), the mean and

covariance parameters are set at once to the mean and covariance of the tuple $(\mathbf{x}, \mathbf{y})$ under the empirical distribution:

$$\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} := \mathcal{E}_{\tilde{p}}\left\{ \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \right\} = \frac{1}{L} \sum_{l=1}^{L} \begin{bmatrix} \mathbf{x}^{(l)} \\ \mathbf{y}^{(l)} \end{bmatrix}; \tag{4.50a}$$

$$\begin{aligned} \begin{bmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_y \end{bmatrix} &:= \mathcal{E}_{\tilde{p}}\left\{ \begin{bmatrix} \mathbf{x}^{(l)} \\ \mathbf{y}^{(l)} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(l)} \\ \mathbf{y}^{(l)} \end{bmatrix}^T \right\} - \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}^T \\ &= \left(\frac{1}{L} \sum_{l=1}^{L} \begin{bmatrix} \mathbf{x}^{(l)} \\ \mathbf{y}^{(l)} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(l)} \\ \mathbf{y}^{(l)} \end{bmatrix}^T \right) - \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}^T . \end{aligned} \tag{4.50b}$$

With VTS As in section 4.4.2, the vector Taylor series approximation by itself only finds compensation for the static parameters. For the dynamic parameters, an additional approximation, the continuous-time approximation, is necessary. (Section 5.4 will estimate the joint distribution without that approximation.)

How to estimate the corrupted speech parameters μ_y, Σ_y was discussed in section 4.4.2, in (4.44). Since the speech is independent of the noise, and the mismatch function in (4.36a) is linearised, the cross-covariance of the speech and the observation does not contain any noise terms, so that (Moreno 1996; Xu *et al.* 2006):

$$\begin{aligned} \Sigma^s_{yx} &\simeq \mathcal{E}\left\{ \left(f_{vts}(x^s, n^s, h^s, 0) - \mu^s_y\right)\left(x^s - \mu^s_x\right)^T \right\} = \mathcal{E}\left\{ J_x\left(x^s - \mu^s_x\right)\left(x^s - \mu^s_x\right)^T \right\} \\ &= J_x \Sigma^s_x. \end{aligned} \tag{4.51a}$$

The cross-covariance for the dynamics follows in the same manner from the approximation of the corrupted speech dynamic coefficients in (4.37):

$$\begin{aligned} \Sigma^\Delta_{yx} &\simeq \mathcal{E}\left\{ \left(y^\Delta - \mu^\Delta_y\right)\left(x^\Delta - \mu^\Delta_x\right)^T \right\} = \mathcal{E}\left\{ J_x\left(x^\Delta - \mu^\Delta_x\right)\left(x^\Delta - \mu^\Delta_x\right)^T \right\} \\ &= J_x \Sigma^\Delta_x. \end{aligned} \tag{4.51b}$$

Similarly to (4.44), the cross-covariance of the feature vector with statics and dynamics then is the concatenation of the parameters of the parts:

$$\Sigma_{yx} := \begin{bmatrix} \Sigma^s_{yx} & 0 \\ 0 & \Sigma^\Delta_{yx} \end{bmatrix}. \tag{4.51c}$$

Alternatively, it is possible to use different approximations to find the joint distribution. This includes the methods that were mentioned at the end of section 4.4.2: second-order VTS (Xu and Chin 2009*b*), the unscented transformation (Xu and Chin 2009*a*), and the trained piecewise linear approximation should also be applicable to estimating the joint. However, the final shape of the joint distribution is still the same, with the diagonal blocks. When the noise is estimated, this limits the amount of improvement different techniques can yield. Section 5.4 will discuss how to estimate a full-covariance joint distribution.

4.4.4 Single-pass retraining

Single-pass retraining (Gales 1995) is a technique that takes a speech recogniser trained on clean speech and retrains it for corrupted speech. It requires *stereo data*, a parallel corpus of clean speech and exactly the same speech in the noisy acoustic environment. Stereo data is only available in laboratory conditions, for example, when noise is artificially added to clean speech. Also, it is unlikely that in practical situations clean speech data is available but not preferred over noise-corrupted speech data as input to a speech recogniser. Single-pass retraining is therefore not a practical technique itself, but one that more practical compensation techniques can be compared to.

Single-pass retraining trains a speech recogniser with expectation–maximisation as normally, but between the expectation step and the maximisation step of the last iteration it replaces the clean audio with artificially corrupted audio that is exactly aligned. One utterance in the stereo data will be denoted $(\mathcal{X}, \mathcal{Y})$, with $\mathcal{X}$ the clean data, and $\mathcal{Y}$ the corrupted data. The empirical distribution representing the whole training data will be denoted with joint distribution $\tilde{p}(\mathcal{X}, \mathcal{Y})$.

The expectation step of EM yields a distribution $p(\mathcal{U}|\mathcal{X})$, based on the clean speech, as in normal training. However, the speech recogniser that is trained is not a distribution over clean observations $q_{\mathcal{U}\mathcal{X}}$, but a distribution over the corrupted speech observations $q_{\mathcal{U}\mathcal{Y}}$. Just as in normal speech recogniser training, this distribution factorises into a distribution over the hidden variables $q_{\mathcal{U}}$ and a distribution over the

observations given the distribution of the hidden variables, here $q_{\mathcal{Y}|\mathcal{U}}$.

The optimisation of the former has the same effect as normal training, in (2.27b), since the observations do not directly enter into the equation. The latter, however, maximises the likelihood of $\mathcal{Y}$ rather than $\mathcal{X}$. Adapted from (2.27), then, the optimisation in the last iteration, K, is given by

$$q_{\mathcal{U}}^{(K)} := \arg\max_{q_{\mathcal{U}}} \int \tilde{p}(\mathcal{X},\mathcal{Y}) \int p(\mathcal{U}|\mathcal{X}) \log q_{\mathcal{U}}(\mathcal{U}) \, d\mathcal{U} \, d(\mathcal{X},\mathcal{Y}); \tag{4.52a}$$

$$q_{\mathcal{Y}|\mathcal{U}}^{(K)} := \arg\max_{q_{\mathcal{Y}|\mathcal{U}}} \int \tilde{p}(\mathcal{X},\mathcal{Y}) \int p(\mathcal{U}|\mathcal{X}) \log q_{\mathcal{Y}|\mathcal{U}}(\mathcal{Y}|\mathcal{U}) \, d\mathcal{U} \, d(\mathcal{X},\mathcal{Y}). \tag{4.52b}$$

Again, $\tilde{p}(\mathcal{X})$ is represented by component–time occupancies $\gamma_t^{(m)}$, found on the clean speech. The output distributions $q^{(m)}$ are then trained on the corresponding corrupted speech vectors (similar to (2.32)):

$$q_{\mathcal{Y}|\mathcal{U}}^{(k)} := \arg\max_{q_{\mathcal{Y}|\mathcal{U}}} \int \tilde{p}(\mathcal{X},\mathcal{Y}) \sum_{m} \sum_{t=1}^{T_{\mathcal{X}}} \gamma_t^{(m)} \log q^{(m)}(\mathbf{y}_t) \, d(\mathcal{X},\mathcal{Y}). \tag{4.53}$$

It is interesting to relate this to model compensation. The clean training data $\tilde{p}(\mathcal{X})$ represents samples from the real distribution of the speech. The corrupted utterance $\mathcal{Y}$ corresponding to each clean speech utterance $\mathcal{X}$, found by artificially adding the noise, is drawn from a distribution $p(\mathcal{Y}|\mathcal{X})$ representing the mismatch. Single-pass retraining therefore effectively trains a speech recogniser on a non-parametric distribution of the noise-corrupted speech that results from combining non-parametric distributions for the clean speech and the noise. This is in contrast with DPMC, discussed in section 4.4.1, which, even though it represents the corrupted speech with an empirical distribution as an intermediate step, assumes parametric distributions for both clean speech and noise, and a known mismatch function. If there is enough data to train the distributions robustly, single-pass retraining yields the optimal parameters for component distributions. A single-pass retrained speech recogniser therefore reflects the corrupted data better than model compensation methods that estimate the same form of component distribution could, because they derive from parametric representations of the speech and the noise. This work will therefore compare model

compensation methods against a single-pass retrained speech recogniser, which gives the ideal compensation for a form of output distribution.

Single-pass retraining is normally applied only in the last iteration of speech recogniser training. With additional training iterations, on just the noisy data, the component–time alignments would shift, and the state model of the speech and that of the noise cease to be independent. In model compensation, the speech and noise models are assumed independent (see figure 4.4). When training reference recognisers, this work will therefore not apply additional training iterations after single-pass retraining.

4.4.4.1 Assessing the quality of model compensation

Normally, word error rates are used to evaluate the performance of speech recognition systems. However, this does not allow a detailed assessment of which aspects of the compensation process are working well and which poorly. An alternative approach is to compare compensated systems' distributions to their ideal counterparts. A well-known tool for estimating the distance between two distributions is the Kullback-Leibler (KL) divergence, discussed in appendix A.2. The only work that has used the KL divergence to investigate the performance of model compensation methods is Gales (1995). However, it will turn out that the KL divergence can help to assess compensation quality with a much finer granularity than word error rates alone.

A useful comparison method is the occupancy-weighted average of the component-for-component KL divergence of the compensated system to the single-pass retrained system (Gales 1995). If $p^{(m)}$ is a Gaussian of the single-pass retrained system, and $q^{(m)}$ is the corresponding Gaussian of the compensated system, then this metric $\mathcal{D}$ is

$$\mathcal{D} \triangleq \frac{1}{\sum_m \gamma^{(m)}} \sum_m \gamma^{(m)} \mathcal{KL}\left(p^{(m)} \middle\| q^{(m)}\right). \tag{4.54}$$

$\gamma^{(m)}$ is the occupancy of component m in the last training iteration, for both the compensated and the single-pass retrained system. Apart from being an obvious measure of compensation quality over a whole speech recognition system, it is also propor-

tional to the expression (in (6.7)) that this thesis will analyse predictive methods such as model compensation methods as aiming to optimise.

Another useful attribute of this form of metric is pointed out in appendix A.2. Depending on the structure of the covariance matrices, it is possible to assess the compensation per coefficient or block of coefficients. When diagonal covariance matrices are used, each dimension may be considered separately. This allows the accuracy of the compensation scheme to be assessed for each dimension. Similarly, block-diagonal compensation can be examined per block of coefficients.

4.5 Observation-dependent methods

Section 4.4 has discussed model compensation methods, which approximate the corrupted speech with a parameterised distribution. This section will describe two methods that use a different approach: they start the computation only when the observation has been seen. The Algonquin algorithm (section 4.5.1) extends the VTS approximation by iteratively updating the expansion point. It comes up with a different Gaussian for each clean speech Gaussian for each observation. It is also possible to approximate the integral over the speech and noise using a piecewise linear approximation (section 4.5.2).

4.5.1 The Algonquin algorithm

The "Algonquin" algorithm (Frey *et al.* 2001*a*; Kristjansson and Frey 2002) is an extension to VTS compensation, which updates the expansion point given the observation. This thesis will view the algorithm from a different slant than the original presentation. The original presentation extended model-based feature enhancement to perform variational inference, which section 4.6.2 will discuss. This section, on the other hand, will discuss how Algonquin iteratively updates its approximation to the corrupted speech distribution for one speech Gaussian, in line with the rest of this chapter. At the same time, the algorithm updates an approximation to the the posterior of the

clean speech and the noise, which will come in useful in section 7.2.

The most important conceptual addition of the Algonquin algorithm is that it takes into account the observation vector. In the following discussion, when an actual observation is meant it will be indicated with $\mathbf{y}_t$. Whereas VTS linearises the mismatch function at the expansion point given by the means of the prior distributions of the speech and the noise, the Algonquin algorithm updates the expansion point iteratively, finding the mode of the posterior of the speech and the noise. It can therefore be seen as an iterative approach to finding the Laplace approximation to the posterior.

For the presentation of the Algonquin algorithm, the convolutional noise will be assumed zero and not written. (As long as it is assumed Gaussian, as in the original paper, the extension is trivial.) Feature vectors will be written $\mathbf{x}, \mathbf{n}, \mathbf{y}$, but they can stand for any type of feature vectors that there exists a linearisable mismatch function for. The original presentation assumed just static coefficients, but a feature vector with statics and dynamics can be used. Section 5.3.4 will introduce a version that uses "extended" feature vectors.

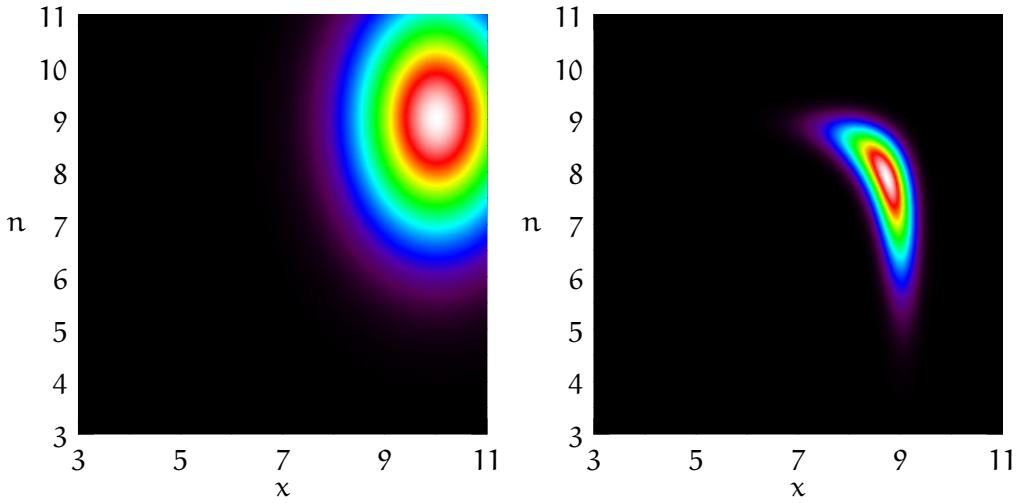
The Algonquin algorithm uses a approximation of the environment model compared to the one discussed in chapter 4.2, in (4.24c). The influence of the phase factor on the observation is captured by a Gaussian around the mode of the distribution for given $\mathbf{x}$ and $\mathbf{n}$. Thus, that distribution is approximated as

$$p(\mathbf{y}|\mathbf{x}, \mathbf{n}) = \int \delta_{\mathbf{f}(\mathbf{x},\mathbf{n},\boldsymbol{\alpha})}(\mathbf{y})\, p(\boldsymbol{\alpha}) d\boldsymbol{\alpha} \simeq \mathcal{N}(\mathbf{y};\, \mathbf{f}(\mathbf{x}, \mathbf{n}),\, \boldsymbol{\Psi})\,, \tag{4.55}$$

where $\mathbf{f}(\mathbf{x}, \mathbf{n}) \triangleq \mathbf{f}(\mathbf{x}, \mathbf{n}, \mathbf{0})$, and $\boldsymbol{\Psi}$ is the fixed covariance that models the uncertainty around the mismatch function.

Figure 4.12 on the next page shows a one-dimensional example. The prior of the speech and the noise is given in the left panel. Their posterior distribution after having seen an observation $\mathbf{y}_t$ is in the right panel. This posterior will be approximated with a Gaussian centred on its mode.

To deal with the non-linearity in the mismatch function, it is linearised (as in VTS compensation). The Algonquin algorithm iteratively updates the linearisation point to the mode of the posterior distribution. The linearisation point in iteration k is denoted



(a) The prior distribution $p(x, n)$. **(b)** The posterior $p(x, n|y = 9)$.

Figure 4.12 *The Algonquin-derived distribution of the clean speech and noise for* $x \sim \mathcal{N}(10, 1)$*;* $n \sim \mathcal{N}(9, 2)$*;* $\psi = 0.04$*;* $y = 9$*.*

by $(\mathbf{x}_0^{(k)}, \mathbf{n}_0^{(k)})$. The linearised mismatch function in iteration k is

$$\mathbf{f}_{\text{vts}}^{(k)}(\mathbf{x}, \mathbf{n}) = \mathbf{f}(\mathbf{x}_0^{(k)}, \mathbf{n}_0^{(k)}) + \mathbf{J}_x^{(k)}(\mathbf{x} - \mathbf{x}_0^{(k)}) + \mathbf{J}_n^{(k)}(\mathbf{n} - \mathbf{n}_0^{(k)}), \tag{4.56}$$

where the Jacobians are

$$\mathbf{J}_x^{(k)} = \left.\frac{dy}{dx}\right|_{\mathbf{x}_0^{(k)}, \mathbf{n}_0^{(k)}}; \qquad \mathbf{J}_n^{(k)} = \left.\frac{dy}{dn}\right|_{\mathbf{x}_0^{(k)}, \mathbf{n}_0^{(k)}}. \tag{4.57}$$

For the first iteration $k = 0$, the linearisation point is set to $(\boldsymbol{\mu}_x, \boldsymbol{\mu}_n)$, so that the mismatch function is equivalent to the one used in VTS compensation in (4.36a), leaving out the phase factor.

Using the linearised mismatch function in (4.56), the speech, the noise, and the observation become jointly Gaussian:

$$q^{(k)}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{n} \\ \mathbf{y} \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{n} \\ \mathbf{y} \end{bmatrix}; \begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_n \\ \boldsymbol{\mu}_y^{(k)} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_x & \mathbf{0} & \boldsymbol{\Sigma}_{xy}^{(k)} \\ \mathbf{0} & \boldsymbol{\Sigma}_n & \boldsymbol{\Sigma}_{ny}^{(k)} \\ \boldsymbol{\Sigma}_{yx}^{(k)} & \boldsymbol{\Sigma}_{yn}^{(k)} & \boldsymbol{\Sigma}_y^{(k)} \end{bmatrix}\right). \tag{4.58}$$

Note that the parameters of the marginal of $\mathbf{x}$ and $\mathbf{n}$ ($\boldsymbol{\mu}_x$, $\boldsymbol{\mu}_n$, $\boldsymbol{\Sigma}_x$, and $\boldsymbol{\Sigma}_n$) are given by the prior and do not depend on the iteration k. On the other hand, the covariance of the observation and the cross-covariances are found through the linearised mismatch function, which changes with every iteration. Those parameters are found as follows. Using the linearised mismatch function and assuming the error $\mathcal{N}(\mathbf{0}, \boldsymbol{\Psi})$ of the mismatch function, the distribution of the corrupted speech is Gaussian $\mathbf{y} \sim \mathcal{N}\left(\boldsymbol{\mu}_y^{(k)}, \boldsymbol{\Sigma}_y^{(k)}\right)$ with parameters similar to those for VTS compensation in (4.42):

$$\boldsymbol{\mu}_y^{(k)} := \mathcal{E}\left\{\mathbf{f}_{\text{vts}}^{(k)}(\mathbf{x}, \mathbf{n})\right\} = \mathbf{f}\left(\mathbf{x}_0^{(k)}, \mathbf{n}_0^{(k)}\right) + \mathbf{J}_x^{(k)}\left(\boldsymbol{\mu}_x - \mathbf{x}_0^{(k)}\right) + \mathbf{J}_n^{(k)}\left(\boldsymbol{\mu}_n - \mathbf{n}_0^{(k)}\right); \tag{4.59a}$$

$$\begin{aligned}
\boldsymbol{\Sigma}_y^{(k)} &:= \mathcal{E}\left\{\left(\mathbf{f}_{\text{vts}}^{(k)}(\mathbf{x}, \mathbf{n}) - \boldsymbol{\mu}_y\right)\left(\mathbf{f}_{\text{vts}}^{(k)}(\mathbf{x}, \mathbf{n}) - \boldsymbol{\mu}_y\right)^T\right\} + \boldsymbol{\Psi} \\
&= \mathcal{E}\left\{\mathbf{J}_x^{(k)}\left(\mathbf{x} - \boldsymbol{\mu}_x\right)\left(\mathbf{J}_x^{(k)}\left(\mathbf{x} - \boldsymbol{\mu}_x\right)\right)^T + \mathbf{J}_n^{(k)}\left(\mathbf{n} - \boldsymbol{\mu}_n\right)\left(\mathbf{J}_n^{(k)}\left(\mathbf{n} - \boldsymbol{\mu}_n\right)\right)^T\right\} + \boldsymbol{\Psi} \\
&= \mathbf{J}_x^{(k)}\mathcal{E}\left\{\left(\mathbf{x} - \boldsymbol{\mu}_x\right)\left(\mathbf{x} - \boldsymbol{\mu}_x\right)^T\right\}\mathbf{J}_x^{(k)^T} + \mathbf{J}_n^{(k)}\mathcal{E}\left\{\left(\mathbf{n} - \boldsymbol{\mu}_n\right)\left(\mathbf{n} - \boldsymbol{\mu}_n\right)^T\right\}\mathbf{J}_n^{(k)^T} + \boldsymbol{\Psi} \\
&= \mathbf{J}_x^{(k)}\boldsymbol{\Sigma}_x\mathbf{J}_x^{(k)^T} + \mathbf{J}_n^{(k)}\boldsymbol{\Sigma}_n\mathbf{J}_n^{(k)^T} + \boldsymbol{\Psi}.
\end{aligned} \tag{4.59b}$$

The cross-covariance between the speech and the observation and between the noise and the observation can be derived similarly:

$$\begin{aligned}
\boldsymbol{\Sigma}_{yx}^{(k)} &:= \mathcal{E}\left\{\left(\mathbf{f}_{\text{vts}}^{(k)}(\mathbf{x}, \mathbf{n}) - \boldsymbol{\mu}_y\right)\left(\mathbf{x} - \boldsymbol{\mu}_x\right)^T\right\} \\
&= \mathcal{E}\left\{\left(\mathbf{J}_x^{(k)}\left(\mathbf{x} - \boldsymbol{\mu}_x\right)\right)\left(\mathbf{x} - \boldsymbol{\mu}_x\right)^T\right\} \\
&= \mathbf{J}_x^{(k)}\mathcal{E}\left\{\left(\mathbf{x} - \boldsymbol{\mu}_x\right)\left(\mathbf{x} - \boldsymbol{\mu}_x\right)^T\right\} = \mathbf{J}_x^{(k)}\boldsymbol{\Sigma}_x;
\end{aligned} \tag{4.60a}$$

$$\boldsymbol{\Sigma}_{yn}^{(k)} := \mathbf{J}_n^{(k)}\boldsymbol{\Sigma}_n. \tag{4.60b}$$

Note that the original implementation (Frey *et al.* 2001*a*; Kristjansson 2002) diagonalised the covariances and the cross-covariances. This requires that the Jacobians are diagonalised too, otherwise the relationship between speech, noise, and observation is invalid.

Assuming the joint distribution of the speech, noise, and observations in (4.58), the posterior distribution of the speech and the noise conditioned on an observation $\mathbf{y}_t$ follows from a standard result (derived in appendix A.1.3, (A.10c)). This ap-

proximation to the posterior distribution will be written $q^{(k)}$:

$$q^{(k)}\left(\begin{bmatrix} x \\ n \end{bmatrix} \middle| y_t\right) = \mathcal{N}\Bigg(\begin{bmatrix} x \\ n \end{bmatrix}; \begin{bmatrix} \mu_x \\ \mu_n \end{bmatrix} + \begin{bmatrix} \Sigma_{xy}^{(k)} \\ \Sigma_{ny}^{(k)} \end{bmatrix} \Sigma_y^{(k)^{-1}} (y_t - \mu_y^{(k)}), \\ \begin{bmatrix} \Sigma_x & 0 \\ 0 & \Sigma_n \end{bmatrix} - \begin{bmatrix} \Sigma_{xy}^{(k)} \\ \Sigma_{ny}^{(k)} \end{bmatrix} \Sigma_y^{(k)^{-1}} \begin{bmatrix} \Sigma_{yx}^{(k)} & \Sigma_{yn}^{(k)} \end{bmatrix}\Bigg). \quad (4.61)$$

Note that the speech and noise priors are not correlated, but the posteriors are.

Algonquin sets the expansion point for the next iteration to the mean of this approximation to the posterior:

$$\begin{aligned} x_0^{(k+1)} &= \mu_x + \Sigma_{xy}^{(k)} \Sigma_y^{(k)^{-1}} (y - \mu_y^{(k)}); \\ n_0^{(k+1)} &= \mu_n + \Sigma_{ny}^{(k)} \Sigma_y^{(k)^{-1}} (y - \mu_y^{(k)}), \end{aligned} \quad (4.62)$$

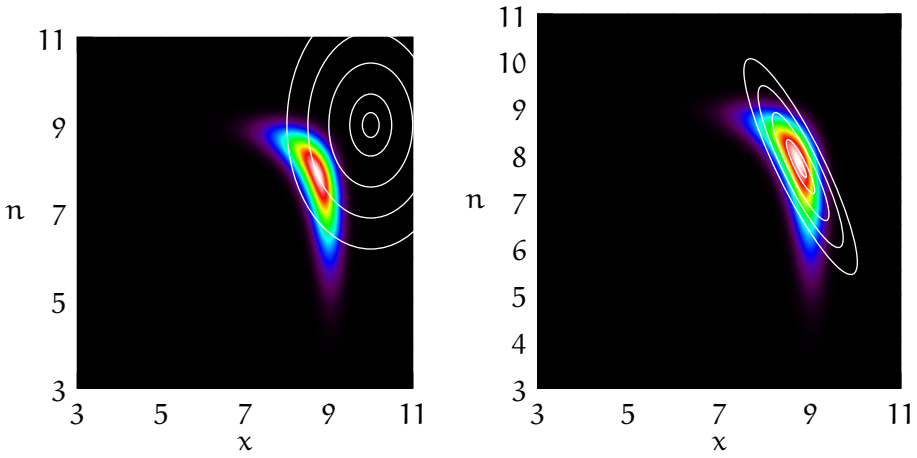
so that the expansion point is updated at every iteration, and the Gaussian approximation to the posterior is moved. There is no guarantee that the mode of the approximation converges to the mode of the real posterior: the algorithm may overshoot. A damping factor could be introduced for this, but this appears to slow down convergence without benefit (Kristjansson 2002).

After K iterations, the Gaussian approximation q to the distribution of y is found from (4.58) and (4.59):

$$\begin{aligned} q_{y_t}^{(K)}(y) &= \mathcal{N}\big(y; \mu_y^{(K)}, \Sigma_y^{(K)}\big) \\ &= \mathcal{N}\Big(y; f\big(x_0^{(K)}, n_0^{(K)}\big) + J_x^{(K)}\big(\mu_x - x_0^{(K)}\big) + J_n^{(K)}\big(\mu_n - n_0^{(K)}\big), \\ &\qquad J_x^{(K)} \Sigma_x J_x^{(K)^T} + J_n^{(K)} \Sigma_n J_n^{(K)^T} + \Psi\Big). \end{aligned} \quad (4.63)$$

Figure 4.13 on the following page shows a one-dimensional simulation of the Algonquin algorithm in (x, n)-space. The left panel shows the prior of the clean speech and the additive noise, and the right panel the Algonquin approximation to the posterior. Note again that the priors of the clean speech and noise are not correlated, but the posterior is.

Algonquin applied to the model compensates each Gaussian separately for each observation. It is not clear from the original presentation that this happens, so it is



(a) Initialisation: clean speech and noise prior.

(b) Two iterations of Algonquin: Gaussian approximation of the posterior.

Figure 4.13 *Iterations of Algonquin.*

proven in appendix D. The problem with using this Gaussian approximation is that the effective distribution is not normalised. Even though q in (4.63) is a normalised Gaussian if estimated iteratively for one $\mathbf{y}_t$, in practice it would be estimated and then applied to the same observation $\mathbf{y}_t$. Thus, in general,

$$\int q(\mathbf{y}_t) d\mathbf{y}_t \neq 1. \tag{4.64}$$

However, Algonquin applied to the model compensates each Gaussian separately for each observation. The output distribution is therefore optimised differently for each component, and there is no reason to assume their densities at one position can be compared in the way decoding normally does. The original Algonquin algorithm works around the problem that $q(\mathbf{y}_t)$ is not a normalised distribution by finding the minimum mean square error estimate of the clean speech. Section 4.6.2 will discuss this.

4.5.2 *Piecewise linear approximation*

The compensation methods in the previous sections have used a single Gaussian to represent the observation distribution. However, the actual observations are not Gaussian distributed even when the speech and noise are. It is possible to approximate the integral that gives the likelihood of the observation. Myrvoll and Nakamura (2004) use a piecewise linear approximation for this as a step in estimating the noise model for feature enhancement. Like for the Algonquin algorithm, the model for the interaction of the speech, noise, and observation the original work uses is somewhat simpler than the one in section 4.2. The following will adhere to the original presentation, because it provides good insight in both the main idea and the main limitation.

The main idea is to transform the integral over the speech and the noise into another space. It then becomes easier to apply a piecewise linear approximation. The main limitation is that the method works on a single dimension, and uses log-spectral domain coefficients. In the log-spectral domain, coefficients are highly correlated. Appendix E.2 shows that it is theoretically possible to perform the transformation of the integral in more dimensions. However, if the piecewise linear approximation in the one-dimensional case requires 8 line segments, the d-dimensional case requires 8^d plane segments. This makes the scheme infeasible for correlated feature vectors.

In one dimension, the scheme works as follows. Speech x, noise n, and observation y are assumed deterministically related, with

$$\exp(y) = \exp(x) + \exp(n)\,, \tag{4.65a}$$

which is equivalent to (4.9) where the convolutional noise and the phase factor are assumed 0. If y is observed to be y_t, and x is changed, n automatically changes too. A substitute variable u is therefore introduced to replace both x and n in the integration in the likelihood expression. It is defined

$$u = 1 - \exp(x - y_t)\,, \tag{4.65b}$$

so that (see section E.1 for details)

$$n = y_t + \log(u)\,; \tag{4.65c}$$

$$x = y_t + \log(1-u)\,. \tag{4.65d}$$

Given a speech coefficient, the distribution of the observation can be written in terms of the distribution of the noise. As this is a transformation of the variable of a probability distribution, it is a standard result (see section A.1.1) that a Jacobian is introduced:

$$p(y_t|x) = \left|\frac{\partial n(x, y_t)}{\partial y_t}\right| p(n(x, y_t))\,, \tag{4.66}$$

where $p(n(x, y_t))$ is the prior of n evaluated at the point implied by the setting of x and y_t.

The likelihood of y_t can be expressed as an integral over x. It follows from (4.65a) that $x < y_t$. It can then be transformed into an integral over $u \in [0, 1]$ as follows:

$$\begin{aligned} p(y_t) &= \int_{-\infty}^{y_t} p(y_t|x)\, p(x) dx \\ &= \int_{-\infty}^{y_t} \left|\frac{\partial n(x, y)}{\partial y}\Big|_{y_t}\right| p(n(x, y_t))\, p(x) dx \\ &= \int_0^1 \left|\frac{\partial n(x, y)}{\partial y}\Big|_{y_t}\right| p(n(u, y_t)) \left|\frac{\partial x(u, y_t)}{\partial u}\right| p(x(u, y_t)) du, \end{aligned} \tag{4.67}$$

where $p(n(u, y_t))$ is the prior of n evaluated at the point implied by the setting of u and y_t, and similar for $p(x(u, y_t))$.

Appendix E gives the complete derivation. The likelihood can be rewritten

$$\begin{aligned} p(y_t) = \exp&\left(\frac{1}{2}\sigma_n^2 + \frac{1}{2}\sigma_x^2 - \mu_n - \mu_x + 2y_t\right) \\ &\int_0^1 \mathcal{N}\Big(\log(u);\, \mu_n - \sigma_n^2 - y_t,\, \sigma_n^2\Big)\, \mathcal{N}\Big(\log(1-u);\, \mu_x - \sigma_x^2 - y_t,\, \sigma_x^2\Big)\, du. \end{aligned} \tag{4.68}$$

The variables of the two Gaussian, $\log(u)$ and $\log(1-u)$, can be approximated with piecewise linear functions. Myrvoll and Nakamura (2004) use 8 line segments. For any observation y_t, the expression then becomes a sum of integrals of a fixed factor times an integral over a Gaussian, for which well-known approximations exist.

This derivation crucially depends on the assumption that coefficients can be considered separately. To model the likelihood well, the priors $p(\mathbf{x})$ and $p(\mathbf{n})$ need to model correlations between coefficients. In the cepstral domain where correlations are not usually modelled, they become more important as the signal-to-noise ratio drops (Gales and van Dalen 2007). But any generalisation of the derivation to vectors of cepstral coefficients will need to convert to log-spectral vectors anyway, and turn diagonal-covariance priors into full-covariance ones. Note that though Myrvoll and Nakamura (2004) give the derivation for log-spectral coefficients, they apply the method to cepstral coefficients.

Appendix E.2 gives the generalisation of the algorithm in Myrvoll and Nakamura (2004) to d-dimensional log-spectral vectors. It turns out that the integral in (4.68) becomes an integral over $[0, 1]^d$. This means that the 8 line segments for the single-dimensional case become 8^d hyperplanes. It is infeasible to apply this to a standard 24-dimensional log-spectral feature space. Section 7.3 will therefore use a similar idea but a different transformation and a different approximation to the integral. It will use a Monte Carlo method, sequential importance resampling, to approximate the integral.

4.6 Model-based feature enhancement

As section 4.1 has discussed, a faster but less principled technique for noise-robustness than model compensation is feature enhancement. The objective of feature enhancement is to reconstruct the clean speech. An advantage of this is that it sidesteps issue of finding compensation for dynamics. Early schemes used spectral subtraction (Boll 1979). However, knowledge of the clean speech and noise distributions make a statistical approach possible (Ephraim 1992). The distributions of the noise and especially the speech are best given in the log-spectral or cepstral domain. Usually, a minimum mean square error (MMSE) estimate of the clean speech given an observation $\mathbf{y}_t$ is

found (Ephraim 1990):

$$\hat{\mathbf{x}} = \mathcal{E}\{\mathbf{x}|\mathbf{y}_t\}\,. \tag{4.69}$$

This requires $p(\mathbf{x}|\mathbf{y})$, and therefore $p(\mathbf{x})$, a model for the speech, which is simplified from a speech recogniser. Normally, a mixture of Gaussians is used for the joint distribution of the clean and corrupted speed:

$$\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} \sim \sum_r \pi^{(r)} \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_x^{(r)}\\ \boldsymbol{\mu}_y^{(r)}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_x^{(r)} & \boldsymbol{\Sigma}_{xy}^{(r)}\\ \boldsymbol{\Sigma}_{yx}^{(r)} & \boldsymbol{\Sigma}_y^{(r)}\end{bmatrix}\right). \tag{4.70}$$

The parameters of this joint distribution can be found with the methods described in section 4.4.3.1. Given the model in (4.70), the estimate in (4.69) is found by marginalising out the front-end component identity. It is a known result, which is derived in appendix A.1.3, that from a joint Gaussian the distribution of one variable conditioned on another is Gaussian with parameters

$$\mathbf{x}|\mathbf{y}, r \sim \mathcal{N}\left(\boldsymbol{\mu}_x^{(r)} + \boldsymbol{\Sigma}_{xy}^{(r)}\boldsymbol{\Sigma}_y^{(r)^{-1}}(\mathbf{y} - \boldsymbol{\mu}_y^{(r)}), \boldsymbol{\Sigma}_x^{(r)} - \boldsymbol{\Sigma}_{xy}^{(r)}\boldsymbol{\Sigma}_y^{(r)^{-1}}\boldsymbol{\Sigma}_{yx}^{(r)}\right) \tag{4.71}$$

The expected value of the clean speech for one component is the mean of the conditional distribution in (4.71):

$$\begin{aligned}\hat{\mathbf{x}} &= \sum_r P(r|\mathbf{y}_t)\,\mathcal{E}\{\mathbf{x}|\mathbf{y}_t, r\}\\ &= \sum_r P(r|\mathbf{y}_t)\left(\boldsymbol{\mu}_x^{(r)} + \boldsymbol{\Sigma}_{xy}^{(r)}\boldsymbol{\Sigma}_y^{(r)^{-1}}(\mathbf{y}_t - \boldsymbol{\mu}_y^{(r)})\right).\end{aligned} \tag{4.72}$$

The posterior responsibilities $P(r|\mathbf{y}_t)$ are found with the component-conditional marginal distribution of $\mathbf{y}$:

$$P(r|\mathbf{y}_t) \propto P(r)\,p(\mathbf{y}_t|r) = \pi^{(r)}\mathcal{N}\big(\mathbf{y}_t;\, \boldsymbol{\mu}_y^{(r)},\, \boldsymbol{\Sigma}_y^{(r)}\big). \tag{4.73}$$

This can be written as a affine transformation $\{\mathbf{A}_{t,\mathrm{mmse}}, \mathbf{b}_{t,\mathrm{mmse}}\}$ that depends on the observation vector. It is a linear interpolation between affine transformations $\left\{\mathbf{A}_{\mathrm{mmse}}^{(r)}, \mathbf{b}_{\mathrm{mmse}}^{(r)}\right\}$ that can be precomputed with

$$\mathbf{A}_{\mathrm{mmse}}^{(r)} = \boldsymbol{\Sigma}_{xy}^{(r)}\boldsymbol{\Sigma}_y^{(r)^{-1}}; \qquad \mathbf{b}_{\mathrm{mmse}}^{(r)} = \boldsymbol{\mu}_x^{(r)} - \mathbf{A}_{\mathrm{mmse}}^{(r)}\boldsymbol{\mu}_y^{(r)}. \tag{4.74}$$

The estimate of the clean speech then becomes

$$\hat{\mathbf{x}} = \mathbf{A}_{t,\text{mmse}}\mathbf{y}_t + \mathbf{b}_{t,\text{mmse}}, \tag{4.75}$$

where the interpolation weights are given by components' posterior probabilities $P(r|\mathbf{y}_t)$:

$$\mathbf{A}_{t,\text{mmse}} = \sum_r P(r|\mathbf{y}_t)\,\mathbf{A}^{(r)}_{\text{mmse}}; \qquad \mathbf{b}_{t,\text{mmse}} = \sum_r P(r|\mathbf{y}_t)\,\mathbf{b}^{(r)}_{\text{mmse}}. \tag{4.76}$$

When decoding with the clean speech estimate $\hat{\mathbf{x}}$ as the input vector for speech recogniser, the likelihood for component m is computed with

$$q^{(m)}(\mathbf{y}_t) = p^{(m)}(\hat{\mathbf{x}}) = p^{(m)}\big(\mathbf{A}_{t,\text{mmse}}\mathbf{y}_t + \mathbf{b}_{t,\text{mmse}}\big). \tag{4.77}$$

It is possible to write this as a transformation of the whole speech recogniser which is different for every observation vector. However, this transformation does not have a probabilistic interpretation.

4.6.1 Propagating uncertainty

A problem that has been recognised (Arrowood and Clements 2002; Stouten *et al.* 2004*a*) is that the clean speech estimate $\hat{\mathbf{x}}$ is a point estimate which does not carry any information about its uncertainty. A number of approaches have been suggested. It is possible to propagate the uncertainty of the posterior $p(\mathbf{x}|\mathbf{y}, r)$ of the clean speech reconstruction (Arrowood and Clements 2002; Stouten *et al.* 2004*a*). This uses the covariance of the Gaussian conditionals in (4.71), and effectively computes likelihoods as

$$q^{(m)}(\mathbf{y}_t) = \int p(\mathbf{x}|\mathbf{y}_t)\, p^{(m)}(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{4.78}$$

which is not mathematically consistent (Gales 2011).

An alternative is to propagate the conditional distribution $p(\mathbf{y}_t|\mathbf{x})$ (Droppo *et al.* 2002):

$$q^{(m)}(\mathbf{y}_t) = \int p(\mathbf{y}_t|\mathbf{x})\, p^{(m)}(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{4.79}$$

where

$$p(\mathbf{y}_t|\mathbf{x}) = \sum_r P(r|\mathbf{x})\, p(\mathbf{y}_t|\mathbf{x}, r)\,. \tag{4.80}$$

The problem with this is that the component posterior $P(r|\mathbf{x})$ depends on the clean speech, because it is conditioned on latent variable and must therefore be approximated.

4.6.2 *Algonquin*

So far, the joint mixture of Gaussians in (4.46) has been assumed fixed. If the joint was trained on stereo data, then this is sensible. However, if it was estimated with VTS, then the linearisation points may not be optimal. Section 4.5.1 has introduced Algonquin for model compensation, which iteratively updates the linearisation points towards the mode of the posterior given the observation. The original Algonquin algorithm (Frey *et al.* 2001*a*) applies feature enhancement, which finds the minimum mean square error estimate of the clean speech. Algonquin extends this idea by at the same time as updating the observation distributions, finding an approximation to the component posteriors of the mixture of Gaussians.

The algorithm replaces the static joint distribution in (4.46) by an approximation (Kristjansson 2002)

$$\mathbf{x}, \mathbf{n}, \mathbf{y} \sim \sum_r \pi^{(m)} \cdot p(\mathbf{x}, \mathbf{n}|r) \cdot q^{(r)(k)}(\mathbf{y}|\mathbf{x}, \mathbf{n})\,. \tag{4.81}$$

At each iteration k, the component-dependent observation distribution $q^{(r)(k)}$ is updated by re-estimating the expansion point as in section 4.5.1. The component distribution of the speech and the noise is Gaussian, and $q^{(r)(k)}$ is assumed Gaussian because of the mismatch function. Therefore, the posterior distribution $q^{(k)(m)}(\mathbf{x}, \mathbf{n}|\mathbf{y}_t)$ of the speech and the noise given the observation also becomes Gaussian. Its mean gives the expansion point for the next iteration.

After a number of iterations, the MMSE estimate for the clean speech is found analogously to (4.72)

$$\hat{\mathbf{x}} = \sum_r q(r) \cdot \mathcal{E}\{ q^{(r)(k)}(\mathbf{x}|\mathbf{y}_t) \}, \tag{4.82a}$$

where the expectation is the mean of the Gaussian posterior. The posterior responsibilities $q(r)$ are found with, analogously to (4.73),

$$q(r) \propto \pi^{(m)} q^{(r)(k)}(\mathbf{y}_t). \tag{4.82b}$$

4.7 Noise model estimation

The discussion so far has assumed that a distribution of the noise is known. In practice, however, this is seldom the case. The noise model must therefore be estimated. The usual approach is to apply expectation–maximisation in an unsupervised-adaptation framework discussed in section 3.1. The aim then is to optimise the noise model parameters to improve the likelihood using the form of model compensation that is used for decoding.

Conceptually, this moves the interface of the model with the real world. It does not matter any more whether the noise model matches the actual noise. What matters is that the parameters can be estimated robustly and that they allow the resulting model to match the real world reasonably well. Discrepancies between the model and the real process therefore become allowable: the noise model estimate can absorb some of the mismatch.

The noise model $\mathcal{M}_n = \{\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n, \boldsymbol{\mu}_h\}$ comprises the parameters of the additive noise, assumed Gaussian with $\mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$, and the convolutional noise $\boldsymbol{\mu}_h$, which is assumed constant. The parameters are of the form

$$\boldsymbol{\mu}_n = \begin{bmatrix} \boldsymbol{\mu}_n^s \\ 0 \end{bmatrix}; \qquad \boldsymbol{\Sigma}_n = \begin{bmatrix} \operatorname{diag}(\boldsymbol{\Sigma}_n^s) & 0 \\ 0 & \operatorname{diag}(\boldsymbol{\Sigma}_n^\Delta) \end{bmatrix}; \qquad \boldsymbol{\mu}_h = \begin{bmatrix} \boldsymbol{\mu}_h^s \\ 0 \end{bmatrix}. \tag{4.83}$$

The expected value of the dynamic coefficients of the additive noise are zero because this work assumes that the noise model has no state changes. Since the convolutional

noise is assumed constant, its dynamic parameters are also zero. The noise covariance is normally assumed diagonal, just like for clean speech Gaussians. With standard feature vectors, with 13 statics and 13 delta and 13 delta-delta coefficients, the noise model has only 65 parameters, as opposed to CMLLR's 1560 per class. This means that methods for noise-robustness can adapt to a few seconds of data, in situations where applying CMLLR decreases performance (for a comparison, see Flego and Gales 2009).

Though noise model estimation can conceptually be seen as adaptation, the generic derivation is different from the derivation of adaptation in section 3.1. The noise is explicitly a hidden variable distributed according to some distribution, whose parameters must be trained. Training uses expectation–maximisation (see section 2.3.2.1) (Rose *et al.* 1994). Here, the hidden variables $\mathcal{U}$ consist not only of the component sequence m_t, but also the sequence of the noise vectors $\mathbf{n}_t$ and $\mathbf{h}_t$. The expressions for the expectation and maximisation steps are (after (2.22) and (2.27b))

$$\rho^{(k)} := q^{(k)}_{\mathcal{U}|\mathcal{Y}} \propto q^{(k)}_{\mathcal{U}\mathcal{Y}}; \tag{4.84a}$$

$$q^{(k)}_{\mathcal{U}} := \arg\max_{q_{\mathcal{U}}} \int \tilde{p}(\mathcal{Y}) \int \rho^{(k)}(\mathcal{U}|\mathcal{Y}) \log q_{\mathcal{U}}(\mathcal{U}) \, d\mathcal{U} \, d\mathcal{Y}. \tag{4.84b}$$

For noise estimation, the posterior distribution over the hidden variables is a joint distribution over the components and the noise vectors, which can be expressed as

$$\begin{aligned} \rho(\mathcal{U}|\mathcal{Y}) &= \rho\big(\{m_t, \mathbf{n}_t, \mathbf{h}_t\}\big|\mathcal{Y}\big) \\ &= \rho\big(\{m_t\}\big|\mathcal{Y}\big)\rho\big(\{\mathbf{n}_t, \mathbf{h}_t\}\big|\{m_t\}, \mathcal{Y}\big). \end{aligned} \tag{4.85}$$

The distribution over the components, $\rho\big(\{m_t\}\big|\mathcal{Y}\big)$, is the same as for speech recogniser training, and again it is sufficient to keep component–time occupancies $\gamma^{(m)}_t$. The distribution over the noise consists of a distribution for each component. The relationship between the noise and the noise-corrupted speech is non-linear. This relationship is captured by $q_{\mathcal{Y}|\mathcal{U}}$, which the computation of the hidden variable posterior in (4.84a) uses (see section 2.3.2.1). The posterior distribution of the noise for one component m_t is $\rho\big(\{\mathbf{n}_t, \mathbf{h}_t\}\big|\{m_t\}, \mathcal{Y}\big)$ therefore does not have a closed form.

Noise estimation is of practical value. It is therefore not surprising that most methods that have been proposed work on a practical compensation method, which lin-

earises of the mismatch function. This is mostly either VTS compensation or joint uncertainty decoding. Methods use either of two ways of approximating the hidden variable posterior. Some fix the linearisation of the influence of the noise on the observation to make computing the parameters possible. Others approximate the real posterior given the mismatch function with numerical or Monte Carlo methods, but apply the linearisation when compensating anyway. Neither is guaranteed to yield an increase of the likelihood.

4.7.1 Linearisation

Both VTS compensation and joint uncertainty decoding linearise the influence of the noise (and the other sources) on the corrupted speech. It is possible to find a new noise model estimate that is guaranteed not to decrease the likelihood as long as the VTS expansion point of the noise is fixed. However, the linearisation depends on the noise model: the expansion point of the additive noise is normally set to the noise mean (see section 4.4.2). As soon as the expansion point changes, therefore, the guarantee drops away. This can lead to oscillations. However, it is always possible to evaluate the result of the likelihood function before accepting a new noise model estimate. Then, a back-off strategy can decrease the step size until the likelihood does increase. Alternatively, the noise model mean and the noise expansion point can be disconnected, so that the guarantee about the likelihood remains valid, even though the VTS approximation becomes less close to the real distribution. Schemes that combine these strategies adaptively are possible. The actual estimation of the new parameters can work in two ways: either by setting up a joint Gaussian distribution of noise and corrupted speech, yielding a factor analysis-style solution, or by iteratively optimising the likelihood function directly.

Since the additive noise is assumed Gaussian, the linearisation makes the additive noise and the noise-corrupted speech jointly Gaussian for each speech component (independently introduced by Kim *et al.* 1998; Frey *et al.* 2001*b*). This results in a

distribution of the form

$$\begin{bmatrix} \mathbf{n} \\ \mathbf{y} \end{bmatrix} \middle| \mathfrak{m} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_n \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_n & \boldsymbol{\Sigma}_{ny} \\ \boldsymbol{\Sigma}_{yn} & \boldsymbol{\Sigma}_y \end{bmatrix} \right). \tag{4.86}$$

This has only been used for statics, but dynamic compensation with the continuous-time approximation yields a linearised relationship of noise and observation too, so the principle can be extended to vectors with statics and dynamics.

From this joint distribution, the distribution of the noise conditional on an observation $\mathbf{y}$ is also Gaussian (see appendix A.1.3). This yields a factor analysis-style solution for the optimal noise distribution for each time instance and component.

However, finding the convolutional noise parameters is not possible in this framework if the convolutional noise is assumed constant, because the posterior distribution of it cannot move from its prior estimate. It is possible to assume a convolutional noise covariance while estimating the parameters and then ignore it when compensating, but that again yields no guarantee of finding the optimal likelihood.

A greater problem, however, is that some blocks of the joint distribution in (4.86) are usually diagonalised. The noise estimate is usually constrained to have a diagonal covariance, and so is the resulting corrupted speech covariance. Diagonalising the corrupted speech covariance, however, happens after compensation. Since it is not an intrinsic property of the process, it cannot be reversed. To infer a distribution over $\mathbf{n}$ from a given value of $\mathbf{y}$, the joint distribution needs to be valid. In the case of the joint Gaussian, the cross-covariance $\boldsymbol{\Sigma}_{yn}$ must be diagonal as well. Diagonalising $\boldsymbol{\Sigma}_{yn}$ is equivalent to diagonalising $\mathbf{J}_n$. Thus, this assumes that the relationship between noise and corrupted speech is per coefficient. However, as section 4.4.2 has pointed out, the Jacobian that gives that relationship is not diagonal. If it is constrained to be diagonal in estimation, this, again, must be applied in compensation as well for the estimation process to yield a guaranteed improvement in likelihood. However, diagonalising the Jacobian is an additional approximation that reduces the quality of the compensation.

An alternative is to directly optimise the noise model parameters. The static noise means can be updated at the same time using a fixed-point iteration (Moreno 1996).

The additive noise covariance, however, is more complex to estimate. It is possible to estimate it on the parts of the waveform known to contain noise without speech. Another options is to use gradient ascent to find an estimate for the additive noise variance for VTS with the continuous-time approximation (Liao and Gales 2006). This needs to be alternated with the estimation of the noise mean. This scheme does allow the convolutional noise to be estimated, and it does not require the Jacobians to be diagonalised. This is the approach that this work will take. The resulting noise model estimate maximises the likelihood of model compensation with VTS. Thus, the parameters do not necessarily correspond to the actual noise or to a consistent sequence of static observations.

4.7.2 Numerical approximation

It is possible to find a numerical approximation to the hidden variable posterior. Myrvoll and Nakamura (2003) propose a method that uses a consistent approximation for estimation and compensation (see section 4.5.2), which does not apply the vector Taylor series approximation. However, it assumes all dimensions independent, and it is not feasible to generalise it to multiple dimensions (see appendix E).

Alternatively, it is possible to approximate the noise posterior with an empirical distribution (Faubel and Klakow 2010). The empirical noise posterior is acquired with importance sampling (see appendix A.4.2). Samples $\mathbf{n}_t^{(l)}$ are drawn from the noise prior for the previous iteration. They are then re-weighted to give an approximation to the noise posterior $p(\mathbf{n}_t|\mathcal{Y})$.[4] Since the noise is assumed identically distributed, the number of samples required per time frame is low. Faubel and Klakow (2010) apply this per dimension, but it may be possible to extended the method to more dimensions. However, note that this method improves the likelihood under the assumption that the exact distribution (given the mismatch function) is used. This has the advantage over using the VTS assumption that no overshoot happens because of the linear-

[4] Faubel and Klakow (2010) uses Parzen windows to approximate the importance weights, but this can be replaced by a straightforward analytical solution. The required expression is (4.66) with $\mathbf{x}$ and $\mathbf{n}$ exchanged.

isation. However, in practice, a different form of compensation will be applied, so that the method does not optimise the likelihood for the actual form of compensation.

4.8 Summary

This chapter has discussed existing approaches for noise-robust speech recognition. Section 4.2.1 has derived the mismatch function, which relates the speech, noise, and the corrupted speech. This resulted (in section 4.3) in an expression for the corrupted speech which has no closed form. Model compensation, the topic of section 4.4, approximates the corrupted speech distribution with a parameterised density. The state-of-the-art VTS compensation finds one corrupted speech Gaussian for one clean speech Gaussian. The Gaussian is diagonalised, because of the imprecise estimation of the off-diagonals and decoding speed. These two issues will be dealt with in chapters 5 and 6, respectively. The methods in section 4.5 use a different approach: they find a different approximation to the corrupted speech distribution for every observation. This philosophy will also apply to the scheme that chapter 7 will introduce. Section 4.6 has described model-based feature enhancement, which only uses a linear transformation to the feature vector. Section 4.7 then cast methods for noise-robustness in an adaptation framework, by estimating the noise model. The interesting aspect is that a standard noise model has only 65 parameters, as opposed to at least 1560 for CMLLR. This means that methods for noise-robustness need far less adaptation data than general adaptation methods.

Part II

Contributions

Chapter 5

Compensating correlations

This chapter will describe the first contribution of this thesis.[1]

The previous chapter has described model compensation methods. They diagonalise the Gaussians they produce, because the off-diagonals are more sensitive to approximations. For the state-of-the-art VTS compensation, the *continuous-time approximation* it applies for the dynamic coefficients will turn out problematic.

This chapter will propose a new approach for compensating the dynamic parameters so that the correlations can be estimated accurately, which, section 5.1 will show, is important. Section 5.2 will then explain how the dynamic coefficients can be expressed as a linear transformation over a vector with all static feature coefficients in a window. When a distribution over this "extended" feature vector is known, then the distribution of the static and dynamic parameters can often be found by linearly transforming the parameters of the distribution over the extended feature vector. Section 5.2.2 will explore under which conditions this is valid. In the same fashion as standard model compensation, there is a range of schemes, that section 5.3 will introduce, that can be used to combine the extended clean speech and noise distributions

[1] Extended DPMC, and its application to joint uncertainty decoding, was introduced in van Dalen and Gales (2008). Extended VTS was introduced in van Dalen and Gales (2009*b*;*a*; 2011). Van Dalen and Gales (2010*a*;*b*) mentioned briefly the form in which this chapter will present them, with a distribution for the phase factor. Extended IDPMC also got only a brief mention. This thesis newly introduces the extended Algonquin algorithm.

to yield the extended corrupted speech distribution. In particular, section 5.3.3 will introduce "extended VTS", a faster method that approximates the mismatch function for each time instance with a vector Taylor series. Section 5.4 will discuss how to estimate extended parameters for joint uncertainty decoding, which compensates a base class at a time. Section 5.5 will discuss how to find robust estimates for speech and noise distributions over extended feature vectors, which have more parameters than normal ones. Estimates for the extended noise distribution can be found from estimates for statics and dynamics. Alternatively, because the off-diagonals can now be estimated, an expectation–maximisation approach is possible.

5.1 Correlations under noise

Feature correlations change under noise. Figure 5.1 on the next page shows the overall correlations of the zeroth and first cepstral coefficients in Toshiba in-car data from different noise conditions (for details, see section 8.1.3). The differences in the orientation of the ellipses indicate differences in correlations. How this comes about can be seen by considering feature correlations of only speech, and of only noise. If the feature space is optimised to reduce correlations for clean speech, which MFCCs make an attempt to do, correlations will appear under noise. However, in the limit as the noise masks the speech, the correlation pattern will be that of the noise. It is therefore important for noise-robustness that these correlations are modelled.

Though correlations could be modelled with full covariance matrices, speech recogniser distributions are normally assumed Gaussian with diagonal covariance matrices. This is true for the clean speech distributions and so ingrained that methods for noise-robustness are proposed without even mentioning that diagonalisation is performed (Kim *et al.* 1998; Acero *et al.* 2000). When estimating corrupted-speech distributions on stereo data, full covariance matrices have been shown to increased performance (Liao and Gales 2005). However, stereo data is seldom available.

To compensate for correlation changes under noise in realistic scenarios, a noise

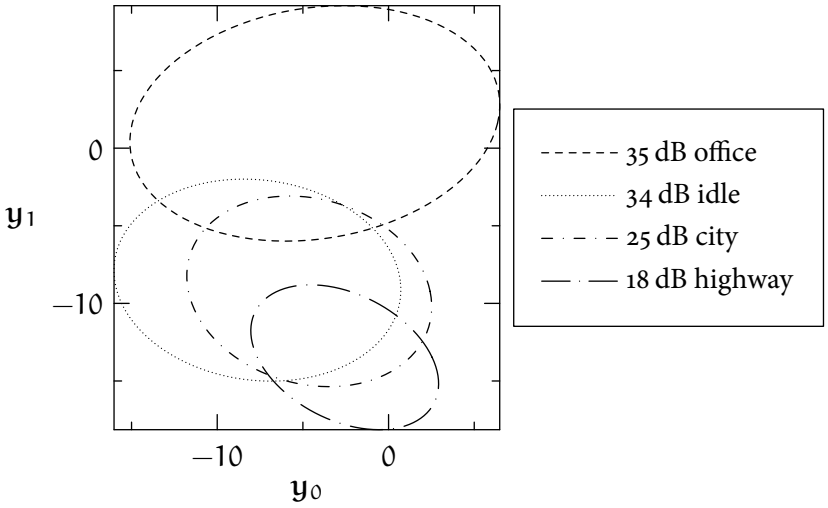


Figure 5.1 *Overall correlations of the zeroth and first cepstral coefficients in Toshiba in-car data (see section 8.1.3) for different noise conditions.*

model must be estimated, and full-covariance compensation computed with a model compensation method like VTS, discussed in section 4.4.2. As mentioned in that section, VTS compensation is normally diagonalised. This is for two reasons: decoding speed, and compensation quality. Chapter 6 will introduce forms of compensation that model correlations but are fast to decode with. This chapter will look into the quality of compensation for correlations. The estimates for full-covariance Gaussians are expected to be more sensitive to approximations in the compensation process than for diagonal covariance matrices. In particular, and section 8.1.1.1 will show this by comparing against a single-pass retrained recogniser, the continuous-time approximation, which standard VTS compensation uses for dynamic coefficients and was introduced in section 4.2.3, does not provide good compensation.

In HMM-based speech recognition systems, dynamic features (usually delta and delta-delta coefficients) are appended to the static features to form the feature vectors (see section 2.1.2). The continuous-time approximation makes the assumption that the dynamic coefficients are the time derivatives of the statics. For VTS, the form of compensation for the dynamic parameters is then closely related to the static parameters. Though compensation with the continuous-time approximation can generate

block-diagonal covariance matrices (as section 4.4.2 has shown), the estimates are not accurate enough to yield an increase in performance.

An advantage of the continuous-time approximation is that it is possible to find compensation for any form of dynamic parameters, both those computed with linear regression and with simple differences. A scheme for dynamic parameter compensation with DPMC stores extra clean speech statistics (see section 4.4.1), but is only applicable to simple differences. As section 2.1.2 has mentioned, state-of-the-art speech recognisers compute dynamic parameters with linear regression. Another scheme that attempts to improve compensation by using additional statistics, but in the log-spectral domain, is described in de la Torre *et al.* (2002). However, as section 5.3.3.1 will show, this approach involves approximations that negate any potential improvements and basically yields the same form as the continuous-time approximation. Though there are known limitations to the use of the continuous time approximation, it is still the form used in the vast majority of model compensation schemes (Acero *et al.* 2000; Liao and Gales 2007; Li *et al.* 2007).

5.2 Compensating dynamics with extended feature vectors

This section will describe an alternative method to using the continuous-time approximation for compensating the dynamic model parameters. The key intuition is that the dynamic coefficients are a linear combination of consecutive static feature vectors. Thus, it is possible to model the effect of the noise separately per time instance, and only then combine the time instances. This applies the same linear transformation that speech recognisers apply to find dynamic coefficients from a range of statics. At which point in the process it is valid to apply the linear transformation depends on the details of the compensation methods. This will become clear in this section.

For simplicity, a window of ± 1 and only first-order dynamic coefficients will be considered. An *extended* feature vector $\mathbf{y}_t^e$, containing the static feature vectors in the

surrounding window, is given by $\mathbf{y}_t^e = \left[\mathbf{y}_{t-1}^{s\,T}\ \mathbf{y}_t^{s\,T}\ \mathbf{y}_{t+1}^{s\,T}\right]^T$.[2] The transformation of the extended feature vector $\mathbf{y}_t^e$ to the standard feature vector with static and dynamic parameters $\mathbf{y}_t$ can be expressed as the linear projection $\mathbf{D}$ that was introduced in section 2.1.2 (analogous to (2.7b)):

$$\mathbf{y}_t = \begin{bmatrix} \mathbf{y}_t^s \\ \mathbf{y}_t^\Delta \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} \\ -\frac{\mathbf{I}}{2} & \mathbf{0} & \frac{\mathbf{I}}{2} \end{bmatrix} \begin{bmatrix} \mathbf{y}_{t-1}^s \\ \mathbf{y}_t^s \\ \mathbf{y}_{t+1}^s \end{bmatrix} = \mathbf{D}\mathbf{y}_t^e, \tag{5.1a}$$

The second row of $\mathbf{D}$ applies the transformation from a window of statics to yield the standard delta features. Similarly,

$$\mathbf{x}_t = \mathbf{D}\mathbf{x}_t^e; \quad \mathbf{n}_t = \mathbf{D}\mathbf{n}_t^e; \quad \mathbf{h}_t = \mathbf{D}\mathbf{h}_t^e; \quad \boldsymbol{\alpha}_t = \mathbf{D}\boldsymbol{\alpha}_t^e, \tag{5.1b}$$

where extended feature vectors $\cdot_t^e$ all contain consecutive static feature vectors $\cdot_{t-1}^s, \cdot_t^s, \cdot_{t+1}^s$. The form of their distributions will be discussed in section 5.5.

Model compensation, described in section 4.4, approximates the predicted distribution of the noise-corrupted speech for one component (from (4.24d); the dependency on the component will not be written in this chapter)

$$p(\mathbf{y}) = \int\int\int\int \delta_{\mathbf{f}(\mathbf{x},\mathbf{n},\mathbf{h},\boldsymbol{\alpha})}(\mathbf{y})\, p(\mathbf{x},\mathbf{n},\mathbf{h},\boldsymbol{\alpha}) \mathrm{d}\boldsymbol{\alpha}\mathrm{d}\mathbf{h}\mathrm{d}\mathbf{n}\mathrm{d}\mathbf{x}. \tag{5.2}$$

To model the effect of the noise on each time instance separately, an extended mismatch function $\mathbf{f}^e$ can be defined as

$$\mathbf{f}^e(\mathbf{x}_t^e, \mathbf{n}_t^e, \mathbf{h}_t^e, \boldsymbol{\alpha}_t^e) = \begin{bmatrix} \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{n}_{t-1}, \mathbf{h}_{t-1}, \boldsymbol{\alpha}_{t-1}) \\ \mathbf{f}(\mathbf{x}_t, \mathbf{n}_t, \mathbf{h}_t, \boldsymbol{\alpha}_t) \\ \mathbf{f}(\mathbf{x}_{t+1}, \mathbf{n}_{t+1}, \mathbf{h}_{t+1}, \boldsymbol{\alpha}_{t+1}) \end{bmatrix}, \tag{5.3a}$$

where $\mathbf{f}(\cdot)$ is the per-time instance mismatch function defined in (4.20). Section 4.2.3 has given the mismatch function for dynamics, which used a projection matrix $\mathbf{D}^\Delta$.

[2] It is straightforward to extend this to handle both second-order dynamics and linear-regression coefficients over a larger window of $\pm w$, so that $\mathbf{y}_t^e = \left[\mathbf{y}_{t-w}^{s\,T} \ldots \mathbf{y}_{t+w}^{s\,T}\right]^T$.

The full noise-corrupted speech vector with statics and dynamics can be found with projection matrix $\mathbf{D}$:

$$\mathbf{y}_t = \mathbf{D}\mathbf{f}^e(\mathbf{x}^e_t, \mathbf{n}^e_t, \mathbf{h}^e_t, \boldsymbol{\alpha}^e_t). \tag{5.3b}$$

To use this to express the distribution over the corrupted speech, the integral in (5.2) is rewritten in terms of extended feature vectors:

$$p(\mathbf{y}) = \int\int\int\int \delta_{\mathbf{D}\mathbf{f}^e(\mathbf{x}^e,\mathbf{n}^e,\mathbf{h}^e,\boldsymbol{\alpha}^e)}(\mathbf{y})\, p(\mathbf{x}^e, \mathbf{n}^e, \mathbf{h}^e, \boldsymbol{\alpha}^e) d\boldsymbol{\alpha}^e d\mathbf{h}^e d\mathbf{n}^e d\mathbf{x}^e. \tag{5.3c}$$

It is possible to approximate the quadruple integral in (5.3c) by sampling. The Dirac delta yields corresponding samples with statics and dynamics, in a similar way to standard DPMC in section 4.4.1. Extended DPMC and extended iterative DPMC, which sections 5.3.1 and 5.3.2 will introduce, train on these samples.

However, DPMC (not iterative DPMC) can also be viewed from a different perspective, which corresponds to the original presentation (in van Dalen and Gales 2008). It will turn out possible to estimate a Gaussian over extended feature vectors, and only then convert to a Gaussian over standard feature vectors with statics and dynamics. This will allow section 5.3.3 to introduce extended VTS, a method that applies a vector Taylor series approximation for every time instance of extended feature vectors. Extended VTS therefore has a reasonable time complexity. The following will detail how a Gaussian over extended feature vectors can be converted into one over statics and dynamics. Section 5.2.2 will prove under what condition optimising a distribution over extended feature vectors is equivalent to optimising one over statics and dynamics.

5.2.1 The extended Gaussian

It is interesting to look at the structure of a Gaussian distribution over extended feature vectors, $\mathbf{y}^e \sim \mathcal{N}(\boldsymbol{\mu}^e_y, \boldsymbol{\Sigma}^e_y)$. The mean $\boldsymbol{\mu}^e_y$ of the concatenation of consecutive static feature vectors is simply a concatenation of static means at time offsets $-1, 0, +1$. For the corrupted speech, these will be written $\boldsymbol{\mu}^s_{y_{-1}}, \boldsymbol{\mu}^s_{y_0}, \boldsymbol{\mu}^s_{y_{+1}}$. The covariance $\boldsymbol{\Sigma}^e_y$ contains the covariance between statics at different time offsets. The covariance between

offsets -1 and $+1$, for example, is written $\boldsymbol{\Sigma}_{y_{-1}y_{+1}}$. Thus, the full parameters of the extended distribution are

$$\boldsymbol{\mu}_y^e = \begin{bmatrix} \boldsymbol{\mu}_{y_{-1}}^s \\ \boldsymbol{\mu}_{y_0}^s \\ \boldsymbol{\mu}_{y_{+1}}^s \end{bmatrix}; \qquad \boldsymbol{\Sigma}_y^e = \begin{bmatrix} \boldsymbol{\Sigma}_{y_{-1}y_{-1}}^s & \boldsymbol{\Sigma}_{y_{-1}y_0}^s & \boldsymbol{\Sigma}_{y_{-1}y_{+1}}^s \\ \boldsymbol{\Sigma}_{y_0y_{-1}}^s & \boldsymbol{\Sigma}_{y_0y_0}^s & \boldsymbol{\Sigma}_{y_0y_{+1}}^s \\ \boldsymbol{\Sigma}_{y_{+1}y_{-1}}^s & \boldsymbol{\Sigma}_{y_{+1}y_0}^s & \boldsymbol{\Sigma}_{y_{+1}y_{+1}}^s \end{bmatrix}. \tag{5.4}$$

An extended vector and its equivalent with statics and dynamics are related with $\mathbf{y} = \mathbf{D}\mathbf{y}^e$. As $\mathbf{D}$ is a linear transformation, if the distribution of the extended corrupted speech $\mathbf{y}_t^e$ is assumed Gaussian, then the extended corrupted speech distribution can be transformed to a distribution over statics and dynamics with

$$\mathbf{y} = \mathbf{D}\mathbf{y}^e \sim \mathcal{N}\left(\mathbf{D}\boldsymbol{\mu}_y^e, \mathbf{D}\boldsymbol{\Sigma}_y^e\mathbf{D}^\top\right) \tag{5.5}$$

The internal structure of the matrices will have ramifications for how to store statistics and how to compensate distributions. For example, the covariance in (5.5), substituting $\boldsymbol{\Sigma}_y^e$ from (5.4) and $\mathbf{D}$ from (5.1a), can be expressed as

$$\boldsymbol{\Sigma}_y = \mathbf{D}\boldsymbol{\Sigma}_y^e\mathbf{D}^\top = \begin{bmatrix} 0 & I & 0 \\ -\frac{I}{2} & 0 & \frac{I}{2} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_{y_{-1}y_{-1}}^s & \boldsymbol{\Sigma}_{y_{-1}y_0}^s & \boldsymbol{\Sigma}_{y_{-1}y_{+1}}^s \\ \boldsymbol{\Sigma}_{y_0y_{-1}}^s & \boldsymbol{\Sigma}_{y_0y_0}^s & \boldsymbol{\Sigma}_{y_0y_{+1}}^s \\ \boldsymbol{\Sigma}_{y_{+1}y_{-1}}^s & \boldsymbol{\Sigma}_{y_{+1}y_0}^s & \boldsymbol{\Sigma}_{y_{+1}y_{+1}}^s \end{bmatrix} \begin{bmatrix} 0 & -\frac{I}{2} \\ I & 0 \\ 0 & \frac{I}{2} \end{bmatrix}. \tag{5.6}$$

5.2.2 *Validity of optimising an extended distribution*

It would be interesting to find under what conditions optimising the distribution over extended feature vectors of the noise-corrupted speech, for example, the Gaussian in section 5.2.1, optimises the distribution over statics and dynamics. The standard feature vector $\mathbf{y}$ with statics and dynamics is related to the extended feature vector $\mathbf{y}^e$ by (repeated from (5.1a))

$$\mathbf{y} = \mathbf{D}\mathbf{y}^e. \tag{5.7}$$

The distribution q^e over extended feature vectors would therefore approximate a distribution similar to the one in (5.3c). There, the integrals were over extended feature

vectors, and the resulting distribution over standard feature vectors. Alternatively, as in section 5.2.1, the resulting distribution is over extended feature vectors as well:

$$p^e(\mathbf{y}^e) = \int\int\int\int \delta_{\mathbf{f}^e(\mathbf{x}^e,\mathbf{n}^e,\mathbf{h}^e,\boldsymbol{\alpha}^e)}(\mathbf{y}^e)\, p(\mathbf{x}^e, \mathbf{n}^e, \mathbf{h}^e, \boldsymbol{\alpha}^e) d\boldsymbol{\alpha}^e d\mathbf{h}^e d\mathbf{n}^e d\mathbf{x}^e. \tag{5.8}$$

The question that the following will answer is under what conditions approximating p^e (in (5.8)) with q^e is equivalent to optimising the approximation q to p (in (5.3c)) directly:

$$\hat{q} := \arg\min_q \mathcal{KL}(p\|q)\,; \tag{5.9a}$$

$$\hat{q}^e := \arg\min_{q^e} \mathcal{KL}(p^e\|q^e)\,. \tag{5.9b}$$

To relate distributions over $\mathbf{y}$ and $\mathbf{y}^e$, the determinant of the Jacobian of the conversion is necessary. Since the relation is linear, this would be the determinant of $\mathbf{D}$ if it were square. As a trick, extra dimensions can be appended onto $\mathbf{y}$ to increase its dimensionality. These irrelevant dimensions are similar to the "nuisance" dimensions for HLDA (discussed in section 3.3.2). They will be written $\mathbf{y}_{()}$, and the vector with these appended $\mathbf{y}_+$, so that

$$\mathbf{y}_+ = \begin{bmatrix} \mathbf{y} \\ \mathbf{y}_{()} \end{bmatrix}. \tag{5.10}$$

Similarly, the projection $\mathbf{D}$ from extended feature vectors to ones with statics and dynamics is extended:

$$\mathbf{D}_+ = \begin{bmatrix} \mathbf{D} \\ \mathbf{D}_{()} \end{bmatrix}. \tag{5.11}$$

Provided $\mathbf{D}_+$ is full-rank, it is irrelevant what entries $\mathbf{D}_{()}$ has exactly, because it is merely a mathematical construct. Then,

$$\mathbf{y}_+ = \mathbf{D}_+\mathbf{y}^e. \tag{5.12}$$

The distributions p and its approximation q are similarly extended:

$$p_+(\mathbf{y}_+) = p(\mathbf{y}) \cdot p_{()}\big(\mathbf{y}_{()}\big|\mathbf{y}\big)\,; \qquad q_+(\mathbf{y}_+) = q(\mathbf{y}) \cdot q_{()}\big(\mathbf{y}_{()}\big|\mathbf{y}\big) \tag{5.13}$$

Again, it is irrelevant how the distribution over the nuisance dimensions $p_{()}$ is defined or what its approximation $q_{()}$ is optimised to. In any case, the distributions over extended feature vectors and over standard feature vectors plus nuisance dimensions are related by the determinant of the Jacobian (a well-known equality, in (A.1)):

$$p^e(\mathbf{y}^e) = |\mathbf{D}_+| \cdot p_+(\mathbf{y}_+); \qquad q^e(\mathbf{y}^e) = |\mathbf{D}_+| \cdot q_+(\mathbf{y}_+). \tag{5.14}$$

The question now is whether when q^e is optimised, q is optimised in the process. This can be taken in two parts. The first question is whether optimising q^e is equivalent to optimising q_+. This is the case since $|\mathbf{D}|$ is constant, so that it drops out of the optimisation, when it is rewritten with (5.14):

$$\begin{aligned}
\hat{q}^e &= \arg\max_{q^e} \int p^e(\mathbf{y}^e) \log(q^e(\mathbf{y}^e)) \, d\mathbf{y}^e \\
&= \arg\max_{q^e} \int \left|\mathbf{D}_+^{-1}\right| |\mathbf{D}_+| \, p_+(\mathbf{y}_+) \log(|\mathbf{D}_+| \, q_+(\mathbf{y}_+)) \, d\mathbf{y}_+ \\
&= \arg\max_{q^e} \int p_+(\mathbf{y}_+) \log(q_+(\mathbf{y}_+)) \, d\mathbf{y}_+ .
\end{aligned} \tag{5.15a}$$

The second part of the question is when optimising q_+ also optimises q. To express this, substitute (5.14) into (5.15a):

$$\begin{aligned}
\hat{q}^e &= \arg\max_{q^e} \int p_+(\mathbf{y}_+) \log(q_+(\mathbf{y}_+)) \, d\mathbf{y}_+ \\
&= \arg\max_{q^e} \int p(\mathbf{y}) \cdot \int p_{()}(\mathbf{y}_{()}|\mathbf{y}) \log\big(q(\mathbf{y}) \cdot q_{()}(\mathbf{y}_{()}|\mathbf{y})\big) \, d\mathbf{y}_{()} \, d\mathbf{y} \\
&= \arg\max_{q^e} \Bigg[\int p(\mathbf{y}) \log(q(\mathbf{y})) \, d\mathbf{y} \\
&\qquad + \int p(\mathbf{y}) \int p_{()}(\mathbf{y}_{()}|\mathbf{y}) \log\big(q_{()}(\mathbf{y}_{()}|\mathbf{y})\big) \, d\mathbf{y}_{()} \, d\mathbf{y} \Bigg].
\end{aligned} \tag{5.15b}$$

Therefore, if the parameters of q and $q_{()}$ can be set independently, then finding the optimal q^e means finding the optimal q.

For a Gaussian q^e, the parameters of q and $q_{()}$ can indeed be set independently. Appendix A.1.3 details the well-known composition of a multi-variate Gaussian into the marginal of some dimensions (here, $\mathbf{y}$) and a conditional of the other variables (here, $\mathbf{y}_{()}$) given the first set. The projection $\mathbf{D}_+$ changes the parameter space, so that

this factorisation is not explicit when optimising q^e. However, since the projection is full-rank, the optimisation in one feature space gives the optimal parameters in another feature space, and the argument still applies. This therefore proves that it is possible to optimise a Gaussian over the extended corrupted speech distribution and convert it to a distribution over statics and dynamics.

5.3 Compensating extended distributions

The following will introduce four ways of estimating an extended distribution for $\mathbf{y}^e$. The first, extended DPMC, uses sampling. Its variant, extended iterative DPMC, finds a mixture of Gaussians. A faster scheme, extended VTS, applies a vector Taylor series approximation to every time instance. Extended Algonquin uses the same approximation, but iteratively updates the expansion point.

They all assume that the extended speech $\mathbf{x}^e$ and noise $\mathbf{n}^e$ are Gaussian, and that the convolutional noise $\mathbf{h}^e$ is constant. The elements of $\boldsymbol{\alpha}$ are assumed Gaussian distributed but constrained to $[-1, +1]$. Section 5.5 will discuss the form of parameters for their distributions.

It would be possible to apply the extended feature vector approach to other approaches that find Gaussian compensation. Indeed, an appendix of van Dalen and Gales (2009*a*) uses the unscented transformation to find a Gaussian extended corrupted speech distribution. However, as pointed out at the end of section 4.4.2, the difference between model compensation schemes that come up with the same form of distribution largely disappears when the noise model is estimated. The models then differ only in the exact parameterisation that the optimisation uses. The only schemes that this thesis will introduce that find one fixed Gaussian for the corrupted speech are therefore extended DPMC, a sampling scheme that in the limit produces the optimal Gaussian, and extended VTS, based on the state-of-the-art VTS.

5.3.1 *Extended* DPMC

The first method of finding a Gaussian for the extended corrupted speech distribution is extended DPMC (EDPMC). It derives from the integral of the exact expression for the extended corrupted speech distribution analogously to standard DPMC (section 4.4.1). However, there is also a second perspective on DPMC, which converts samples to standard feature vectors immediately and trains a distribution on those. The first perspective ties in with extended VTS, which section 5.3.3 will introduce. The second perspective makes it possible to extend the algorithm to mixtures of Gaussians, for extended iterative DPMC (section 5.3.2).

The first perspective on DPMC finds an extended corrupted speech Gaussian. It derives from the integral over extended feature vectors in (5.8):

$$p^e(\mathbf{y}^e) = \int\int\int\int \delta_{\mathbf{f}^e(\mathbf{x}^e,\mathbf{n}^e,\mathbf{h}^e,\boldsymbol{\alpha}^e)}(\mathbf{y})\, p(\mathbf{x}^e, \mathbf{n}^e, \mathbf{h}^e, \boldsymbol{\alpha}^e) \mathrm{d}\boldsymbol{\alpha}^e \mathrm{d}\mathbf{h}^e \mathrm{d}\mathbf{n}^e \mathrm{d}\mathbf{x}^e. \tag{5.16}$$

Extended DPMC approximates this by representing this distribution p^e by an empirical distribution $\tilde{p}^e$. To find the empirical distribution, sample tuples $(\mathbf{x}^{e(l)}, \mathbf{n}^{e(l)}, \mathbf{h}^{e(l)}, \boldsymbol{\alpha}^{e(l)})$ can be drawn from the extended distributions of the clean speech, noise, and phase factor. These are joint samples over consecutive feature vectors, so that (again, assuming a window ± 1)

$$\mathbf{x}_t^{e(l)} = \begin{bmatrix} \mathbf{x}_{t-1}^{s(l)} \\ \mathbf{x}_t^{s(l)} \\ \mathbf{x}_{t+1}^{s(l)} \end{bmatrix}; \quad \mathbf{n}_t^{e(l)} = \begin{bmatrix} \mathbf{n}_{t-1}^{s(l)} \\ \mathbf{n}_t^{s(l)} \\ \mathbf{n}_{t+1}^{s(l)} \end{bmatrix}; \quad \mathbf{h}_t^{e(l)} = \begin{bmatrix} \mathbf{h}_{t-1}^{s(l)} \\ \mathbf{h}_t^{s(l)} \\ \mathbf{h}_{t+1}^{s(l)} \end{bmatrix}; \quad \boldsymbol{\alpha}_t^{e(l)} = \begin{bmatrix} \boldsymbol{\alpha}_{t-1}^{s(l)} \\ \boldsymbol{\alpha}_t^{s(l)} \\ \boldsymbol{\alpha}_{t+1}^{s(l)} \end{bmatrix}. \tag{5.17}$$

The distribution of the extended feature vectors over the corrupted speech $\mathbf{y}^e$ is then defined by applying the mismatch function on each time instance, as in (5.3a). The mismatch function $\mathbf{f}$ for the static parameters in (4.10b) (for log-spectral feature vectors) or (4.20) (for cepstral feature vectors) is applied to each time offset. This yields

an extended corrupted speech sample $\mathbf{y}^{e(l)}$:

$$\mathbf{y}_t^{e(l)} = \begin{bmatrix} \mathbf{y}_{t-1}^{s(l)} \\ \mathbf{y}_t^{s(l)} \\ \mathbf{y}_{t+1}^{s(l)} \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{x}_{t-1}^{s(l)}, \mathbf{n}_{t-1}^{s(l)}, \mathbf{h}_{t-1}^{s(l)}, \boldsymbol{\alpha}_{t-1}^{s(l)}) \\ \mathbf{f}(\mathbf{x}_t^{s(l)}, \mathbf{n}_t^{s(l)}, \mathbf{h}_t^{s(l)}, \boldsymbol{\alpha}_t^{s(l)}) \\ \mathbf{f}(\mathbf{x}_{t+1}^{s(l)}, \mathbf{n}_{t+1}^{s(l)}, \mathbf{h}_{t+1}^{s(l)}, \boldsymbol{\alpha}_{t+1}^{s(l)}) \end{bmatrix}. \tag{5.18}$$

The empirical distribution then has delta spikes at the positions of these samples:

$$\tilde{p}^e = \frac{1}{L} \sum_l \delta_{\mathbf{y}_t^{e(l)}}. \tag{5.19}$$

This approximation can be substituted in for p^e in (5.9b):

$$\begin{aligned} \hat{q}^e &:= \arg\min_{q^e} \mathcal{KL}(\tilde{p}^e \| q^e) \\ &= \arg\max_{q^e} \int \tilde{p}^e(\mathbf{y}^e) \log q^e(\mathbf{y}^e) \, d\mathbf{y}^e \\ &= \arg\max_{q^e} \frac{1}{L} \sum_l \log q^e(\mathbf{y}^e). \end{aligned} \tag{5.20}$$

This is equivalent to finding a maximum-likelihood solution on the samples, which has a well-known procedure for many distributions. For a Gaussian $q \sim \mathcal{N}(\boldsymbol{\mu}_y^e, \boldsymbol{\Sigma}_y^e)$, maximum-likelihood estimates of the extended corrupted speech parameters $\boldsymbol{\mu}_y^e$ and $\boldsymbol{\Sigma}_y^e$ can then be found with

$$\boldsymbol{\mu}_y^e = \frac{1}{L} \sum_{l=1}^{L} \mathbf{y}^{e(l)}; \tag{5.21a}$$

$$\boldsymbol{\Sigma}_y^e = \left(\frac{1}{L} \sum_{l=1}^{L} \mathbf{y}^{e(l)} \left[\mathbf{y}^{e(l)} \right]^T \right) - \boldsymbol{\mu}_y^e \boldsymbol{\mu}_y^{eT}. \tag{5.21b}$$

The samples have been generated from distributions in which the time instances are correlated. Therefore, the time instances in the corrupted speech sample will also be dependent. The cross-correlations of $\boldsymbol{\Sigma}_y^e$ (with the structure in (5.4)) will therefore be estimated correctly.

Having thus found Gaussian parameters for the extended corrupted speech distribution, distributions over statics and dynamics can be found with (5.5):

$$\mathbf{y} = \mathbf{D}\mathbf{y}^e \sim \mathcal{N}\left(\mathbf{D}\boldsymbol{\mu}_y^e, \mathbf{D}\boldsymbol{\Sigma}_y^e \mathbf{D}^T \right). \tag{5.22}$$

In the limit as the number of samples goes to infinity, this finds the optimal Gaussian for the corrupted speech distribution.

An alternative perspective on DPMC gives less insight in the cross-correlations of the corrupted speech distribution but allows other distribution than Gaussians to be trained from the samples. It is possible to directly express the corrupted speech distribution with statics and dynamics while integrating over extended distributions (from (5.3c)):

$$p(\mathbf{y}) = \int\int\int\int \delta_{\mathbf{D}\mathbf{f}^e(\mathbf{x}^e,\mathbf{n}^e,\mathbf{h}^e,\boldsymbol{\alpha}^e)}(\mathbf{y})\, p(\mathbf{x}^e, \mathbf{n}^e, \mathbf{h}^e, \boldsymbol{\alpha}^e) d\boldsymbol{\alpha}^e d\mathbf{h}^e d\mathbf{n}^e d\mathbf{x}^e. \tag{5.23}$$

Note that the Dirac delta has a projection $\mathbf{D}$ added, so that the delta spike is directly in the standard domain. The Monte Carlo approximation of this integral is very similar to that of the first perspective on DPMC. Extended corrupted speech samples $\mathbf{y}^{e(l)}$ are found exactly as in (5.18), but then immediately converted to samples with statics and dynamics:

$$\mathbf{y}^{(l)} = \mathbf{D}\mathbf{y}^{e(l)}. \tag{5.24}$$

The empirical distribution then is similar to $\tilde{p}^e$ in (5.19):

$$\tilde{p} = \frac{1}{L}\sum_{l} \delta_{\mathbf{y}_t^{(l)}} = \frac{1}{L}\sum_{l} \delta_{\mathbf{D}\mathbf{y}_t^{e(l)}}. \tag{5.25}$$

This distribution is in the domain with statics and dynamics, so that the procedure from here follows that of standard DPMC, in section 4.4.1. The parameters of Gaussian q are trained on samples $\mathbf{y}_t^{(l)}$ in exactly the same way. Substituting (5.24) into (4.31),

$$\boldsymbol{\mu}_y := \frac{1}{L}\sum_{l=1}^{L} \mathbf{y}^{(l)} = \frac{1}{L}\sum_{l=1}^{L} \mathbf{D}\mathbf{y}^{e(l)}; \tag{5.26a}$$

$$\boldsymbol{\Sigma}_y := \left(\frac{1}{L}\sum_{l=1}^{L} \mathbf{y}^{(l)}\mathbf{y}^{(l)^T}\right) - \boldsymbol{\mu}_y\boldsymbol{\mu}_y^T = \left(\frac{1}{L}\sum_{l=1}^{L} \mathbf{D}\mathbf{y}^{e(l)}\mathbf{y}^{e(l)^T}\mathbf{D}^T\right) - \boldsymbol{\mu}_y\boldsymbol{\mu}_y^T. \tag{5.26b}$$

This is equal to the parameters of DPMC viewed from the first perspective (combining (5.21) and (5.22)). As section 5.2.2 has proven, this is because q^e, and therefore q,

is Gaussian, and $\mathbf{D}$ is a linear projection. The next section will introduce extended iterative DPMC, which trains a mixture of Gaussians rather than one Gaussian, using the second perspective.

5.3.2 *Extended* IDPMC

Extended iterative DPMC is an extension of DPMC to train a mixture of Gaussians, as iterative DPMC is to DPMC (see section 4.4.1). It should be possible to train a mixture of Gaussians over extended feature vectors, and convert each of the Gaussians to be over standard vectors with statics and dynamics afterwards. However, as section 5.2.2 has shown, there is a requirement for this to be equivalent to optimising the distribution in the standard domain. When the extended distribution is transformed to a distribution with as many dimensions, some of which are the statics and dynamics, and the rest the nuisance dimensions, the distribution over the nuisance dimensions must be separate from the other distribution. In the case of a mixture of Gaussians, the nuisance dimensions are not allowed to depend on the hidden variable which indicates which component has produced the observation. This would mean that while training the mixture of Gaussians, some dimensions, in a different feature space, must be tied across components. A more straightforward way of deriving iterative DPMC is from the second perspective on DPMC, which converts each sample to the standard domain (in (5.24)) first, and then trains the distribution on those samples.

Training a mixture of Gaussians from extended samples without tying dimensions is not guaranteed to be optimal, whereas training it on samples with statics and dynamics is. This is straightforward to see from the procedure of training iterative DPMC. Training mixtures of Gaussians uses expectation–maximisation, with in each iteration assigns responsibilities (component-sample posteriors) to train the parameters of each Gaussian. The iteration is guaranteed not to decrease the likelihood, i.e. not to increase the KL divergence with the empirical distribution. Since recognition uses statics and dynamics, it is the likelihoods in that domain that the responsibilities should be computed for to optimise the KL divergence between p and q.

Extended iterative DPMC, then, uses the samples $\mathbf{y}^{(l)}$, found with (5.24). Training the mixture of Gaussians then applies exactly the iteration in (4.35):

$$\rho^{(k)}(m,l) := \frac{\pi_m^{(k-1)} q^{(m)(k-1)}\left(\mathbf{y}^{(l)}\right)}{\sum_{m' \in \Omega^{(\theta)}} \pi_{m'}^{(k-1)} q^{(m')(k-1)}\left(\mathbf{y}^{(l)}\right)}; \tag{5.27a}$$

$$\pi_m^{(k)} := \frac{1}{L} \sum_l \rho^{(k)}(m,l); \tag{5.27b}$$

$$\boldsymbol{\mu}_y^{(m)(k)} := \frac{1}{\sum_l \rho^{(k)}(m,l)} \sum_l \rho^{(k)}(m,l) \mathbf{y}^{(l)}; \tag{5.27c}$$

$$\boldsymbol{\Sigma}_y^{(m)(k)} := \left(\frac{1}{\sum_l \rho^{(k)}(m,l)} \sum_l \rho^{(k)}(m,l) \mathbf{y}^{(l)} \mathbf{y}^{(l)\mathsf{T}} \right) - \boldsymbol{\mu}_y^{(m)(k)} \boldsymbol{\mu}_y^{(m)(k)\mathsf{T}}. \tag{5.27d}$$

Analogously to IDPMC, a good initial setting for extended IDPMC, which this thesis will use, is extended DPMC. To increase the number of components, it will again apply "mixing up": the component with the greatest mixture weight is split into two components. Then, a few iterations of expectation–maximisation are run, and the procedure is repeated until the desired number of components has been reached.

In the limit as the number of components M and the number of samples L go to infinity, the modelled distribution of the corrupted speech can become equal to the real one. However, as explained in section 4.4.1, to train the mixture model well, the parameters of each component need to be trained on a large enough portion of the samples. Therefore, as the number of components M is increased, L must grow faster than M. One iteration of expectation–maximisation takes $\mathcal{O}(LM)$ time. The number of iterations of mixing up and expectation–maximisation increases linearly with M. In practice, therefore, the time complexity of IDPMC is at least $\mathcal{O}\left(M^3\right)$.

It is important to realise the difference with normal training of speech recognisers, where the amount of training data is limited, and the number of components is therefore limited. Here, an unlimited number of samples can be generated within machine limitations, and a high number of components can be chosen. To represent the noise-corrupted speech as well as possible, a much larger number of samples is useful than is usually required for training a speech recogniser, especially to estimate full covariance

matrices. However, it is not a priori clear how many Gaussians is enough to correctly represent the corrupted speech distribution in 39 dimensions. The experiments in section 8.2, which will model the distribution as exactly as possible, will increase the number of components, and therefore the number of samples, to the machine limits.

5.3.3 *Extended* VTS

Another method of finding Gaussian compensation adapts VTS compensation (section 4.4.2) to extended feature vectors. Just like the first presentation of extended DPMC in section 5.3.1, extended VTS (EVTS) will approximate the extended corrupted speech with a Gaussian, and then convert the Gaussian to the standard domain. Section 5.2.1 has shown that, if the approximation is Gaussian, minimising the KL divergence between the extended distribution and its approximation is equivalent to optimising the KL divergence in the standard domain with static and dynamic features.

Since the extended feature vector is a concatenation of static feature vectors, it is possible to use a linearised version of the static mismatch function for each time offset to yield an overall mismatch function for $\mathbf{y}^e$. This alleviates the need for the continuous-time approximation for dynamics, so that no distribution over dynamics is required for the phase factor. It is possible to define a distribution for the extended phase factor $\boldsymbol{\alpha}^e$. The elements of $\boldsymbol{\alpha}^e$ are approximately Gaussian distributed but constrained to $[-1, +1]$ (see section 5.5.3). To simplify the distribution of $\mathbf{y}^s$, the constraint can be ignored, so that $\boldsymbol{\alpha}^e \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\alpha^e)$.

An extension to static compensation using VTS can be used to find the extended corrupted speech distribution. The first-order vector Taylor series approximation in (4.36a) is applied to each time instance separately. Thus the expansion point for each time instance is given by the static means at the appropriate time offsets. These are obtained from the extended distributions distributions over the clean speech $\mathbf{x}^e$, noise $\mathbf{n}^e$, and the phase factor $\boldsymbol{\alpha}^e$. Thus, using the form of the VTS approximation in

(4.36a) per time instance:

$$\mathbf{f}^{e}_{\text{vts}}(\mathbf{x}^{e}_{t}, \mathbf{n}^{e}_{t}, \mathbf{h}^{e}_{t}, \boldsymbol{\alpha}^{e}_{t}) \triangleq \begin{bmatrix} \mathbf{f}_{t-1,\text{vts}}(\mathbf{x}_{t-1}, \mathbf{n}_{t-1}, \mathbf{h}_{t-1}, \boldsymbol{\alpha}_{t-1}) \\ \mathbf{f}_{t,\text{vts}}(\mathbf{x}_{t}, \mathbf{n}_{t}, \mathbf{h}_{t}, \boldsymbol{\alpha}_{t}) \\ \mathbf{f}_{t+1,\text{vts}}(\mathbf{x}_{t+1}, \mathbf{n}_{t+1}, \mathbf{h}_{t+1}, \boldsymbol{\alpha}_{t+1}) \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{f}\left(\boldsymbol{\mu}^{s}_{x_{-1}}, \boldsymbol{\mu}^{s}_{n_{-1}}, \boldsymbol{\mu}^{s}_{h_{-1}}, \mathbf{0}\right) + \mathbf{J}_{x_{-1}}\left(\mathbf{x}^{s}_{t-1} - \boldsymbol{\mu}^{s}_{x_{-1}}\right) + \mathbf{J}_{n_{-1}}\left(\mathbf{n}^{s}_{t-1} - \boldsymbol{\mu}^{s}_{n_{-1}}\right) + \mathbf{J}_{\alpha_{-1}}\boldsymbol{\alpha}^{s}_{t-1} \\ \mathbf{f}\left(\boldsymbol{\mu}^{s}_{x_{0}}, \boldsymbol{\mu}^{s}_{n_{0}}, \boldsymbol{\mu}^{s}_{h_{0}}, \mathbf{0}\right) + \mathbf{J}_{x_{0}}\left(\mathbf{x}^{s}_{t} - \boldsymbol{\mu}^{s}_{x_{0}}\right) + \mathbf{J}_{n_{0}}\left(\mathbf{n}^{s}_{t} - \boldsymbol{\mu}^{s}_{n_{0}}\right) + \mathbf{J}_{\alpha_{0}}\boldsymbol{\alpha}^{s}_{t} \\ \mathbf{f}\left(\boldsymbol{\mu}^{s}_{x_{+1}}, \boldsymbol{\mu}^{s}_{n_{+1}}, \boldsymbol{\mu}^{s}_{h_{+1}}, \mathbf{0}\right) + \mathbf{J}_{x_{+1}}\left(\mathbf{x}^{s}_{t+1} - \boldsymbol{\mu}^{s}_{x_{+1}}\right) + \mathbf{J}_{n_{+1}}\left(\mathbf{n}^{s}_{t+1} - \boldsymbol{\mu}^{s}_{n_{+1}}\right) + \mathbf{J}_{\alpha_{+1}}\boldsymbol{\alpha}^{s}_{t+1} \end{bmatrix}, \tag{5.28}$$

where the offset-dependent Jacobians are given by

$$\begin{aligned} \mathbf{J}_{x_{-1}} &= \left.\frac{\mathrm{d}\mathbf{y}^{s}}{\mathrm{d}\mathbf{x}^{s}}\right|_{\boldsymbol{\mu}^{s}_{n_{-1}}, \boldsymbol{\mu}^{s}_{x_{-1}}, \boldsymbol{\mu}^{s}_{h_{-1}}, \mathbf{0}}; \qquad & \mathbf{J}_{x_{0}} &= \left.\frac{\mathrm{d}\mathbf{y}^{s}}{\mathrm{d}\mathbf{x}^{s}}\right|_{\boldsymbol{\mu}^{s}_{n_{0}}, \boldsymbol{\mu}^{s}_{x_{0}}, \boldsymbol{\mu}^{s}_{h_{0}}, \mathbf{0}}; \\ \mathbf{J}_{x_{+1}} &= \left.\frac{\mathrm{d}\mathbf{y}^{s}}{\mathrm{d}\mathbf{x}^{s}}\right|_{\boldsymbol{\mu}^{s}_{n_{+1}}, \boldsymbol{\mu}^{s}_{x_{+1}}, \boldsymbol{\mu}^{s}_{h_{+1}}, \mathbf{0}}, \end{aligned} \tag{5.29}$$

and similar for $\mathbf{J}_{n}$ and $\mathbf{J}_{\alpha}$. Note that $\mathbf{J}_{x_0}$ is equal to the Jacobian for standard VTS in (4.36b).

As in standard VTS (in (4.41)), approximating the corrupted speech distribution works by substituting the vector Taylor series approximation of the mismatch function. Again, using this approximation, the extended corrupted speech distribution drops out as Gaussian. The approximation q^{e} to the extended corrupted speech distribution then is

$$\begin{aligned} \mathrm{q}^{e}(\mathbf{y}^{e}) &:= \int\int\int \delta_{\mathbf{f}^{e}_{\text{vts}}(\mathbf{x}^{e}, \mathbf{n}^{e}, \mathbf{h}^{e}, \boldsymbol{\alpha}^{e})}(\mathbf{y}^{e})\, \mathrm{p}(\mathbf{x}^{e}, \mathbf{n}^{e}, \mathbf{h}^{e}, \boldsymbol{\alpha}^{e})\, \mathrm{d}\boldsymbol{\alpha}^{e}\, \mathrm{d}\mathbf{n}^{e}\, \mathrm{d}\mathbf{x}^{e} \\ &= \int\int\int \delta_{\mathbf{f}^{e}_{\text{vts}}(\mathbf{x}^{e}, \mathbf{n}^{e}, \mathbf{h}^{e}, \boldsymbol{\alpha}^{e})}(\mathbf{y}^{e})\, \mathcal{N}\left(\mathbf{x}^{e}; \boldsymbol{\mu}^{e}_{x}, \boldsymbol{\Sigma}^{e}_{x}\right) \\ &\qquad \mathcal{N}(\mathbf{n}^{e}; \boldsymbol{\mu}^{e}_{n}, \boldsymbol{\Sigma}^{e}_{n})\, \mathcal{N}(\boldsymbol{\alpha}^{e}; \boldsymbol{\mu}^{e}_{\alpha}, \boldsymbol{\Sigma}^{e}_{\alpha})\ \mathrm{d}\boldsymbol{\alpha}^{e}\, \mathrm{d}\mathbf{n}^{e}\, \mathrm{d}\mathbf{x}^{e} \\ &\triangleq \mathcal{N}\left(\mathbf{y}^{e}; \boldsymbol{\mu}^{e}_{y}, \boldsymbol{\Sigma}^{e}_{y}\right). \end{aligned} \tag{5.30}$$

This Gaussian is found by compensating each block of the mean and covariance separately with the appropriately linearised mismatch function. The structure of the para-

meters of q^e is (repeated from (5.4))

$$\mu_y^e = \begin{bmatrix} \mu_{y_{-1}}^s \\ \mu_{y_0}^s \\ \mu_{y_{+1}}^s \end{bmatrix}; \qquad \Sigma_y^e = \begin{bmatrix} \Sigma_{y_{-1}y_{-1}}^s & \Sigma_{y_{-1}y_0}^s & \Sigma_{y_{-1}y_{+1}}^s \\ \Sigma_{y_0y_{-1}}^s & \Sigma_{y_0y_0}^s & \Sigma_{y_0y_{+1}}^s \\ \Sigma_{y_{+1}y_{-1}}^s & \Sigma_{y_{+1}y_0}^s & \Sigma_{y_{+1}y_{+1}}^s \end{bmatrix}. \tag{5.31}$$

The parameters of this extended corrupted speech distribution can be found by computing expectations over the Gaussians. The mean for time offset $+1$, for example, uses the linearisation for that time instance:

$$\mu_{y_{+1}}^s = \mathcal{E}\{f_{t+1,vts}(x_{t+1}^s, n_{t+1}^s, h_{t+1}^s, \alpha_{t+1}^s)\} = f(\mu_{x_{+1}}^s, \mu_{n_{+1}}^s, \mu_{h_{+1}}^s). \tag{5.32}$$

The covariance matrix Σ_y^e requires the correlations between all time offsets in the window. The covariance between offsets 0 and $+1$, for example, uses the linearisations for time instances 0 and $+1$:

$$\begin{aligned} \Sigma_{y_0y_{+1}}^s &= \mathcal{E}\{(f_{t,vts}(x_0^s, n_0^s, h_0^s, \alpha_0^s) - \mu_{y_0})(f_{t+1,vts}(x_{+1}^s, n_{+1}^s, h_{+1}^s, \alpha_{+1}^s) - \mu_{y_{+1}})^T\} \\ &= \mathcal{E}\Big\{ \big(J_{x_0}(x_t^s - \mu_{x_0}^s) + J_{n_0}(n_t^s - \mu_{n_0}^s) + J_{\alpha_0}\alpha_t^s\big) \\ &\qquad \big(J_{x_{+1}}(x_{t+1}^s - \mu_{x_{+1}}^s) + J_{n_{+1}}(n_{t+1}^s - \mu_{n_{+1}}^s) + J_{\alpha_{+1}}\alpha_{t+1}^s\big)^T \Big\} \\ &= J_{x_0}\Sigma_{x_0x_{+1}}^s J_{x_{+1}}^T + J_{n_0}\Sigma_{n_0n_{+1}}^s J_{n_{+1}}^T + J_{\alpha_0}\Sigma_{\alpha_0\alpha_{+1}}^s J_{\alpha_{+1}}^T. \end{aligned} \tag{5.33}$$

This is applied for each of the possible time offset blocks in Σ_y^e. Thus, each block (t, t') in the covariance matrix combines Jacobians at time offsets t and t' and cross-covariances of the speech, noise, and phase factor for t and t'. This is the main difference between standard VTS (which uses the continuous-time approximation) and extended VTS. Section 5.3.3.1 will examine standard VTS in the light of this. Section 5.3.3.2 will discuss the time complexity of extended VTS.

5.3.3.1 Relationship with VTS

It is interesting to examine the relationship between standard VTS and extended VTS described in the previous section. It is possible to describe the mismatch function for

the dynamic coefficients of standard VTS, which uses the continuous-time approximation for the dynamic coefficients, in terms of extended VTS. To be able to compare with standard VTS at all, the phase factor has to be assumed 0.

The approximation that standard VTS uses for the static coefficients is exactly the same as the one extended VTS uses for the statics at the centre time offset. Therefore, the compensated static mean and covariance that standard VTS finds are the same as the ones extended VTS finds for time offset 0. However, dynamic parameter compensation with standard VTS uses the continuous-time approximation. This uses the vector Taylor series expansion point of the static coefficients for all the dynamic coefficients. When the vector Taylor series expansion uses the same clean speech and noise means $\mu^s_{x_0}, \mu^s_{n_0}$ and thus the same Jacobian $\mathbf{J}_0$ for all time offsets in (5.28), it becomes

$$\begin{bmatrix} y^s_{t-1} \\ y^s_t \\ y^s_{t+1} \end{bmatrix} \simeq \begin{bmatrix} f(\mu^s_{x_0}, \mu^s_{n_0}, \mu^s_{h_0}, 0) + \mathbf{J}_{x_0}(x^s_{t-1} - \mu^s_{x_0}) + \mathbf{J}_{n_0}(n^s_{t-1} - \mu^s_{n_0}) \\ f(\mu^s_{x_0}, \mu^s_{n_0}, \mu^s_{h_0}, 0) + \mathbf{J}_{x_0}(x^s_t - \mu^s_{x_0}) + \mathbf{J}_{n_0}(n^s_t - \mu^s_{n_0}) \\ f(\mu^s_{x_0}, \mu^s_{n_0}, \mu^s_{h_0}, 0) + \mathbf{J}_{x_0}(x^s_{t+1} - \mu^s_{x_0}) + \mathbf{J}_{n_0}(n^s_{t+1} - \mu^s_{n_0}) \end{bmatrix}. \quad (5.34)$$

This approximation results in the following when substituted in the expression for computing dynamic coefficients in (2.8):

$$\begin{aligned} y^\Delta_t &= \frac{\sum_{\tau=1}^w \tau(y^s_{t+\tau} - y^s_{t-\tau})}{2\sum_{\tau=1}^w \tau^2} \\ &= \frac{\sum_{\tau=1}^w \tau(\mathbf{J}_{x_0} x^s_{t+\tau} + \mathbf{J}_{n_0} n^s_{t+\tau} - \mathbf{J}_{x_0} x^s_{t-\tau} - \mathbf{J}_{n_0} n^s_{t-\tau})}{2\sum_{\tau=1}^w \tau^2} \\ &= \frac{\mathbf{J}_{x_0}\sum_{\tau=1}^w \tau(x^s_{t+\tau} - x^s_{t-\tau}) + \mathbf{J}_{n_0}\sum_{\tau=1}^w \tau(n^s_{t+\tau} - n^s_{t-\tau})}{2\sum_{\tau=1}^w \tau^2} \\ &= \mathbf{J}_{x_0} x^\Delta_t + \mathbf{J}_{n_0} n^\Delta_t. \end{aligned} \quad (5.35)$$

This is exactly the same expression as the continuous-time approximation when applied to VTS compensation (in (4.37)). Extended VTS compensation therefore becomes equivalent to standard VTS compensation when the expansion point is chosen equal for all time offsets. Extended VTS performs the transformation from extended to standard parameters after compensation. This allows extended VTS to use a different vector Taylor series expansion point for every time offset to find more accurate compensation.

	VTS		eVTS	
Statistics	diag.	block-d.	striped	full
Decoding	diag.	block-d.	diag.	full
Jacobians	s^3		es^3	
Compensation	$d^\Delta s^2$	$d^\Delta s^3$	$e^2 s^2$	$e^2 s^3$
Projection	—		$d^\Delta s e^2$	$d^{\Delta^2} s^2 e^2$

Table 5.1 *Computational complexity $\mathcal{O}(\cdot)$ per component for compensation with VTS and with extended VTS, for diagonal blocks and for full blocks.*

5.3.3.2 *Computational cost*

Compensation with extended VTS is more computationally expensive than VTS with the continuous-time approximation. This section examines the differences in detail. The computational complexity per component will be expressed in terms of the size of the static feature vector s (typically 13), the total width of the window $e = 4w + 1$ (typically 9), and the number of orders of statics and dynamics d^Δ (typically 3). Since the calculation of the covariance matrices dominates the computation time, the analysis will not explicitly consider the means.

Table 5.1 gives a comparison of the computational complexity for the three operations that can be distinguished in extended VTS compensation. The first one is computing the Jacobian of the mismatch function, which takes $\mathcal{O}(s^3)$. Standard VTS compensation uses one linearisation point per component, and therefore needs to compute the Jacobian only once. Extended VTS, however, uses a different linearisation point for all e time offset in the window, and computes a Jacobian for each of these.

Compensation of the covariance matrix is done one $s \times s$ block at a time. The expression for standard VTS (see (4.42b)) is of the form:

$$\boldsymbol{\Sigma}_y^s := \mathbf{J}_x \boldsymbol{\Sigma}_x \mathbf{J}_x^\mathsf{T} + \mathbf{J}_n \boldsymbol{\Sigma}_n \mathbf{J}_n^\mathsf{T}. \tag{5.36}$$

The expression for extended VTS compensation has the same form (but different variables) for each block of the covariance matrix. It has time complexity $\mathcal{O}(s^2)$ if the blocks for the noise $\boldsymbol{\Sigma}_n^s$, the clean speech $\boldsymbol{\Sigma}_x^s$ and the corrupted speech $\boldsymbol{\Sigma}_y^s$ are diagonal. For standard VTS, this happens when the covariances for statistics and decoding

are all diagonal; for extended VTS, when the blocks in the covariance matrices for statistics are diagonal ("striped"), and covariances for decoding are diagonal. When either the statistics or compensation uses full covariance matrices, then compensation takes $\mathcal{O}(s^3)$. For VTS, the d^Δ blocks along the diagonal are compensated; for extended VTS, for the $\frac{1}{2}e(e+1)$ blocks in the extended covariance matrix. The row labelled "Compensation" in table 5.1 summarises this.

Extended VTS projects the compensated distribution onto the standard feature space with statics and dynamics. Since the blocks of the projection matrix $\mathbf{D}$ are diagonal, computing one entry of the resulting covariance matrix $\boldsymbol{\Sigma}_y^e = \mathbf{D}\boldsymbol{\Sigma}_y^e\mathbf{D}^T$ takes $\mathcal{O}(e^2)$. For diagonal-covariance decoding, $d^\Delta s$ entries need to be computed; for full-covariance decoding, ${d^\Delta}^2 s^2$.

Thus for full-covariance compensation, the computational complexity of EVTS is significantly higher than standard VTS. However, in practice per-Gaussian compensation is often too costly even when the standard version of VTS is used. Joint uncertainty decoding (JUD) (Liao 2007, section 4.4.3) addresses this by computing compensation per base class rather than per Gaussian component. Section 5.4 will detail how to find a joint distribution using extended VTS. Chapter 6 will deal with another important issue: the computational cost of decoding. If full-covariance compensation is found, joint uncertainty decoding still compensates for changes in the correlations by decoding with full covariance matrices. This is slow. Predictive linear transformations can solve this issue by applying transformation to the feature vectors that eliminate the need to decode with full covariance matrices.

5.3.4 Algonquin with extended feature vectors

Section 4.5.1 has discussed the Algonquin algorithm, which applies a vector Taylor series approximation but iteratively updates the expansion point to fit the observation better. This used the joint distribution of the sources, the speech and the noise, and the observation. The way it was originally presented, it only acted on the static part of feature vectors. This is all that is necessary for feature enhancement. The covari-

ance matrix of the observation was supposed to be diagonal. To make the process consistent, the relation between the sources and the observation must then also be assumed independent, which is an additional approximation. If this approximation is to be removed, then the observation covariance must be full, which runs into the same problem as for standard VTS compensation: compensation for dynamics, which applies the continuous-time approximation, is not accurate enough.

However, it is possible to apply the Algonquin algorithm to extended feature vectors so as not to rely on the continuous-time approximation. This makes it possible to express the influence that one coefficient in the speech vector, for example, has on another coefficient in the observation vector. The main intuition is that once the mismatch function is linearised for a component, the extended speech and noise vectors and the observation vector with statics and dynamics are jointly Gaussian. The following will detail how this intuition can be used to find an extended Algonquin algorithm.

Extended VTS applies the mismatch function per time instance. One way of viewing the resulting transformation from extended speech and noise to extended observation vectors is as a function $\mathbf{f}^{e}_{vts}$ defined in (5.28):

$$\mathbf{y}^{e} \simeq \mathbf{f}^{e}_{vts}(\mathbf{x}^{e}, \mathbf{n}^{e}, \mathbf{h}^{e}, \boldsymbol{\alpha}^{e}). \tag{5.37}$$

Just like in the original Algonquin algorithm, the phase factor will not be assumed to have a distribution, but assumed $\mathbf{0}$, and instead the uncertainty will be modelled with an error term $\mathcal{N}(\mathbf{0}, \boldsymbol{\Psi})$ on the observation. Also, for notational convenience, the convolutional noise $\mathbf{h}^{e}$ will again be omitted. It is handled the same as the additive noise $\mathbf{n}^{e}$. The linearised mismatch function $\mathbf{f}^{e(k)}_{vts}$ at iteration k relates the extended vectors of the sources to the extended observation vector $\mathbf{y}^{e}$, which in turn is related to the observation vector with statics and dynamics by the linear projection $\mathbf{D}$ as in (5.1a), plus error term $\mathcal{N}(\mathbf{0}, \boldsymbol{\Psi})$. This implies that if $\mathbf{x}^{e}$ and $\mathbf{n}^{e}$ are Gaussian, $\mathbf{x}^{e}$, $\mathbf{n}^{e}$ and $\mathbf{y}$ are

jointly Gaussian, similarly to (4.58):

$$q_{y_t}^{(k)}\left(\begin{bmatrix} x^e \\ n^e \\ y \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} x^e \\ n^e \\ y \end{bmatrix}; \begin{bmatrix} \mu_x^e \\ \mu_n^e \\ \mu_y^{(k)} \end{bmatrix}, \begin{bmatrix} \Sigma_x^e & 0 & \Sigma_{xy}^{(k)} \\ 0 & \Sigma_n^e & \Sigma_{ny}^{(k)} \\ \Sigma_{yx}^{(k)} & \Sigma_{yn}^{(k)} & \Sigma_y^{(k)} \end{bmatrix}\right). \tag{5.38}$$

μ_x^e, μ_n^e, Σ_x^e, and Σ_n^e are taken directly from the priors of the speech and the noise. The Jacobians that related the extended speech and noise with the observation with statics and dynamics, y, are

$$J_x^{(k)} = \left.\frac{dy}{dx}\right|_{x_0^{(k)}} = \left.\frac{dy}{dy^e}\,\frac{dy^e}{dx}\right|_{x_0^{(k)}} = D\begin{bmatrix} J_{x_{-1}}^{s(k)} & 0 & 0 \\ 0 & J_{x_0}^{s(k)} & 0 \\ 0 & 0 & J_{x_{+1}}^{s(k)} \end{bmatrix}; \tag{5.39a}$$

$$J_n^{(k)} = \left.\frac{dy}{dn}\right|_{n_0^{(k)}} = \left.\frac{dy}{dy^e}\,\frac{dy^e}{dn}\right|_{n_0^{(k)}} = D\begin{bmatrix} J_{n_{-1}}^{s(k)} & 0 & 0 \\ 0 & J_{n_0}^{s(k)} & 0 \\ 0 & 0 & J_{n_{+1}}^{s(k)} \end{bmatrix}. \tag{5.39b}$$

The parameters of the joint distribution that depend on the linearisation at iteration k are then

$$\Sigma_{yx}^{(k)} = J_x^{e(k)}\Sigma_x^e; \tag{5.40a}$$

$$\Sigma_{yn}^{(k)} = J_n^{e(k)}\Sigma_n^e; \tag{5.40b}$$

$$\Sigma_y^{(k)} = J_x^{e(k)}\Sigma_x^e J_x^{e(k)^T} + J_n^{e(k)}\Sigma_n^e J_n^{e(k)^T} + \Psi. \tag{5.40c}$$

Having set up the joint distribution in (5.38), the Algonquin algorithm proceeds as in section 4.5.1. From the joint distribution, each observation gives a Gaussian approximation of the posterior distribution of the extended speech and the noise. The expansion point is set to the mean of this distribution, which yields a newly linearised mismatch function f_{vts}^e, and therefore a new joint distribution. After a few iterations of this process, the expansion point should be centred on the actual posterior of the speech and the noise. The advantage of applying this to extended feature vectors is that the distribution of the corrupted speech with dynamic coefficients can be modelled with a full covariance matrix. The original algorithm diagonalised the corrupted

speech distribution. To make the joint distribution valid, the Jacobians must then also be assumed diagonal, which is an additional approximation compared to VTS compensation. Using extended feature vectors, compensation is of good enough quality to find full covariance matrices.

5.4 Extended joint uncertainty decoding

Section 4.4.3 has discussed joint uncertainty decoding, which applies a compensation method to a base class at once. The Gaussian joint distribution can be estimated using any model compensation method, with an appropriate extension. Given the joint distribution per base class, it is possible to compensate the components in that base class more quickly than by applying the compensation method on each component separately.

Applying extended VTS to each component is slower than normal VTS. Therefore, applying extended VTS per base class leads to an even greater increase in speed than applying standard VTS per base class. However, one of the important aspects of extended VTS is that it can generate full-covariance compensation. This leads joint uncertainty decoding to produce full-covariance compensation as well, which slows down decoding. Therefore, section 6.3 will present predictive linear transformations, which enable fast decoding from full-covariance joint uncertainty decoding. This section will produce joint Gaussian distributions with extended DPMC and extended VTS (repeated from (4.46)):

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_x & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_y \end{bmatrix} \right). \tag{5.41}$$

Since this joint distribution over statics and dynamics is Gaussian, like in section 5.2, finding the optimal extended Gaussian distribution and then converting it yields the optimal distribution over statics and dynamics. The extended joint distribution is

$$\begin{bmatrix} \mathbf{x}^e \\ \mathbf{y}^e \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x^e \\ \boldsymbol{\mu}_y^e \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_x^e & \boldsymbol{\Sigma}_{xy}^e \\ \boldsymbol{\Sigma}_{yx}^e & \boldsymbol{\Sigma}_y^e \end{bmatrix} \right). \tag{5.42}$$

To transform the joint distribution over extended feature vectors in (5.41) into the one in (5.42), the relation between the joint vectors applies to the Gaussian:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} D & 0 \\ 0 & D \end{bmatrix} \begin{bmatrix} x^e \\ y^e \end{bmatrix}. \tag{5.43}$$

Therefore, the same transformation can be applied to the joint extended distribution in (5.42):

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} D\mu_x^e \\ D\mu_y^e \end{bmatrix}, \begin{bmatrix} D\Sigma_x^e D^\top & D\Sigma_{xy}^e D^\top \\ D\Sigma_{yx}^e D^\top & D\Sigma_y^e D^\top \end{bmatrix} \right). \tag{5.44}$$

Given these parameters per base class, decoding uses the same form as standard joint uncertainty decoding, in section 4.4.3.

The rest of this section will therefore estimate the parameters of the joint extended distribution in (5.42). Just like for standard joint uncertainty decoding, the original compensation methods (here, extended DPMC and extended VTS) already provide the clean and corrupted speech marginals $x^e \sim \mathcal{N}(\mu_x^e, \Sigma_x^e)$, $y^e \sim \mathcal{N}\left(\mu_y^e, \Sigma_y^e\right)$. The clean speech is given, and the corrupted speech Gaussian is what a model compensation method finds. The cross-covariance Σ_{xy}^e is what the extension needs to find.

With extended DPMC Section 4.4.3.1 has detailed how to find a standard joint distribution with DPMC. The procedure for producing an extended joint distribution with extended DPMC is analogous.

Section 5.3.1 has discussed how to draw extended samples $y^{e(l)}$ from the noise-corrupted speech distribution. To train the joint distribution, sample pairs of both the clean speech and the corrupted speech are retained. The empirical distribution has L delta spikes at positions $(x^{e(l)}, y^{e(l)})$, analogous to (5.19):

$$\tilde{p}(x^e, y^e) = \frac{1}{L} \sum_l \delta_{(x^{e(l)}, y^{e(l)})}. \tag{5.45}$$

Just like in (5.21), the parameters of the Gaussian are set to minimise the KL divergence to the empirical distribution. This is equivalent to maximising the likelihood

of the resulting distributions on the samples. However, for the joint distribution, the mean and covariance parameters are set at once to the mean and covariance of the tuple $(\mathbf{x}, \mathbf{y})$ under the empirical distribution:

$$\begin{bmatrix} \boldsymbol{\mu}^{e}_{x} \\ \boldsymbol{\mu}^{e}_{y} \end{bmatrix} := \frac{1}{L} \sum_{l=1}^{L} \begin{bmatrix} \mathbf{x}^{e(l)} \\ \mathbf{y}^{e(l)} \end{bmatrix}; \tag{5.46a}$$

$$\begin{bmatrix} \boldsymbol{\Sigma}^{e}_{x} & \boldsymbol{\Sigma}^{e}_{xy} \\ \boldsymbol{\Sigma}^{e}_{yx} & \boldsymbol{\Sigma}^{e}_{y} \end{bmatrix} := \left(\frac{1}{L} \sum_{l=1}^{L} \begin{bmatrix} \mathbf{x}^{e(l)} \\ \mathbf{y}^{e(l)} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{e(l)} \\ \mathbf{y}^{e(l)} \end{bmatrix}^{T} \right) - \begin{bmatrix} \boldsymbol{\mu}^{e}_{x} \\ \boldsymbol{\mu}^{e}_{y} \end{bmatrix} \begin{bmatrix} \boldsymbol{\mu}^{e}_{x} \\ \boldsymbol{\mu}^{e}_{y} \end{bmatrix}^{T}. \tag{5.46b}$$

With extended VTS When finding the joint distribution with extended DPMC, the estimating of the cross-covariance is implicit. When using extended VTS for the same purpose, however, the structure of the cross-covariance has to be considered. It is analogous to (5.4):

$$\boldsymbol{\Sigma}^{e}_{yx} = \begin{bmatrix} \boldsymbol{\Sigma}^{s}_{y_{-1}x_{-1}} & \boldsymbol{\Sigma}^{s}_{y_{-1}x_{0}} & \boldsymbol{\Sigma}^{s}_{y_{-1}x_{+1}} \\ \boldsymbol{\Sigma}^{s}_{y_{0}x_{-1}} & \boldsymbol{\Sigma}^{s}_{y_{0}x_{0}} & \boldsymbol{\Sigma}^{s}_{y_{0}x_{+1}} \\ \boldsymbol{\Sigma}^{s}_{y_{+1}x_{-1}} & \boldsymbol{\Sigma}^{s}_{y_{+1}x_{0}} & \boldsymbol{\Sigma}^{s}_{y_{+1}x_{+1}} \end{bmatrix}. \tag{5.47}$$

The blocks of this can each be found analogously to (5.33). For example, for the cross-covariance between the corrupted speech at time instance 0 and the clean speech at time instance $+1$, noting that the clean speech is assumed independent of the noise and the phase factor,

$$\begin{aligned} \boldsymbol{\Sigma}^{s}_{y_{0}x_{+1}} &= \mathcal{E}\left\{ (\mathbf{f}_{vts}(\mathbf{x}^{s}_{0}, \mathbf{n}^{s}_{0}, \mathbf{h}^{s}_{0}, \boldsymbol{\alpha}^{s}_{0}) - \boldsymbol{\mu}_{y_{0}}) \left(\mathbf{x}_{t+1} - \boldsymbol{\mu}_{x_{+1}}\right)^{T} \right\} \\ &= \mathcal{E}\left\{ \left(\mathbf{J}_{x_{0}} (\mathbf{x}^{s}_{t} - \boldsymbol{\mu}^{s}_{x_{0}}) + \mathbf{J}_{n_{0}} (\mathbf{n}^{s}_{t} - \boldsymbol{\mu}^{s}_{n_{0}}) + \mathbf{J}_{\alpha_{0}} \alpha^{s} \right) \left(\mathbf{x}^{s}_{t} - \boldsymbol{\mu}^{s}_{x_{+1}} \right)^{T} \right\} \\ &= \mathbf{J}_{x_{0}} \boldsymbol{\Sigma}^{s}_{x_{0}x_{+1}}. \end{aligned} \tag{5.48}$$

The equivalent computation can be performed for all blocks of $\boldsymbol{\Sigma}^{e}_{yx}$. This gives the complete joint distribution in (5.42).

5.5 Extended statistics

A practical issue when compensating extended feature vectors is the form of the statistics for the clean speech and the noise. For standard VTS, the clean speech statistics are usually taken from the recogniser trained on clean speech and the noise model is usually estimated with maximum-likelihood estimation, as discussed in section 4.7. In contrast, extended VTS and extended DPMC require distributions over the extended clean speech and noise vectors. As these have more parameters than standard statistics, robustness and storage requirements need to be carefully considered. A model for the phase factor is also needed.

5.5.1 Clean speech statistics

Model compensation schemes, such as VTS, use the Gaussian components from the uncompensated system as the clean speech distributions. For extended VTS and extended DPMC, however, distributions over the extended clean speech vector are required. For one extended clean speech Gaussian $\mathcal{N}(\boldsymbol{\mu}_x^e, \boldsymbol{\Sigma}_x^e)$, the parameters are of the same form as in 5.4:

$$\boldsymbol{\mu}_x^e = \begin{bmatrix} \boldsymbol{\mu}_{x_{-1}}^s \\ \boldsymbol{\mu}_{x_0}^s \\ \boldsymbol{\mu}_{x_{+1}}^s \end{bmatrix}; \qquad \boldsymbol{\Sigma}_x^e = \begin{bmatrix} \boldsymbol{\Sigma}_{x_{-1}x_{-1}}^s & \boldsymbol{\Sigma}_{x_{-1}x_0}^s & \boldsymbol{\Sigma}_{x_{-1}x_{+1}}^s \\ \boldsymbol{\Sigma}_{x_0x_{-1}}^s & \boldsymbol{\Sigma}_{x_0x_0}^s & \boldsymbol{\Sigma}_{x_0x_{+1}}^s \\ \boldsymbol{\Sigma}_{x_{+1}x_{-1}}^s & \boldsymbol{\Sigma}_{x_{+1}x_0}^s & \boldsymbol{\Sigma}_{x_{+1}x_{+1}}^s \end{bmatrix}. \tag{5.49}$$

In common with standard model compensation schemes, when there is no noise the compensated system should be the same as the original clean system trained with expectation–maximisation. To ensure that this is the case single-pass retraining (Gales 1995) should be used to obtain the extended clean speech distributions. Here the same posteriors (associated with the complete data set for expectation–maximisation) of the last standard clean speech training iteration (with static and dynamic parameters) are used to accumulate extended feature vectors around every time instance.

Another problem with using the extended statistics is ensuring robust estimation. The extended feature vectors contain more coefficients than the standard static and

dynamic ones. Hence, the estimates of their distributions will be less robust and take up more memory. If full covariance matrices for $\boldsymbol{\Sigma}_x^e$ are stored and used, both first- and second-order dynamic parameters use window widths of ± 2, and there are s static parameters, this requires estimating a $9s \times 9s$ covariance matrix for every component. This is memory-intensive and singular matrices and numerical accuracy problems can occur. One solution is to reduce the number of Gaussian components or states in the system. However, the precision of the speech model then decreases. Also, this makes it hard to compare the performance of compensation with extended VTS and standard VTS.

An alternative approach is to modify the structure of the covariance matrices, in the same fashion as diagonalising the standard clean speech covariance model. To maintain some level of inter-frame correlations, which may be useful for computing the dynamic parameters, each block is diagonalised. This yields the following structure:

$$\boldsymbol{\Sigma}_x^e = \begin{bmatrix} \operatorname{diag}\left(\boldsymbol{\Sigma}_{x_{-1}x_{-1}}^s\right) & \operatorname{diag}\left(\boldsymbol{\Sigma}_{x_{-1}x_0}^s\right) & \operatorname{diag}\left(\boldsymbol{\Sigma}_{x_{-1}x_{+1}}^s\right) \\ \operatorname{diag}\left(\boldsymbol{\Sigma}_{x_0x_{-1}}^s\right) & \operatorname{diag}\left(\boldsymbol{\Sigma}_{x_0x_0}^s\right) & \operatorname{diag}\left(\boldsymbol{\Sigma}_{x_0x_{+1}}^s\right) \\ \operatorname{diag}\left(\boldsymbol{\Sigma}_{x_{+1}x_{-1}}^s\right) & \operatorname{diag}\left(\boldsymbol{\Sigma}_{x_{+1}x_0}^s\right) & \operatorname{diag}\left(\boldsymbol{\Sigma}_{x_{+1}x_{+1}}^s\right) \end{bmatrix}. \tag{5.50}$$

For each Gaussian component, the ith element of the static coefficients for a time instance is then assumed correlated with only itself and the ith element of other time instances. This causes $\boldsymbol{\Sigma}_x^e$ to have a striped structure with only $45s$ parameters rather than $9s(9s+1)/2$ for the full case. This type of covariance matrix will be called "striped". It is a simple instantiation of structured precision matrix modelling,[3] discussed in section 3.3.1, with the special attribute that when there is no noise it will still yield the standard static and dynamic clean speech parameters.

[3] A striped matrix can be expressed as a block-diagonal matrix transformed by a permutation matrix. This can straightforwardly be expressed in terms of basis matrices.

5.5.2 *Noise model estimation*

A noise model with extended feature vectors is necessary to perform compensation with extended VTS. This noise model is of the form $\mathcal{M}_n^e = \{\boldsymbol{\mu}_n^e, \boldsymbol{\Sigma}_n^e, \boldsymbol{\mu}_h^e\}$, with parameters

$$\mathbf{n}^e = \begin{bmatrix} \mathbf{n}_{t-1}^s \\ \mathbf{n}_t^s \\ \mathbf{n}_{t+1}^s \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{n_{-1}}^s \\ \boldsymbol{\mu}_{n_0}^s \\ \boldsymbol{\mu}_{n_{+1}}^s \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{n_{-1}n_{-1}}^s & \boldsymbol{\Sigma}_{n_{-1}n_0}^s & \boldsymbol{\Sigma}_{n_{-1}n_{+1}}^s \\ \boldsymbol{\Sigma}_{n_0n_{-1}}^s & \boldsymbol{\Sigma}_{n_0n_0}^s & \boldsymbol{\Sigma}_{n_0n_{+1}}^s \\ \boldsymbol{\Sigma}_{n_{+1}n_{-1}}^s & \boldsymbol{\Sigma}_{n_{+1}n_0}^s & \boldsymbol{\Sigma}_{n_{+1}n_{+1}}^s \end{bmatrix} \right); \tag{5.51}$$

$$\mathbf{h}^e = \begin{bmatrix} \mathbf{h}_{t-1}^s \\ \mathbf{h}_t^s \\ \mathbf{h}_{t+1}^s \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}_{h_{-1}}^s \\ \boldsymbol{\mu}_{h_0}^s \\ \boldsymbol{\mu}_{h_{+1}}^s \end{bmatrix}. \tag{5.52}$$

In this thesis, and the majority of other work, the noise model consists of a single Gaussian component. The distribution for each time offset therefore is by definition the same. This means that the extended means for the additive and convolutional noise simply repeat the static means. The structure of the extended covariance $\boldsymbol{\Sigma}_n^e$ is also known. Since the noise is assumed identically distributed for all time instances at the same distance, the correlation between time instances is always the same. Thus, all diagonals of the covariance matrix repeat the same entries. Let $\boldsymbol{\Sigma}_{n_0}^s, \boldsymbol{\Sigma}_{n_1}^s, \boldsymbol{\Sigma}_{n_2}^s$ indicate the cross-correlation between noise that is 0, 1, or 2 time instances apart. The extended noise model then has the following form:

$$\boldsymbol{\mu}_n^e = \begin{bmatrix} \boldsymbol{\mu}_n^s \\ \boldsymbol{\mu}_n^s \\ \boldsymbol{\mu}_n^s \end{bmatrix}; \qquad \boldsymbol{\Sigma}_n^e = \begin{bmatrix} \boldsymbol{\Sigma}_{n_0}^s & \boldsymbol{\Sigma}_{n_1}^{sT} & \boldsymbol{\Sigma}_{n_2}^{sT} \\ \boldsymbol{\Sigma}_{n_1}^s & \boldsymbol{\Sigma}_{n_0}^s & \boldsymbol{\Sigma}_{n_1}^{sT} \\ \boldsymbol{\Sigma}_{n_2}^s & \boldsymbol{\Sigma}_{n_1}^s & \boldsymbol{\Sigma}_{n_0}^s \end{bmatrix}; \qquad \boldsymbol{\mu}_h^e = \begin{bmatrix} \boldsymbol{\mu}_h^s \\ \boldsymbol{\mu}_h^s \\ \boldsymbol{\mu}_h^s \end{bmatrix}. \tag{5.53}$$

In theory these noise parameters could be found using maximum likelihood estimation. However, this would make the noise estimation process inconsistent between standard VTS and extended VTS. It would be useful to use the standard noise estimation schemes and map the parameters to the ones in the extended forms above. This has the additional advantage of limiting the number of parameters to be trained, thus ensuring robust estimation on small amounts of data. These standard noise paramet-

ers are (repeated from (4.83))

$$\mu_n = \begin{bmatrix} \mu_n^s \\ 0 \end{bmatrix}; \qquad \Sigma_n = \begin{bmatrix} \mathrm{diag}(\Sigma_n^s) & 0 \\ 0 & \mathrm{diag}(\Sigma_n^\Delta) \end{bmatrix}; \qquad \mu_h = \begin{bmatrix} \mu_h^s \\ 0 \end{bmatrix}. \tag{5.54}$$

The extended noise means are straightforward functions of the static means of the standard noise model (5.54). Similarly, $\Sigma_{n_0}^s$, the covariance between noise 0 time instances apart, is the static noise covariance Σ_n^s. Computing the off-diagonals of the extended covariance, however, is not as straightforward. The next subsections will discuss two forms of extended noise covariance from the standard noise covariance: the diagonal form, and a smooth reconstruction.

A simple way of reconstructing the extended noise covariance from a standard noise model assumes that the noise is uncorrelated between time instances. This is done by setting the off-diagonal elements are set to zero, which yields

$$\Sigma_n^e = \begin{bmatrix} \Sigma_n^s & 0 & 0 \\ 0 & \Sigma_n^s & 0 \\ 0 & 0 & \Sigma_n^s \end{bmatrix}. \tag{5.55}$$

This only uses the static elements of the estimated noise covariance. For very low signal-to-noise ratio (SNR) conditions this form of extended noise distribution will not yield the standard noise distributions for the dynamic parameters.

Another option is to use the dynamic parameters of the noise model to find a reconstruction of the extended noise covariance from (5.54). A problem is that the mapping from the extended feature domain to statics and dynamics is straightforward, but the reverse mapping is under-specified. The standard feature vector n_t is related to one in the extended domain n_t^e (analogously to (5.1a)):

$$n_t = \begin{bmatrix} n_t^s \\ n_t^\Delta \end{bmatrix} = \begin{bmatrix} 0 & I & 0 \\ -\frac{I}{2} & 0 & \frac{I}{2} \end{bmatrix} \begin{bmatrix} n_{t-1}^s \\ n_t^s \\ n_{t+1}^s \end{bmatrix} = D n_t^e. \tag{5.56}$$

To reconstruct n_t^e from n_t, extra constraints are necessary as D is not square, and therefore not invertible. These constraints should yield an extended feature vector that

represents a plausible sequence of static feature vectors. The Moore-Penrose pseudo-inverse of $\mathbf{D}$ could be used. However, this would result in the $\mathbf{n}_t^e$ with the smallest norm. For the three-dimensional example used here, the reconstruction would set $\mathbf{n}_{t-1}^e = -\mathbf{n}_{t+1}^e$ without any reference to the value of the static coefficients $\mathbf{n}_t^s$. Thus, the Moore-Penrose pseudoinverse might lead to reconstructions with large changes in coefficients from one time to the next.

The need for smooth changes from time instance to time instance can be used as additional constraints. Thus the aim is to find a smooth reconstruction whilst satisfying the constraints to yield the standard static and dynamic distributions. To implement this constraint, rows representing higher-frequency changes are added to $\mathbf{D}$ and zeros added to $\mathbf{n}_t$ to indicate their desired values. The extension of the projection matrix $\mathbf{D}$, $\mathbf{E}$, can then be made invertible. Thus

$$\begin{bmatrix} \mathbf{n}_t^s \\ \mathbf{n}_t^\Delta \\ 0 \end{bmatrix} = \mathbf{E}\mathbf{n}_t^e; \qquad \mathbf{E}^{-1}\begin{bmatrix} \mathbf{n}_t^s \\ \mathbf{n}_t^\Delta \\ 0 \end{bmatrix} = \mathbf{n}_t^e. \tag{5.57}$$

For the extra rows of $\mathbf{E}$, the corresponding rows from the discrete cosine transform (DCT) matrix are appropriate, since they indicate higher-order frequencies and are independent. The entries of a $N \times N$ DCT matrix $\mathbf{C}$ are given by (see (2.3))

$$c_{ij} = \sqrt{\frac{2}{N}} \cos \frac{(2j-1)(i-1)\pi}{2N}. \tag{5.58}$$

The form of $\mathbf{E}$, $\mathbf{D}$ with DCT-derived blocks appended, is:

$$\mathbf{E} = \begin{bmatrix} 0 & \mathbf{I} & 0 \\ -\frac{\mathbf{I}}{2} & 0 & \frac{\mathbf{I}}{2} \\ c_{31}\mathbf{I} & c_{32}\mathbf{I} & c_{33}\mathbf{I} \end{bmatrix}. \tag{5.59}$$

Because the dynamic mean of the additive noise is zero, $\mathbf{E}^{-1}\boldsymbol{\mu}_n$ is equal to the extended mean in (5.53) (and similar for the convolutional noise). To reconstruct the extended covariance $\boldsymbol{\Sigma}_n^e$ from a standard noise model, the cross-covariance between statics and dynamics can be ignored, and the higher-order covariance terms set to zero to make

the reconstruction as smooth as possible. This results in the following expression:

$$\begin{bmatrix} \boldsymbol{\Sigma}_n^s & & \\ & \boldsymbol{\Sigma}_n^\Delta & \\ & & 0 \end{bmatrix} = \mathbf{E}\boldsymbol{\Sigma}_n^e\mathbf{E}^\top = \mathbf{E}\begin{bmatrix} \boldsymbol{\Sigma}_{n_0}^s & \boldsymbol{\Sigma}_{n_1}^{s\top} & \boldsymbol{\Sigma}_{n_2}^{s\top} \\ \boldsymbol{\Sigma}_{n_1}^s & \boldsymbol{\Sigma}_{n_0}^s & \boldsymbol{\Sigma}_{n_1}^{s\top} \\ \boldsymbol{\Sigma}_{n_2}^s & \boldsymbol{\Sigma}_{n_1}^s & \boldsymbol{\Sigma}_{n_0}^s \end{bmatrix}\mathbf{E}^\top, \tag{5.60}$$

where the empty entries on the left-hand side are ignored. This is a system with three sets of matrix equalities, which can be simply solved.[4] In this work, the estimated noise covariance matrix $\boldsymbol{\Sigma}_n$ is diagonal, so that $\boldsymbol{\Sigma}_{n_0}^s, \boldsymbol{\Sigma}_{n_1}^s, \boldsymbol{\Sigma}_{n_2}^s$ are also diagonal. This results in a striped matrix for $\boldsymbol{\Sigma}_n^e$.

5.5.2.1 Zeros in the noise variance estimate

An additional issue that can occur when estimating the noise model using maximum likelihood, is that noise variances estimates for some dimensions can become very small, or zero. Though this value may optimise the likelihood, it does not necessarily reflect the "true" noise variance. This can lead to the following problem in compensation.

One problem for the small noise variance estimates is that the clean speech silence models are never really estimated on silence. In practice, even for clean speech there are always low levels of background noise. Thus, the estimated noise is really only relative to this clean background level. At very high SNRs the noise may be at a similar level to the clean silence model. This will cause very small noise variance values. Another problem results from the form of the covariance matrix compensation. For the static parameters this may be written as (repeated from (4.42b))

$$\boldsymbol{\Sigma}_y^{s(m)} := \mathbf{J}_x^{(m)}\boldsymbol{\Sigma}_x^{s(m)}\mathbf{J}_x^{(m)\top} + \mathbf{J}_n^{(m)}\boldsymbol{\Sigma}_n^s\mathbf{J}_n^{(m)\top}. \tag{5.61}$$

At low SNRs $\mathbf{J}_x^{(m)} \to 0$ and $\mathbf{J}_n^{(m)} \to \mathbf{I}$, so the corrupted distribution tends to the noise distribution. Conversely, at high SNRs, $\mathbf{J}_x^{(m)} \to \mathbf{I}$ and $\mathbf{J}_n^{(m)} \to 0$, as the corrupted speech distribution tends to the clean speech distribution. The impact of this when estimating the noise covariance matrix $\boldsymbol{\Sigma}_n^s$ in high SNR conditions is that changes in

[4] This implicitly sets $\boldsymbol{\Sigma}_{n_0}^s$ to $\boldsymbol{\Sigma}_n^s$.

the form of the noise covariance matrix have little impact on the final compensated distribution.

When VTS with the continuous-time approximation, along with diagonal corrupted speech covariance matrices, is used during both noise estimation and recognition then the process is self-consistent. However if the noise estimates are used with EVTS to find full compensated covariance matrices this is not the case. This slight mismatch can cause problems. To address this issue a back-off strategy can be used. When the estimated noise variance has very low values rather than using full compensated covariance matrices, diagonal compensated variances can be used. This will occur at high SNRS, where the correlation changes compared to the clean speech conditions should be small. In this condition little gain is expected from full compensated covariance matrices.

5.5.2.2 Estimating an extended noise model

An alternative approach to address this problem is to make the noise estimation and decoding consistent for EVTS. This would mean: estimating the parameters of the extended noise distribution directly. Some of the methods in section 4.7.1 can be extended. The most important difference with the circumstances in that section is that now the compensated components have full covariances. Section 4.7.1 discussed two types of methods: one type modelled the noise as a hidden variable in the expectation–maximisation framework, and the other directly optimised the noise model. Directly optimising the noise model has become harder, because of the full covariance matrices. However, the biggest problem with seeing the noise as a hidden variable was the inconsistency arising from the diagonalisation. Since the resulting component distributions are not diagonalised any more, this ceases to be a problem. This section will sketch how estimation of the extended noise distribution could proceed.

Just like in section 4.7, training uses expectation–maximisation (see section 2.3.2.1). Here, the hidden variables $\mathcal{U}$ consist of the component sequence m_t and the sequence of extended noise vectors $\mathbf{n}^e_t$ and $\mathbf{h}^e_t$. At an abstract level, the expressions for the ex-

pectation and maximisation steps are the same as in (4.84):

$$\rho^{(k)} := q^{(k)}_{\mathcal{U}|\mathcal{Y}} \propto q^{(k)}_{\mathcal{U}\mathcal{Y}}; \tag{5.62a}$$

$$q^{(k)}_{\mathcal{U}} := \arg\max_{q_{\mathcal{U}}} \int \tilde{p}(\mathcal{Y}) \int \rho^{(k)}(\mathcal{U}|\mathcal{Y}) \log q_{\mathcal{U}}(\mathcal{U}) \, d\mathcal{U} \, d\mathcal{Y}. \tag{5.62b}$$

For simplicity, assume only additive noise. The expectation step again is approximate: the linearisation of the mismatch function from the last iteration is used. With extended feature vectors, each time instance is linearised separately. For time instance $t+1$, for example, the influence of $\mathbf{n}^s_{t+1}$ on $\mathbf{y}^s_{t+1}$ is defined by $\mathbf{J}_{n_{t+1}}$. The extended feature vector $\mathbf{y}^e_t$, which consists of time instances $\mathbf{y}^s_{t-w} \ldots \mathbf{y}^s_{t+w}$, is related to the observation with statics and dynamics with another linear projection $\mathbf{D}$. The influence of the extended noise vector on the observation vector with statics and dynamics is therefore linear. The details of the relationship are the same as for extended Algonquin (see section 5.3.4). Both the noise and the observation are modelled Gaussian per component m, so that their relation can be expressed as a joint distribution:

$$\begin{bmatrix} \mathbf{n}^e \\ \mathbf{y} \end{bmatrix} \middle| m \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}^e_n \\ \boldsymbol{\mu}^{(m)(k)}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}^e_n & \boldsymbol{\Sigma}^{(m)(k)}_{ny} \\ \boldsymbol{\Sigma}^{(m)(k)}_{yn} & \boldsymbol{\Sigma}^{(m)(k)}_y \end{bmatrix} \right), \tag{5.63}$$

where $\boldsymbol{\Sigma}_{ny}$ is the cross-covariance of the extended noise vector and the conventional observation vector. None of the blocks of the covariance matrix is diagonal. The posterior distribution of the noise vector for component m at time t, $\rho^{(m)}_t(\mathbf{n}^e_t) = \rho\left(\mathbf{n}^e_t \middle| \mathbf{y}_t, m_t\right)$, is then Gaussian. Its parameters can be found as in appendix A.1.3.

In the maximisation step, the noise parameters are set to the empirical mean and covariance of ρ. This requires summing over all time slices and components and weighting the distributions by the posterior component–time occupancy $\gamma^{(m)}_t$ defined in (2.31a):

$$\boldsymbol{\mu}^{e(k)}_n := \mathcal{E}_\rho\{\mathbf{n}^e\} = \frac{1}{\int \tilde{p}(\mathcal{Y}) T_{\mathcal{Y}} \, d\mathcal{Y}} \int \tilde{p}(\mathcal{Y}) \sum_{t=1}^{T_{\mathcal{Y}}} \sum_m \gamma^{(m)}_t \mathcal{E}_{\rho^{(m)}_t}\{\mathbf{n}^e\} d\mathcal{Y}; \tag{5.64a}$$

$$\begin{aligned}\boldsymbol{\Sigma}_n^{e(k)} &:= \mathcal{E}_\rho\{\mathbf{n}^e\mathbf{n}^{eT}\} - \boldsymbol{\mu}_n^{e(k)}\boldsymbol{\mu}_n^{e(k)T} \\ &= \left(\frac{1}{\int \tilde{p}(\mathcal{Y})T_\mathcal{Y}d\mathcal{Y}}\int \tilde{p}(\mathcal{Y})\sum_{t=1}^{T_\mathcal{Y}}\sum_m \gamma_t^{(m)}\mathcal{E}_{\rho_t^{(m)}}\{\mathbf{n}^e\mathbf{n}^{eT}\}d\mathcal{Y}\right) - \boldsymbol{\mu}_n^{e(k)}\boldsymbol{\mu}_n^{e(k)T}.\end{aligned} \tag{5.64b}$$

Additionally, the convolutional noise should be estimated, and the structure of the noise model must be constrained. This work does not investigate this. Instead, the extended noise model will derive from a noise model estimated with standard VTS. By using the same noise estimates for both VTS and EVTS, only differences in the compensation process are examined, rather than any differences in the noise estimation process. It should be emphasised that the results presented for EVTS may a slight underestimate of the possible performance if a fully integrated noise estimate was used.

5.5.3 *Phase factor distribution*

The phase factor $\boldsymbol{\alpha}^e$ is assumed to have independent dimensions (within and between time instances). It is

$$\boldsymbol{\alpha}^e = \begin{bmatrix}\boldsymbol{\alpha}_{t-1}^s \\ \boldsymbol{\alpha}_t^s \\ \boldsymbol{\alpha}_{t+1}^s\end{bmatrix}, \tag{5.65}$$

where every dimension is independent and distributed as in 4.18:

$$p(\alpha_i) \propto \begin{cases}\mathcal{N}(\alpha_i;\, 0,\, \sigma_{\alpha,i}^2) & \alpha_i \in [-1, +1]; \\ 0 & \text{otherwise.}\end{cases} \tag{5.66}$$

5.6 Summary

This chapter has described the first contribution of this thesis. It has extended model compensation methods from chapter 4 to produce full-covariance compensation.

The most important insight is that full covariance matrices require higher-quality compensation for dynamics. This claim will be validated in section 8.1.1.1. Section 5.2

has therefore shown how from a distribution over the statics in a window (an *extended* feature vector) a distribution over dynamics can be found. This uses the same linear projection that feature extraction uses. Section 5.3 has introduced instances of compensation methods that model the effect of the noise separately for each time instance in the extended feature vector. They are therefore capable of generating accurate full covariance matrices. Section 5.5.2 has shown how to find an extended noise model from a standard noise model, so that model compensation with extended feature vectors needs as little adaptation data as standard model compensation.

The same principle of compensation can apply to a base class at once: section 5.4 has detailed how joint uncertainty decoding can be extended. The choice of the number of base classes gives a trade-off between speed and accuracy. However, decoding with full covariance Gaussians is still slow. The next chapter will present approximations to deal with this.

Chapter 6

Predictive transformations

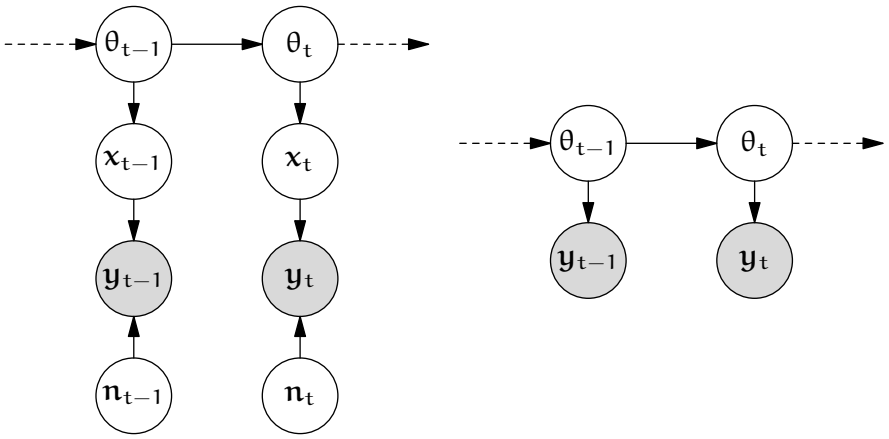
This chapter will describe the second contribution of this thesis.[1]

The previous chapter has introduced methods of model compensation that need little data to train, but find full-covariance model compensation, which it is slow to decode with. This chapter will introduce *predictive methods*, which approximate a distribution predicted with one model with another, differently parameterised model. For example, this enables fast transformations to be trained from full-covariance compensation.

Section 6.1 will formalise predictive methods. They can combine the advantages of one method and those of another. For example, section 6.2 will introduce predictive linear transformations, which like their adaptive versions (which were discussed in sections 3.2 and 3.3) allow fast decoding. The interesting aspect for this work is that they can model correlations without the computational burden of full-covariance

[1] Though van Dalen (2007); Gales and van Dalen (2007) introduced predictive linear transformations, they derived the predicted statistics by intuition. This chapter introduces a rigorous framework for predictive transformation. Additionally, this chapter will introduce a number of new schemes. Front-end CMLLR schemes, joint work with Federico Flego, in section 6.4 were published as van Dalen *et al.* (2009). Section 6.3.5 will newly introduce predictive HLDA.

Since the introduction of predictive linear transformations, a number of variants of specific transformations have been proposed. Xu *et al.* (2009) used joint uncertainty decoding in a different feature space for estimating predictive CMLLR transforms. This chapter will give the theoretical underpinnings for this. Another interesting line of work combines predicted statistics with statistics from data (Flego and Gales 2009; Breslin *et al.* 2010)



(a) The predicted noise-corrupted speech, from figure 4.5 on page 72.

(b) Compensated hidden Markov model, from figure 4.7 on page 75.

Figure 6.1 *Model compensation as a predictive method: the predicted corrupted speech is approximated with a standard* HMM.

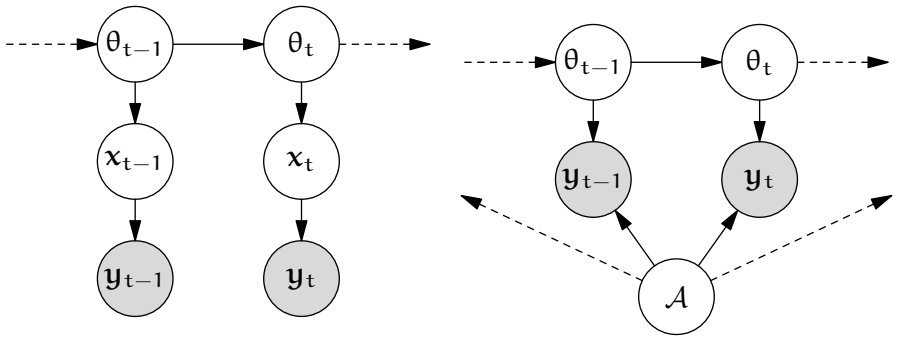
Gaussians. Section 6.3 will use full-covariance joint uncertainty decoding (discussed in section 5.4) as the predicted distribution.

Another use of the predictive framework is for fast feature transformations. Unlike conventional feature transformations for noise-robustness, which aim to reconstruct the clean speech, the methods that section 6.4 will introduce aim to model the corrupted speech distribution.

6.1 Approximating the predicted distribution

Predictive methods are methods that train parameterised distributions from distributions predicted with another model. This is useful when it is impossible or impractical to use the former model. This section will introduce the general framework for predictive methods. It will give a formalisation of predictive transformations as minimising the KL divergence between the predicted distributions and the model set.

For example, section 4.4 has introduced model compensation, which approximates the predicted model for the corrupted speech. Figure 6.1a shows that predicted



(a) Predicted corrupted speech with joint uncertainty decoding.

(b) A hidden Markov model with a linear transformation.

Figure 6.2 *Predictive linear transformations: a JUD-compensated HMM is approximated with a linear transformation.*

model, which results in an integral (in (4.24)) that has no closed form. Figure 6.1b shows the approximated model. The abstract idea of interpreting model compensation as a predictive method was proposed in Gales (1998*b*).

In many cases, it is possible to sample from the predicted distribution, and train the approximated distribution from those samples. Indeed, extended DPMC (see section 5.3.1) drops out when the parameters of all Gaussians of the approximate distribution are trained separately, and extended iterative DPMC when the state-conditional mixtures of Gaussians are trained. These schemes are slow but in the limit yield the optimal parameters for their parameterisations.

Faster predictive methods, such as speech recogniser transformations, are possible. Section 6.2 will introduce an application of the predictive framework that finds a speech recogniser transformation that approximates another distribution. Predictive transformations could be estimated with Monte Carlo. However, this would negate the reason for estimating transformations, which is speed. Figure 6.2a shows a graphical model for joint uncertainty decoding, a fast compensation method discussed in section 4.4.3. Section 6.3 will discuss how to find an approximation to this with linear transformations (discussed in 3.2), depicted in figure 6.2b. This combines the advantages of joint uncertainty decoding, which needs little data to adapt but can generate

full-covariance compensation (see section 5.4), and linear transformations, which are fast in decoding.

The rest of this section will discuss the general framework of predictive transformations. They are estimated by minimising the KL divergence between the predicted distributions and the target transformation of the model set. Section 6.1.1 will introduce a form of predicted distribution that predicts and approximates distributions per component (normally, a Gaussian). This is the form that most model compensation methods presented in the chapter use, and will be the form used in the rest of this work. However, it is also possible to approximate the distributions per sub-phone state. The formal derivation of this will be introduced in section 6.1.2. Iterative DPMC, introduced in section 4.4.1, is the only method in this thesis that applies this.

6.1.1 *Per component*

The idea of predictive transformations is that they are analogous to adaptive transformations, but are estimated on predicted statistics rather than statistics from data. Section 2.3.2.1 wrote the maximisation step of expectation–maximisation as minimising the KL divergence between the inferred distribution of the complete data and the model to be trained. Here, the idea is to minimise a KL divergence in the same way, but between the predicted distribution and the model.

The predicted distribution over the hidden variables $\mathcal{U}$ and test data observations $\mathcal{Y}$ will be written

$$p(\mathcal{U}, \mathcal{Y}) = p(\mathcal{U})p(\mathcal{Y}|\mathcal{U})\,. \tag{6.1}$$

For speech recognition, $\mathcal{U}$ is the component sequence $\{m_t\}$, which no transformation method in this thesis changes. $p(\mathcal{Y}|\mathcal{U})$ consists of the predicted component-conditional distributions $p^{(m)}$. This replaces the empirical distribution from data and the inferred distribution over the hidden parameters in (2.24) on page 32, which was $p(\mathcal{U}, \mathcal{X}) = \tilde{p}(\mathcal{X})\rho(\mathcal{U}|\mathcal{X})$.

The distribution to be trained is $q_{\mathcal{U}\mathcal{Y}}$, which for speech recognition factorises into a distribution over the hidden variables $q_{\mathcal{U}}$, and one over the observed variables given the hidden ones $q_{\mathcal{Y}|\mathcal{U}}$ (as in (2.26)):

$$q_{\mathcal{U}\mathcal{Y}}(\mathcal{U},\mathcal{Y}) = q_{\mathcal{U}}(\mathcal{U})\, q_{\mathcal{Y}|\mathcal{U}}(\mathcal{Y}|\mathcal{U})\,. \tag{6.2}$$

Of these, only $q_{\mathcal{Y}|\mathcal{U}}$, the component-conditional distributions $q^{(m)}$, will be trained, so that

$$\begin{aligned}
\arg\min_{q_{\mathcal{U}\mathcal{Y}}} \mathcal{KL}(p\|q_{\mathcal{U}\mathcal{Y}}) &= \arg\min_{q_{\mathcal{U}\mathcal{Y}}} \int\!\!\int p(\mathcal{U},\mathcal{Y}) \log \frac{p(\mathcal{U},\mathcal{Y})}{q_{\mathcal{U}\mathcal{Y}}(\mathcal{U},\mathcal{Y})}\, d\mathcal{Y}\, d\mathcal{U} \\
&= \arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \int\!\!\int p(\mathcal{U})p(\mathcal{Y}|\mathcal{U}) \log \frac{p(\mathcal{U})p(\mathcal{Y}|\mathcal{U})}{q_{\mathcal{U}}(\mathcal{U})\, q_{\mathcal{Y}|\mathcal{U}}(\mathcal{Y}|\mathcal{U})}\, d\mathcal{Y}\, d\mathcal{U} \\
&= \arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \int\!\!\int p(\mathcal{U})p(\mathcal{Y}|\mathcal{U}) \log \frac{p(\mathcal{Y}|\mathcal{U})}{q_{\mathcal{Y}|\mathcal{U}}(\mathcal{Y}|\mathcal{U})}\, d\mathcal{Y}\, d\mathcal{U}.
\end{aligned} \tag{6.3}$$

In this case, the minimisation is performed per component distribution, so that $\mathcal{U}$ represents just the component identity m, and $\mathcal{Y}$ just the observation $\mathbf{y}$ it generates. The output distribution $q_{\mathcal{Y}|\mathcal{U}}$ factorises per time step, so that the expression becomes

$$\begin{aligned}
\arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \mathcal{KL}(p\|q_{\mathcal{U}\mathcal{Y}}) &= \arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \int \sum_m p(m)\, p^{(m)}(\mathbf{y}) \log \frac{p^{(m)}(\mathbf{y})}{q^{(m)}(\mathbf{y})}\, d\mathbf{y} \\
&= \arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \sum_m p(m)\, \mathcal{KL}\big(p^{(m)} \big\| q^{(m)}\big).
\end{aligned} \tag{6.4}$$

A maximum-likelihood estimate of the prior distribution on the component can be found from the training data. The expectation step of expectation–maximisation gives the total component occupancy $\gamma^{(m)}$, which in (2.31b) was defined as

$$\gamma^{(m)} \triangleq \int \tilde{p}(\mathcal{X}) \sum_{t=1}^{T_{\mathcal{X}}} \int \rho(\mathcal{U}|\mathcal{X})\, 1(m_t = m)\, d\mathcal{U}\, d\mathcal{X}. \tag{6.5}$$

The maximum-likelihood estimated prior of the component is

$$p(m) := \frac{\gamma^{(m)}}{\sum_{m'} \gamma^{(m')}}. \tag{6.6}$$

This can straightforwardly be substituted into (6.4). However, the normalisation term $1/\sum_{m'} \gamma^{(m')}$ does not make a difference for the minimisation. Therefore, and be-

cause without the normalisation the minimand turns out to be easier to relate to transformations trained on data, the minimisation in (6.4) will be written

$$\arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \mathcal{KL}(p\|q_{\mathcal{U}\mathcal{Y}}) = \arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \sum_m \gamma^{(m)} \mathcal{KL}\big(p^{(m)}\big\|q^{(m)}\big). \tag{6.7}$$

The objective of predictive transforms, then, turns out to be to minimise the occupancy-weighted KL divergence between the predicted component distributions and the distributions used for decoding. For the model compensation methods that were discussed in the previous chapters, the parameters of each component distribution $q^{(m)}$ were independent, so that the optimisation (in (4.28)) was separate for each component. For the predictive linear transformations that section 6.2 will introduce, however, the component distributions $q^{(m)}$ cannot be optimised separately, because they share parameters. The weighting by training data occupancy $\gamma^{(m)}$ therefore is necessary.

The per-component KL divergence consists of the entropy of $p^{(m)}$ and the cross-entropy of $p^{(m)}$ and $q^{(m)}$ (see appendix A.2), of which only the cross-entropy can be optimised. To solve (6.7), it therefore suffices to find

$$\begin{aligned}\arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \mathcal{KL}(p\|q_{\mathcal{U}\mathcal{Y}}) &= \arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \sum_m \gamma^{(m)} \mathcal{H}\big(p^{(m)}\big\|q^{(m)}\big) \\ &= \arg\max_{q_{\mathcal{Y}|\mathcal{U}}} \sum_m \gamma^{(m)} \int p^{(m)}(\mathbf{y}) \log q^{(m)}(\mathbf{y}) \mathrm{d}\mathbf{y}. \end{aligned} \tag{6.8}$$

The minimisation of the cross-entropy turns into a maximisation of something that can be interpreted as the expected log-likelihood.

6.1.2 *Per sub-phone state*

The discussion so far has assumed that the hidden variables are components m. However, it is also possible to match the decoding process more closely. Section 2.4 has discussed how Viterbi decoding finds a state sequence Θ, and marginalises out over the component sequences M. Finding a transformed distribution that minimises the per-component KL divergence is often sub-optimal compared to minimising a per-state KL divergence. The following first presents the expression for the per-state op-

timisation, and then a method to approximate this for the specific case of two mixture models.

By following the derivation in section 6.1.1, but using the sub-phone state sequence $\Theta = \{\theta_t\}$ as the hidden variables $\mathcal{U}$ and per-sub-phone output distributions for the predictions and approximations, $p_{\mathcal{Y}|\mathcal{U}} = \left\{p^{(\theta)}\right\}$, $q_{\mathcal{Y}|\mathcal{U}} = \left\{q^{(\theta)}\right\}$, the expression to optimise becomes (analogous to (6.7)):

$$\arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \mathcal{KL}(p \| q_{\mathcal{U}\mathcal{Y}}) = \arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \sum_{\theta} \gamma^{(\theta)} \mathcal{KL}\left(p^{(\theta)} \middle\| q^{(\theta)}\right), \tag{6.9}$$

where $\gamma^{(\theta)}$ is the sub-phone occupancy, $p^{(\theta)}$ is the predicted distribution for state θ, and $q^{(\theta)}$ the transformed speech recogniser model for that state.

For the case where $p^{(\theta)}$ and $q^{(\theta)}$ are mixture models, this expression normally has no analytic solution. However, assuming that it is possible to compute and improve the divergence between pairs of components of the mixtures, the divergence between pairs of components of the mixtures, the divergence between sub-phone state pairs can be improved starting from the per-component one (Yu 2006; Dognin *et al.* 2009). This uses the upper bound on the cross-entropy presented in section A.2.2. After finding the tightest upper bound, $q_{\mathcal{Y}|\mathcal{U}}$ is set to improve the upper bound by setting variational parameters. This process can be repeated a number of times to iteratively improve the bound.

To apply this, (6.9) must be written as a minimisation of only one half of the KL divergence, the cross-entropy $\mathcal{H}\left(p^{(\theta)} \middle\| q^{(\theta)}\right)$, which is analogous to (6.8):

$$\arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \mathcal{KL}(p \| q_{\mathcal{U}\mathcal{Y}}) = \arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \sum_{\theta} \gamma^{(\theta)} \mathcal{H}\left(p^{(\theta)} \middle\| q^{(\theta)}\right). \tag{6.10}$$

This is the minimisation that iterative DPMC approximates with a sampling method (in (4.33)). The following will optimise an upper bound on this. For simplicity of notation, it will assume that the components of the mixture distribution $p^{(\theta)}$ are not shared with other distributions $p^{(\theta')}$, and similar for components of $q^{(\theta)}$. The algorithm is straightforward to extend to the general case.

The distributions $p^{(\theta)}$ and $q^{(\theta)}$ are assumed mixture distributions with component sets $\Omega^{(\theta)}$ and $\widetilde{\Omega}^{(\theta)}$:

$$p^{(\theta)}(\mathbf{y}) = \sum_{m \in \Omega^{(\theta)}} \pi_m^{(\theta)} p^{(m)}(\mathbf{y}); \qquad q^{(\theta)}(\mathbf{y}) = \sum_{n \in \widetilde{\Omega}^{(\theta)}} \omega_n^{(\theta)} q^{(n)}(\mathbf{y}). \tag{6.11}$$

The sub-phone occupancy can be written as the sum of the occupancies of the components:

$$\gamma^{(\theta)} = \sum_{m \in \Omega^{(\theta)}} \gamma^{(m)}. \tag{6.12}$$

The algorithm for finding the upper bound to the cross-entropy between two mixtures (Yu 2006; Hershey and Olsen 2007) is discussed in appendix A.2.2. The algorithm optimises a probabilistic mapping between the components from one mixture and components from the other mixture. This mapping is represented by variational parameters $\phi_n^{(m)}$, with (repeated from (A.16))

$$\sum_n \phi_n^{(m)} = 1, \qquad \phi_n^{(m)} \geq 0. \tag{6.13}$$

Appendix A.2.2 shows that the cross-entropy between mixtures can be upper-bounded by a function of the sub-weights and the KL divergence between all component pairs. The expression is given in (A.17b). Applied to one sub-phone state pair it is

$$\begin{aligned} \mathcal{H}\big(p^{(\theta)} \big\| q^{(\theta)}\big) &\leq \sum_{m \in \Omega^{(\theta)}} \sum_{n \in \widetilde{\Omega}^{(\theta)}} \pi_m^{(\theta)} \phi_n^{(m)} \left(\mathcal{H}\big(p^{(m)} \big\| q^{(n)}\big) + \log \frac{\phi_n^{(m)}}{\omega_n^{(\theta)}} \right) \\ &\triangleq \mathcal{F}\big(p^{(\theta)}, q^{(\theta)}, \boldsymbol{\phi}\big). \end{aligned} \tag{6.14}$$

To find an upper bound on the KL divergence of the whole model set, this can be substituted in the expression optimised in (6.10):

$$\begin{aligned} &\sum_\theta \gamma^{(\theta)} \mathcal{H}\big(p^{(\theta)} \big\| q^{(\theta)}\big) \\ &\leq \sum_\theta \gamma^{(\theta)} \left(\sum_{m \in \Omega^{(\theta)}} \sum_{n \in \widetilde{\Omega}^{(\theta)}} \pi_m^{(\theta)} \phi_n^{(m)} \left(\mathcal{H}\big(p^{(m)} \big\| q^{(n)}\big) + \log \frac{\phi_n^{(m)}}{\omega_n^{(\theta)}} \right) \right) \\ &\triangleq \mathcal{F}\big(p_{\mathcal{Y}|\mathcal{U}}, q_{\mathcal{Y}|\mathcal{U}}, \boldsymbol{\phi}\big). \end{aligned} \tag{6.15}$$

function OPTIMISE-PREDICTIVE-DISTRIBUTION($p_{\mathcal{Y}|\mathcal{U}}$)
  Initialise $q_{\mathcal{Y}|\mathcal{U}}$
  repeat
    for all m, n **do**
      $\phi_n^{(m)} \leftarrow \frac{\omega_n^{(\theta)} \exp\left(-\mathcal{H}(p^{(m)} \| q^{(n)})\right)}{\sum_{n'} \omega^{(n')} \exp\left(-\mathcal{H}(p^{(m)} \| q^{(n')})\right)}$
    $q_{\mathcal{Y}|\mathcal{U}} \leftarrow \arg\max_{q_{\mathcal{Y}|\mathcal{U}}} \mathcal{F}\left(p_{\mathcal{Y}|\mathcal{U}}, q_{\mathcal{Y}|\mathcal{U}}, \boldsymbol{\phi}\right)$
  until convergence
  return $q_{\mathcal{Y}|\mathcal{U}}$

Algorithm 3 *Optimising the upper bound to KL divergence to a predicted distribution for mixture models.*

Optimising the state-for-state KL divergence works as in algorithm 3. The initialisation for approximate distribution $q_{\mathcal{Y}|\mathcal{U}}$ could be the component-for-component optimum, or another setting. The variational parameters are then optimised to tighten the upper bound as in (A.20), after which the upper bound is improved by setting $q_{\mathcal{Y}|\mathcal{U}}$. Both these steps are guaranteed not to increase the cross-entropy, so iterating over them finds a local optimum.

Since for most of the methods in the rest of this thesis, the components of the predicted distribution and its approximation both derive from the same clean speech component, the component-for-component optimisation from the previous section will be used most. It is interesting to see how the component-for-component optimisation relates to the state-for-state optimisation. It is straightforward to set parameters $\phi_n^{(m)}$ so that (6.15) is equal to the component-for-component KL divergence in (6.7) (repeated from (A.21)):

$$\phi_n^{(m)} = \begin{cases} 1, & m = n; \\ 0, & m \neq n. \end{cases} \tag{6.16}$$

Since the inequality in (6.15) still holds under this setting, optimising the component-for-component KL divergence gives is an upper bound on the optimal state-for-state KL divergence with variational parameters $\phi_n^{(m)}$. It is called the *matched-pair bound* (see section A.2.2 and Hershey and Olsen 2007). This means that even when optimising

the component-for-component KL divergence, this optimises an upper bound on the state-for-state KL divergence, which is consistent with the decoding process.

6.2 Predictive linear transformations

Adaptive linear transformations have been discussed in section 3.2. They do not have a model of the environment, but restrict the number of parameters that need to be trained, compared to training a full speech recogniser, by estimating linear transformations. However, they still require more data than methods for noise-robustness. However, decoding is often hardly slowed down at all. For examples, CMLLR (section 3.2.1) transforms every observation feature by a small number of transformations, which is fast. It then feeds the differently-transformed feature vectors to different groups of components.

No such tricks are possible for model compensation methods. Methods that find Gaussian compensation, like standard VTS and extended VTS, apply a different transformation to each component. Converting them into fewer linear transformations, however, could leverage the fast adaptation of extended VTS, and the fast decoding of linear transformations.

As section 3.1 has discussed, adaptation uses expectation–maximisation. The maximisation step is equivalent to minimising the KL divergence between the empirical distribution and the modelled distribution. The framework of predictive transformations, introduced in section 6.1, approximates one distribution by another, also minimising the KL divergence between them. That the optimisation in both cases is similar makes converting adaptive transformations into predictive transformations relatively straightforward. The derivation of, for example, predictive CMLLR, runs parallel to the derivation of standard CMLLR. The main difference will be that the statistics from data will be replaced by predicted statistics.

Predictive transformations approximate a predicted distribution p. The approximate distribution $q_{\mathcal{Y}|\mathcal{U}}$ over observed variables $\mathcal{Y}$ given hidden variables $\mathcal{U}$ is set to

(repeated from (6.7) and (6.8)):

$$\begin{aligned}\arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \mathcal{KL}(p\|q_{\mathcal{U}\mathcal{Y}}) &= \arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \sum_m \gamma^{(m)} \mathcal{KL}\left(p^{(m)}\middle\|q^{(m)}\right) \\ &= \arg\min_{q_{\mathcal{Y}|\mathcal{U}}} \sum_m \gamma^{(m)} \mathcal{H}\left(p^{(m)}\middle\|q^{(m)}\right) \\ &= \arg\max_{q_{\mathcal{Y}|\mathcal{U}}} \sum_m \gamma^{(m)} \int p^{(m)}(\mathbf{y}) \log q^{(m)}(\mathbf{y}) d\mathbf{y}, \end{aligned} \tag{6.17}$$

where $\gamma^{(m)}$ is the total occupancy on the training data for component m. Linear transformations are defined by a set of transformations $\mathcal{A}$, one of which acts on each component. The output distribution of transformed component m is written $q^{(m)}(\mathbf{y}|\mathcal{A})$. The optimisation in (6.17) then becomes

$$\begin{aligned}\mathcal{A} &:= \arg\min_{\mathcal{A}} \sum_m \gamma^{(m)} \mathcal{KL}\left(p^{(m)}\middle\|q^{(m)}\right) \\ &= \arg\max_{\mathcal{A}} \sum_m \gamma^{(m)} \int p^{(m)}(\mathbf{y}) \log q^{(m)}(\mathbf{y}|\mathcal{A}) d\mathbf{y}. \end{aligned} \tag{6.18}$$

This expression is similar to the optimisation for adaptive linear transformations in (3.2):

$$\mathcal{A}^{(k)} := \arg\max_{\mathcal{A}} \int \tilde{p}(\mathcal{Y}) \sum_m \sum_{t=1}^{T_{\mathcal{Y}}} \gamma_t^{(m)} \log q^{(m)}(\mathbf{y}_t|\mathcal{A}) d\mathcal{Y}. \tag{6.19}$$

Alternatively, the per-state KL divergence could be optimised using the technique in section 6.1.2. This would require interleaving the optimisation of the variational parameters of the upper bound on the KL divergence per state, and the optimisation of the transformation. Since the upper bound (in (6.15)) is a linear combination of the per-component cross-entropies, optimising the transformation still has the form in (6.18). For clarity, therefore, the following will use that form.

Both predictive CMLLR and predictive covariance MLLR derive from (6.18) in the same way as their adaptive versions derive from (6.19). It will turn out the only change is in the statistics, which are predicted rather than gathered from data. This is convenient, because the procedure for estimating the transformations presented in chapter 3.2 can be re-used. It is also satisfying, because the predicted statistics that drop out correspond to the intuitive expressions. They are the same as Gales and van Dalen (2007) derived by intuition.

One difference between the adaptive linear transformations and the predictive versions is in the regression classes. Adaptation needs to carefully control for the available amount of data. One tool for this is a regression class tree that expands nodes as long as there is enough data to train the corresponding transformation robustly. For predictive transformations, on the other hand, there is a predicted distribution, parameterised for the instantiations in this thesis. This corresponds to an infinite amount of data, so that data sparsity is not an issue.

An interesting case is when there is only one component in a base class. If additionally the predicted distributions are Gaussian, then the algorithms for predictive CMLLR and predictive semi-tied covariance matrices will find a transformation that sets $q^{(m)}$ exactly equal to $p^{(m)}$. The choice of the number of base classes therefore gives a trade-off between speed and accuracy.

Since each component is assigned to one base class only, the optimisation expressions for each base class are independent. All of the derivations in the next sections will therefore simplify notation by assuming only one base class, and summing over all components. To convert these into expressions that do use base classes, the sums over components should only by over components in the base class that the transformation is estimated for.

6.2.1 *Predictive* CMLLR

Predictive CMLLR (PCMLLR) uses the exact same form of transformation that CMLLR uses. It is called "constrained" because the linear transformation of the means and covariances are equal. There is also a bias on the mean. As section 3.2.1 has shown, this can be written with the inverse transformation, which then works on the feature vector. The likelihood computation becomes (as in (3.4b))

$$q^{(m)}(\mathbf{y}|\mathcal{A}) = |\mathbf{A}| \cdot \mathcal{N}\left(\mathbf{A}\mathbf{y} + \mathbf{b};\, \boldsymbol{\mu}_x^{(m)},\, \boldsymbol{\Sigma}_x^{(m)}\right), \tag{6.20}$$

where $\boldsymbol{\mu}_x^{(m)}$ and $\boldsymbol{\Sigma}_x^{(m)}$ are Gaussian parameters for the clean speech. Though for clarity of notation it will not be written explicitly here, there is normally a set of R transform-

ations $\left\{\boldsymbol{A}^{(r)}, \mathbf{b}^{(r)}\right\}$, one for each base class (in the adaptive case, regression class). Each component is assigned to one base class. A fast implementation can therefore transform each observation vector $\mathbf{y}_t$ into R transformed versions $\boldsymbol{A}^{(r)}\mathbf{y}_t + \mathbf{b}^{(r)}$ and pass each component the appropriately transformed versions. The determinant $\left|\boldsymbol{A}^{(r)}\right|$ in (6.20) can be precomputed. This makes decoding fast, both for the adaptive and predictive versions of CMLLR.

The algorithm for predictive CMLLR turns out to be the same as that for adaptive CMLLR. Appendix B.1.2 derives this. Both are expressed in terms of the same statistics, but the difference is in how these statistics are acquired. For the predictive version they are

$$\gamma \triangleq \sum_m \gamma^{(m)}; \tag{6.21a}$$

$$\mathbf{k}^{(i)} \triangleq \sum_m \frac{\gamma^{(m)}\mu_{x,i}^{(m)}}{\sigma_{x,ii}^{(m)}} \left[\, \mathcal{E}_{p^{(m)}}\left\{\mathbf{y}^{\mathsf{T}}\right\} \;\; 1 \,\right]; \tag{6.21b}$$

$$\mathbf{G}^{(i)} \triangleq \sum_m \frac{\gamma^{(m)}}{\sigma_{x,ii}^{(m)}} \begin{bmatrix} \mathcal{E}_{p^{(m)}}\left\{\mathbf{y}\mathbf{y}^{\mathsf{T}}\right\} & \mathcal{E}_{p^{(m)}}\{\mathbf{y}\} \\ \mathcal{E}_{p^{(m)}}\left\{\mathbf{y}^{\mathsf{T}}\right\} & 1 \end{bmatrix}. \tag{6.21c}$$

The form of the statistics for predictive CMLLR is intuitively related to the form of the statistics for the adaptive version, in (3.5). γ is the total occupancy, which for adaptive CMLLR is found from the distribution over the hidden variables, and here is derived from the clean training data. $\mathbf{k}^{(i)}$ is a function of the expected value of the predicted distribution; its equivalent in (3.5b) can be viewed as the mean observation vector for component m under the empirical distribution, and similarly for $\mathbf{G}^{(i)}$.

6.2.2 *Predictive covariance* MLLR

Covariance MLLR (see section 3.2.2) is a technique that tries to find the best covariance transformation to model the data. Since the overall aim of this chapter is to find transformations that help model correlations without the decoding cost of full covariances, a predictive variant of covariance MLLR should be useful. The derivation of predictive covariance MLLR follows the same structure as that of predictive CMLLR,

in the previous section. The KL divergence between the predicted distributions and the transformed speech recogniser is minimised. This will result in the same expression as for adaptive covariance MLLR, but with the statistics replaced by the predicted equivalents.

The transformed likelihood is exactly the one in (3.6b):

$$q^{(m)}(\mathbf{y}|\mathcal{A}) = |\mathbf{A}| \cdot \mathcal{N}\big(\mathbf{A}\mathbf{y};\, \mathbf{A}\boldsymbol{\mu}_x^{(m)},\, \boldsymbol{\Sigma}_x^{(m)}\big). \tag{6.22}$$

As explained in section 3.2.2, this expression transforms feature and means, because that makes decoding faster than transforming the covariance with the inverse.

The derivation of predictive covariance MLLR is in appendix B.2.2. As for predictive CMLLR, the only change is in the statistics, which again are intuitively related to those for the adaptive version in (3.7):

$$\gamma \triangleq \sum_m \gamma^{(m)}; \tag{6.23a}$$

$$\mathbf{G}^{(i)} \triangleq \sum_m \frac{\gamma^{(m)}}{\sigma_{x,ii}^{(m)}} \mathcal{E}_{p^{(m)}}\Big\{\big(\mathbf{y} - \boldsymbol{\mu}_x^{(m)}\big)\big(\mathbf{y} - \boldsymbol{\mu}_x^{(m)}\big)^{\mathsf{T}}\Big\}. \tag{6.23b}$$

6.2.3 *Predictive semi-tied covariance matrices*

Predictive covariance MLLR does not change the mean, nor the covariance matrices beyond the linear transformation it applies. However, it is possible to adjust the covariance matrices as well as the linear transformation that is applied to them. Semi-tied covariance matrices, discussed in section 3.3.1, do exactly that, for the training data. It is possible to train the same transformation on predicted statistics. The derivation is analogous to the one for standard semi-tied covariance matrices.

The likelihood expression for predictive semi-tied covariance matrices is analogous to (3.12b):

$$q^{(m)}(\mathbf{y}) = |\mathbf{A}| \cdot \mathcal{N}\big(\mathbf{A}\mathbf{y};\, \mathbf{A}\boldsymbol{\mu}_y^{(m)},\, \widetilde{\boldsymbol{\Sigma}}_{y,\mathrm{diag}}^{(m)}\big). \tag{6.24}$$

This expression is similar to the one for predictive covariance MLLR, in (3.6b). It also uses a transformation $\mathbf{A}$ to find a feature space in which diagonal covariance matrices are used. However, there are two important differences. First, the mean is $\boldsymbol{\mu}_y^{(m)}$ (instead of $\boldsymbol{\mu}_x^{(m)}$): it is set to the mean of the predicted corrupted speech distributions. This is not explicitly necessary for standard semi-tied covariance matrices, since they are normally trained on the clean training data. The second difference is that, just like with normal semi-tied covariance matrices, the component covariance is also re-estimated, and here indicated by $\widetilde{\boldsymbol{\Sigma}}_{y,\text{diag}}^{(m)}$. For standard semi-tied covariance matrices, the reason covariances could be re-estimated was that the training data was used (rather than test data, as for adaptation transformations). Without data sparsity, over-training was not a problem. For the predictive version, the training data statistics will be replaced by predicted statistics, so that over-training is again not a problem.

The three types of parameters to be estimated are $\boldsymbol{\mu}_y^{(m)}$, $\boldsymbol{\Sigma}_y^{(m)}$, and $\mathbf{A}$. The means are straightforwardly estimated. Even when it is transformed by $\mathbf{A}$, $\mathbf{y}$ is transformed by the same matrix, so that $\mathbf{A}\boldsymbol{\mu}_y$ is still the mean, in the transformed space. The covariances, on the other hand, are diagonal, and if $\mathbf{A}$ changes, then they are sub-optimal, and vice versa. They will therefore, like for standard semi-tied covariance matrices, be estimated in an iterative fashion. Every step is guaranteed to not decrease the KL divergence, so that the algorithm finds a local optimum.

The full derivation is in B.3.2. The statistics required are (repeated from (B.42))

$$\boldsymbol{W}^{(m)} \triangleq \mathcal{E}_{p^{(m)}}\left\{\left(\mathbf{y}-\boldsymbol{\mu}_y^{(m)}\right)\left(\mathbf{y}-\boldsymbol{\mu}_y^{(m)}\right)^{\mathsf{T}}\right\}; \tag{6.25a}$$

$$\gamma \triangleq \sum_m \gamma^{(m)}; \tag{6.25b}$$

$$\mathbf{G}^{(i)} \triangleq \sum_m \frac{\gamma^{(m)}}{\widetilde{\sigma}_{y,ii}^{(m)}} \boldsymbol{W}^{(m)}. \tag{6.25c}$$

Given the statistics, the process of estimating the component parameters and the transformations is basically the same as for standard semi-tied covariance matrices, in section 3.3.1. In addition, the means are first estimated. The complete process is given in algorithm 4.

function PREDICTIVE-SEMI-TIED-COVARIANCE-MATRICES($\{p^{(m)}, \gamma^{(m)}\}, \gamma$)
  for all components m **do**
    $\mu_y^{(m)} \leftarrow \mathcal{E}_{p^{(m)}}\{y\}$
    $W^{(m)} \leftarrow \mathcal{E}_{p^{(m)}}\{(y - \mu_y^{(m)})(y - \mu_y^{(m)})^T\}$
    $\widetilde{\Sigma}_{y,\text{diag}}^{(m)} \leftarrow \text{diag}(W^{(m)})$
  $A \leftarrow I$
  repeat
    $G^{(i)} \leftarrow \sum_m \frac{\gamma^{(m)}}{\widetilde{\sigma}_{y,ii}^{(m)}} W^{(m)}$
    $A \leftarrow$ ESTIMATE-COVARIANCE-MLLR($\gamma, G^{(i)}$)
    for all components m **do**
      $\widetilde{\Sigma}_{y,\text{diag}}^{(m)} \leftarrow \text{diag}(AW^{(m)}A^T)$
  until convergence
  return $\{\mu_y^{(m)}, \widetilde{\Sigma}_{y,\text{diag}}^{(m)}\}, A$

Algorithm 4 *Estimating predictive semi-tied covariance matrices.*

This scheme is computationally expensive, because it alternates over updating A and $\widetilde{\Sigma}_{y,\text{diag}}^{(m)}$, and updating A requires iterating over its rows. An alternative is to update only A, by stopping after the call to ESTIMATE-COVARIANCE-MLLR in algorithm 4. The form of the likelihood is then the same as in (6.24), but the covariance matrices on the models are diagonalised predicted covariances in the original feature space, unlike covariance MLLR, where the original covariance matrices are retained. A is optimised for this feature space. This form will be referred to as "half-iteration predictive semi-tied".

6.3 Correlation modelling for noise

Section 6.2 has presented predictive linear transform agnostic to the form of the predicted distribution. Indeed, they can be trained from any distribution that yields the required statistics. This work applies predictive linear transformations to methods for noise-robustness. However, since the introduction of the general framework (Gales and van Dalen 2007), other forms of predictive transformations have been proposed,

like VTLN (Breslin *et al.* 2010).

This section will confine itself to estimating predictive linear transformations from joint uncertainty decoding as discussed in sections 4.4.3 and 5.4. The advantage of the form of distribution that joint uncertainty decoding predicts is that the components are Gaussian-distributed, and that it uses base classes.

The statistics that the predictive methods from the previous chapter require are straightforwardly expressed in terms of the means and covariances of the component distributions. As discussed in section 4.3, the real noise-corrupted speech distributions do not have a closed form. Model compensation methods normally already approximate the component distributions as Gaussians, so that no additional approximations are required to find the statistics from joint uncertainty decoding.

That joint uncertainty decoding shares compensation parameters across a whole base class is particularly useful if the predictive transform uses the same base class. It will turn out to be possible to express the statistics that joint uncertainty decoding predicts in a component-dependent part that can be computed off-line, and a base-class-dependent part that changes with the noise parameters. Accumulating statistics from all components is therefore not necessary at run-time. This saves storage space and computation time.

This section will use the convention that base classes for joint uncertainty decoding and for predictive transformations are the same. As before, since the estimation is per base class, the notation will assume only one base class. The distribution for component m that joint uncertainty decoding predicts was given in (4.48a):

$$p^{(m)}(\mathbf{y}) = \left|\mathbf{A}_{\mathsf{jud}}\right| \cdot \mathcal{N}\left(\mathbf{A}_{\mathsf{jud}}\mathbf{y} + \mathbf{b}_{\mathsf{jud}};\, \boldsymbol{\mu}_x^{(m)},\, \boldsymbol{\Sigma}_x^{(m)} + \boldsymbol{\Sigma}_{\mathsf{bias}}\right), \tag{6.26}$$

where $\mathbf{A}_{\mathsf{jud}}$, $\mathbf{b}_{\mathsf{jud}}$, and $\boldsymbol{\Sigma}_{\mathsf{bias}}$ are computed from the joint distribution of the clean speech and the observation as in section 4.4.3. Of particular interest is the case where $\boldsymbol{\Sigma}_{\mathsf{bias}}$ is full, to compensate for changes in feature correlations, which arises when the joint distribution has a full covariance.

The predictive transformations in this section will use the predicted distribution

of the JUD-transformed observation $\hat{\mathbf{y}}$,

$$\hat{\mathbf{y}} = \mathbf{A}_{\text{jud}}\mathbf{y} + \mathbf{b}_{\text{jud}}. \tag{6.27}$$

Other options are possible (e.g. Xu *et al.* 2009; 2011), because JUD compensation can be written without feature transformation, but with the inverse transformation on the mean and covariance. However, from joint uncertainty decoding with a full transformation and full covariance bias, predictive CMLLR could find the exact same maximum likelihood solution (it would if it found the global optimum). Predictive covariance MLLR keeps the original covariance $\boldsymbol{\Sigma}_x$, so transforming it to a different feature space first would defeat the purpose. Predictive semi-tied covariance matrices could, just like predictive CMLLR, find the same solution in whatever feature space, as long as the the means are re-estimated as in section 6.2.3.

The statistics that the predictive transforms require from the predicted distributions consist solely of the following elements, which are straightforward to derive from the Gaussian in (6.26):

$$\mathcal{E}_{p^{(m)}}\{\hat{\mathbf{y}}\} = \boldsymbol{\mu}_x^{(m)}; \tag{6.28a}$$

$$\text{Var}_{p^{(m)}}\{\hat{\mathbf{y}}\} = \boldsymbol{\Sigma}_x^{(m)} + \boldsymbol{\Sigma}_{\text{bias}}; \tag{6.28b}$$

$$\mathcal{E}_{p^{(m)}}\left\{\hat{\mathbf{y}}\hat{\mathbf{y}}^{\mathsf{T}}\right\} = \boldsymbol{\Sigma}_x^{(m)} + \boldsymbol{\Sigma}_{\text{bias}} + \boldsymbol{\mu}_x^{(m)}\boldsymbol{\mu}_x^{(m)\mathsf{T}}. \tag{6.28c}$$

The next sections will discuss instantiations of the predictive versions of CMLLR, covariance MLLR, and semi-tied covariance matrices. In each case, it will turn out to be possible to express the statistics so that most of the accumulation can be performed off-line. This is because in (6.28) the statistics derived from the clean speech component, $\boldsymbol{\mu}_x^{(m)}$ and $\boldsymbol{\Sigma}_x^{(m)}$, do not depend on the noise model, whereas $\boldsymbol{\Sigma}_{\text{bias}}$ does depend on the noise model, but not on the component distributions.

6.3.1 *Predictive* CMLLR

Predictive CMLLR trained on joint uncertainty decoding results in the following likelihood calculation:

$$q^{(m)}(\mathbf{y}) = \left|\mathbf{A}_{\text{cmllr}}\right| \cdot \left|\mathbf{A}_{\text{jud}}\right| \cdot \mathcal{N}\Big(\mathbf{A}_{\text{cmllr}}\big(\mathbf{A}_{\text{jud}}\mathbf{y} + \mathbf{b}_{\text{jud}}\big) + \mathbf{b}_{\text{cmllr}};\, \boldsymbol{\mu}_x^{(m)},\, \boldsymbol{\Sigma}_x^{(m)}\Big). \tag{6.29}$$

Compared to (6.26), this lacks covariance bias $\boldsymbol{\Sigma}_{\text{bias}}$. To make up for that, $\mathbf{A}_{\text{cmllr}}$ and $\mathbf{b}_{\text{cmllr}}$ are trained to minimise the KL divergence with p in (6.26). This uses statistics $\mathbf{k}^{(i)}$ and $\mathbf{G}^{(i)}$ defined in (6.21) in section 6.2.1. Using (6.28), they become

$$\mathbf{k}^{(i)} \triangleq \sum_m \frac{\gamma^{(m)}\mu_{x,i}^{(m)}}{\sigma_{x,ii}^{(m)}} \Big[\, \mathcal{E}_{p^{(m)}}\{\hat{\mathbf{y}}^{\mathsf{T}}\} \;\; 1 \,\Big] = \sum_m \frac{\gamma^{(m)}\mu_{x,i}^{(m)}}{\sigma_{x,ii}^{(m)}} \Big[\, {\mu_x^{(m)}}^{\mathsf{T}} \;\; 1 \,\Big]; \tag{6.30a}$$

$$\begin{aligned}\mathbf{G}^{(i)} &\triangleq \sum_m \frac{\gamma^{(m)}}{\sigma_{x,ii}^{(m)}} \begin{bmatrix} \mathcal{E}_{p^{(m)}}\{\hat{\mathbf{y}}\hat{\mathbf{y}}^{\mathsf{T}}\} & \mathcal{E}_{p^{(m)}}\{\hat{\mathbf{y}}\} \\ \mathcal{E}_{p^{(m)}}\{\hat{\mathbf{y}}^{\mathsf{T}}\} & 1 \end{bmatrix} \\ &= \sum_m \frac{\gamma^{(m)}}{\sigma_{x,ii}^{(m)}} \begin{bmatrix} \left(\boldsymbol{\Sigma}_x^{(m)} + \boldsymbol{\Sigma}_{\text{bias}} + \boldsymbol{\mu}_x^{(m)}{\boldsymbol{\mu}_x^{(m)}}^{\mathsf{T}}\right) & \boldsymbol{\mu}_x^{(m)} \\ {\boldsymbol{\mu}_x^{(m)}}^{\mathsf{T}} & 1 \end{bmatrix}. \end{aligned} \tag{6.30b}$$

It is interesting that $\mathbf{k}^{(i)}$ does not depend on the parameters of joint uncertainty decoding. It can therefore be computed off-line and cached in its entirety. $\mathbf{G}^{(i)}$, on the other hand, does depend on a JUD parameter: $\boldsymbol{\Sigma}_{\text{bias}}$. However, it can be rewritten to be largely cacheable:

$$\mathbf{G}^{(i)} = \underbrace{\sum_m \frac{\gamma^{(m)}}{\sigma_{x,ii}^{(m)}} \begin{bmatrix} \left(\boldsymbol{\Sigma}_x^{(m)} + \boldsymbol{\mu}_x^{(m)}{\boldsymbol{\mu}_x^{(m)}}^{\mathsf{T}}\right) & \boldsymbol{\mu}_x^{(m)} \\ {\boldsymbol{\mu}_x^{(m)}}^{\mathsf{T}} & 1 \end{bmatrix}}_{\text{cached}} + \begin{bmatrix} \boldsymbol{\Sigma}_{\text{bias}} & 0 \\ 0 & 0 \end{bmatrix} \underbrace{\sum_m \frac{\gamma^{(m)}}{\sigma_{x,ii}^{(m)}}}_{\text{cached}}. \tag{6.30c}$$

This completely removes the need to iterate over components at run-time. Since there is a $\mathbf{G}^{(i)}$ for every dimension i, the total computational cost per base class for computing them is $\mathcal{O}\big(Rd^3\big)$.

6.3.2 *Predictive covariance* MLLR

Predictive covariance MLLR results in the following likelihood calculation when trained on joint uncertainty decoding:

$$q^{(m)}(\mathbf{y}) = \left|\boldsymbol{A}_{\text{mllrcov}}\right| \cdot \left|\boldsymbol{A}_{\text{jud}}\right| \cdot \mathcal{N}\left(\boldsymbol{A}_{\text{mllrcov}}\left(\boldsymbol{A}_{\text{jud}}\mathbf{y} + \mathbf{b}_{\text{jud}}\right); \boldsymbol{A}_{\text{mllrcov}}\boldsymbol{\mu}_x^{(m)}, \boldsymbol{\Sigma}_x^{(m)}\right). \tag{6.31}$$

The transformation $\boldsymbol{A}_{\text{mllrcov}}$ on the features and on the means is equivalent to the inverse transformation on the covariance, as discussed in section 3.2.2.

Compared to the distribution $p^{(m)}$ in (6.26) that $q^{(m)}$ aims to approximate, this lacks the covariance bias $\boldsymbol{\Sigma}_{\text{bias}}$. Therefore, $\boldsymbol{A}_{\text{mllrcov}}$ is trained to make up for that. This allows decoding to use unchanged diagonal covariances while still modelling some of the predicted correlations. Training predictive covariance MLLR uses statistics $\mathbf{G}^{(i)}$ defined in (B.22) in section 6.2.2. Using (6.28), and noting that the mean of $\hat{\mathbf{y}}$ is $\boldsymbol{\mu}_x^{(m)}$, they become

$$\begin{aligned}\mathbf{G}^{(i)} &\triangleq \sum_m \frac{\gamma^{(m)}}{\sigma_{x,ii}^{(m)}} \mathcal{E}_{p^{(m)}}\left\{\left(\hat{\mathbf{y}} - \boldsymbol{\mu}_x^{(m)}\right)\left(\hat{\mathbf{y}} - \boldsymbol{\mu}_x^{(m)}\right)^{\mathsf{T}}\right\} = \sum_m \frac{\gamma^{(m)}}{\sigma_{x,ii}^{(m)}} \text{Var}_{p^{(m)}}\{\hat{\mathbf{y}}\} \\ &= \sum_m \frac{\gamma^{(m)}}{\sigma_{x,ii}^{(m)}}\left(\boldsymbol{\Sigma}_x^{(m)} + \boldsymbol{\Sigma}_{\text{bias}}\right).\end{aligned} \tag{6.32a}$$

Just like for predictive CMLLR trained on joint uncertainty decoding, this expression can be written so that iterations over the components can be cached. This works similarly to (6.30c):

$$\mathbf{G}^{(i)} \triangleq \underbrace{\sum_m \frac{\gamma^{(m)}}{\sigma_{x,ii}^{(m)}} \boldsymbol{\Sigma}_x^{(m)}}_{\text{cached}} + \boldsymbol{\Sigma}_{\text{bias}} \underbrace{\sum_m \frac{\gamma^{(m)}}{\sigma_{x,ii}^{(m)}}}_{\text{cached}}. \tag{6.32b}$$

The total on-line cost of computing $\mathbf{G}^{(i)}$ for all dimensions i is therefore $\mathcal{O}\left(Rd^3\right)$. This does not depend on the number of components. The model means also need to be transformed, by $\boldsymbol{A}_{\text{mllrcov}}$, the complexity of which does depend on the number of components: $\mathcal{O}\left(Md^2\right)$.

6.3.3 *Predictive semi-tied covariance matrices*

Predictive semi-tied covariance matrices, when trained on statistics predicted by joint uncertainty decoding, result in the following likelihood expression:

$$q^{(m)}(\mathbf{y}) = \left|\mathbf{A}_{st}\right| \cdot \left|\mathbf{A}_{jud}\right| \cdot \mathcal{N}\Big(\mathbf{A}_{st}\big(\mathbf{A}_{jud}\mathbf{y} + \mathbf{b}_{jud}\big)\,;\, \mathbf{A}_{st}\boldsymbol{\mu}_x^{(m)},\, \widetilde{\boldsymbol{\Sigma}}_{y,diag}^{(m)}\Big)\,. \tag{6.33}$$

As in the case of predictive covariance MLLR, the transformation $\mathbf{A}_{st}$ is equivalent to the inverse transformation on the covariance matrix. Semi-tied covariance matrices mean to find a feature space, specified by $\mathbf{A}_{st}$, in which a diagonal covariance matrix is a reasonable assumption. Unlike for covariance MLLR, the component covariances are also adapted to minimise the KL divergence with the predicted distribution. $\mathbf{A}_{st}$ and the $\widetilde{\boldsymbol{\Sigma}}_{y,diag}^{(m)}$ are estimated in an iterative process.

The difference with the predicted distribution, in (6.26), is that the covariance matrix $\widetilde{\boldsymbol{\Sigma}}_{y,diag}^{(m)}$ is diagonal. The transformation $\mathbf{A}_{st}$ needs to make up only for the off-diagonal entries of the covariance matrix, unlike for predictive covariance MLLR, which has a similar form, but uses the original covariance matrices. Training predictive semi-tied covariance matrices uses statistics $\boldsymbol{\mu}_y^{(m)}$, $\boldsymbol{W}^{(m)}$ and $\mathbf{G}^{(i)}$ defined in (6.25). Using (6.28), $\boldsymbol{\mu}_y^{(m)}$ and $\boldsymbol{W}^{(m)}$ become

$$\boldsymbol{\mu}_y^{(m)} \triangleq \mathcal{E}_{p^{(m)}}\{\hat{\mathbf{y}}\} = \boldsymbol{\mu}_x^{(m)}; \tag{6.34a}$$

$$\begin{aligned}\boldsymbol{W}^{(m)} &\triangleq \mathcal{E}_{p^{(m)}}\Big\{\big(\hat{\mathbf{y}} - \boldsymbol{\mu}_y^{(m)}\big)\big(\hat{\mathbf{y}} - \boldsymbol{\mu}_y^{(m)}\big)^{\mathsf{T}}\Big\} = \mathrm{Var}_{p^{(m)}}\{\hat{\mathbf{y}}\} \\ &= \boldsymbol{\Sigma}_x^{(m)} + \boldsymbol{\Sigma}_{bias}.\end{aligned} \tag{6.34b}$$

$\mathbf{G}^{(i)}$ can be defined in terms of this, exactly as in (6.25c):

$$\mathbf{G}^{(i)} \triangleq \sum_m \frac{\gamma^{(m)}}{\widetilde{\sigma}_{y,ii}^{(m)}} \boldsymbol{W}^{(m)}. \tag{6.34c}$$

These statistics can again be formulated in such a way that the on-line computational cost is less than for a direct implementation. This assumes that the original speech

covariance matrices $\boldsymbol{\Sigma}_x^{(m)}$ are diagonal. $\mathbf{G}^{(i)}$ can be written

$$\mathbf{G}^{(i)} = \underbrace{\sum_m \frac{\gamma^{(m)}}{\widetilde{\sigma}_{y,ii}^{(m)}} \boldsymbol{\Sigma}_x^{(m)}}_{\mathcal{O}(Md)} + \underbrace{\boldsymbol{\Sigma}_{\text{bias}} \sum_m \frac{\gamma^{(m)}}{\widetilde{\sigma}_{y,ii}^{(m)}}}_{\mathcal{O}(M+d^2)} . \tag{6.35a}$$

The total cost of finding these for d dimensions and R base classes is $\mathcal{O}(Md^2 + Rd^3)$. A similar optimisation can be applied to the covariance update in (B.40):

$$\begin{aligned} \widetilde{\boldsymbol{\Sigma}}_{y,\text{diag}}^{(m)} &:= \text{diag}\left(\boldsymbol{A}\boldsymbol{W}^{(m)}\boldsymbol{A}^T\right) = \text{diag}\left(\boldsymbol{A}\left(\boldsymbol{\Sigma}_x^{(m)} + \boldsymbol{\Sigma}_{\text{bias}}\right)\boldsymbol{A}^T\right) \\ &= \underbrace{\text{diag}\left(\boldsymbol{A}\boldsymbol{\Sigma}_x^{(m)}\boldsymbol{A}^T\right)}_{\mathcal{O}(d^2)} + \underbrace{\text{diag}\left(\boldsymbol{A}\boldsymbol{\Sigma}_{\text{bias}}\boldsymbol{A}^T\right)}_{\mathcal{O}(d^3)} . \end{aligned} \tag{6.35b}$$

Since the left-hand term is component-dependent, it needs to be computed separately for each component. However, $\boldsymbol{\Sigma}_x^{(m)}$ is diagonal and the result is constrained to be diagonal, so that the cost per component is only $\mathcal{O}(d^2)$. The right-hand term is the same for a whole base class, but since $\boldsymbol{\Sigma}_{\text{bias}}$ is full, the calculation takes $\mathcal{O}(d^3)$. The overall complexity of updating $\widetilde{\boldsymbol{\Sigma}}_{y,\text{diag}}^{(m)}$ for all M components in R base classes is therefore $\mathcal{O}(Md^2 + Rd^3)$.

For the full scheme, the complexities are multiplied by the number of outer iterations K. The half-iteration scheme for predictive semi-tied covariance matrices only finds a transformation matrix $\mathbf{A}_{\text{st}}$ and only initialises the covariances $\widetilde{\boldsymbol{\Sigma}}_{y,\text{diag}}^{(m)}$, apart from adding the covariance bias, which is $\mathcal{O}(Md)$.

6.3.4 *Computational complexity*

Table 6.1 on the next page details the time requirements for the approximations to joint uncertainty decoding discussed in the previous sections. The naive implementation for calculating the cofactors takes $\mathcal{O}(Rd^4)$ per iteration, but using the Sherman-Morrison matrix-inversion lemma this can be reduced to $\mathcal{O}(Rd^3)$ per iteration (Gales and van Dalen 2007). Inverting $\mathbf{G}^{(i)}$ takes $\mathcal{O}(Rd^4)$ per iteration.[2] In all cases, by al-

[2] By using the average of the diagonal of $\widetilde{\boldsymbol{\Sigma}}_{y,\text{diag}}^{(m)}$ rather than different $\widetilde{\sigma}_{y,ii}^{(m)}$ for $1 \leq i \leq d$, it is possible to reduce this to $\mathcal{O}(Rd^3)$ (Gales and van Dalen 2007) at the cost of some loss in accuracy. This has not been shown in the table.

	CMLLR	MLLR covariance	Half semi-tied	Full semi-tied
Estimation				
Statistics	Rd^3	Rd^3	$Md^2 + Rd^3$	$K(Md^2 + Rd^3)$
Inverting $\mathbf{G}^{(i)}$	Rd^4	Rd^4	Rd^4	KRd^4
Calculating $\mathbf{c}_i$	LRd^3	LRd^3	LRd^3	$KLRd^3$
Compensation				
Features	TRd^2	TRd^2	TRd^2	TRd^2
Means	0	Md^2	Md^2	Md^2
Variances	0	0	Md	KMd^2

Table 6.1 *The complexity of estimating predictive transforms from joint uncertainty decoding.* d *is the size of the feature vector;* M *is the number of components;* R *is the number of base classes;* L *is the number of inner iterations;* K *is the number of outer iterations (see section 3.3.1 on page 50).*

lowing for diagonal covariances on the models compensated for noise, the complexity associated with decoding T observations with joint uncertainty decoding is reduced by a factor of d.

6.3.5 *Predictive* HLDA

This chapter has so far presented predictive methods that start with a predicted distribution over feature vectors with statics and dynamics. Those distributions will have been derived from distributions over extended feature vectors as in chapter 5. The conversion from extended feature vectors, or distributions over them, to ones with statics and dynamics used a linear transformation $\mathbf{D}$. The predictive linear transformations in this chapter have estimated another linear feature transformations $\mathbf{A}$. An interesting avenue would be to not assume projection $\mathbf{D}$ and replace the pair of transformations $\mathbf{A} \cdot \mathbf{D}$, of which only $\mathbf{A}$ is estimated, by one transformation that at the same time reduces the feature dimensionality and transforms to a good feature space for noise at the same time.

To formulate distributions over the feature space with reduced dimensionality, a square Jacobian is necessary. Dimensionality reduction therefore needs, at least math-

ematically, to find a new feature space of the same dimensionality. Some of those dimensions are useful dimensions, which are used for discrimination, and some are *nuisance* dimensions, the distributions over which should be tied over all components so they do not discriminate. This requirement was also an issue for estimating an approximate distribution over extended feature vectors. As section 5.2.2 has discussed, this is only equivalent to optimising in the projected space under certain conditions. Extended IDPMC (section 5.3.2), for example, which estimates mixtures of Gaussians, explicitly bypassed the issue and estimates parameters directly on samples with statics and dynamics. This trick does not apply here, so the nuisance dimensions must be handled explicitly.

This section will therefore sketch how to apply heteroscedastic linear discriminant analysis (Kumar 1997), discussed in section 3.3.2, to extended feature vectors. This will be called "predictive HLDA". This explicitly splits the feature transformation up in useful and nuisance dimensions, and ties the distribution over the nuisance dimensions. Both heteroscedastic linear discriminant analysis (HLDA) and semi-tied covariance matrices can be seen as instances of multiple HLDA (MHLDA). MHLDA is HLDA with different transformations for different classes; semi-tied covariance matrices is MHLDA without dimensionality reduction. The scheme sketched here could straightforwardly be extended to multiple base classes.

The likelihood calculation for predictive HLDA is similar to the one for predictive semi-tied covariance matrices (in (6.24)):[3]

$$q^{(m)}(\mathbf{y}^e) = \mathcal{N}\big(\boldsymbol{A}\mathbf{y}^e; \widetilde{\boldsymbol{\mu}}_y^{(m)}, \widetilde{\boldsymbol{\Sigma}}_{y,\mathsf{diag}}^{(m)}\big). \tag{6.36}$$

The two differences are that this expression uses extended feature vectors $\mathbf{y}^e$, and that $\boldsymbol{A}$ is a non-square matrix that reduces the dimensionality as well as finding a feature space in which it is a decent approximation to make $\widetilde{\boldsymbol{\Sigma}}_{y,\mathsf{diag}}^{(m)}$ diagonal.

HLDA needs statistics similar to semi-tied covariance matrices; analogously, predictive HLDA needs statistics similar to predictive semi-tied covariance matrices, but

[3] The factor $|\boldsymbol{A}|$ in (6.24) does not need to be computed if there is only one base class, like here, since it affects all likelihood equally.

over extended feature vectors. Denoting the predicted distribution for component m with $p^{e(m)}$, the predicted covariance for component m is (analogous to (6.34b))

$$\boldsymbol{W}^{(m)} \triangleq \mathcal{E}_{p^{e(m)}}\{\mathbf{y}^e\mathbf{y}^{e\mathsf{T}}\} - \mathcal{E}_{p^{e(m)}}\{\mathbf{y}^e\}\mathcal{E}_{p^{e(m)}}\{\mathbf{y}^e\}^{\mathsf{T}}. \tag{6.37}$$

Given the statistics, following the procedure for computing semi-tied covariance matrices would yield a transformation that aims to find an optimal square linear transformation. HLDA on the other hand applies dimensionality reduction as well. Unlike with predictive semi-tied covariance matrices, the new component parameters must be derived directly from the extended statistics. The means, for example, are set to the transformed mean of the predicted distribution:

$$\widetilde{\boldsymbol{\mu}}_y^{(m)} := \boldsymbol{A}\mathcal{E}_{p^{e(m)}}\{\mathbf{y}^e\}, \tag{6.38a}$$

and the component covariance matrices use (6.37):

$$\widetilde{\boldsymbol{\Sigma}}_{y,\mathsf{diag}}^{(m)} := \mathrm{diag}\left(\boldsymbol{A}\boldsymbol{W}^{(m)}\boldsymbol{A}^{\mathsf{T}}\right). \tag{6.38b}$$

Optimising the transformation $\boldsymbol{A}$ as in Kumar (1997) yields an interesting variant of predictive linear transformations that transforms the extended feature space as well as selecting useful features from it. For noise-corrupted speech, this makes intuitive sense. Some features may be completely masked under low signal-to-noise ratios. Predictive HLDA will then transform the feature vector used for recognition so as to replace those features by more useful features for that specific noise condition. However, predictive HLDA could also be applied to distributions predicted from other models.

6.4 Front-end PCMLLR

Predictive linear transformations are flexible. This thesis has motivated their use from the perspective of reducing the computational load of decoding with full covariances. However, predictive CMLLR finds a component-dependent feature transformation, and

it is possible to use this for approximating diagonal-covariance predicted distributions. This section will apply PCMLLR-like transformations to features without reference to the component identity.[4] An interesting practical consequence will be that the resulting transformation can be similar to that of model-based feature enhancement (see section 4.6), but is motivated differently. Whereas feature enhancement aims to reconstruct the clean speech, here, the transformations aim to minimise the KL divergence between the predicted distribution and the effective speech recogniser distribution. This uses more precise information about the clean speech distribution, the speech recogniser components rather than the front-end components. Many statistics can again be cached, so that this is computationally very efficient.

The framework of predictive transformations can also be used to speed up compensation for noise with diagonal covariance matrices. The methods in this section will derive from predictive CMLLR, which, like standard CMLLR, applies a component-dependent feature transformation to the observation vector. The transformation that both forms of CMLLR find is a set of affine transformations for each base class r: $\mathcal{A} = \{\mathcal{A}^{(r)}\} = \left\{\mathbf{A}^{(r)}, \mathbf{b}^{(r)}\right\}$. To keep the notation uncluttered, this thesis has not explicitly written the dependency on the base class, but this section will, because it will be vital. Both adaptive and predictive CMLLR decode with (from (6.20), with the base class explicit)

$$q^{(m)}(\mathbf{y}|\mathcal{A}) = \left|\mathbf{A}^{(r)}\right| \cdot \mathcal{N}\left(\mathbf{A}^{(r)}\mathbf{y} + \mathbf{b}^{(r)}; \boldsymbol{\mu}_x^{(m)}, \boldsymbol{\Sigma}_x^{(m)}\right). \tag{6.39a}$$

Each of the components is assigned one base class r, and only one feature transformation $\mathcal{A}^{(r)}$ is estimated for a base class. Therefore, which transformation $\mathcal{A}^{(r)}$ is chosen from the set of transformations depends on the component. However, it is also possible to find a transformation that takes the observation into account when minimising the KL divergence to the effective decoding distribution from the predicted noise-corrupted speech distribution. This could make the transformation more appropriate for the acoustic region that the observation is in.

[4] The work in this section, section 6.4, is joint work with Federico Flego, published as van Dalen *et al.* (2009).

The next sections will introduce methods of finding a component-independent transformation $\mathcal{A}_t = \{\mathbf{A}_t, \mathbf{b}_t\}$ at each time instance t. The distribution of component m becomes

$$q^{(m)}(\mathbf{y}_t) = |\mathbf{A}_t| \cdot \mathcal{N}\left(\mathbf{A}_t\mathbf{y} + \mathbf{b}_t;\, \boldsymbol{\mu}_x^{(m)},\, \boldsymbol{\Sigma}_x^{(m)}\right). \tag{6.39b}$$

The simplest scheme estimates a global transformation. It is equivalent to predictive CMLLR with the number of base classes set to $R = 1$. This makes the scheme trivially independent of the component. It is interesting as a baseline method, because the methods that this section will introduce all reduce to it when there is only one base class. The optimisation in (6.18) becomes

$$\mathcal{A} := \arg\min_{\mathcal{A}} \sum_m \gamma^{(m)} \mathcal{KL}\left(p^{(m)} \middle\| q^{(m)}\right). \tag{6.40}$$

Section 6.4.1 will introduce a method that re-trains the transformation with predictive CMLLR for every feature vector. Section 6.4.2 will introduce two methods that estimate the base class posterior to combine precomputed transformations.

6.4.1 *Observation-trained transformation*

The global transformation estimated with (6.40) gives an optimal overall transformation. However, when the observation is known, the distribution over the component identity can be approximated better. A scheme that estimates one PCMLLR transform from the posterior predicted distribution will be called "observation-trained PCMLLR".

To give a more robust estimate of the component distribution, the update of the component distribution is not performed per component, but per base class. To do this, the component occupancy is factorised into the occupancy of the base class $\gamma^{(r)}$ and the occupancy of the component given the base class:

$$\gamma^{(m)} = \gamma^{(r)} \left(\frac{\gamma^{(m)}}{\gamma^{(r)}} \right), \qquad \gamma^{(r)} = \sum_{m \in \Omega^{(r)}} \gamma^{(m)}, \tag{6.41a}$$

where $\Omega^{(r)}$ is the set of all components in a base class.

The base class occupancy can be replaced by a weight estimated using the observation $\mathbf{y}_t$. An obvious choice is the posterior of the front-end mixture of Gaussians, $P(r|\mathbf{y}_t)$. Effectively, the posterior distribution of the corrupted speech vector $\mathbf{y}$ given the observation becomes

$$p(\mathbf{y}|\mathbf{y}_t) = \sum_r P(r|\mathbf{y}_t)\, P(\mathbf{y}|r)\,. \tag{6.41b}$$

The base class posterior then replaces the prior base class occupancy in (6.41a):

$$P(m|\mathbf{y}_t) := P(r|\mathbf{y}_t)\left(\frac{\gamma^{(m)}}{\gamma^{(r)}}\right) = \frac{P(r|\mathbf{y}_t)}{\gamma^{(r)}}\gamma^{(m)}. \tag{6.41c}$$

Each of the components m of the speech recogniser GMM is weighted or deweighted by the same amount as its associated front-end component r, so that (6.40) becomes

$$\mathcal{A}_t := \arg\min_{\mathcal{A}} \sum_r \frac{P(r|\mathbf{y}_t)}{\gamma^{(r)}} \sum_{m\in\Omega^{(r)}} \gamma^{(m)} \mathcal{KL}\big(p^{(m)} \big\| q^{(m)}\big). \tag{6.41d}$$

$\mathcal{A}_t$ in (6.41d) is retrained for every feature vector. Apart from the weighting, this is the same expression as the optimisation for generic predictive transformations in (6.18). The algorithm therefore is the same as for predictive CMLLR discussed in section 6.2.1, but with modified statistics. They are similar to (6.21), with the component occupancies $\gamma^{(m)}$ replaced by their posteriors $P(m|\mathbf{y}_t)$ in (6.41c):

$$\mathbf{k}^{(i)} \triangleq \sum_r \frac{P(r|\mathbf{y}_t)}{\gamma^{(r)}} \sum_{m\in\Omega^{(r)}} \frac{\gamma^{(m)}\mu^{(m)}_{x,i}}{\sigma^{(m)}_{x,ii}} \Big[\, \mathcal{E}_{p^{(m)}}\{\mathbf{y}^\mathsf{T}\} \;\; 1 \,\Big]; \tag{6.42a}$$

$$\mathbf{G}^{(i)} \triangleq \sum_r \frac{P(r|\mathbf{y}_t)}{\gamma^{(r)}} \sum_{m\in\Omega^{(r)}} \frac{\gamma^{(m)}}{\sigma^{(m)}_{x,ii}} \begin{bmatrix} \mathcal{E}_{p^{(m)}}\{\mathbf{y}\mathbf{y}^\mathsf{T}\} & \mathcal{E}_{p^{(m)}}\{\mathbf{y}\} \\ \mathcal{E}_{p^{(m)}}\{\mathbf{y}^\mathsf{T}\} & 1 \end{bmatrix}. \tag{6.42b}$$

If implemented directly, it is computationally expensive to accumulate these statistics. However, given that speech recogniser components are weighted a whole base class at a time, the necessary statistics are weighted sums of per-base class statistics, inside the sums $\sum_{m\in\Omega^{(r)}}$. These per-base class statistics are exactly the same as the statistics that standard CMLLR (predictive and adaptive) uses, in (6.21), to estimate per-base class transformations. The number of base classes is normally only a fraction of the number of back-end components, so computing the statistics in (6.42b) is fast.

6.4.2 *Posterior-weighted transformations*

Rather than estimating a global transform, whether using the speech recogniser component priors or their updated versions in (6.4.1), it is possible to estimate a set of transforms appropriate to regions of the acoustic space with PCMLLR, and from those construct a transformation that is observation-specific. A simple way of doing this is to pick the transformation associated with the most likely front-end component:

$$\mathcal{A}_t := \mathcal{A}^{(r^*)}, \qquad r^* = \arg\max_r P(r|\mathbf{y}_t)\,. \tag{6.43}$$

This scheme, "hard-decision PCMLLR", yields a piecewise linear transformation of the feature space.

A more sophisticated approach is similar to front-end CMLLR (Liao and Gales 2005). This interpolates the transforms, each optimised to minimise the KL divergence to the predicted back-end distributions in an acoustic region. The front-end component posterior is an approximate measure of how likely an observation is to have been generated by a back-end component associated with front-end component r. The transformation becomes

$$\mathbf{A}_t = \sum_r P(r|\mathbf{y}_t)\,\mathbf{A}^{(r)}; \tag{6.44a}$$

$$\mathbf{b}_t := \sum_r P(r|\mathbf{y}_t)\,\mathbf{b}^{(r)}. \tag{6.44b}$$

This will be called "interpolated PCMLLR". The form of the compensation is then the same as for model-based feature enhancement (see section 4.6), so that decoding is equally fast. However, interpolated PCMLLR is properly viewed as transforming the models, aiming to minimise the KL divergence between the predicted distributions and the speech recogniser models, rather than reconstructing the clean speech. Post-processing of the transformed observation as if it represents the clean speech is therefore not possible.

6.5 Summary

This chapter has described the second contribution of this thesis. It has introduced a general framework for approximating one speech recogniser parameterisation with another. Its objective is to minimise the KL divergence to a predicted distribution. This framework of predictive methods subsumes model compensation methods. Section 6.2 has derived a number of new predictive versions of linear transformations that chapter 3 had discussed. Section 6.3 has detailed how to approximate joint uncertainty decoding with full covariances (from section 5.4). This combines joint uncertainty decoding's ability to train on limited data and linear transformations' ability to model correlations without reducing decoding speed. The resulting chain of methods (extended VTS, joint uncertainty decoding, and predictive semi-tied covariance matrices) is what this thesis proposes as a feasible compensation scheme for real-world speech recognition.

To show off the versatility of predictive transformations, section 6.4 has introduced a number of schemes based off predictive CMLLR, which apply only feature transformations, but make the resulting speech recogniser match the predicted distributions as well as possible.

Chapter 7

Asymptotically exact likelihoods

This chapter will present the third contribution of this thesis, which is more theoretical.[1]

Section 4.1 has argued that using the exact corrupted speech distribution would lead to optimal classification. Given standard models for the speech, noise, and mismatch function, the distributions of the corrupted speech does not have a closed form. Model compensation methods therefore approximate it with a parameterised distribution, usually a Gaussian. Even for a mixture of full-covariance Gaussians, estimated as in section 5.3.2, the number of Gaussians required to approximate the real distribution well may be high, and the complexity is essentially cubic in that number. No previous work has investigated how well speech recognition would perform without parameterised distributions.

This chapter will introduce a sampling method that, given distributions over feature vectors for the speech and the noise and a mismatch function, in the limit calculates the corrupted speech likelihood exactly. What the method yields, therefore, is an upper bound on model compensation. Section 7.1 will introduce the model that this chapter assumes. Section 7.2 will discuss how to approximate the distribution with importance sampling from a Gaussian over the speech and noise. However, when

[1] This work has been published as van Dalen and Gales (2010*b*) and a summary in van Dalen and Gales (2010*a*).

applied to more than a few dimensions this becomes infeasible. Section 7.3 will therefore first transform the integral in one dimension, and apply importance sampling. It will then introduce factorisations for the multi-dimensional integrand, and approximate the integral with sequential importance sampling. This scheme is too slow to implement in a speech recogniser. An alternative metric for assessing model compensation methods, more fine-grained than word error rates, would be how closely model compensation methods match the distribution they approximate. Section 7.4 will therefore introduce a method to compute the KL divergence up to a constant of the approximation of the corrupted speech distribution to the real distribution.

7.1 Likelihood evaluation

This section uses a different approach to approximating the corrupted speech distribution from standard model compensation. Model compensation methods, like the ones discussed in section 4.4, find a parameterised approximation for the corrupted speech distribution for recognition. However, no expression for the full density is needed: while recognising speech only likelihoods for vectors that are observed are required. Therefore, similarly to the methods in section 4.5, in this section the likelihood is approximated for a specific observation $\mathbf{y}_t$. For simplicity of notation, the convolutional noise will be assumed zero in this section.[2] Substituting $\mathbf{y}_t$ for $\mathbf{y}$ in (4.24b),

$$p(\mathbf{y}_t) = \int\int p(\mathbf{y}_t|\mathbf{n},\mathbf{x})\, p(\mathbf{n})\mathrm{d}\mathbf{n} p(\mathbf{x})\mathrm{d}\mathbf{x} \tag{7.1}$$

Since, given $\mathbf{y}_t$, this is essentially of the form

$$p(\mathbf{y}_t) = \int \phi(\mathbf{n},\mathbf{x}) p(\mathbf{n},\mathbf{x})\mathrm{d}(\mathbf{x},\mathbf{n}), \tag{7.2}$$

where the desired quantity is the integral of test function ϕ under distribution p, the obvious approach would be to apply one of a number of standard approaches to solving this form of problem.

[2] In the mismatch function in (4.10a) the convolutional noise just causes an offset on the speech signal.

Laplace's method may be used. This approximates the complete integrand by placing a Gaussian q on its mode. This is not possible analytically, but the Algonquin algorithm (see section 4.5.1) approximates this with an iterative algorithm. However, this does not yield any guarantee about the resulting likelihood.

Another approach would seem to find a variational lower-bound q to the integrand. An obstacle to this, however, is the form of the test function $\phi(\mathbf{n}, \mathbf{x}) = p(\mathbf{y}_t|\mathbf{n}, \mathbf{x})$. It represents the probability that corrupted speech vector $\mathbf{y}_t$ is generated from clean speech and noise vectors $\mathbf{x}, \mathbf{n}$. Though the phase factor introduces some uncertainty about this mapping, most values for $\mathbf{x}$ and $\mathbf{n}$ are incompatible for given $\mathbf{y}_t$, and $\phi(\mathbf{n}, \mathbf{x})$ is then 0. It is therefore not possible to lower-bound the integrand with a Gaussian, and not obvious which other distribution (the shape of the integrand will be detailed in section 7.2, and depicted in figure 7.1a on page 187) to choose.

A third approach may be to approximate the integral with Monte Carlo. (7.2) is written in a form that makes it obvious how to do so. It is straightforward to sample $(\mathbf{n}, \mathbf{x})$ from the prior $p(\mathbf{n}, \mathbf{x})$. However, the shape of $p(\mathbf{n}, \mathbf{x})$ is not always a good match for the shape of ϕ, so that most samples are drawn in vain. To alleviate this problem, it is possible to use importance sampling, which is discussed in appendix A.4.2, to evaluate the integral over $\phi(\mathbf{n}, \mathbf{x})p(\mathbf{n}, \mathbf{x})$ at once. Why sampling from $p(\mathbf{n}, \mathbf{x})$ is so inefficient will also be analysed from this perspective.

Importance sampling requires a *proposal distribution*, the distribution that the algorithm draws from instead of the actual distribution. It makes up for the difference by assigning each sample a weight. The proposal distributions needs to match the target density well, otherwise the weights will have a high variance, and too many samples will be required to arrive at a good estimate. The number of samples required grows exponentially with the number of dimensions.

To apply importance sampling, the proposal distribution will be either the prior, which reduces to straightforward Monte Carlo, or an Algonquin-derived approximation of the posterior. Both essentially use a joint Gaussian distribution over the speech and the noise. The prior distribution is often far away from the integrand, but even the

Algonquin-generated Gaussian cannot approximate the curved integrand well. Therefore, the number of samples required makes it infeasible to use this approximation. Section 7.3 will then introduce a transformation of the integrand. The integral over $\mathbf{x}$, $\mathbf{n}$, and $\boldsymbol{\alpha}$ can then be expressed as an integral over substitute variable $\mathbf{u}$ and $\boldsymbol{\alpha}$. This new expression is still exact, but more amenable to being approximated with importance sampling. After considering the one-dimensional case, the higher-dimensional space will use the same techniques for each dimension, and apply sequential importance resampling.

This chapter will focus on one Gaussian for the speech and one for the noise:

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)\,; \tag{7.3a}$$

$$\mathbf{n} \sim \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)\,. \tag{7.3b}$$

The mismatch function is the one given in (4.10a). This mismatch function uses a state-of-the-art model of the phase factor (see section 4.2.1.1). However, all that the scheme relies on is that it is possible to draw samples from the phase factor distribution. Any other distribution for $p(\alpha)$ can be plugged in as long as it can be sampled from.

In this work, the speech and noise are modelled with Gaussians. However, different distributions would be possible, provided the form of the distributions satisfies two requirements. First, a *proposal distribution* must be found that is sufficiently close to the one-dimensional distribution that importance sampling is possible. Second, a reasonable approximation must be available for the marginal of one dimension given settings for a subset of all other dimensions. For Gaussians, both of these requirements will appear to be possible, though far from straightforward, to fulfil. One aspect that makes the second requirement hard is that the speech and noise will be modelled with correlated dimensions, through full covariances.

Since the speech and noise priors are full-covariance Gaussians, it make little difference whether they represent cepstral or log-spectral features: the domains are related by a linear transformation, the DCT (see section 2.1). In the log-spectral domain, the mismatch function works dimension per dimension (see section 4.2.1), so this

chapter will assume log-spectral domain speech and noise priors. The vectors in this chapter will consist of just statics, and will be denoted with $\mathbf{x}$, $\mathbf{n}$, $\boldsymbol{\alpha}$, $\mathbf{y}$.

The main difference with earlier work with a similar aim is that here, the speech and noise priors have full covariance matrices, so that dimensions can not be treated separately. Myrvoll and Nakamura (2004) used a piecewise linear approximation in one dimension, which, as section 4.5.2 has shown, does not generalise to multiple dimensions. At the same conference as work in this thesis was presented (van Dalen and Gales 2010*a*), Hershey *et al.* (2010) independently introduced a similar strategy. As will be discussed in section 7.3.1.3, it also treats dimensions separately.

The aim of the method that this chapter will introduce is the opposite of that of single-pass retraining (see section 4.4.4), though both can be seen as ideal model compensation. Single-pass retraining does not use speech and noise distributions or a mismatch function, but it does assume that the corrupted speech is Gaussian. The method in this chapter assumes the speech and noise priors to be Gaussian, and the mismatch function to be given, but does not assume a form of distribution of the corrupted speech.

7.2 Importance sampling over the speech and noise

This section presents the first approach, which is to approximate the integration over $\mathbf{x}$ and $\mathbf{n}$. Importance sampling requires that the integrand can be evaluated at any point $(\mathbf{x}, \mathbf{n})$. For this, part of the observation likelihood needs to be rewritten more explicitly. The corrupted speech likelihood is ((4.24b) and (4.24c) with $\mathbf{y}_t$ substituted for $\mathbf{y}$, as in (7.1)):

$$p(\mathbf{y}_t) = \int\int p(\mathbf{y}_t|\mathbf{x},\mathbf{n})\, p(\mathbf{n})\mathrm{d}\mathbf{n} p(\mathbf{x})\mathrm{d}\mathbf{x} \tag{7.4a}$$

$$= \int\int\int \delta_{\mathbf{f}(\mathbf{x},\mathbf{n},\boldsymbol{\alpha})}(\mathbf{y}_t)\, p(\boldsymbol{\alpha})\mathrm{d}\boldsymbol{\alpha} p(\mathbf{n})\mathrm{d}\mathbf{n} p(\mathbf{x})\mathrm{d}\mathbf{x}. \tag{7.4b}$$

The value of the observation vector $\mathbf{y}_t$ is not deterministic given the speech and the noise, because the phase factor $\boldsymbol{\alpha}$ is a random variable. However, $\boldsymbol{\alpha}$ is deterministic

given $\mathbf{x}$, $\mathbf{n}$, and $\mathbf{y}_t$, so that $p(\mathbf{y}_t|\mathbf{x},\mathbf{n})$ can be written in terms of the distribution of the phase factor. The phase factor that a specific setting of $\mathbf{x}$, $\mathbf{n}$, and $\mathbf{y}_t$ implies will be written $\boldsymbol{\alpha}(\mathbf{x},\mathbf{n},\mathbf{y}_t)$. It is a standard result (in appendix A.1.1 and, e.g., Bishop 2006, 11.1.1) that transforming the space of a probability distribution requires multiplying by the determinant of the Jacobian:

$$p(\mathbf{y}_t|\mathbf{x},\mathbf{n}) = \left|\left.\frac{\partial\boldsymbol{\alpha}(\mathbf{x},\mathbf{n},\mathbf{y})}{\partial\mathbf{y}}\right|_{\mathbf{y}_t}\right| \cdot p(\boldsymbol{\alpha}(\mathbf{x},\mathbf{n},\mathbf{y}_t))\,, \tag{7.5}$$

where $p(\boldsymbol{\alpha}(\mathbf{x},\mathbf{n},\mathbf{y}))$ denotes the density of $p(\boldsymbol{\alpha})$ at the value of $\boldsymbol{\alpha}$ implied by $\mathbf{x}$, $\mathbf{n}$, and $\mathbf{y}$.

The value of the phase factor as a function of the other variables follows from (4.9). The relation is defined per coefficient (i.e. frequency bin) i of the variables:

$$\alpha_i = \frac{\exp\left(y_i^{\log}\right) - \exp\left(x_i^{\log}\right) - \exp\left(n_i^{\log}\right)}{2\exp\left(\frac{1}{2}x_i^{\log} + \frac{1}{2}n_i^{\log}\right)}, \tag{7.6a}$$

and its partial derivative with respect to y_i is

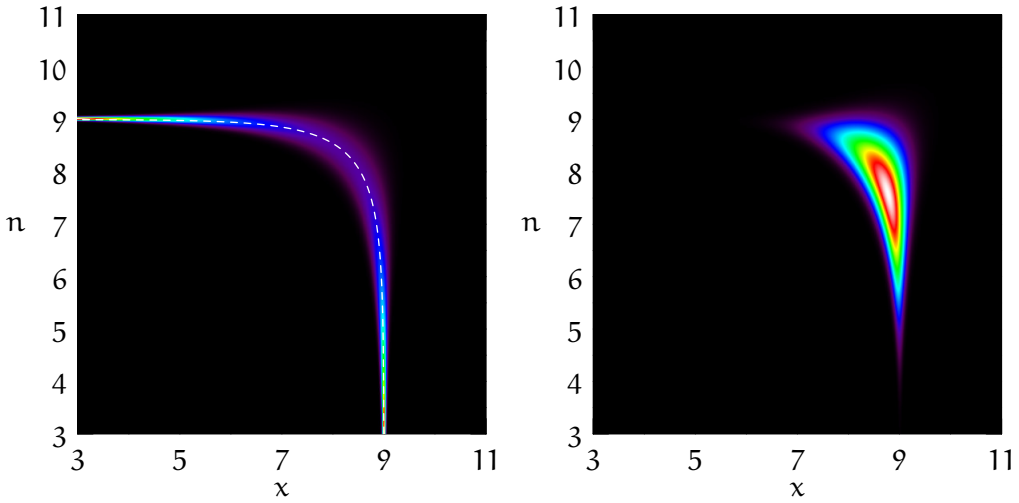
$$\frac{\partial\alpha(x_i,n_i,y_i)}{\partial y_i} = \frac{\exp\left(y_i^{\log}\right)}{2\exp\left(\frac{1}{2}x_i^{\log} + \frac{1}{2}n_i^{\log}\right)}. \tag{7.6b}$$

The diagonal elements of the Jacobian $\frac{\partial\boldsymbol{\alpha}(\mathbf{x},\mathbf{n},\mathbf{y})}{\partial\mathbf{y}}$ are given by these partial derivatives with respect to y_i. The off-diagonal entries of the Jacobian are 0.

The the distribution of the corrupted speech in (7.4a) can then be written as

$$\begin{aligned}
p(\mathbf{y}_t) &= \int\int p(\mathbf{y}_t|\mathbf{x},\mathbf{n})\,p(\mathbf{n})\mathrm{d}\mathbf{n}p(\mathbf{x})\mathrm{d}\mathbf{x} \\
&= \int\int \left|\left.\frac{\partial\boldsymbol{\alpha}(\mathbf{x},\mathbf{n},\mathbf{y})}{\partial\mathbf{y}}\right|_{\mathbf{y}_t}\right| \cdot p(\boldsymbol{\alpha}(\mathbf{x},\mathbf{n},\mathbf{y}_t))\,p(\mathbf{n})\mathrm{d}\mathbf{n}p(\mathbf{x})\mathrm{d}\mathbf{x} \\
&= \int\int \left\{\left(\prod_i \frac{\exp\left(y_{t,i}^{\log}\right)}{2\exp\left(\frac{1}{2}x_i^{\log} + \frac{1}{2}n_i^{\log}\right)}\right) \cdot p(\boldsymbol{\alpha}(\mathbf{x},\mathbf{n},\mathbf{y}_t))\,p(\mathbf{n})\,p(\mathbf{x})\right\}\mathrm{d}\mathbf{n}\mathrm{d}\mathbf{x} \\
&\triangleq \int\int \gamma(\mathbf{x},\mathbf{n})\mathrm{d}\mathbf{n}\mathrm{d}\mathbf{x}.
\end{aligned} \tag{7.7}$$

This expression is exact. Though the integrand, the expression in curly braces and denoted with γ, is now straightforward to evaluate at any given point $(\mathbf{x},\mathbf{n})$ if $p(\boldsymbol{\alpha})$

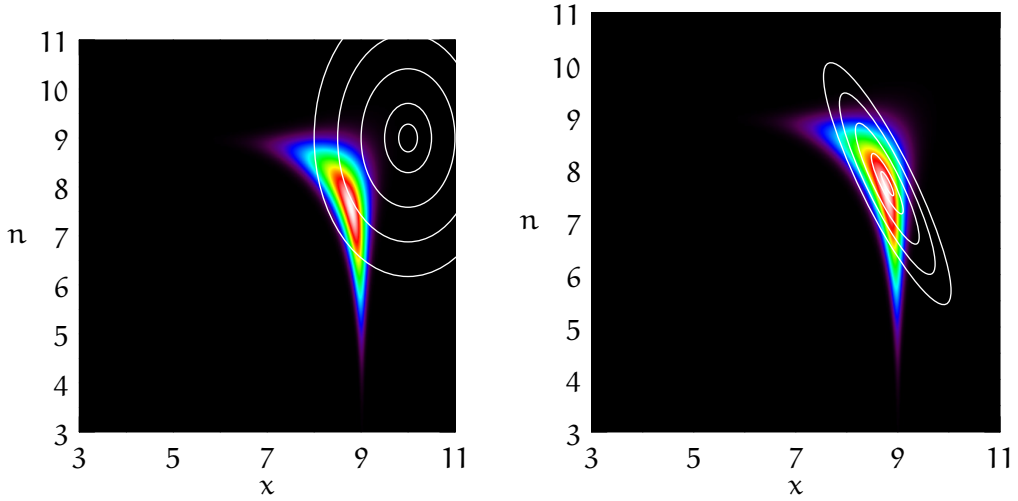


(a) The distribution $p(y_t|x, n)$. **(b)** The distribution $p(x, n, y_t)$.

Figure 7.1 *The distribution of the clean speech and noise for* $x \sim \mathcal{N}(10, 1)$*;* $n \sim \mathcal{N}(9, 2)$*;* $\alpha \sim \mathcal{N}(0, 0.04)$*;* $y_t = 9$.

can be evaluated, the integral has no closed form. It can, however, be approximated using importance sampling with γ as the target density. Importance sampling requires a proposal distribution that is close to the target. Figure 7.1a illustrates a one-dimensional version of the density $p(y_t|x, n)$. The density lies around the curve that relates x and n for given α and y_t, shown as a dashed line. This curve is given by $\exp(x) + \exp(n) = \exp(y_t)$. That the sum of the exponents of x and n is fixed causes a bend in the curve. Figure 7.1b shows the density in figure 7.1a multiplied by the priors of x and n, which gives $p(x, n, y_t) = \gamma(x, n)$.

Importance sampling uses a proposal distribution ρ, which it draws L samples $(\mathbf{x}^{(l)}, \mathbf{n}^{(l)})$ from and weights them to make up for the difference between proposal and target densities. This integral to be approximated is the normalisation constant of γ. The derivation is in appendix A.4.2, and in particular (A.43b), but intuitively it



(a) The prior $p(\mathbf{x}, \mathbf{n})$.

(b) Gaussian approximation to the posterior $p(\mathbf{x}, \mathbf{n}|\mathbf{y}_t)$.

Figure 7.2 *The distribution of the clean speech and noise for* $x \sim \mathcal{N}(10, 1)$; $n \sim \mathcal{N}(9, 2)$; $\alpha \sim \mathcal{N}(0, 0.04)$; $y = 9$.

can be written as

$$\int\int \gamma(\mathbf{x}, \mathbf{n}) d\mathbf{x} d\mathbf{n} = \int\int \frac{\gamma(\mathbf{x}, \mathbf{n})}{\rho(\mathbf{x}, \mathbf{n})} \rho(\mathbf{x}, \mathbf{n}) d\mathbf{x} d\mathbf{n} \simeq \sum_{l=1}^{L} \frac{\gamma(\mathbf{x}^{(l)}, \mathbf{n}^{(l)})}{\rho(\mathbf{x}^{(l)}, \mathbf{n}^{(l)})}, \qquad (7.8)$$
$$(\mathbf{x}^{(l)}, \mathbf{n}^{(l)}) \sim \rho.$$

The fraction of the target and proposal densities, γ/ρ, gives the weight of the samples. This weight makes up for the difference between the two densities.

To approximate the integral under γ, this section considers two options for the proposal distribution: the prior, and the Algonquin approximation to the posterior. Both are Gaussian.

A priori, the speech and the noise are independent. Given Gaussian prior distributions for $p(\mathbf{x})$ and $p(\mathbf{n})$ (in (7.3a) and (7.3b), respectively), their joint prior becomes

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{n} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_n \end{bmatrix}, \begin{bmatrix} \Sigma_x & 0 \\ 0 & \Sigma_n \end{bmatrix} \right). \qquad (7.9)$$

This Gaussian is shown as white lines on top of the actual posterior of $\mathbf{x}$ and $\mathbf{n}$ in figure 7.2a on the facing page.

An alternative approach would be to use the Gaussian approximation to the posterior that the Algonquin algorithm (presented section 4.5.1) finds. Unlike the one in (7.9), this distribution does not model the speech and the noise as independent. Figure 7.2b shows it superimposed on the actual posterior.

The main problem areas for either Gaussian as proposal distribution are the regions of space where proposal and target do not match well. Where the proposal distribution has a higher value than the target distribution, more samples will be drawn that are assigned lower weights: a waste of computational time. Conversely, where the proposal distribution is much lower than the target distribution, samples will seldom be drawn, and when they do, they are assigned high weights. In this case, the number of samples that needs to be drawn to get sufficient coverage becomes very high.

These two problems are exacerbated by high dimensionality: for every dimension, either of these cases can occur. The probability that neither problem arises for a sample decreases exponentially, so that the number of samples required increases exponentially. Both approximations shown in figure 7.2 suffer from this problem, so that it is not feasible to apply them to a 24-dimensional log-spectral problem. The next section will therefore transform the space so that the target distribution per dimension can be approximated better.

7.3 Importance sampling in a transformed space

The problem with the scheme in the previous section is the hard-to-approximate bend in the distribution of x and n given y_t. This section will overcome this by transforming the space, and then approximating the integral. Conceptually, this is similar to Myrvoll and Nakamura (2004), which was discussed in section 4.5.2. However, there the approximation was constrained to one dimension. Here, the mismatch function contains an extra variable (the phase factor), a different transformation is used, and

the approximation uses sequential importance sampling. Initially, a one-dimensional space will be considered. Section 7.3.2 will discuss how to generalise this to the multi-dimensional case and how to deal with the additional complications that arise.

7.3.1 *Single-dimensional*

In one dimension, the mismatch function is ((4.9) without indices and with y_t substituted for y)

$$\exp\left(y_t^{\log}\right) = \exp\left(x^{\log}\right) + \exp\left(n^{\log}\right) + 2\alpha \exp\left(\tfrac{1}{2}x^{\log} + \tfrac{1}{2}n^{\log}\right). \tag{7.10}$$

As discussed in section 4.2.1.1 on page 65, α is a weighted average of cosines of the angle between the signals of the frequencies in one bin, and as such constrained to be between -1 and $+1$.

Since this equality relates four variables deterministically, if three are known, then in many cases the fourth is known as well. The objective of this section is to find an approximation to $p(y_t)$ for a given observation y_t. Since four variables are linked deterministically and one is known (y_t), the integration will be over two variables. This was also the case in section 7.2. There, the obvious choice of integrating over x and n did not work out well because of the shape of the density in (x, n)-space. This section introduces a variable u that represents a pair (x, n) given y_t and α. Changing u traverses the curve in (x, n)-space, so that the bend is not a problem any more. The integral will then be over α and the new variable u.

A property of (7.10) that complicates the derivation of the transformed integral is that when three variables are known, the fourth can in some cases have two values. (7.10) is quadratic in $\exp\left(\frac{1}{2}x\right)$ and $\exp\left(\frac{1}{2}n\right)$, so it can have two solutions for x and n. This can be seen in figure 7.3 on the facing page, which shows how x and n are related for various values of α. For $n = 10, \alpha = 0.99$, for example, x has two solutions.

However, for given y_t and α, it is possible to define one variable that unambiguously identifies a point on the curve. The substitute variable will be called u with

$$u = n - x. \tag{7.11}$$

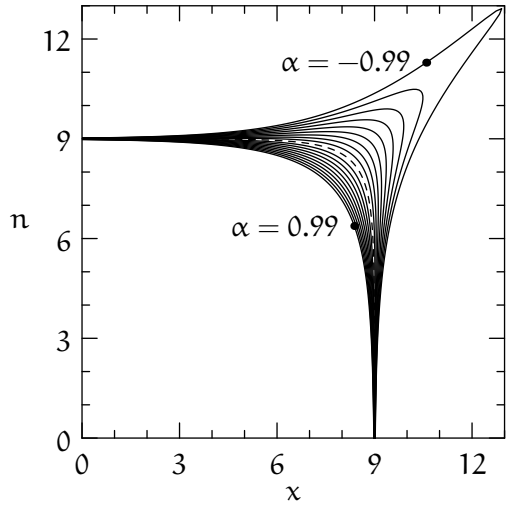


Figure 7.3 *The relation between clean speech* x *and additive noise* n *for* $y_t = 9$ *and evenly-spaced values of* α.

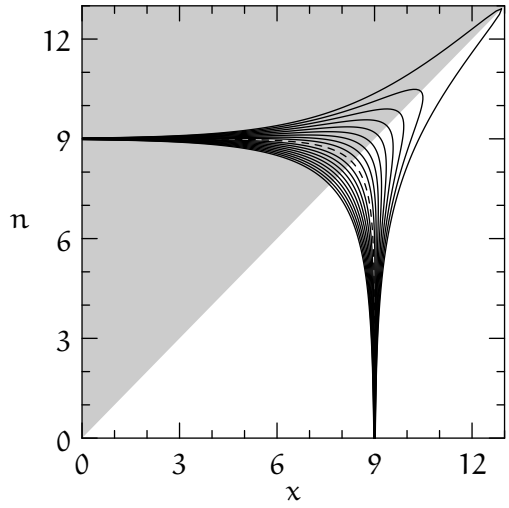


Figure 7.4 *The region of the* (x, n) *that the integral is explicitly derived for:* $x \leq n$.

The value of u is related to the signal-to-noise ratio. If u is a large negative number, x is close to y_t and n is large and negative. This represents a very low signal-to-noise ratio. The converse is true if u is a large positive number: then n is close to y_t and x is large and negative.

This substitution will be used to define an integral that yields $p(y_t)$. First, $p(y_t|\cdot)$ will be re-expressed using $p(n|\cdot)$ or $p(x|\cdot)$. Since neither of these variables is known

deterministically for all values of the other variables, the integral will be partitioned in two parts. n has one possible value given a setting for (x, y_t, α) when x is constrained to be smaller than n, which is the shaded region in figure 7.4 on the previous page. In the complementary region, x has one possible value given fixed (n, y_t, α). The likelihood can be written as a sum of both these regions:

$$p(y_t) = p(y_t, x \leq n) + p(y_t, n < x)\,. \tag{7.12}$$

Because of the symmetry between these two regions, only the derivation for the region $x \leq n$ will be given explicitly. The derivation for $n < x$ is analogous.

The additive noise that follows from setting the variables x, y_t, α will be denoted with $n(x, y_t, \alpha)$. This is deterministic in the region where $x \leq n$. The variable of the probability distribution will be changed from y_t to n. This requires multiplication by a Jacobian (see section A.1.1 on page 255 or, for example, Bishop 2006, 11.1.1). This Jacobian, the partial derivative of n with respect to y and keeping x and α fixed, will be written $\frac{\partial n(x,y,\alpha)}{\partial y}$, and be evaluated at y_t.

$$p(y_t, x \leq n|x, \alpha) = \left| \frac{\partial n(x, y, \alpha)}{\partial y}\bigg|_{y_t} \right| \cdot 1(x \leq n) \cdot p(n(x, y_t, \alpha))\,. \tag{7.13}$$

Here, $1(\cdot)$ denotes the indicator function, which evaluates to 1 if its argument is true, and 0 otherwise. $p(n(x, y_t, \alpha))$ is the probability distribution of n evaluated at $n(x, y_t, \alpha)$, the value of n corresponding to the setting of (x, y_t, α).

The evaluation of the half of the likelihood for $x \leq n$ can then be rewritten with (7.13) and by then replacing the variable of the integration by u. The predicate $x \leq n$ is equivalent to $0 \leq u$, which can be expressed using bounds on the integral.

$$\begin{aligned}
p(y_t, x \leq n) &= \int p(\alpha) \int p(x)\, p(y_t, x \leq n|x, \alpha) \mathrm{d}x \mathrm{d}\alpha \\
&= \int p(\alpha) \int p(x) \cdot \left| \frac{\partial n(x, y, \alpha)}{\partial y}\bigg|_{y_t} \right| \cdot 1(x \leq n) \cdot p(n(x, y_t, \alpha)) \mathrm{d}x \mathrm{d}\alpha \\
&= \int p(\alpha) \int_0^\infty \left| \frac{\partial x(u, y_t, \alpha)}{\partial u} \right| \cdot \left| \frac{\partial n(x, y, \alpha)}{\partial y}\bigg|_{y_t, x(u, y_t, \alpha)} \right| \\
&\qquad \cdot p(x(u, y_t, \alpha)) \cdot p(n(u, y_t, \alpha))\ \mathrm{d}u\, \mathrm{d}\alpha.
\end{aligned} \tag{7.14}$$

Here, $p(x(n, y_t, \alpha))$ is the probability distribution of x evaluated at $x(n, y_t, \alpha)$, the value of x corresponding to the setting of (n, y_t, α). Appendix F.1 on page 304 gives the derivations for the Jacobians in (7.14), and $x(u, y_t, \alpha)$ and $n(u, y_t, \alpha)$ for the region $x \leq n$. From (F.7c) the product of the Jacobians is -1. By substituting 1 for the absolute value of the product of the Jacobians in (F.7c) into (7.14), one half of the likelihood in (7.12) becomes

$$\begin{aligned} p(y_t, x \leq n) &= \int p(\alpha) \int_0^\infty \left| \frac{\partial x(u, y_t, \alpha)}{\partial u} \right| \cdot \left| \frac{\partial n(x, y, \alpha)}{\partial y} \Big|_{y_t, x(u, y_t, \alpha)} \right| \\ &\qquad p(x(u, y_t, \alpha))\, p(n(u, y_t, \alpha))\ du\, d\alpha \\ &= \int p(\alpha) \int_0^\infty p(x(u, y_t, \alpha))\, p(n(u, y_t, \alpha)) du d\alpha. \end{aligned} \tag{7.15a}$$

The integral of u is over the area where $0 \leq u$, i.e. where $x \leq n$. The equivalent integration over the region where $u < 0$ could be derived by exchanging x and n, and replacing u with $-u$. Applying this to all derivations in section F.1 on page 304 and to (7.15a) yields the other half of the likelihood in (7.12). Because of the symmetry of n and x and because the Jacobians cancel out, the result is identical to (7.15a) save for the range of u:

$$p(y_t, n < x) = \int p(\alpha) \int_{-\infty}^0 p(x(u, y_t, \alpha))\, p(n(u, y_t, \alpha)) du d\alpha. \tag{7.15b}$$

The sum of (7.15a) and (7.15b) yields the total likelihood of y_t. The integrand will be called γ.

$$\begin{aligned} p(y_t) &= p(y_t, x \leq n) + p(y_t, n < x) \\ &= \int p(\alpha) \int_{-\infty}^\infty p(x(u, y_t, \alpha))\, p(n(u, y_t, \alpha)) du d\alpha \\ &\triangleq \int p(\alpha) \int \gamma(u|\alpha) du d\alpha \qquad (7.16) \\ &\triangleq \int\int \gamma(u, \alpha) du d\alpha. \qquad (7.17) \end{aligned}$$

Thus the integral has been expressed in terms of u and α, rather than x and n as in (7.7). This derivation is exact and holds for any form of priors for the speech and

noise $p(x)$ and $p(n)$. Just like after rewriting $p(y_t)$ in (7.7), the integrand can be evaluated at any given point (u, α), assuming that $p(\alpha)$ can be evaluated, but the integral has no closed form. The outer integral is straightforward to approximate with plain Monte Carlo (see section A.4 on page 268). This works by drawing samples $\alpha^{(l)}$ from $p(\alpha)$ and averaging over sampling approximations for the inner integral given $\alpha^{(l)}$. The problem with approximating the inner integral is that it is impossible to draw samples from $\gamma(u|\alpha)$. Therefore, importance sampling is necessary. This requires a proposal distribution $\rho(u|\alpha)$ that it is possible to draw samples from, and is close to γ. Section A.4.2 on page 269 gives a detailed description of importance sampling. However, intuitively the double integral can be replaced by a summation over L samples $\alpha^{(l)}$ from $p(\alpha)$ and corresponding samples for $u^{(l)}$ drawn from $\rho(u|\alpha^{(l)})$:

$$\begin{aligned}
\int p(\alpha) \int \gamma(u|\alpha) \mathrm{d}u \mathrm{d}\alpha &= \int p(\alpha) \int \frac{\gamma(u|\alpha)}{\rho(u|\alpha)} \rho(u|\alpha) \mathrm{d}u \mathrm{d}\alpha \\
&\simeq \frac{1}{L} \sum_{l=1}^{L} \int \frac{\gamma\left(u\middle|\alpha^{(l)}\right)}{\rho\left(u\middle|\alpha^{(l)}\right)} \rho\left(u\middle|\alpha^{(l)}\right) \mathrm{d}u \qquad (7.18a) \\
&\simeq \frac{1}{L} \sum_{l=1}^{L} \frac{\gamma\left(u^{(l)}\middle|\alpha^{(l)}\right)}{\rho\left(u^{(l)}\middle|\alpha^{(l)}\right)}, \qquad (7.18b) \\
\alpha^{(l)} \sim p(\alpha)\,, \quad u^{(l)} &\sim \rho\left(u\middle|\alpha^{(l)}\right).
\end{aligned}$$

The next section will detail the shape of $\gamma(u|\alpha^{(l)})$ and find appropriate forms for $\rho(u|\alpha^{(l)})$ for it.

7.3.1.1 *The shape of the integrand*

To apply importance sampling, a proposal distribution is required, which will be tailored to the parameters of the target distribution. As discussed in section 7.2, it is important that the proposal distribution matches the integrand closely, or too many samples will be required for a good approximation. This section will find proposal distributions with well-matching shapes. The scaling of the density graphs in this section will be arbitrary.

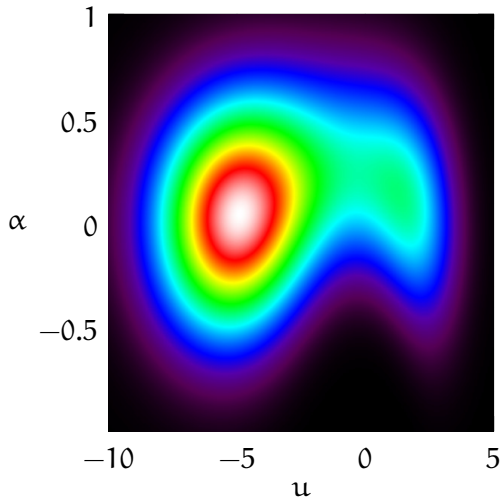


Figure 7.5 *The density of* $\gamma(u, \alpha) = p(\alpha)\,\gamma(u|\alpha)$ *for* $y_t = 9, x \sim \mathcal{N}(7, 1)\,, n \sim \mathcal{N}(4, 4)\,, \sigma_\alpha^2 = 0.13$.

So far, the derivation has not assumed any specific distributions for x or n, and has been valid for any distribution. However, for different distributions, different proposal functions are required. With Gaussian for the speech and noise, the integrand becomes

$$\begin{aligned}\gamma(u, \alpha) &= p(\alpha)\,\mathcal{N}(x(u, \alpha, y_t);\, \mu_x,\, \sigma_x^2)\mathcal{N}(n(u, \alpha, y_t);\, \mu_n,\, \sigma_n^2) \\ &\triangleq p(\alpha)\,\gamma(u|\alpha)\,. \end{aligned} \tag{7.19}$$

Figure 7.5 gives an example of the target distribution $\gamma(u, \alpha)$. As shown in (7.18b), samples for one dimension, α, can be directly drawn from the distribution for α. It is the other dimension, u, that requires importance sampling, and therefore a proposal distribution ρ. The following examples will assume the mode of $p(\alpha)$, $\alpha = 0$, and consider representative shapes for $\gamma(u|\alpha = 0)$.

$\gamma(u|\alpha)$ consists of a factor $\mathcal{N}(x(u, \alpha, y_t);\, \mu_x,\, \sigma_x^2)$ related to the clean speech and a factor $\mathcal{N}(n(u, \alpha, y_t);\, \mu_n,\, \sigma_n^2)$ related to the noise. Both terms are Gaussians, but the variables of the Gaussians (x and n) are non-linear functions of u. Figure 7.6 on the following page depicts the relationship between x and n. Different values of u represent different positions on the curve. When u is negative, n tends towards u and

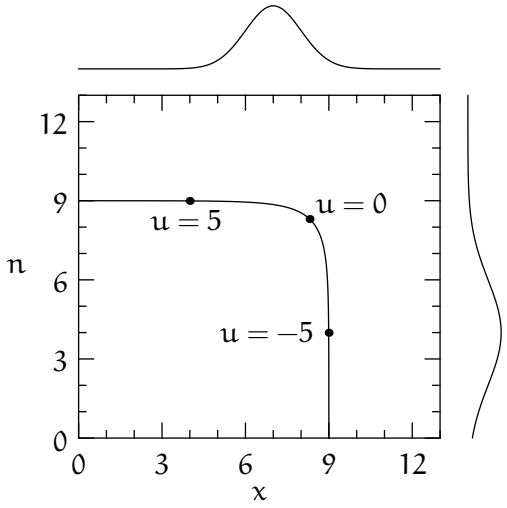


Figure 7.6 *Values of* x, n *for* $y_t = 9, \alpha = 0$*, and varying* u*. At the top of the frame* $\mathcal{N}(x;\ \mu_x,\ \sigma_x^2)$ *in* γ*; right* $\mathcal{N}(n;\ \mu_n,\ \sigma_n^2)$.

x tends towards y_t. When u is positive, x tends towards $-u$ and n tends towards y_t. Around $u = 0$ there is a soft cut-off. This graph provides an intuitive connection to noise masking schemes, which assume that either the speech or the noise dominates (Klatt 1976; Holmes and Sedgwick 1986). This would yield a curve with a sharp angle, so that in all cases either the speech of the noise is equal to the observation.

The two factors of $\gamma(u|\alpha)$ are the Gaussians depicted on top and on the side of the graph. They are evaluated at the appropriate values of x and n. When the Gaussians are plotted with respect to u, the soft cut-off leads to a Gaussian distribution that is infinitely extended on one side. Figure 7.7 illustrates the shape of the two factors. As u tends to $-\infty$, x tends to y_t. In this example, as $u \to -\infty$, $\mathcal{N}(x(u, \alpha, y_t);\ \mu_x,\ \sigma_x^2)$ therefore tends to

$$\mathcal{N}(x(u, \alpha, y_t);\ \mu_x,\ \sigma_x^2) \to \mathcal{N}(y_t;\ \mu_x,\ \sigma_x^2) = \mathcal{N}(9;\ 7,\ 1) = e^{-2}/\sqrt{2\pi} \simeq 0.054. \tag{7.20}$$

$\mathcal{N}(x(u, \alpha, y_t);\ \mu_x,\ \sigma_x^2)$ in figure 7.7 on the next page therefore is a Gaussian-like distribution with a soft cut-off on its left tail, so that it converges to a non-zero constant. Analogously, $\mathcal{N}(n(u, \alpha, y_t);\ \mu_n,\ \sigma_n^2)$ is Gaussian-like but cut off at its right tail, where

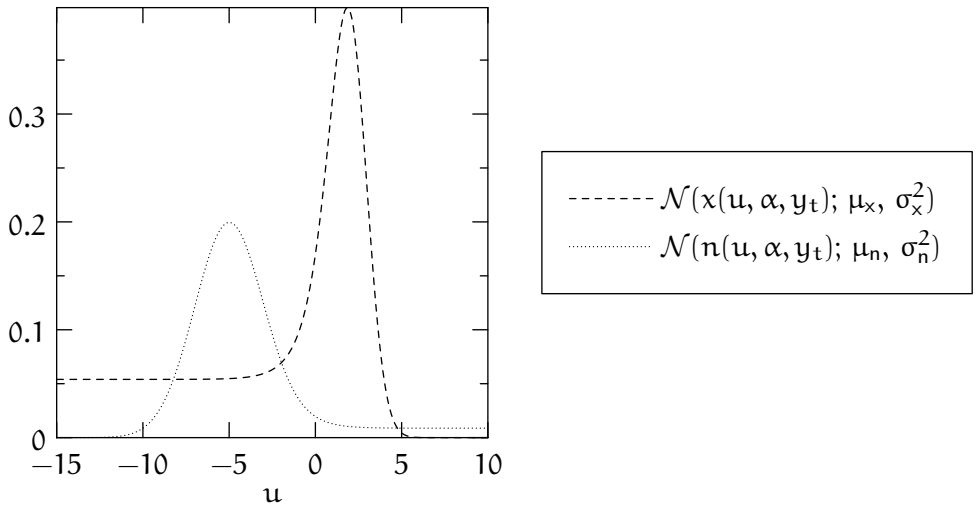


Figure 7.7 *The factors of* $\gamma(u|\alpha = 0)$ *separately. For* $y_t = 9, x \sim \mathcal{N}(7, 1), n \sim \mathcal{N}(4, 4)$.

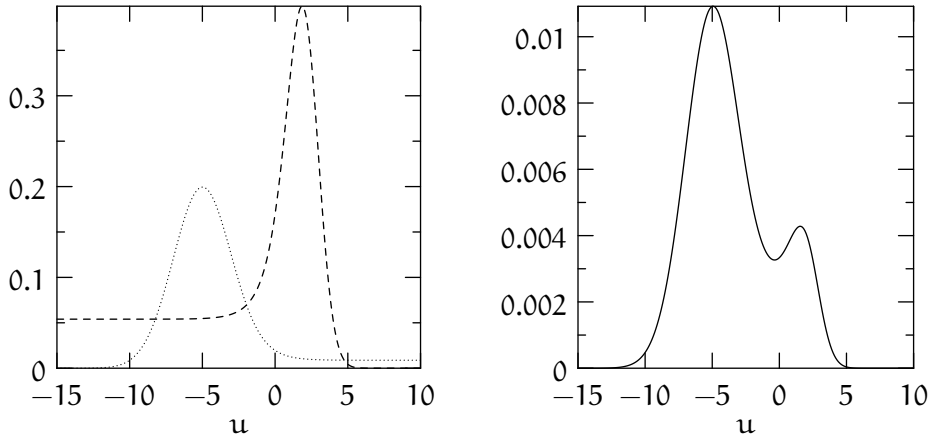
it converges to a non-zero constant.

Figure 7.8 on the following page shows examples for the three types of shape of $\gamma(u|\alpha = 0)$. Each time, the left graph contains the shape of the two factors, and the right graph their product. Figure 7.8a uses the example from figure 7.7. The integrand, the product of two cut-off Gaussians, is bimodal. When u tends to $\pm\infty$, one term of γ tends to a non-zero constant, but the other one tends to 0. γ therefore tends to 0 as well.[3]

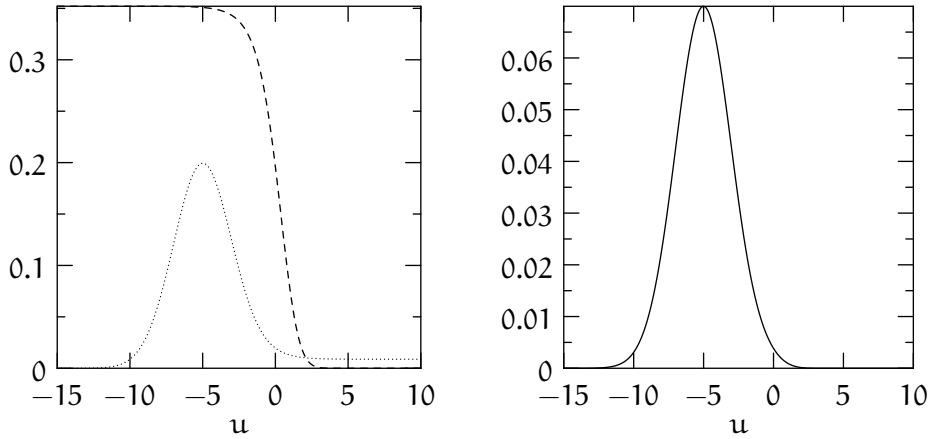
In figure 7.8b, $\mu_x > y_t$, so that the graph of $\mathcal{N}(x(u, \alpha, y_t);\ \mu_x,\ \sigma_x^2)$ is cut off right of its maximum. The product is similar to $\mathcal{N}(n(u, \alpha, y_t);\ \mu_n,\ \sigma_n^2)$, except that the right tail goes to zero. The result is almost Gaussian.

In the last example, figure 7.8c, both μ_x and μ_n are greater than y_t, so that both Gaussians are cut off before their maximum. Their product has a lop-sided Gaussian-like shape around the point of the soft cut-off, by definition $u = 0$.

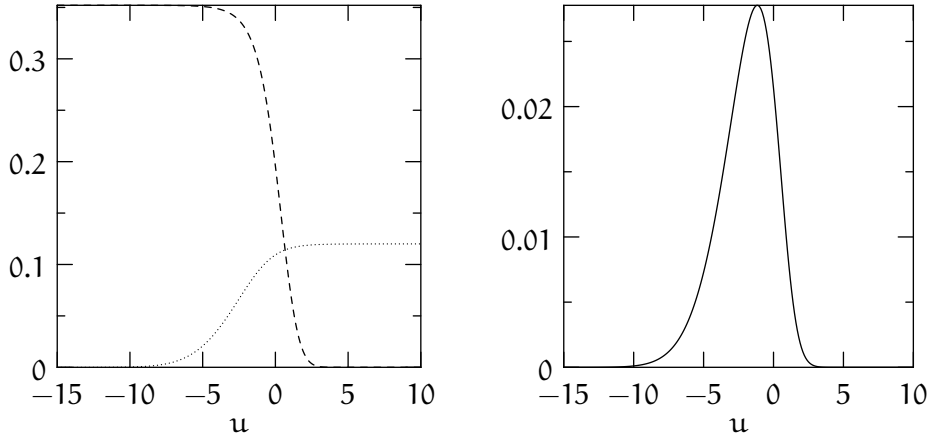
[3] This must be true also because $\int \gamma(u|\alpha = 0)\,du$ is equal to $p(y_t)$ evaluated at a point, which cannot be infinite if either σ_x^2 or σ_n^2 is non-zero.



(a) $\mathcal{N}(x(u,0,y_t);\,7,\,1), \mathcal{N}(n(u,0,y_t);\,4,\,4)$; their product $\gamma(u|\alpha)$.



(b) $\mathcal{N}(x(u,0,y_t);\,9.5,\,1), \mathcal{N}(n(u,0,y_t);\,4,\,4)$; their product $\gamma(u|\alpha)$.



(c) $\mathcal{N}(x(u,0,y_t);\,9.5,\,1), \mathcal{N}(n(u,0,y_t);\,10,\,10)$; their product $\gamma(u|\alpha)$.

Figure 7.8 $\gamma(u|\alpha = 0)$ *for different cases: left the two factors, right their product.*

7.3.1.2 *Importance sampling from the integrand*

A proposal distribution that it is simple to draw a sample from is a Gaussian mixture model. To find a mixture of Gaussian densities that approximates γ, the three types of γ from figure 7.8 are considered separately. Because γ has these different shapes, the approximation will be different for each of these cases. the proposal distributions must be defined over u, so that it is useful to find the value of u corresponding to a specific setting of x, α, and y_t (and n, α, and y_t). This will be denoted $u(x, \alpha, y_t)$ (and $u(n, \alpha, y_t)$). The expressions are derived in appendix F.4, (F.25) and (F.26f).

Figure 7.9 on the next page shows the proposal distributions. Their magnitudes are scaled to equalise the areas under the target and proposal densities. The proposal distribution is chosen differently depending on the mean of the terms of γ as follows:

1. $\mu_x < y_t$ and $\mu_n < y_t$. This produces a shape of γ as in figure 7.8a. The shape of γ being close to the product of two Gaussians, a Gaussian mixture model with the parameters of these two Gaussians would form a good proposal distribution.

   As a proposal distribution, a mixture of two Gaussians is chosen with means at the approximate modes, and covariances set to σ_x^2 and σ_n^2, respectively. These Gaussians are illustrated in figure 7.9a. They approximate the terms $\mathcal{N}(n(u, \alpha, y_t);\ \mu_n,\ \sigma_n^2)$ and $\mathcal{N}(x(u, \alpha, y_t);\ \mu_x,\ \sigma_x^2)$ with

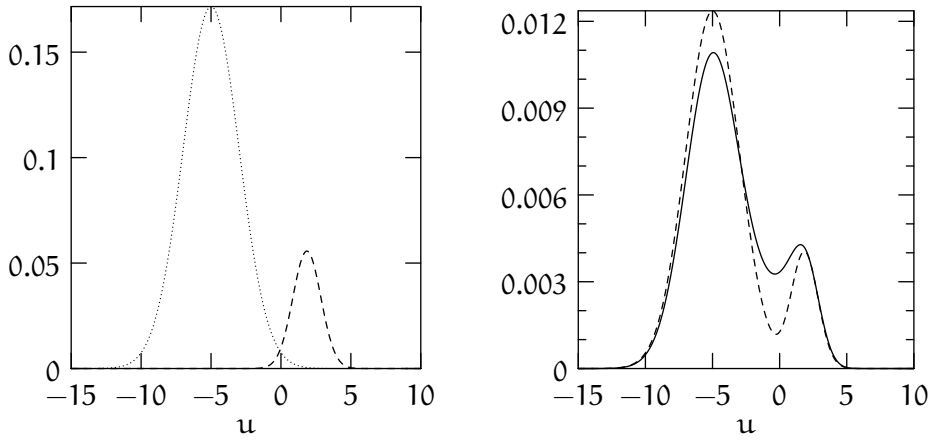
   $$\mathcal{N}\big(u;\ u(\mu_n, \alpha, y_t),\ \sigma_n^2\big); \tag{7.21a}$$

   $$\mathcal{N}\big(u;\ u(\mu_x, \alpha, y_t),\ \sigma_x^2\big). \tag{7.21b}$$

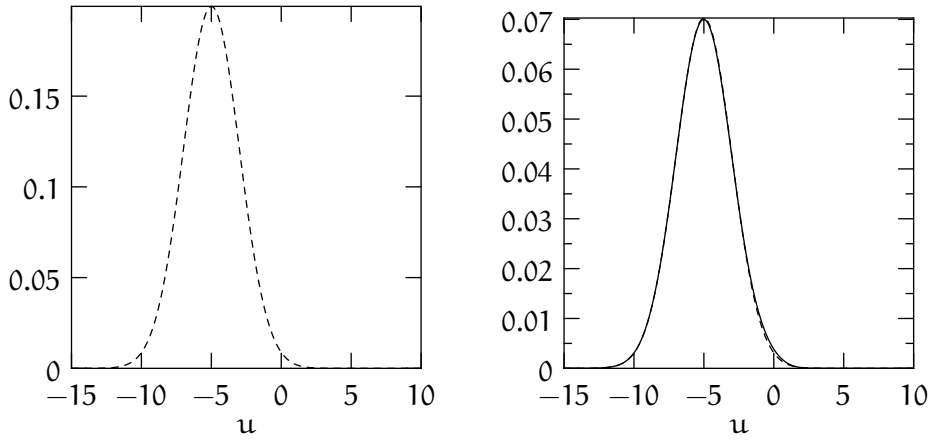
   As was seen in figure 7.7 on page 197, each Gaussian is essentially scaled by the extended tail of the other one. The weights of the Gaussians of the proposal distribution can be set to the value that the tail of the other one converges to, which can be computed as in (7.20):

   $$\pi_n \propto \mathcal{N}(y_t;\ \mu_x,\ \sigma_x^2); \tag{7.22a}$$

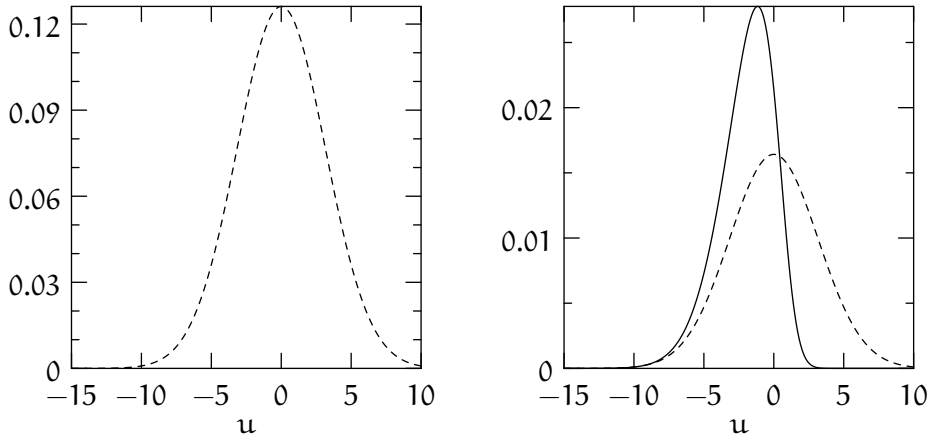
   $$\pi_x \propto \mathcal{N}(y_t;\ \mu_n,\ \sigma_n^2). \tag{7.22b}$$



(a) Proposal (GMM) for $y_t = 9, x \sim \mathcal{N}(7,1)\,, n \sim \mathcal{N}(4,4)$.



(b) Proposal (one Gaussian) for $y_t = 9, x \sim \mathcal{N}(9.5,1)\,, n \sim \mathcal{N}(4,4)$.



(c) Proposal (one Gaussian) for $y_t = 9, x \sim \mathcal{N}(9.5,1)\,, n \sim \mathcal{N}(10,10)$.

Figure 7.9 *The proposal distribution for $\gamma(u|\alpha)$ for different cases: left the components of the proposal distribution, right γ (solid line) and proposal distribution ρ (dashed line, scaled so the area under the curve matches).*

These weights are normalised so that they sum to 1. The distribution becomes

$$p(u) = \pi_n \mathcal{N}\big(u;\, u(\mu_n, \alpha, y_t),\, \sigma_n^2\big) + \pi_x \mathcal{N}\big(u;\, u(\mu_x, \alpha, y_t),\, \sigma_x^2\big). \quad (7.23)$$

2. $\mu_x > y_t$ and $\mu_n < y_t$ (or its mirror image, $\mu_x < y_t$ and $\mu_n > y_t$). Figure 7.8b on page 198 has shown that $\mathcal{N}(x(u, \alpha, y_t);\, \mu_x,\, \sigma_x^2)$ is cut off before its peak, and converges to its maximum in the limit as $u \to -\infty$. This results in a Gaussian distribution except for one tail. The proposal distribution is therefore set to this Gaussian:

$$p(u) = \mathcal{N}\big(u;\, u(\mu_n, \alpha, y_t),\, \sigma_n^2\big). \quad (7.24)$$

Figure 7.9b on the facing page shows the near-perfect match of this proposal distribution.

3. $\mu_n > y_t$ and $\mu_x > y_t$. Both terms of γ are cut off before their peaks, resulting in a shape as in figure 7.8c on page 198. The product is a distribution around $u = 0$ with Gaussian-like tails, one derived from $\mathcal{N}(n(u, \alpha, y_t);\, \mu_n,\, \sigma_n^2)$ and another one derived from $\mathcal{N}(x(u, \alpha, y_t);\, \mu_x,\, \sigma_x^2)$. The proposal distribution is therefore set to a Gaussian with mean 0. Its variance is set to the largest of the variances of the speech and the noise:

$$p(u) = \mathcal{N}\big(u;\, 0,\, \max(\sigma_n^2, \sigma_x^2)\big). \quad (7.25)$$

As figure 7.9c on the facing page shows, this provides good coverage but overestimation on part of the space. This means that some samples will receive a very low weight.

Thus, by transforming the space of the integration from (x, n) to (u, α), much better proposal distributions for importance sampling can be found than in (x, n)-space, like in section 7.2. The sample weights will therefore vary less, so that good approximations to the integral will be found with a much smaller number of samples. The next section will extend the techniques applied in this chapter to the multi-dimensional case.

7.3.1.3 *Related method*

At the same time as van Dalen and Gales (2010*a*), a similar method was proposed (Hershey *et al.* 2010). There are three differences. First, the model is different: no mel-bins are used, so that the phase factor model in this work effectively is replaced by a per-frequency cosine of the angle between speech and noise in the complex plane (see section 4.2.1.1). Also, the variable transformation is different. The biggest difference, however, is that the method treats dimensions as independent. It therefore fails to take into account the correlations in the distributions of the speech and noise. The next section will introduce multi-dimensional sampling. The strategy it uses may also apply to the method in Hershey *et al.* (2010).

7.3.2 *Multi-dimensional*

In this chapter, the log-spectral domain is used. This has the advantage that the interaction between the speech and the noise can be modelled separately per dimension. However, there are strong correlations between log-spectral coefficients. Therefore, the Gaussian priors for the speech and the noise have full covariance matrices. This section will build on the techniques used in the previous section. It will generalise the transformation of the integral to multi-dimensional $(\mathbf{u}, \boldsymbol{\alpha})$. Rather than standard importance sampling, it will then apply sequential importance resampling to approximate the integral.

The relations between the single-dimensional variables in the previous section hold per dimension for the multi-variate case. The substitute variable $\mathbf{u}$ that is introduced to represent a point $(\mathbf{x}, \mathbf{n})$ given observation $\mathbf{y}_t$ and phase factor vector $\boldsymbol{\alpha}$ is therefore defined as

$$\mathbf{u} = \mathbf{n} - \mathbf{x}. \tag{7.26}$$

The expressions for $\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y})$ and $\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y})$, the values for the speech and noise that result from setting $(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y})$, are given in appendix F.2.

There was a complication in transforming the one-dimensional integral in section 7.3.1 from (x, n) to (u, α): for some x, multiple values for n were possible, and vice versa. Because transforming the integral needed a deterministic link, the integral was split into two parts, for two regions of (x, n)-space. In the multi-dimensional case it is necessary to do this for each of the dimensions. Appendix F.2 gives the full derivation. The integration is therefore first split up into conditional distributions per dimension i, and then into regions. The integrals for the two regions over (x_i, n_i) are rewritten as an integral over (u_i, α_i). Collapsing the dimensions (see equation (F.16)) then yields an unsurprising generalisation of (7.17):

$$p(\mathbf{y}_t) = \int p(\boldsymbol{\alpha}) \int p(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t))\, p(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t)) \mathrm{d}\mathbf{u} \mathrm{d}\boldsymbol{\alpha} \tag{7.27a}$$

$$\triangleq \int \int \gamma(\mathbf{u}, \boldsymbol{\alpha}) \mathrm{d}\mathbf{u} \mathrm{d}\boldsymbol{\alpha}, \tag{7.27b}$$

and it is convenient to factorise the integrand $\gamma(\mathbf{u}, \boldsymbol{\alpha})$ as

$$\gamma(\mathbf{u}, \boldsymbol{\alpha}) = \gamma(\boldsymbol{\alpha}) \gamma(\mathbf{u}|\boldsymbol{\alpha})\,, \tag{7.27c}$$

$$\gamma(\boldsymbol{\alpha}) = p(\boldsymbol{\alpha})\,; \tag{7.27d}$$

$$\gamma(\mathbf{u}|\boldsymbol{\alpha}) = p(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t))\, p(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t))\,. \tag{7.27e}$$

This derivation is valid whatever the form of the speech and noise priors, $p(\mathbf{x})$ and $p(\mathbf{n})$. In this work, they are Gaussians with (repeated from (7.3a) and (7.3b))

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)\,; \qquad \mathbf{n} \sim \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)\,, \tag{7.28}$$

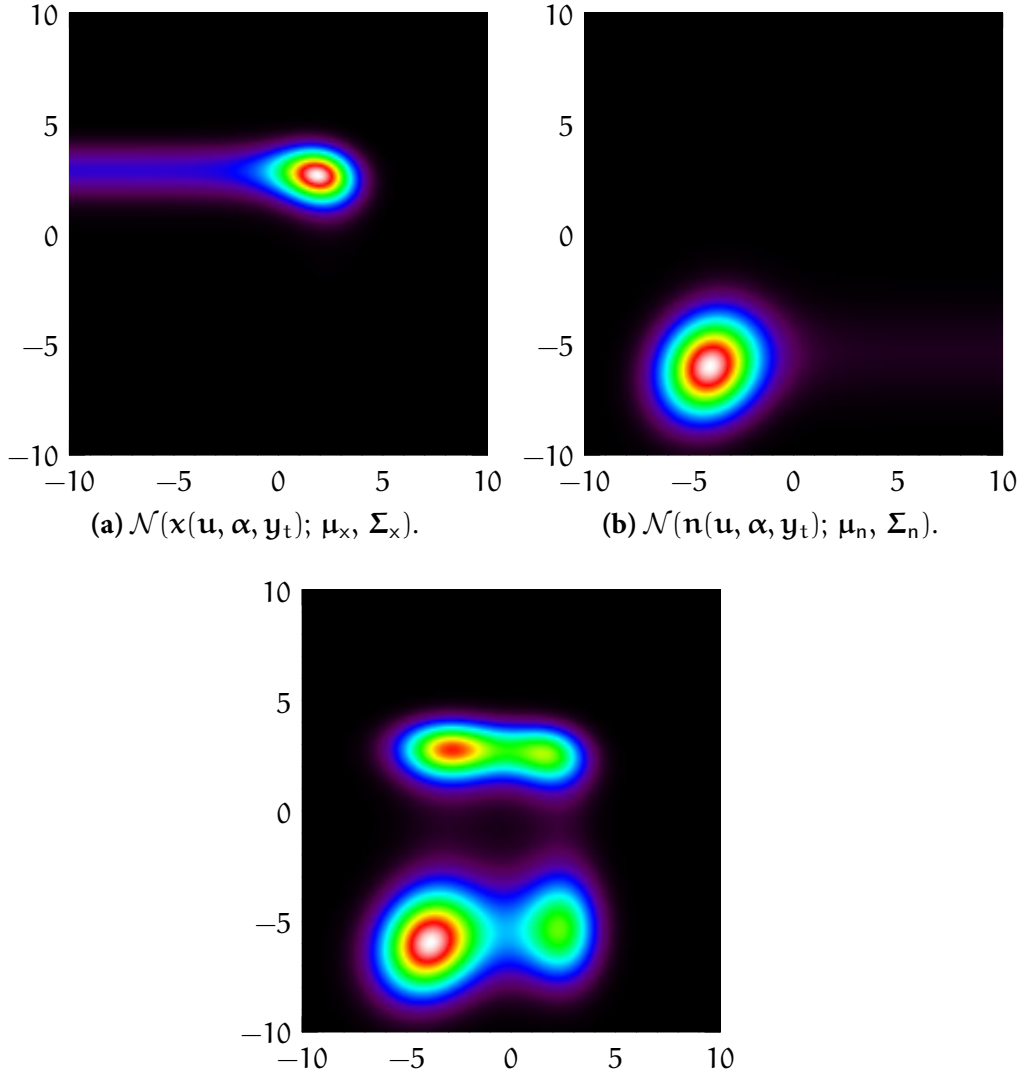
with full covariance matrices $\boldsymbol{\Sigma}_x$ and $\boldsymbol{\Sigma}_n$. $\gamma(\mathbf{u}|\boldsymbol{\alpha})$ then becomes

$$\gamma(\mathbf{u}|\boldsymbol{\alpha}) = \mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t);\, \boldsymbol{\mu}_x,\, \boldsymbol{\Sigma}_x)\, \mathcal{N}(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t);\, \boldsymbol{\mu}_n,\, \boldsymbol{\Sigma}_n)\,, \tag{7.29a}$$

so that

$$\gamma(\mathbf{u}, \boldsymbol{\alpha}) = p(\boldsymbol{\alpha})\, \mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t);\, \boldsymbol{\mu}_x,\, \boldsymbol{\Sigma}_x)\, \mathcal{N}(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t);\, \boldsymbol{\mu}_n,\, \boldsymbol{\Sigma}_n)\,. \tag{7.29b}$$

To approximate this integral, conventional importance sampling could again be used. Just like in section 7.2, intuitively, the integration over two variables can be ap-



(a) $\mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t);\, \boldsymbol{\mu}_x,\, \boldsymbol{\Sigma}_x)$.

(b) $\mathcal{N}(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t);\, \boldsymbol{\mu}_n,\, \boldsymbol{\Sigma}_n)$.

(c) The product: $\gamma(\mathbf{u}|\boldsymbol{\alpha})$.

Figure 7.10 *The integrand $\gamma(\mathbf{u}|\boldsymbol{\alpha})$ for $\boldsymbol{\alpha} = \mathbf{0}$: the two factors, and their product.*

proximated by drawing samples $(\mathbf{u}^{(l)}, \boldsymbol{\alpha}^{(l)})$ from a proposal distribution ρ:

$$\begin{aligned}\int\int \gamma(\mathbf{u}, \boldsymbol{\alpha}) \mathrm{d}\mathbf{u}\mathrm{d}\boldsymbol{\alpha} &= \int\int \frac{\gamma(\mathbf{u}, \boldsymbol{\alpha})}{\rho(\mathbf{u}, \boldsymbol{\alpha})} \rho(\mathbf{u}, \boldsymbol{\alpha}) \mathrm{d}\mathbf{u}\mathrm{d}\boldsymbol{\alpha} \\ &\simeq \frac{1}{L} \sum_{l=1}^{L} \frac{\gamma(\mathbf{u}^{(l)}, \boldsymbol{\alpha}^{(l)})}{\rho(\mathbf{u}^{(l)}, \boldsymbol{\alpha}^{(l)})}, \qquad (\mathbf{u}^{(l)}, \boldsymbol{\alpha}^{(l)}) \sim \rho. \end{aligned} \tag{7.30}$$

Figure 7.10 illustrates how the shape of the integrand $\gamma(\mathbf{u}|\boldsymbol{\alpha} = \mathbf{0})$ generalises to

two dimensions of $\mathbf{u}$. The principles are the same as the one-dimensional case in figure 7.8a on page 198. Figure 7.10a contains the factor of γ deriving from the speech prior, $\mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t); \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$, and figure 7.10b the same for the noise prior. They are again Gaussians with a soft cut-off, this time in two directions. By choosing the slightly contrived parameter setting

$$\mathbf{x} \sim \mathcal{N}\left(\begin{bmatrix} 7 \\ 6.3 \end{bmatrix}, \begin{bmatrix} 1 & -0.1 \\ -0.1 & 0.5 \end{bmatrix}\right); \quad \mathbf{n} \sim \mathcal{N}\left(\begin{bmatrix} 5 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 & 0.3 \\ 0.3 & 2 \end{bmatrix}\right); \quad \mathbf{y}_t = \begin{bmatrix} 9 \\ 9 \end{bmatrix}, \tag{7.31}$$

the product of the two factors, in figure 7.10c, turns out to have four maxima. In general, for d dimensions, the integrand can have 2^d modes. Even though this may be unlikely to occur often in practice, it is hard to formulate a proposal distribution for importance sampling. The proposal distribution would need to be close to the integrand, and it must be possible to draw samples from it. A mixture of Gaussians, for example, could need as many components as the integrand has maxima. However, rather than applying normal importance sampling, the integrand will be factorised in dimensions for sequential importance sampling.

7.3.2.1 Per-dimension sampling

Rather than sampling from all dimensions at once, sequential importance sampling (see appendix A.4.3 on page 272) can be used, which samples dimension per dimension. Fundamentally, it is an instantiation of importance sampling for multiple dimensions. First, it draws a set of samples from a distribution over the first dimension, and assigns the samples a weight. Then, for each dimension it extends every partial sample with a value drawn given the value for previous dimensions of that sample. The advantage of this formulation is that between dimensions it allows for resampling: duplicating higher-probability samples from the set and removing lower-probability ones. This concentrates the samples on the most relevant areas of the space.

To be able to apply sequential importance sampling, the target density needs to be factorised into dimensions. If the feature space is d-dimensional, the integration

is over $2d$ dimensions: $\alpha_1, \ldots, \alpha_d, u_1, \ldots, u_d$. It is important to realise that there is no need for the factors to represent conditional probability distributions, normalised or not. It is true that the most informative weights after each dimension i would arise if factors $1 \ldots i$ combined to form the (potentially unnormalised) marginal of partial sample $u_{1:i}$. Resampling would then be most effective. By definition, the factors would be (unnormalised) conditionals. However, this is not a requirement, and this work will compare two different factorisations, both of which should be close to the actual conditionals.

Since the phase factor coefficients are independent (see section 4.2.1.1 on page 65), an obvious factorisation for $\gamma(\boldsymbol{\alpha})$ is

$$\gamma(\boldsymbol{\alpha}) = \gamma_1(\alpha_1) \cdots \gamma_d(\alpha_d) = \prod_{i=1}^{d} \gamma_i(\alpha_i), \qquad \gamma_i(\alpha_i) = p(\alpha_i)\,. \tag{7.32a}$$

Since the u_i are not independent, the factors of $\gamma(\mathbf{u}|\boldsymbol{\alpha})$ must take the full partial sample into account:

$$\begin{aligned}\gamma(\mathbf{u}|\boldsymbol{\alpha}) &= \gamma_1(u_1|\alpha_1)\,\gamma_2(u_2|u_1, \boldsymbol{\alpha}_{1:2})\,\gamma_3(u_3|\mathbf{u}_{1:2}, \boldsymbol{\alpha}_{1:3}) \cdots \gamma_d(u_d|\mathbf{u}_{1:d-1}, \boldsymbol{\alpha}_{1:d}) \\ &= \prod_{i=1}^{d} \gamma_i(u_i|\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i})\,. \end{aligned} \tag{7.32b}$$

Again, the notation of these factors $\gamma_i(u_i|\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i})$ does not mean that they are necessarily related to conditional distributions. They can be any function of the variables before and after the bar, as long as their product $\gamma(\mathbf{u}|\boldsymbol{\alpha})$ yields the correct result. Indeed, the next subsections will present two choices for the factorisation. Both apply the same factorisation to both terms of γ in parallel. The first, which will be called "postponed factorisation", each factor only incorporates the dimensions that are sampled from, and leave other terms for later in the process. The second, which will be called "quasi-conditional factorisation", factorises the two Gaussians separately into conditional distributions per dimension.

The form of the speech and noise prior in this work are standard Gaussians, so

that the factorisation of $\gamma(\mathbf{u}|\boldsymbol{\alpha})$ must satisfy (combining (7.29a) and (7.32b)):

$$\prod_{i=1}^{d} \gamma_i(u_i|\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) = \mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t);\, \boldsymbol{\mu}_x,\, \boldsymbol{\Sigma}_x)\, \mathcal{N}(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t);\, \boldsymbol{\mu}_n,\, \boldsymbol{\Sigma}_n)\,. \quad (7.33)$$

7.3.2.2 *Postponed factorisation*

The Gaussian terms in (7.33) can be written as products with every element of the precision matrices. The precision matrix is the inverse covariance: $\boldsymbol{\Lambda}_x = \boldsymbol{\Sigma}_x^{-1}$. Its elements are denoted $\lambda_{x,ij}$. It is possible to postpone the terms until both dimensions to be related are known. This requires some manipulation, the details of which are in appendix F.3. The integrand is rewritten in (F.20), and the factors γ_i are defined as (from (F.21))

$$\begin{aligned}
&\gamma_1(u_1|\alpha_1) \\
&\quad = |2\pi\boldsymbol{\Sigma}_y|^{-\frac{1}{2}} |2\pi\boldsymbol{\Sigma}_x|^{-\frac{1}{2}} \exp\Big(-\tfrac{1}{2}\lambda_{n,11}(n(u_1, \alpha_1, y_{t,1}) - \mu_{n,1})^2 \\
&\qquad\quad - \tfrac{1}{2}\lambda_{x,11}(x(u_1, \alpha_1, y_{t,1}) - \mu_{x,1})^2\Big); \qquad (7.34a) \\
&\gamma_i(u_i|\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) \\
&\quad = \exp\Big(-\tfrac{1}{2}\lambda_{x,ii}(x(u_i, \alpha_i, y_{t,i}) - \mu_{x,i})^2 - (x(u_i, \alpha_i, y_{t,i}) - \mu_{x,i})\, \nu_{x,i} \\
&\qquad - \tfrac{1}{2}\lambda_{n,ii}(n(u_i, \alpha_i, y_{t,i}) - \mu_{n,i})^2 - (n(u_i, \alpha_i, y_{t,i}) - \mu_{n,i})\, \nu_{n,i}\Big), \qquad (7.34b)
\end{aligned}$$

where the terms containing coordinates of lower dimensions $\mathbf{u}_{1:i-1}$ are defined as (in (F.19b))

$$\nu_{x,i} = \sum_{j=1}^{i-1} \lambda_{x,ij}(x(u_j, \alpha_j, y_{t,j}) - \mu_{x,j})\,; \quad \nu_{n,i} = \sum_{j=1}^{i-1} \lambda_{n,ij}(n(u_j, \alpha_j, y_{t,j}) - \mu_{n,j})\,. \quad (7.35)$$

Note again that $\gamma_i(u_i|\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i})$ is not a conditional distribution.

At every dimension, a proposal distribution ρ_i is required for importance sampling. This distribution needs to have a shape similar to γ_i. There is, however, no need to match the amplitude of γ_i. This is convenient when rewriting γ_i to find the shape of the distribution, as appendix F.3 does.

The factors in (7.34b) turn out to be proportional to two Gaussian distributions that are functions of $x(u_i, \alpha_i, y_{t,i})$ and $n(u_i, \alpha_i, y_{t,i})$ (from (F.23)):

$$\gamma_i(u_i|\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}) \propto \mathcal{N}\left(x(u_i, \alpha_i, y_{t,i});\, \mu_{x,i} - \frac{\nu_{x,i}}{\lambda_{x,ii}},\, \lambda_{x,ii}^{-1}\right) \cdot \mathcal{N}\left(n(u_i, \alpha_i, y_{t,i});\, \mu_{n,i} - \frac{\nu_{n,i}}{\lambda_{n,ii}},\, \lambda_{n,ii}^{-1}\right). \tag{7.36}$$

This expression has the same shape as the one-dimensional integrand in (7.19) in section 7.3.1, and the same proposal distribution as discussed in section 7.3.1.2 can be used.

7.3.2.3 *Quasi-conditional factorisation*

Alternatively, both Gaussians in (7.29b) could be decomposed into the marginal of the first dimension, the marginal of the second dimension given the first, and so on. Appendix A.1.3 decomposes a Gaussian distribution over two vectors into the marginal of the first vector times the distribution of the second conditional on the first. However, since the density γ is the product of two Gaussians with two different non-linear variables, its factors are not normalised or proportional to conditional probabilities.

Crucially, the derivation in (A.7) does not rely on the input variable or normalisation. It is therefore possible to find a factorisation of both speech and noise Gaussians in parallel, even if the factors are not exactly conditionals. For this, both terms of $\gamma_{1:i}$ is factorised recursively as (only the left-hand term is shown)

$$\mathcal{N}(\mathbf{x}_{1:i};\, \boldsymbol{\mu}_{x,1:i},\, \boldsymbol{\Sigma}_{x,1:i,1:i}) = \mathcal{N}(\mathbf{x}_{1:i-1};\, \boldsymbol{\mu}_{x,1:i-1},\, \boldsymbol{\Sigma}_{x,1:i-1,1:i-1}) \cdot \mathcal{N}\big(x_i;\, \mu_{x,i|1:i-1}(\mathbf{x}_{1:i-1}),\, \sigma^2_{x,i|1:i-1}\big), \tag{7.37a}$$

where the parameters for $\mathbf{x}_i$ dependent on $\mathbf{x}_{1:i-1}$ are

$$\mu_{x,i|1:i-1}(\mathbf{x}_{1:i-1}) = \mu_{x,i} - \boldsymbol{\Sigma}_{x,i,1:i-1}\left[\boldsymbol{\Sigma}_{x,1:i-1,1:i-1}\right]^{-1}(\mathbf{x}_{1:i-1} - \boldsymbol{\mu}_{x,1:i-1})\,; \tag{7.37b}$$

$$\sigma^2_{x,i|1:i-1} = \sigma^2_{x,i,i} - \boldsymbol{\Sigma}_{x,i,1:i-1}\left[\boldsymbol{\Sigma}_{x,1:i-1,1:i-1}\right]^{-1}\boldsymbol{\Sigma}_{x,1:i-1,i}. \tag{7.37c}$$

Note that the variance $\sigma^2_{x,i|1:i-1}$ is not a function of $\mathbf{x}$, so that it needs to be computed only once for all samples. Also, in computing the inverses of $\boldsymbol{\Sigma}_{x,1:i,1:i}$ for $i = 1 \ldots d$,

their structure can be exploited. A block-wise inversion can be used. Block-wise inversion is a technique often applied to take advantage of known structure in blocks of the matrix, for example, when a block is diagonal. The trick here, however, is that the intermediate results are useful. If the the inverse of a matrix with one column removed from the right and one row removed from the bottom, $\boldsymbol{\Sigma}_{x,1:i-1,1:i-1}$ is known, the inverse of the matrix with the extra row and column, $\boldsymbol{\Sigma}_{x,1:i,1:i}$, can be computed in $\mathcal{O}(i^2)$ time. An incremental implementation that yields $[\boldsymbol{\Sigma}_{x,1:i,1:i}]^{-1}$ for $i = 1 \ldots d$ thus has a time complexity of only $\mathcal{O}(d^3)$, the same as inverting only the full covariance matrix with a conventional approach.

The fully factorised formulation of the left-hand term of γ is

$$\mathcal{N}(\mathbf{x}_{1:d};\, \boldsymbol{\mu}_{x,1:d},\, \boldsymbol{\Sigma}_{x,1:d,1:d}) = \mathcal{N}\big(x_1;\, \mu_{x,1},\, \sigma^2_{x,1,1}\big) \prod_{i=2}^{d} \mathcal{N}\big(x_i;\, \mu_{x,i|1:i-1}(\mathbf{x}_{1:i-1}),\, \sigma^2_{x,i|1:i-1}\big). \tag{7.38}$$

The analogous factorisation can be applied to the right-hand term of $\gamma(\mathbf{u})$. (7.29a) can then be factorised

$$\begin{aligned}\gamma(\mathbf{u}|\boldsymbol{\alpha}) &= \mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t);\, \boldsymbol{\mu}_x,\, \boldsymbol{\Sigma}_x)\, \mathcal{N}(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t);\, \boldsymbol{\mu}_n,\, \boldsymbol{\Sigma}_n) \\ &= \mathcal{N}\big(x(u_1, \alpha_1, y_1);\, \mu_1,\, \sigma^2_{1,1}\big)\mathcal{N}\big(n(u_1, \alpha_1, y_1);\, \mu_1,\, \sigma^2_{1,1}\big) \\ &\quad \prod_{i=2}^{d} \mathcal{N}\Big(x(u_i, \alpha_i, y_i);\, \mu_{x,i|1:i-1}\big(\mathbf{x}(\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}, \mathbf{y}_{1:i-1})\big),\, \sigma^2_{x,i|1:i-1}\Big) \\ &\qquad \mathcal{N}\Big(n(u_i, \alpha_i, y_i);\, \mu_{n,i|1:i-1}\big(\mathbf{n}(\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}, \mathbf{y}_{1:i-1})\big),\, \sigma^2_{n,i|1:i-1}\Big).\end{aligned} \tag{7.39}$$

The factors of γ then become[4]

$$\begin{aligned}&\gamma_i(u_i|\mathbf{u}_{1:i-1}) \\ &\quad = \mathcal{N}\Big(x(u_i, \alpha_i, y_i);\, \mu_{x,i|1:i-1}\big(\mathbf{x}(\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}, \mathbf{y}_{1:i-1})\big),\, \sigma^2_{x,i|1:i-1}\Big) \\ &\qquad \mathcal{N}\Big(n(u_i, \alpha_i, y_i);\, \mu_{n,i|1:i-1}\big(\mathbf{n}(\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}, \mathbf{y}_{1:i-1})\big),\, \sigma^2_{n,i|1:i-1}\Big),\end{aligned} \tag{7.40}$$

[4] This formulation assumes (in γ_1) that multiplying a 1×0 matrix by a 0×0 matrix by a 0×1 matrix yields a 1×1 matrix $[0]$.

where $\mu_{x,i|1:i-1}(\mathbf{x}_{1:i-1})$ and $\sigma^2_{x,i|1:i-1}$ are defined in (7.37a). Again, the factors have the form of density as the one-dimensional γ in (7.19), so that the proposal distribution given in section 7.3.1.2 can be used.

7.3.2.4 *Applying sequential importance resampling*

Whichever factorisation of $\gamma(\mathbf{u}|\boldsymbol{\alpha})$ is chosen, the application of sequential importance resampling is the same. The integral $\int\int \gamma(\mathbf{u}, \boldsymbol{\alpha}) \mathrm{d}\boldsymbol{\alpha} \mathrm{d}\mathbf{u}$, the value of interest, is the normalisation constant of $\gamma(\mathbf{u}, \boldsymbol{\alpha})$, which will be called Z. To find Z by stepping through dimensions, it can be expressed as a sequence of incremental normalisation constants Z_i/Z_{i-1} (see appendix A.4.3). Given a sample set $\left\{\left(\mathbf{u}^{(l)}_{1:i-1}, \boldsymbol{\alpha}^{(l)}_{1:i-1}\right)\right\}$, the approximation of the incremental normalisation constant is[5] (when resampling is used)

$$\frac{\widetilde{Z_i}}{Z_{i-1}} = \frac{1}{L}\sum_{l=1}^{L} \frac{\gamma_i\left(\alpha_i^{(l)}\right)}{\rho_i\left(\alpha_i^{(l)}\right)} \frac{\gamma_i\left(u_i^{(l)}\middle|\mathbf{u}^{(l)}_{1:i-1}, \boldsymbol{\alpha}^{(l)}_{1:i}\right)}{\rho_i\left(u_i^{(l)}\middle|\mathbf{u}^{(l)}_{1:i-1}, \boldsymbol{\alpha}^{(l)}_{1:i}\right)}, \tag{7.41}$$

where samples $\alpha_i^{(l)}$ are drawn from proposal distribution $\rho_i\left(\alpha_i^{(l)}\right)$ and samples $u_i^{(l)}$ from the appropriate $\rho_i\left(u_i\middle|\mathbf{u}^{(l)}_{1:i-1}, \boldsymbol{\alpha}^{(l)}_{1:i}\right)$.

The shape of the density $\gamma\left(u_i\middle|\mathbf{u}^{(l)}_{1:i-1}, \boldsymbol{\alpha}^{(l)}_{1:i}\right)$ depends on the current partial sample $\left(\mathbf{u}^{(l)}_{1:i-1}, \boldsymbol{\alpha}^{(l)}_{1:i}\right)$ and the type of factorisation. The factorisations in sections 7.3.2.2 and 7.3.2.3 both result in factors of the form

$$\begin{aligned}\gamma_i(u_i|\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) = \mathcal{N}\left(x(u_i, \alpha_i, y_{t,i});\, \hat{\mu}_{x,i},\, \hat{\sigma}^2_{x,i}\right) \\ \cdot \mathcal{N}\left(n(u_i, \alpha_i, y_{t,i});\, \hat{\mu}_{n,i},\, \hat{\sigma}^2_{n,i}\right),\end{aligned} \tag{7.42}$$

where the parameters $(\hat{\mu}_{x,i}, \hat{\sigma}^2_{x,i}, \hat{\mu}_{n,i}, \hat{\sigma}^2_{n,i})$ depend on the type of factorisation and the current partial sample $\left(\mathbf{u}^{(l)}_{1:i-1}, \boldsymbol{\alpha}^{(l)}_{1:i}\right)$. Appropriate proposal distributions for this type of density have been discussed in section 7.3.1.2. These distributions over u_i take the form of one Gaussian or a mixture of two. They are therefore straightforward to draw a sample from and slot into (7.41).

The density $\gamma_i\left(\alpha_i\right)$ is set to $p(\alpha_i)$ defined in (4.18), which has a Gaussian shape, but constrained to $[-1, 1]$. It is straightforward to draw a sample directly from this

[5] Since samples for one dimension of $\boldsymbol{\alpha}$ and one of $\mathbf{u}$ are drawn, this could be written Z_{2i}/Z_{2i-2} to be exactly compatible with appendix A.4.3.

distribution, by sampling from the Gaussian and rejecting samples outside $[-1, 1]$ (see section 4.2.1.1). Therefore, $\rho_i(\alpha_i)$ can be set to $\gamma_i(\alpha_i)$. this means that $\gamma_i(\alpha_i)/\rho_i(\alpha_i)$ in (7.41) becomes 1, and hybrid sequential importance resampling of algorithm 7 on page 278 can be applied.

Resampling is discussed in appendix A.4.4. In short, it duplicates higher-weight samples from the sample set and removes lower-weight ones between every dimension. This reduces the variance of the sample weights; conceptually, it focuses effort on higher-probability regions.

When using resampling, the order in which the dimensions are traversed becomes important. The longer ago samples for one dimensions were drawn, the more likely they are to have duplicate entries for that dimension. The sample set will therefore be less varied for earlier dimensions. This is not a big problem when, as here, the interest is not in the samples, but in the normalisation constant. However, when drawing samples for one dimension, it still makes sense to have last drawn the dimensions which the new dimension depends on most.

For example, it might seem obvious to draw $\alpha_1^{(l)} \dots \alpha_d^{(l)}$ first, and then go through $u_1^{(l)} \dots u_d^{(l)}$. However, in $i - 1$ rounds of resampling, the set of samples $\alpha_1^{(l)} \dots \alpha_d^{(l)}$ may become considerably less varied. For higher i, $u_i^{(l)}$ may be drawn with only a few or one unique $\alpha_i^{(l)}$, which limits the accuracy of the approximation of the normalisation constant. In this work, the order of sampling is therefore $\alpha_1, u_1, \alpha_2, u_2, \dots \alpha_d, u_d$. This works best if u_i and u_j are less dependent when $j-i$ is greater. The order in which samples for the dimensions are drawn could also be determined on the fly by considering the values of the off-diagonals in $\mathbf{\Sigma}_x$ and $\mathbf{\Sigma}_n$, but this work does not investigate this.

This section has discussed a transformation of the integral that gives the corrupted speech likelihood $p(\mathbf{y}_t)$, two different factorisations of the integrand, and how to apply sequential importance sampling to approximate the integral. The estimate from the sampling scheme is consistent, but not unbiased. This means that for a small sample cloud, the approximated value for $p(\mathbf{y}_t)$ may be generally overestimated of

underestimated. However, as the sample cloud size increases, the resulting value converges to the real likelihood.

7.4 Approximate cross-entropy

It is standard practice in speech recognition research to judge speech recognition methods by word error rates. However, this chapter has the explicit aim of modelling the predicted corrupted speech distribution as accurately as possible. There is a more direct way of testing performance on this criterion than the word error rate: the Kullback-Leibler divergence between the predicted distribution and the approximation. This is similar to the KL divergence to the single-pass retrained system discussed in section 4.4.4.1, but here the reference distribution is not parametric, and the speech and noise distributions are. For a detailed assessment and for efficiency, it is useful to focus on one speech Gaussian.

The KL divergence is always non-negative; it is 0 if and only if the distributions are the same. The KL divergence to real distribution p from approximation q over $\mathbf{y}$ is defined as (from (A.11))

$$\mathcal{KL}(p\|q) = \int p(\mathbf{y}) \log \frac{p(\mathbf{y})}{q(\mathbf{y})} d\mathbf{y}. \tag{7.43}$$

Here, q is an approximation to the noise-corrupted speech distribution, found for example with VTS, DPMC, or transformed-space sampling. The problem in computing this divergence is the one that motivates this whole chapter: the real distribution p has no closed form, and neither does $\mathcal{KL}(p\|q)$. This problem can be worked around in two steps.

First, the KL divergence can be decomposed as (from (A.12))

$$\mathcal{KL}(p\|q) = \mathcal{H}(p\|q) - \mathcal{H}(p)\,, \tag{7.44a}$$

where the cross-entropy of p and q is defined as

$$\mathcal{H}(p\|q) = -\int p(\mathbf{y}) \log q(\mathbf{y}) d\mathbf{y} \tag{7.44b}$$

and the entropy of p as

$$\mathcal{H}(p) = -\int p(\mathbf{y}) \log p(\mathbf{y}) d\mathbf{y}. \tag{7.44c}$$

The entropy of p is constant when only q changes. The cross-entropy is then equal to the KL divergence up to a constant. For comparing different approximations q against a fixed p, therefore, the cross-entropy $\mathcal{H}(p\|q)$ suffices. It does not, however, give an absolute divergence. When q becomes equal to p, the cross-entropy becomes equal to the entropy, but the latter cannot be computed for the noise-corrupted speech distribution.[6]

The second problem is that $\mathcal{H}(p\|q)$ cannot be computed either. However, it is straightforward to draw samples from p: section 4.3.1 has shown the algorithm. L samples drawn from p give the delta spikes in the empirical distribution $\tilde{p}$:

$$\tilde{p} = \sum_{l} \delta_{\mathbf{y}^{(l)}}, \qquad \mathbf{y}^{(l)} \sim p(\mathbf{y}). \tag{7.45}$$

Then, plain Monte Carlo (see section A.4) can approximate the cross-entropy between p and q with the cross-entropy between $\tilde{p}$ and q.

$$\mathcal{H}(p\|q) \simeq \mathcal{H}(\tilde{p}\|q) = -\int \tilde{p}(\mathbf{y}) \log q(\mathbf{y}) d\mathbf{y} = -\frac{1}{L} \sum_{l} \log q\big(\mathbf{y}^{(l)}\big). \tag{7.46}$$

The cross-entropy in (7.44b) can be viewed as the expectation of the log-likelihood $\log q(\mathbf{y})$ under p, which is approximated with Monte Carlo (as in (A.37)). The result is (7.46), which can be seen as the negative log-likelihood of q for the set of samples $\{\mathbf{y}^{(l)}\}$. This gives another motivation for using the cross-entropy as a metric for comparing compensation methods. Note that when q is the transformed-space sampling method from section 7.3, for every sample $\mathbf{y}^{(l)}$ another level of sampling takes place inside the evaluation of $q\big(\mathbf{y}^{(l)}\big)$.

[6] The entropy could be rewritten to

$$\mathcal{H}(p) = \int p(\mathbf{y}) \log p(\mathbf{y}) d\mathbf{y} = \int \Big(\iint p(\mathbf{x}, \mathbf{n}, \mathbf{y}) d\mathbf{n} d\mathbf{x} \Big) \log \Big(\iint p(\mathbf{x}, \mathbf{n}, \mathbf{y}) d\mathbf{n} d\mathbf{x} \Big) d\mathbf{y},$$

which has no obvious closed form.

The cross-entropy results in section 8.2 will use this Monte Carlo approximation. It has one caveat: the distribution $\mathbf{q}$ is assumed to be normalised, and if not, then the result is not valid. This means that the Algonquin approximation, which does not yield a normalised distribution over $\mathbf{y}$, cannot be assessed in this way. As pointed out in section 7.3.2, the likelihood approximation of transformed-space sampling is biased, but consistent. This means that as the size of its sample cloud increases, $\mathbf{q}$ converges to being normalised.

7.5 Summary

This chapter has described the third contribution of this thesis.

This chapter has introduced a new technique for computing the likelihood of a corrupted speech observation vector. It does not use a parametric density, like the schemes in chapters 4 and 5, but a sampling method. The integral over speech, noise, and phase factor that the likelihood consists of is transformed to allow importance sampling to be applied. As the number of samples goes to infinity, this approximation converges to the real likelihood. This work uses it with specific distributions (Gaussian speech and noise, a constrained Gaussian for the phase factor), but the method could also be applied to other distributions. Though the method is too slow to embed in a speech recogniser, it is possible to find the KL divergence from an approximated corrupted speech distribution to the real one up to a constant. Section 8.2 will use it to assess how close to ideal compensation methods are, and the effect of approximations such as assuming the corrupted speech Gaussian.

Chapter 8

Experiments

This thesis has looked into two ways of improving statistical models for noise-robustness. The experimental results will therefore be in two parts.

The first part is about modelling correlations within speech recogniser components. The theory has considered two aspects: estimating full-covariance compensation (chapter 5), and decoding with that compensation but without the computational cost (chapter 6). These aspects will be demonstrated in section 8.1.

The second part is more theoretical. Chapter 7 has introduced a method that, given speech and noise priors and a mismatch function, computes the corrupted speech likelihood exactly in the limit. Though it is too slow for decoding, it makes it possible to find how well current methods for model compensation do. This will use an approximation to the KL divergence. It is interesting to see how well that predicts speech recogniser performance. It also becomes possible to investigate specific approximations that model compensation methods make. Section 8.2 will examine, for example, the influence of the assumption that the corrupted speech distribution is Gaussian and diagonalising that Gaussian's covariance. It will also assess the impact of a common approximation to the mismatch function for VTS compensation, namely setting the phase factor to a fixed value.

8.1 Correlation modelling

Realistic methods specifically for noise-robustness are meant to deal with short, noise-corrupted utterances, for example, voice commands to a car navigation system. With little adaptation data, it is vital to have as few parameters as possible to estimate. A Gaussian noise model can be estimated on a few seconds of data. A generic adaptation method, on the other hand, would do better only if the noise is constant for minutes (for a comparison between VTS and CMLLR on short noisy utterances, see Flego and Gales 2009). The scenario that this section will consider is, therefore, that of short utterances with varying noise.

Model compensation methods using extended feature vectors (like extended VTS or extended DPMC) model dynamics better. Therefore, they are able to find better full-covariance compensation. This section will examine the effects of the improvements that extended VTS makes over standard VTS. For this, it will use the Resource Management task, which allows per-speaker noise estimation, so that it is feasible to run EDPMC. Then, results on AURORA will indicate whether the improvements carry over to this well-known task. Finally, results on the Toshiba in-car data will show performance on data recorded in a real noisy environment.

For all recognition systems, clean training data is used to train the speech models. 39-dimensional feature vectors are used: 12 MFCCs and the zeroth coefficient, augmented with deltas and delta-deltas. Unless indicated otherwise, the MFCCs are found from the magnitude spectrum with HTK (Young *et al.* 2006) and the deltas and delta-deltas are computed over a window of 2 observations left and 2 right, making the total window width 9.

The state of the art in model compensation is VTS compensation with the continuous-time approximation. Section 4.4.2 has discussed how it uses a vector Taylor series approximation of the mismatch function. This makes it possible to estimate the noise model, because the influence of the noise is locally linearised. To keep in line with standard VTS compensation, for this section (not for section 8.2) the same noise estimates and the same mismatch function will be used for all per-component

compensation methods in this section. The phase factor will be assumed 0 in the magnitude spectrum. (Appendix C.2 shows how this is roughly equivalent to setting the phase factor to 1 when working with the power spectrum.)

When the noise model is estimated, maximum-likelihood estimation (Liao and Gales 2006), as described in section 4.7, is used for a clean system with VTS and the continuous-time approximation. The initial noise model's Gaussian for the additive noise is the maximum-likelihood estimate from the first 20 and last 20 frames of the utterance, which are assumed to contain no speech. The initial convolutional noise estimate is $\mathbf{0}$. Given this initial noise estimate for an utterance, a recognition hypothesis is found. This is used to find component–time posteriors. Then, the noise means and the additive noise covariance are re-estimated. Decoding with this noise model yields the final hypothesis.

8.1.1 Resource Management

To assess compensation quality and the effect of noise estimation, initial experiments are on a task of reasonable complexity, but with artificial noise. The first corpus used is the 1000-word Resource Management corpus (Price *et al.* 1988). Operations Room noise from the NOISEX-92 database (Varga and Steeneken 1993) is added at 20 and 14 dB.

The RM database contains read sentences associated with a naval resource management task. This task contains 109 training speakers reading 3990 sentences, a total of 3.8 hours of data. The original database contains clean speech recorded in a sound-isolated booth, which was used for training the recognisers. All results are averaged over three of the four available test sets, February 89, October 89, and February 91 (September 1992 is not used), a total of 30 test speakers and 900 utterances.

The NOISEX-92 database provides recording samples of various artificial, pedestrian and military noise environments recorded at 20 kHz with 16-bit resolution. The Destroyer Operations Room noise is sampled at random intervals and added to the clean speech data scaled to yield signal-to-noise ratios of 20 dB and 14 dB.

State-clustered triphone models with six components per mixture are built using the HTK RM recipe (Young *et al.* 2006). After four iterations of embedded re-estimation, the monophone models are cloned to produce a single-component triphone system. After two iterations of embedded training, the triphones are clustered at the state level. The number of distinct states is about 1600. These are then mixed up to six components, yielding about 9500 components in total. The language model for recognition is a word-pair grammar.

For initial experiments, an equivalent system is trained with one component per state from the one-but-last six-component system. The single-component one will be used for initial experiments, because data sparsity becomes an issue when estimating full covariance matrices over extended feature vectors. The six-component system is the standard one. On clean data, it produces a word error rate of 3.1 %. At the 20 dB word error rate, however, it yields 38.1 %. Since the additive background noise is known, it is possible to generate stereo data (clean and artificially corrupted) and use single-pass retraining (see section 4.4.4) to obtain an idealised compensated system. The word error rate then becomes 7.4 %. It is also possible to extract the true noise model.

When the noise model is estimated, this is done per speaker. With maximum-likelihood estimation for the noise model, there is no guarantee that this compensates only for noise: it may implicitly also adapt for speaker characteristics. For example, the voice quality will influence the parameters of the convolutional noise, like cepstral mean normalisation does.

10 000 samples per distribution are used for DPMC. (Performance does not improve with additional samples.) The noise covariance estimate does not contain any zero entries, so back-off as discussed in section 5.5.2.1 is not necessary.

Section 8.1.1.1 will investigate the closeness of compensation methods to the ideal distribution. Section 8.1.1.2 will look into reconstructing an extended noise model from a noise model with statics and dynamics. Section 8.1.1.3 will then apply extended feature vector compensation to components. The computational complexity of

compensation and decoding will be the topic of section 8.1.1.4.

8.1.1.1 Compensation quality

The compensation methods with extended feature vectors in this thesis aim to model the corrupted speech distribution more accurately, with the objective to improve performance. Whether they succeed can be verified in two stages. This section will assess compensation quality with the KL divergence; later sections will assess recognition performance.

Compensation quality will be measured with the component-per-component KL divergence to the single-pass retrained system as explained in section 4.4.4.1. That section has also mentioned that, depending on the structure of the covariance matrix, the KL divergence can be computed separately for coefficients or blocks of components. The following will therefore examine single coefficients, for diagonal-covariance compensation, first, and then blocks of coefficients for block-diagonal matrices.

The results derive from a Resource Management system with one component per state so full-covariance speech statistics can be robustly estimated. The total number of Gaussians is 1600. NOISEX-92 Operations Room noise is artificially added at a 14 dB SNR. Hence it is possible to obtain the correct noise distribution, for both the standard and extended feature vector cases. The noise models also have full covariance matrices.[1]

Diagonal compensation Normally, VTS-compensated covariance matrices are diagonalised. Thus it is interesting to initially examine this configuration. Using diagonal covariance matrices also allows each dimension to be assessed. Figure 8.1 on the following page contrasts the accuracy of an uncompensated system, and three forms of compensation: standard VTS, extended VTS, and, as an indication of maximum possible performance, extended DPMC. This graph is for a 14 dB SNR, but graphs for other SNRs are very similar. The horizontal axis has the feature dimensions: 13 static

[1] Similar trends are observed when striped noise statistics, consistent with diagonal standard noise models for VTS, are used for EVTS.

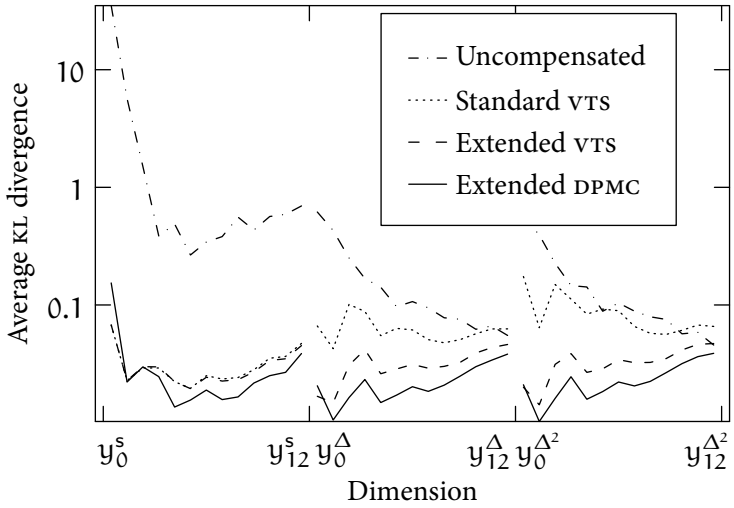


Figure 8.1 *Average Kullback-Leibler divergence between compensated systems and a single-pass retrained system.*

MFCCs $\mathbf{y}^s$, 13 first-order dynamics $\mathbf{y}^\Delta$, and 13 second-order dynamics $\mathbf{y}^{\Delta^2}$. As expected, the uncompensated system is furthest away from the single-pass retrained system, and extended DPMC provides the most accurate compensation given the speech and noise models. The difference between standard VTS and extended VTS is interesting. By definition, both yield the same compensation for the statics.[2] For the deltas and especially the delta-deltas, however, the continuous-time approximation does not consistently decrease the distance to the single-pass retrained system. Extended VTS, though not as accurate as extended DPMC, provides a substantial improvement over standard VTS.

Block-diagonal compensation The previous section used diagonal covariance matrices. To compensate for changing correlations under noise, more complex covariance matrix structures, such as full or block-diagonal, may be used. VTS with the continuous-time approximation can also be used to generate block-diagonal covariance matrices for the output distributions. The form of this was shown in (4.44). Normally, EVTS produces full covariance matrices, and so can single-pass retraining, but they

[2]Small differences are because HTK gathers statistics for statics and dynamics differently from for extended feature vectors.

Compensation	—	VTS	EVTS	EDPMC
$\mathbf{y}^{s}$	58.8	1.0	1.0	1.0
$\mathbf{y}^{\Delta}$	3.3	1.4	0.7	0.5
$\mathbf{y}^{\Delta^2}$	3.2	1.7	0.7	0.5

Table 8.1 *Resource Management task: average KL divergence to a block-diagonal single-pass retrained system for VTS (continuous time), EVTS and DPMC at 14 dB SNR.*

can be constrained to be block-diagonal by setting other entries to zero. It then is possible to compute the KL divergence to a single-pass retrained system per block. This allows the compensation of each of the blocks of features to be individually assessed: the statics, and first- and second-order dynamics. VTS compensation uses block-diagonal statistics for both the clean speech and noise models. For EVTS the extended statistics have full covariance matrices, to be equivalent to the statistics for standard VTS.

Table 8.1 shows the average KL divergence between a system compensated with block-diagonal VTS with the continuous time approximation and the block-diagonal single-pass retrained system. For other signal-to-noise ratios, numbers are very similar. VTS finds compensated parameters close to the single-pass retrained system for the static features: the KL divergence goes from 58.8 to 1.0. However, the dynamic parameters are not compensated as accurately, though both the delta and delta-delta parameters are still somewhat closer to the SPR system than the uncompensated model set (3.2 to 1.7 for the delta-deltas). Similarly to diagonal compensation (see figure 8.1), with block-diagonal covariances standard VTS finds good compensation for the static parameters, but less good for the deltas and delta-deltas.

EVTS has the same compensation as standard VTS for the statics. As in the diagonal-covariance case, however, for dynamic parameters compensation it is more accurate. It does yield a clear improvement over the uncompensated system (3.2 to 0.7 for the delta-deltas) and is close to EDPMC, which in the limit yields the best obtainable Gaussian compensation given the speech and noise models.

Σ_n^e		Σ_y diag	Σ_y full
known	diag	16.4	16.0
	smoothed	15.9	15.2
	full	15.9	14.9
estimated	diag	12.0	11.2
	smoothed	12.5	12.0

Table 8.2 *Resource Management task: word error rates for reconstructing an extended noise model at 14 dB SNR.*

8.1.1.2 Extended noise reconstruction

In practice, labelled samples of the noise to estimate the noise model are not always available. As discussed in section 5.5.2, when using the standard ML estimated noise models there are two approaches to mapping the noise model parameters to the extended noise model parameters: one with a diagonal covariance matrix for the additive noise, and one with a smoothly reconstructed matrix (which is striped). This section contrasts the performance of the two. It applies extended VTS to the system with six striped-covariance Gaussian components per state. The standard noise model parameters can either be derived from the actual noise data, the "known" case, or using maximum-likelihood estimation, "estimated". In this section the noise added to the RM task is scaled to yield a 14 dB SNR.

The top three rows of table 8.2 compare diagonal and smoothed reconstructions of the extended noise when the standard noise model is estimated on the actual data. For diagonal-covariance and full-covariance compensation, smoothing results in 0.6 % and 0.8 % absolute improvement in the word error rate. This works almost as well as when the known noise distribution is used (as well, with diagonal-covariance compensation). However, when the standard noise model parameters are found with maximum-likelihood estimation, the bottom two rows in table 8.2, the smoothing process degrades performance.

This degradation from the use of smoothing when using maximum-likelihood estimated noise parameters is caused by the nature of the maximum-likelihood es-

Jacobians	$\boldsymbol{\Sigma}_x^e$	20 dB	14 dB
Standard VTS		6.8	13.7
Fixed	diag	7.8	13.5
Variable	diag	7.5	13.0
Variable	striped	6.2	12.0

Table 8.3 *Resource Management task: word error rates for approximations of EVTS. Diagonal-covariance decoding.*

timates. For the smoothing process it is assumed that there is some true underlying sequence of noise samples that yields the standard noise model parameters. This is guaranteed to be true for the known noise distribution. However, this is not necessarily the case for the estimated noise. The dynamic noise parameters are estimated using the continuous-time approximation. There are no constraints that the estimates reflect the true or a possible sequence of noise samples. Using smoothing, which assumes relationships in the noise sample sequence, to estimate the extended noise covariance matrix may therefore not help. The experiments in the following sections will therefore use the diagonal reconstruction for the extended noise distribution.

8.1.1.3 Per-component compensation

This section investigates recognition performance for per-component compensation with extended VTS. Here, a standard noise model is estimated and a diagonal extended noise model derived from it.

Table 8.3 presents the properties of extended VTS and striped statistics. The first row gives word error rates for standard VTS at 20 and 14 dB. As discussed in section 5.3.3.1, extended VTS becomes standard VTS if the expansion point is chosen equal for all time instances. Varying the expansion point is expected to provide better compensation for dynamics. On the other hand, diagonalising extended clean speech statistics discards information compared to diagonalising standard features with statics and dynamics. For the second line in the table, the expansion points vary, but the Jacobian is fixed. At the lower SNR, the improved modelling helps, but at the higher SNR, where the corrupted speech distributions are closer to the clean speech, diagonal-

Scheme	Speech	Σ_y	Operations Room 20	14	8	Car 20	14	8
VTS	diag Σ_x	diag	6.8	13.7	30.0	5.2	9.1	18.7
	block Σ_x	block	7.0	14.2	31.6	5.3	9.7	20.1
EVTS	striped Σ_x^e	diag	6.2	12.0	27.9	4.8	8.5	18.2
		full	6.3	11.2	26.7	5.0	8.3	17.9
EDPMC	striped Σ_x^e	diag	6.3	11.9	27.9	4.8	8.2	16.4
		full	6.0	11.3	26.3	4.7	7.9	15.9

Table 8.4 *Resource Management task: word error rates for standard VTS, extended VTS and extended DPMC.*

ising the extended clean speech statistics discards vital information. For the third row, the Jacobians are allowed to vary, which gives complete EVTS, if with diagonal speech statistics. The bottom row uses striped statistics, as discussed in section 5.5.1, which discards no information compared to diagonal standard statistics. This leads to a consistent improvement over standard VTS. The following experiments will therefore use striped statistics.

Table 8.4 shows contrasts between compensation with standard VTS and with extended feature vectors using either EVTS or EDPMC. A first thing to note is that all approaches perform better than single-pass retraining (not in the table, at 7.4 % for 20 dB Operations Room noise). This is because the noise estimation can implicitly compensate for some speaker characteristics. The results in the first row are from the standard scheme, diagonal-covariance compensation with VTS. Block-diagonal compensation with standard VTS is also implemented and block-diagonal clean speech statistics are used. The results for this approach are in the second row. The use of the block-diagonal compensation with VTS degrades performance, for example 13.7 % to 14.2 % for Operations Room noise at 14 dB.

Compensation with EVTS (shown in the middle two rows of the table) yields better performance than standard VTS for both diagonal and full compensated covariance matrices. For diagonal-covariance compensation, the relative improvement is 5–10 % (6.8 to 6.2 %; 13.7 to 12.0 %, etc.) over standard VTS. Though at the higher SNR condition, 20 dB, full-covariance compensation does not improve performance over

diagonal-covariance performance, gains are observed at the lower SNRs. For Operations Room noise at 14 dB, full-covariance compensation produces an 11.2 % word error rate, which is a 20 % relative improvement from standard VTS, and 7 % relative gain compared to the diagonal case.

In addition, table 8.4 shows the performance of EDPMC, which in the limit can be viewed as the optimal Gaussian compensation scheme. The results for this approach are shown in the bottom two rows of table 8.4. When compared with EDPMC, the first-order approximation in EVTS degrades performance by up to 0.4 % absolute, except for 8 dB Car noise. However, EVTS is significantly faster than EDPMC.

8.1.1.4 Per-base class compensation

The results in section 8.1.1.3 used per-component compensation, and full covariance matrices. In real recognition systems, this is often impractical: computing per-component compensation is too slow, as is decoding with full covariances. These problems can be solved with two techniques in tandem. Joint uncertainty decoding can perform compensation with extended feature vectors per base class (see section 5.4). This speeds up compensation. Predictive linear transformations, such as predictive semi-tied covariance matrices (see section 6.2.3), can convert full-covariance compensation into diagonal-covariance compensation plus a linear transformation. This speeds up decoding. It is the combination of extended feature vector compensation with joint uncertainty decoding and predictive transformations that make it feasible for real systems.

Since both joint uncertainty decoding (JUD) and predictive semi-tied covariance matrices are approximations, some loss of accuracy is expected in return for the gain in speed. The noise model is estimated to maximise the likelihood for standard VTS and for JUD with standard VTS. Again, a diagonal-covariance noise model with extended feature vectors is found. It should be noted that the performance figures on extended feature vectors give an underestimate of performance if the noise model was estimated consistently. In the results in this work there is a contest between the effect of the

Gaussians	Type	$\boldsymbol{\Sigma}_y$	20 dB	14 dB
9.5 K	VTS	diag	6.8	13.7
	EVTS	diag	6.2	12.0
		full	6.3	11.2
	EDPMC	diag	6.3	11.9
		full	6.0	11.3
16 (JUD)	VTS	diag	7.4	16.4
	EVTS	diag	7.6	15.6
		full	7.5	13.8
		semi-tied	7.5	13.8
	EDPMC	diag	8.0	16.2
		full	7.4	14.6
		semi-tied	7.4	14.7

Table 8.5 *Word error rates for per-component and per-base class compensation.*

quality of compensation (extended DPMC should be better than extended VTS, which should be better than standard VTS), and the effect of the mismatch in the noise model estimation (standard VTS should be better than extended VTS, which should be better than extended DPMC).

Table 8.5 shows the word error rates. The top half of table 8.5 contains results for per-component compensation. They are repeated from table 8.4 on page 224 for reference. The main effects are that methods with extended feature vectors (EVTS and EDPMC) were able to produce more accurate compensation, especially with full covariance matrices and at the lower signal-to-noise ratio. Because the noise model is estimated for standard VTS, to which EVTS compensation is more similar than EDPMC compensation, EDPMC does not consistently outperform EVTS even though it provides more accurate compensation.

The second half of the table contains results with joint uncertainty decoding. To reduce the computational cost substantially, the number of base classes is low: only 16. The noise model is optimised for JUD compensation with VTS, the top row of the second half. Going from per-component to per-base class VTS, the accuracy decreases by 0.6 % and 2.7 %. At 20 dB, extended VTS and extended DPMC do not improve over standard VTS at all. This is caused by the mismatch of the noise model, compounded

by the approximation that JUD uses.

At the 14 dB signal-to-noise ratio, however, the performance differences are more similar to the ones for per-component compensation. However, here extended DPMC performs less well than extended VTS, though it is slower and more accurate given the real noise distribution. This is caused by the larger mismatch between the compensation that the noise model is estimated for and the actual compensation. The mismatch between standard VTS and extended VTS is less great. Extended VTS improves JUD performance by 0.8 % when generating diagonal covariances, and by 2.6 % with full covariances. The latter models the change in feature correlations. However, since it makes the covariance bias of joint uncertainty decoding full, all covariance matrices become full, which is slow in decoding. Semi-tied covariance matrices, in the next row, convert the full-covariance Gaussians to diagonal-covariance ones with a linear transformation per base class. As in earlier work (Gales and van Dalen 2007), this does not impact performance negatively: it stays at 13.8 %.

These results show that it is feasible to reduce the computational load of extended feature vector approaches to a practical level, with only limited negative effect to accuracy. The combination of predictive semi-tied covariance matrices, joint uncertainty decoding, and extended VTS is the set-up that this thesis proposes for practical compensation. Joint uncertainty decoding reduces the number of Gaussians to be compensated. Then, predictive semi-tied covariance matrices provides a form of decoding that is basically as fast as diagonal covariances, but does model correlations. The choice of the number of base classes for joint uncertainty decoding provides a trade-off between speed and accuracy. It is important to note that joint uncertainty decoding with only one component in each base class is equal to the form of compensation it is derived from, for example, extended VTS. The same goes for predictive semi-tied covariance matrices. The results here show the extremes of both, but it is possible to select any point in between for the desired trade-off between speed and accuracy.

8.1.2 AURORA 2

AURORA 2 is a small vocabulary digit string recognition task (Hirsch and Pearce 2000). Though it is less complex than the Resource Management corpus, it is a standard corpus for testing noise-robustness. Utterances are one to seven digits long and based on the TIDIGITS database with noise artificially added. The clean speech training data comprises 8440 utterances from 55 male and 55 female speakers. The test data is split into three sections. Test set A comprises 4 noise conditions: subway, babble, car and exhibition hall. Matched training data is available for these test conditions, but not used in this work. Test set B comprises 4 different noise conditions. For both test set A and B the noise is scaled and added to the waveforms. For the two noise conditions in test set C convolutional noise is also added. Each of the conditions has a test set of 1001 sentences with 52 male and 52 female speakers.

The feature vectors are extracted with the ETSI front-end (Hirsch and Pearce 2000). The delta and delta-delta coefficients use 2 and 3 frames left and right, respectively, for total window of 11 frames. The acoustic models are whole word digit models with 16 emitting states, and 3 mixtures per state and silence. The results presented here use the simple AURORA back-end. Using the simple back-end recogniser rather than one with more Gaussian components per state ensures that block-diagonal and full covariance matrices for the clean speech are robust. Since for this task, as for the Resource Management task, the noise estimates do not contain zero elements in the variance, the back-off strategy for the noise estimate discussed in section 5.5.2.1 is not necessary.

8.1.2.1 *Extended* VTS

Table 8.6 on the next page shows results for compensation with VTS with the continuous-time approximation and EVTS. Both diagonal and block-diagonal forms of VTS are used. VTS with diagonal compensation (trained on diagonal speech statistics) is the standard method. Results for this are shown in the columns labelled "VTS diag" of table 8.6 and are treated as the baseline performance figures (similar results for VTS are given in Li *et al.* (2007)). VTS can also be used to produce block-diagonal cov-

SNR	A			B			C		
Scheme	VTS		EVTS	VTS		EVTS	VTS		EVTS
Comp.	diag	block	full	diag	block	full	diag	block	full
00	28.2	24.3	23.5	26.2	22.9	24.3	25.9	23.6	22.6
05	10.5	8.2	7.1	9.3	7.9	7.2	9.9	8.2	6.9
10	4.3	3.3	2.5	3.9	3.2	2.4	4.4	3.4	2.8
15	2.2	1.9	1.2	2.2	1.8	1.2	2.3	1.8	1.4
20	1.6	1.3	0.8	1.4	1.2	0.7	1.6	1.2	1.0
Avg.	9.4	7.8	7.0	8.6	7.4	7.2	8.8	7.7	6.9

Table 8.6 *AURORA: diagonal compensation with standard VTS and full compensation with extended VTS.*

ariance matrices. The results for this are shown in the columns labelled "VTS block" in table 8.6. Compared to the standard diagonal VTS scheme, this gives, for example, relative reductions in word error rate of 15 % to 22 % at 5 dB SNR. With diagonal-covariance statistics for the clean speech (not in the table), this yields no performance gain. However, performance gains were obtained when using block-diagonal clean-speech models, unlike for Resource Management. This difference in performance between the tasks can be explained by the additional complexity of the RM task compared to AURORA.

The results for EVTS are shown in the last three columns of table 8.6. Here, full-covariance models are used for the extended clean speech to produce compensated full covariance matrices for decoding. The improved compensation for dynamics causes extended VTS to perform better than to block-diagonal standard VTS in all but one noise conditions. At 5 dB again, relative improvements are an extra 3 % to 10 %.

8.1.2.2 *Front-end* PCMLLR

This section makes a different, but interesting, trade-off between speed and accuracy. Section 6.4 has introduced a number of schemes that are based on predictive transformations but are even faster, because they solely use feature transformations. Since the objective of the experiments in this section is to find fast feature transformations, appropriate for embedded real-time systems, the small AURORA task is again used.

	SNR					
Scheme	20	15	10	5	0	Avg.
VTS	1.6	2.2	4.3	10.5	28.2	9.4
JUD	1.7	2.7	4.7	11.4	30.4	10.2
PCMLLR	1.5	2.6	5.2	13.5	34.7	11.5
Model-MMSE	33.8	47.5	61.6	77.4	91.7	62.4

Table 8.7 *AURORA: component-dependent compensation.*

Also, all transformations have diagonal covariance matrices, which makes compensation and decoding very fast.

64 base classes are used, which as in Stouten *et al.* (2004*b*) is about a tenth of the number of back-end components (546). The noise model for each utterance is estimated for every utterance, for compensation on the base class level. This noise model is then used to VTS-compensate the 64-component clean front-end GMM, producing the joint distribution (4.70). This scheme for estimating the joint distribution is the same as used for MMSE feature enhancement in Stouten *et al.* (2004*b*), with the addition of additive noise estimation and explicit compensation of dynamic parameters.

Table 8.7 contains word error rates for component-specific model compensation. VTS is the odd one out in that it compensates every back-end component separately, providing accurate but slow compensation. All other schemes use the joint distribution. JUD actually decodes with the back-end models set to their predicted distributions in (4.48a), trading in some accuracy for speed compared to VTS. The other two schemes, PCMLLR and model-MMSE, both use component-dependent transformations. PCMLLR minimises the KL divergence to JUD compensation. Model-MMSE is applied in the same way, but estimated for MMSE (see section 4.6): every transformation reconstructs the clean speech in an area of acoustic space. PCMLLR brings the adapted models close to the JUD-predicted statistics. That model-MMSE fails to provide meaningful compensation highlights the difference in the nature of PCMLLR and MMSE transforms, even though their form is the same. MMSE transforms, which aim to reconstruct the clean speech, have no meaning when applied separately to each base class of components.

	SNR					
PCMLLR Scheme	20	15	10	5	0	Avg.
Global	2.9	6.5	16.8	40.5	71.8	27.7
Observation-trained	1.4	2.6	5.5	14.2	37.1	12.2
Hard-decision	1.5	3.0	6.5	16.6	40.5	13.6
Interpolated	1.4	2.5	5.0	12.6	32.5	10.8
MMSE	1.5	2.6	5.4	14.4	39.7	12.7

Table 8.8 *AURORA: component-independent transformation: PCMLLR-based schemes and MMSE.*

Table 8.8 shows results of using component-independent transformations. MMSE is the standard feature enhancement scheme that reconstructs a clean speech estimate from the noise-corrupted observation. For higher SNRs, its accuracy is similar to model compensation methods JUD and PCMLLR trained from the same joint distribution. For lower SNRs, however, MMSE's point estimate of the clean speech performs less well than JUD and PCMLLR's compensation of distributions.

Global PCMLLR estimates one transform per noise condition, and is the baseline for component-independent PCMLLR. Observation-trained PCMLLR adapts the predicted corrupted speech distribution for every observation and estimates an appropriate transform. This yields accurate compensation, which for lower SNRs shows the advantages of model compensation.

Hard-decision PCMLLR uses the same transforms as PCMLLR, but picks one based on the feature vector rather than on the back-end component. This simple scheme yields a piecewise linear transformation of the feature space. Interpolated PCMLLR performs better, by interpolating the transforms weighted by the front-end posterior. Interpolated PCMLLR outperforms all other derivatives of PCMLLR, PCMLLR itself, and MMSE, which uses the same form of decoding. The ingredients for its performance are twofold. First, the PCMLLR transforms are trained to minimise the KL divergence of the adapted models to the JUD-predicted corrupted speech distributions in an acoustic region. Secondly, the interpolation smoothly moves between transformations that are appropriate for the acoustic region. PCMLLR itself applies component-dependent

Scheme	Decoding	ENON 35 dB	CITY 25 dB	HWY 18 dB
VTS	diag	1.2	2.5	3.2
eVTS	diag	1.1	2.4	2.8
	full	1.7	2.5	2.4
eVTS	back-off	1.1	2.2	2.4
% utterances		87 %	38 %	11 %

Table 8.9 *Extended VTS on the Toshiba in-car task.*

transformations independently of the feature vector.

8.1.3 Toshiba in-car database

Experiments are also run on a task with real recorded noise: the Toshiba in-car database. This is a corpus collected by Toshiba Research Europe Limited's Cambridge Research Laboratory. It comprises a set of small/medium sized tasks with noisy speech collected in an office and in vehicles driving at various conditions. This work uses three of the test sets containing digit sequences (phone numbers) recorded in a car with a microphone mounted on the rear-view mirror. The ENON set, which consists of 835 utterances, was recorded with the engine idle, and has a 35 dB average signal-to-noise ratio. The CITY set, which consists of 862 utterances, was recorded driving in the city, and has a 25 dB average signal-to-noise ratio. The HWY set, which consists of 887 utterances, was recorded on the highway, and has a 18 dB average signal-to-noise ratio.

The clean speech models are trained on the Wall Street Journal corpus, based on the system described in Liao (2007), but the number of states is reduced to about 650, more appropriate for an embedded system. The acoustic models used are cross-word triphones decision-tree clustered per state, with three emitting states per HMM, twelve components per GMM and diagonal covariance matrices. The number of components is about 7800. Like for Resource Management, extended clean speech statistics for extended VTS are striped for robustness. The language model is an open digit loop. To find the noise model, it is re-estimated twice on a new hypothesis.

Table 8.9 on the facing page shows results on the Toshiba task. The top row contains word error rates for the standard compensation method: VTS trained on diagonal speech statistics. The performance of EVTS using diagonal covariance matrices is shown in the second row. Again EVTS shows gains over VTS, especially at the lowest SNR condition, HWY. In the HWY condition about a 12 % relative reduction in error rate was obtained.

Initially full-covariance matrix compensation with EVTS is evaluated without the use of the back-off scheme described in section 5.5.2.1. Using EVTS with full-covariance decoding yields additional gains compared to diagonal compensation at low SNRS (2.8 % to 2.4 %). However the performance is degraded at higher SNR conditions, for example ENON where performance is degraded from 1.1 % to 1.7 %.

In contrast to the previous tasks, at high SNRS there are found to be zeros in the noise variance estimate. The back-off scheme, labelled "EVTS back-off" in table 8.9, is therefore used. Here diagonal covariance matrix compensation is used if any noise variance estimate falls below 0.05 times the variance floor used for clean speech model training (results are consistent over a range of values from 0.0 to 0.1). The bottom line in table 8.9 shows the percentage of utterances for each of the task where the system is backed off to diagonal covariance matrix compensation. As expected the percentage at high SNRS, 87 %, was far higher than at lower SNRS, 11 %. Using this back-off approach gives consistent gains over using either diagonal or full compensation EVTS alone. Note that as the back-off is based on the ML-estimated noise variances, it is fully automated. Compared to standard VTS, EVTS with back-off gave relative reductions of 8 % in the ENON condition, 12 % in CITY, and 25 % in HWY.

8.2 The effect of approximations

The second part of this chapter is more theoretical. Chapter 7 has found an accurate approximation to the corrupted speech likelihood. To assess its performance, this section will initially consider the cross-entropy to the real distribution for individual

components, as discussed in section 7.4. In the limit, the transformed-space sampling method yields the ideal compensation, which gives the point where the KL divergence is 0. This allows a fine-grained assessment of the approximations that methods for noise-robustness make. This section will consider the effects of parameterisations of the noise-corrupted speech distributions, such as assuming it Gaussian (section 8.2.2), and diagonalising the covariance matrix of this Gaussian (section 8.2.3). It will also investigate the effects of a common approximation to the mismatch function: assuming the phase factor α fixed (section 8.2.4).

8.2.1 Set-up

If the speech and noise models represented the real distributions perfectly, then computing the corrupted speech distribution exactly would yield the best recognition performance. In practice, however, the models are imperfect and improving the KL divergence to the real distribution does not necessarily mean that the speech recognition accuracy will also improve. In this respect, assessing the quality of speech recognition compensation with the KL divergence is conceptually similar to assessing language models by their perplexities. The following sections will therefore also examine how the cross-entropy results relate to word error rates.

However, not all methods discussed in this thesis can be assessed with both of these metrics. The Algonquin algorithm, discussed in section 4.5.1, yields a Gaussian approximation of the corrupted speech distribution specific to an observation. Used as a method to approximate the likelihood of observations, it therefore is not normalised. This makes it impossible to compute the cross-entropy for it. As discussed in section 7.3, the likelihood for transformed-space sampling is not normalised for small sample clouds, but converges to normalisation as the number of samples increases.

This section will also not present word error rates for the transformed-space sampling method introduced in chapter 7.3, because decoding with it is prohibitively slow. The cause of this is a conceptual difference between model compensation methods (e.g. VTS, DPMC, and IDPMC) on the one hand and transformed-space sampling on the

other. Model compensation computes a parametric distribution, and once that is done, running a recogniser or computing a cross-entropy is not necessarily slower than without compensation. Transformed-space sampling, on the other hand, approximates the likelihood given an observation, and cannot precompute anything. To employ it for the recognition task in this section, just for the statics and with a decent-sized sample cloud of 512, it would run at roughly 30 million times real-time.[3] This figure is based on an implementation that is not optimised for speed at all, but even with an optimised implementation running a speech recogniser with it would not be feasible.

However, the approximated likelihood of transformed-space sampling tends to the exact likelihood. In the following section it will become clear that the cross-entropy that transformed-space sampling converges to indicates the minimum value of the cross-entropy to the real distribution. This minimum is by definition where the KL divergence is 0. The distance to this point in a cross-entropy graph therefore shows how far compensation methods are from the ideal compensation.

The cross-entropy experiments will assess compensation quality for the corrupted speech distribution resulting from combining one speech Gaussian with one noise Gaussian. The distributions will just be over statics, to remove the dependence on any additional approximations for the dynamics.

For the speech recognition experiments, the distributions over dynamic features for speech recognition are found with extended feature vectors, as introduced in chapter 5. Not only does this yield better accuracy than approximations, but it also keeps compensation for dynamics most closely related to that for the statics. For robustness, the speech statistics have striped covariance matrices as discussed in section 5.5.1. The estimated Gaussian distributions of the corrupted speech have full covariance matrices.

This section uses a noise-corrupted version of the Resource Management task, the set-up of which was discussed in section 8.1.1, both for evaluating the cross-entropy

[3] Computing the likelihood of one sample for one speech Gaussian takes slightly over 30 seconds on a machine on the Cambridge Engineering Department's compute cluster. Processing one second of speech, with observations every 10 ms, and 9500 speech Gaussians, requires performing this operation 950 000 times. This therefore takes roughly 30 million seconds.

and the word error rate. All experiments use a noise model trained directly on the noise as added to the speech audio, and there is no convolutional noise. This eliminates the influence of the noise estimation algorithm. As discussed in section 4.4, in practice methods for noise robustness can estimate a noise model on little training data compared to generic adaptation techniques. This is because their model of the environment matches the actual environment to some degree. This section examines how close that model can be without considering how to estimate the noise model.

This section uses the mismatch function presented in section 4.2.1 as the real one. It also assumes that the phase factor is Gaussian but constrained to $[-1, 1]$, and independent per time frame and per spectral coefficient. The variances σ_α^2 of the phase factor are found from the actual filter bank weights for the HTK Resource Management recogniser, as discussed in section 4.2.1.1. For the cross-entropy experiments, the schemes that sample from the phase factor distribution, DPMC and transformed-space sampling, use the exact distribution. VTS requires the distribution to be Gaussian (see section 5.3.3) and ignores the domain of the coefficients.

Previous work on model compensation has not modelled the phase factor with a distribution. Work on feature enhancement with VTS, on the other hand, has: to find a minimum mean square error estimate (Deng *et al.* 2004), or with a Kalman filter (Leutnant and Haeb-Umbach 2009*a*). For model compensation with VTS, previous work has fixed α to 0 (Moreno 1996; Acero *et al.* 2000), to 1 (Liao 2007, and section 8.1 of this thesis), mathematically highly improbable, or 2.5 (Li *et al.* 2007), mathematically inconsistent. The original presentation of DPMC (Gales 1995) similarly ignored the phase factor. However, since it trains a distribution on samples drawn from the corrupted speech distribution, it is straightforward to extend it so it uses a distribution for α. The phase factor distribution to draw samples from was given in section 4.2.1.1. The recognition experiments in this section apply extended VTS and DPMC compensation with a distribution over α. This is possible because extended feature vectors are used rather than the continuous-time approximation (see section 5.3.3). Section 8.2.4 will examine the influence of fixing the phase factor.

For the cross-entropy experiments, the full-covariance noise and speech Gaussians are both over 24 log-spectral coefficients. The one for the noise is trained directly on the noise audio. The speech distribution is taken from a trained Resource Management system, single-pass retrained to find Gaussian in the log-spectral domain. A low-energy speech component[4] is chosen, to represent the part of the utterance where the low SNR causes recognition errors. The distance between the speech and the noise means, averaged over the log-spectral coefficients, corresponds to a 10 dB SNR. Except where mentioned, the relative ordering of the approximation methods is the same for all combinations of speech and noise examined. 5000 samples $\mathbf{y}^{(l)}$ are drawn from the corrupted speech distribution.

For the cross-entropy experiments, DPMC trains Gaussians on 50 000 samples. For IDPMC, the average number of samples per Gaussian component is also 50 000, so that the 8-component mixture, for example, is trained on 400 000 samples. For the recognition experiments, the number of samples per component for extended DPMC is set to 100 000. For iterative DPMC, the average number of samples is 100 000: for example, a 6-component mixture is trained on 600 000 samples.

This is many more samples than in section 8.1. This is because as the compensation methods get to the limits of their performance, both for the cross-entropy and speech recognition, small inaccuracies become important. For the cross-entropy experiments, there is no mismatch between training data and predicted distribution, because both are generated with exactly the same models. For both cross-entropy and recognition experiments, over-training can be prevented not by reducing the number of components, but by increasing the number of samples. Training IDPMC, in particular, appears to be sensitive to the number of samples each component has to train on. The number of samples is therefore pushed to machine limits,[5] at which point performance appeared to have converged.

[4] Tied state "ST_uh_4_3", component 2.

[5] Training the recogniser takes several processor-weeks.

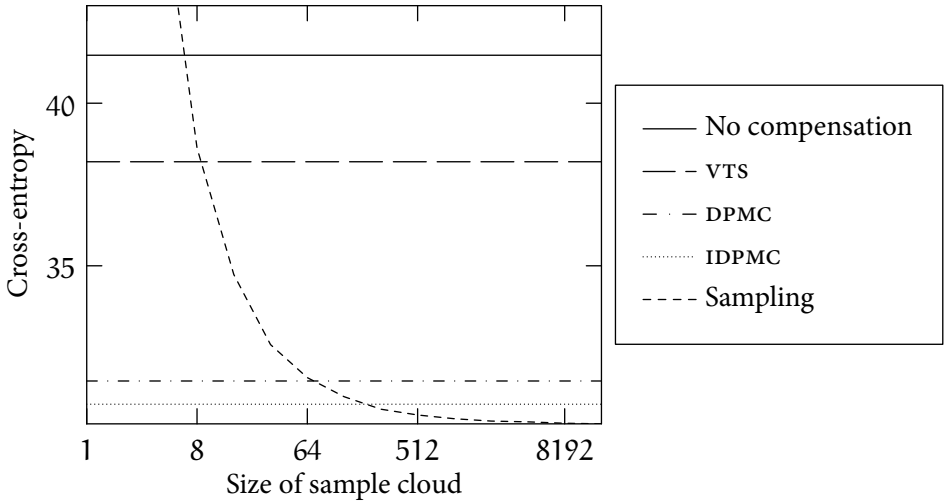


Figure 8.2 *Cross-entropy to the corrupted speech distribution for transformed-space sampling and model compensation methods.*

8.2.2 Compensation methods

The graph in figure 8.2 shows the cross-entropy for different compensation methods. The curved line indicates the cross-entropy between the real distribution and the transformed-space sampling method described in section 7.3, for increasing sample cloud size. The factorisation is the quasi-conditional factorisation from section 7.3.2.3. For distributions other than a three-dimensional toy example, the postponed factorisation discussed in section 7.3.2.2 showed a much slower convergence: in an earlier version of the experiment in figure 8.2 (without the phase factor), even with 16 384 samples it did not perform as well as the VTS-estimated Gaussian.

As the size of the sample cloud increases, the approximation of $p(\mathbf{y}^{(l)})$ found with transformed-space sampling converges to the correct value. This means that the cross-entropy $\mathcal{H}(p\|q)$ converges to the entropy $\mathcal{H}(p)$. The bottom of the graph is set to the point the curve in figure 8.2 converges to, which indicates the entropy of p. Since the KL divergence can be written (in (7.44a)) as $\mathcal{KL}(p\|q) = \mathcal{H}(p\|q) - \mathcal{H}(p)$, this is the point where the KL divergence is 0. Since the KL divergence cannot be negative, this point gives the optimum cross-entropy. It gives a lower bound on how well the

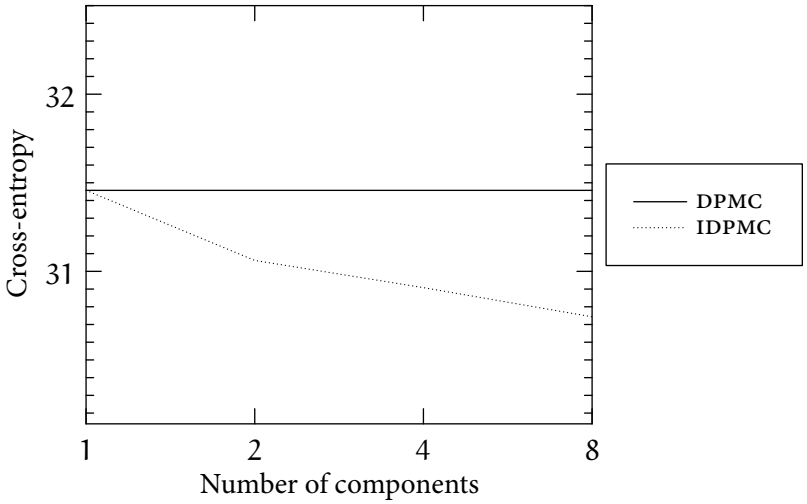


Figure 8.3 *Cross-entropy to the corrupted speech distribution for iterative* DPMC.

real corrupted speech distribution can be matched. The value of the cross-entropy for transformed-space sampling with 16 384 samples, 30.14, will also be the bottom for the other graphs in this section.

The line labelled "DPMC" in figure 8.2 indicates the best match to the real distribution possible with one Gaussian. The Monte Carlo approximation to the cross-entropy, section 7.4 has shown, is equivalent to the negative average log-likelihood on the samples. DPMC finds the Gaussian that maximises its log-likelihood on the samples it is trained on. If the sample sets for training and testing were the same, then DPMC would yield the mathematically optimal Gaussian. Though different sample sets are used (with 50 000 samples for training DPMC and 5000 samples for testing) the cross-entropy has converged. Any other Gaussian approximation will perform worse.

The state-of-the-art VTS compensation finds such a Gaussian analytically, and it is much faster. However, its cross-entropy to the real distribution is far from DPMC's ideal one.

Just like DPMC, IDPMC finds a distribution from samples, but it uses a mixture of Gaussians rather than one Gaussian. The mixture in the graph has 8 components trained on 400 000 samples, and comes close to the correct distribution. As the num-

Compensation	Shape	20 dB	14 dB
—	diag	38.1	83.8
VTS, $\alpha = 1$		8.6	17.3
EVTS	full	11.1	16.5
EDPMC		7.4	13.3
EIDPMC		6.9	12.0
EIDPMC + 6		6.2	11.1
EIDPMC + 12		6.5	11.3

Table 8.10 *Word error rates for various compensation schemes.*

ber of components increases from 1 to 8, keeping the average number of samples for components at 50 000, the cross-entropy decreases, as figure 8.3 on the preceding page illustrates. With an infinite number of components, it would yield the exact distribution. To correctly model the non-Gaussianity in 24 dimensions, however, a large number of components are necessary, which quickly becomes impractical.

To examine the link between the cross-entropy and the word error rate, recognition experiments are run. Improved modelling of the corrupted speech does not guarantee better discrimination, since speech and noise models are not necessarily the real ones. Since transformed-space sampling needs to be run separately for every observation vector for every speech component, it is too slow to use in a speech recogniser.

Table 8.10 contains word error rates at two signal-to-noise ratios for comparison with the cross-entropy results in figure 8.2. Results with the uncompensated system, trained on clean data, are in the top row. Below it, as a reference, is standard VTS. It sets the phase factor α to 1, and finds diagonal-covariance compensation. Standard VTS uses the continuous-time approximation to compensate delta- and delta-delta parameters. This yields inaccurate compensation for off-diagonals. Section 8.1.1.1 has demonstrated that. Using block-diagonal statistics and compensation (not in the table), word error rates for standard VTS are worse than using diagonal-covariance ones: 19.5 % and 38.5 %.

The bottom part of the table contains results on extended VTS (EVTS) and extended

DPMC (EDPMC). They use distributions over extended feature vectors. They also use a distribution over the phase factor α. The covariances of the resulting distributions are full.

EVTS performs less well than standard VTS at 20 dB. This is caused by the interaction of the phase factor with the vector Taylor series approximation, which section 8.2.4 will explore in more detail. At 14 dB, the more precise modelling does pay off. Compared to the uncompensated system extended VTS's performance improves more (38.1 % to 11.1 %) than expected from its improvement in terms of the cross-entropy in figure 8.2. VTS compensation uses a vector Taylor series approximation around the speech and noise means. It therefore models the mode of the corrupted speech distribution better than the tails. This causes the majority of the improvement in discrimination.

However, extended DPMC, which finds the optimal Gaussian given the speech and noise models, does yield better accuracy (7.4 %). Extended DPMC finds one corrupted speech Gaussian for one clean speech Gaussian. The cross-entropy experiment only uses one clean speech Gaussian. Extended IDPMC (EIDPMC), however, trains a mixture of Gaussians from samples, which can be drawn from any distribution. For the recognition experiments, therefore, EIDPMC compensates one state-conditional mixture at a time. Replacing the 6-component speech distribution by a 6-component corrupted speech distribution, EIDPMC increases performance from EDPMC's 7.4 % to 6.9 %. By modelling the the distribution better, with 12 components ("EIDPMC + 6"), performance increases further to 6.2 %. The corrupted speech distribution should be more precise as the number of components increases to 18 ("EIDPMC + 12"). However, even by increasing the number of samples by a factor of 2, to 3600 000, performance does not increase. This can be explained by lack of robustness of the speech statistics, even though they have striped covariance matrices. Since in figure 8.3 the line for IDPMC tends towards the best possible cross-entropy, this should be the best possible word error rate for these clean speech and noise distributions and this noise model.

Going from a Gaussian trained with extended VTS to the optimal Gaussian to a

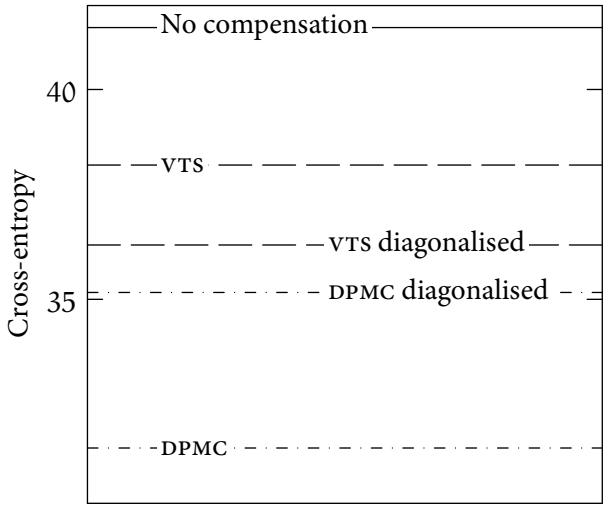


Figure 8.4 *The effect of diagonalisation on the cross-entropy.*

mixture of Gaussians in general improves the precision of the corrupted speech model. This shows in the cross-entropy to the real distribution, and the same effects are observed in the word error rate. Better modelling of the corrupted speech distribution leads to better performance. The next sections will evaluate specific common approximations: diagonalising Gaussians' covariance matrices, and setting α to a fixed value.

8.2.3 *Diagonal-covariance compensation*

The cepstral-domain Gaussians of speech recognisers are often diagonalised. This yields more robust estimates than full covariance matrices, and makes decoding fast. For noisy conditions specifically, it has previously been observed that feature correlations change and it is advantageous to compensate for this. However, it turns out that modelling correlations for the wrong noise conditions is counter-productive (Gales and van Dalen 2007). Also, estimates for off-diagonal elements are much less robust to approximations (see section 5.3.3). This section will relate these effects using the cross-entropy and speech recogniser accuracy.

Diagonalisation usually takes place in the cepstral domain. The theory and the cross-entropy experiments have used log-spectral-domain feature vectors. To emulate

Compensation	Shape	20 dB	14 dB
—	diagonal	38.1	83.8
eVTS	diag	8.3	15.7
	full	11.1	16.5
eDPMC	diag	7.5	14.9
	full	7.4	13.3

Table 8.11 *The effect of diagonalisation on the word error rate.*

diagonalisation in the cepstral domain for log-spectral-domain features, therefore, the Gaussian is first converted to the cepstral domain with a DCT matrix. Normally, cepstral feature vectors are truncated to 13 elements. However, to be able to convert back to the log-spectral domain, here all 24 dimensions are retained. The Gaussian is then diagonalised. To be able to compare the log-likelihoods, the diagonalised Gaussians are converted back to the log-spectral domain with the inverse DCT.

Figure 8.4 compares the cross-entropy to the real distribution of diagonalised and non-diagonalised Gaussians found with DPMC and VTS. As explained in the previous section, DPMC by definition yields the optimal Gaussian, so it must result in the lowest cross-entropy, and diagonalising it makes it perform less well. It is interesting that full-covariance VTS performs less well than its diagonalised form. On this test case, apparently, the off-diagonals in the cepstral domain are not estimated well enough, so that diagonalising lends the distribution robustness.

Table 8.11 investigates speech recognition performance when diagonalising Gaussian compensation. As in the previous section, the compensation methods use a phase factor distribution and extended feature vectors, to model the distributions as precisely as possible. Here, as for the cross-entropy, diagonalising extended VTS compensation improves performance (e.g., 11.1 % to 8.3 %). The off-diagonal covariance entries are not estimated well, so that diagonalisation increases robustness. (The next section will relate this to the model for the phase factor α.) However, eDPMC, which finds the optimal Gaussian compensation, does perform better when it is allowed to model correlations (7.5 % to 7.4 % at 20 dB). As expected from the results in section 8.1.1.3, at a lower signal-to-noise ratio the correlations change more, so that modelling them

becomes more important (14.9 % to 13.3 % at 20 dB).

8.2.4 *Influence of the phase factor*

Model compensation often assumes a mismatch function that is an approximation to the real one as presented in section 4.2.1. However, traditionally the phase factor α, which arises from the interaction between the speech and the noise in the complex plane, has been assumed fixed. As section 8.2.1 has explained, there has recently been interest in modelling the phase factor with a distribution. However, this work has been the first to use a phase factor distribution for model compensation. This section will look into the effect of the approximation of assuming the phase factor fixed.

Two settings for α are of interest. The traditional presentation (Moreno 1996; Acero *et al.* 2000) sets $\alpha = 0$, which is the mode of the actual distribution. The second setting is $\alpha = 1$. As appendix C.2 shows, if the term with α in the mismatch function is ignored and magnitude-spectrum feature vectors are used, this is roughly equivalent to setting $\alpha = 1$ for the power spectrum. This setting has been applied in previous work (e.g. Liao 2007), and in section 8.1 of this thesis.

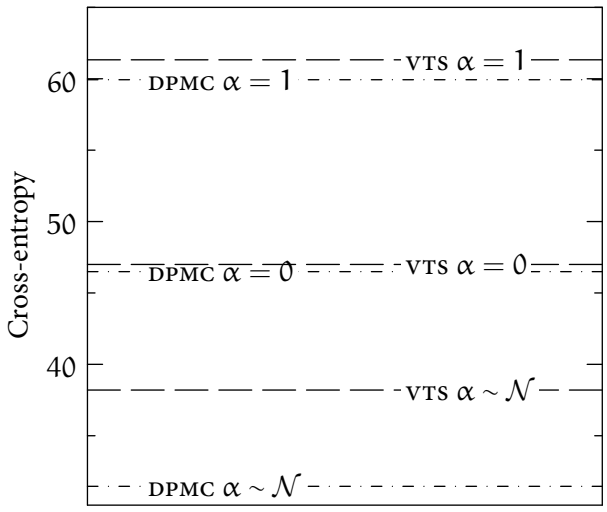


Figure 8.5 *The effect of the phase factor on Gaussian compensation.*

Figure 8.5 shows the cross-entropy for DPMC and VTS with different models for the

Scheme	α	20 dB	14 dB
	0	7.6	13.2
EDPMC	1	8.0	14.7
	$\mathcal{N}$	7.4	13.3
	0	11.4	16.5
EVTS	1	8.7	14.9
	$\mathcal{N}$	11.1	16.5

Table 8.12 *The effect of the phase factor on Gaussian compensation.*

phase factor. Note that the vertical axis uses a larger scale than figure 8.2. The bottom of the graph is still set to the optimal cross-entropy acquired with transformed-space sampling. Both methods generate full covariance matrices. The diagonalised versions (not shown in the table) show the same trends, with smaller distances between cross-entropies. DPMC with the model for α matching the actual distribution ("DPMC $\alpha \sim \mathcal{N}$") yields the lowest cross-entropy by definition. VTS with a Gaussian model ("VTS $\alpha \sim \mathcal{N}$") is at some distance.

The obvious choice for fixing α would be the mode of its actual distribution, 0. With that assumption, both DPMC and VTS end up further away from the ideal distribution. Note that though the cross-entropy lines for "DPMC $\alpha = 0$" and "VTS $\alpha = 0$" are close, the distributions are not necessarily similar. As expected, when α is fixed to 1, the modelled distributions become even further away from the actual ones.

Table 8.12 contains word error rates for the same contrasts. Again, it shows only full-covariance compensation. With diagonal covariances the trends are again the same but less pronounced. For EDPMC, the effect of different phase factor models is as expected. Whether α is distributed around 0 or fixed to 0 mostly affects the covariances. Though this does have an effect on the cross-entropy, since the change to the covariance matrices is fairly uniform across components, this makes little difference for discrimination. However, setting α to the unlikely value 1 affects performance negatively.

The results for VTS are more surprising. Again, there is little difference between setting α to 0 and letting it be distributed around 0. For VTS, this by definition does

not affect components' means, but only their covariances. However, setting it to 1 does improve performance. This may be because overestimation of the mode (see section 4.4.2) improves modelling for some components. Preliminary results suggest that which value of α yields the best cross-entropy varies with different distances between the speech and noise means. A hypothesis is that for different tasks, different settings for α optimise compensation for components at a speech-to-noise ratio where miscompensation is most likely to cause recognition errors. This would explain why the optimal α is different for different corpora (Gales and Flego 2010). However, this is material for future research.

What the results here do show is that while modelling α with a distribution reduces the distance to the actual distribution, as evidenced by the improving cross-entropy, discrimination is not helped. Section 4.4.2 has pointed out that the only effect of using a distribution for the phase factor over a fixed value at the distribution's mode is a fairly equal bias on the covariance, which is unlikely to influence discrimination much. It has also discussed how in practice the noise estimation can subsume this bias. Using a distribution over the phase factor rather than a fixed value as is currently done, is therefore unlikely to result in gains in a practical speech recogniser.

Chapter 9

Conclusion

The theme of this thesis has been to improve modelling of noise-corrupted speech distributions for speech recognition. It has argued (section 4.1) that if the model for speech, noise, and their interaction were exact, then decoding the audio with the exact corrupted speech distribution would yield the best speech recogniser performance.

This thesis has sought not only performance baselines, but also performance ceilings. It has derived an explicit expression for the corrupted speech distribution, which has no closed form. It has then analysed model compensation as aiming to minimise the divergence between the recogniser distribution and this predicted corrupted speech distribution. Two ways of improving modelling of corrupted speech distributions have been proposed.

First, this thesis has introduced methods that can model within-component feature correlations under noise. This is not normally done, because of problems finding accurate compensation, and the computational cost. This thesis has found new approaches for both problems (chapters 5 and 6), resulting in a chain of techniques that provide correlation modelling at a reasonable and tunable computational cost.

Second, this thesis has introduced a method of approximating the real corrupted speech likelihood (chapter 7). Given speech and noise distributions and a mismatch function, it finds a Monte Carlo approximation to the likelihood of one observation vector. Though it is very slow, in the limit it computes the exact likelihood. It there-

fore gives a theoretical bound for noise-robust speech recognition. This has made it possible to assess the effect of approximations that model compensation makes.

The following gives more detail about the contributions of this thesis.

9.1 Modelling correlations

Methods for noise-robust speech recognition, like the state-of-the-art VTS compensation, normally use diagonal-covariance Gaussians, and fail to model within-component feature correlations. The reason for this is twofold: existing methods do not give good estimates for correlations, and the computational cost of decoding with full covariance matrices is prohibitive. This thesis has presented insights and solutions for both problems.

VTS compensation's estimates for the feature correlations of corrupted speech are unreliable because of the *continuous-time approximation*. It assumes that dynamic coefficients are time derivatives of static coefficients, whereas in reality they are merely approximations. Dynamic coefficients are found with linear regression from a window of static feature vectors. Chapter 5 has introduced model compensation methods that apply the mismatch function (or an approximation) to each time instance in the window separately. By only then applying the linear transformation that the dynamic coefficients are extracted with, compensation becomes much more precise. It then becomes possible to generate good full-covariance compensation with as little adaptation data as standard VTS needs.

However, decoding with full covariance matrices is slow. Chapter 6 has therefore derived methods that approximate full-covariance compensation using linear transformations. This makes decoding faster. These *predictive linear transformations* are versions of well-known adaptive linear transformations, which normally require much more adaptation data than methods for noise-robustness. Also, *joint uncertainty decoding* can compensate a whole base class of components at once. Combining joint uncertainty decoding and predictive linear transformations makes many of the re-

quired statistics cacheable, so that the whole process is fast enough to be implemented in a real-world speech recogniser.

Predictive linear transformations had been introduced before (van Dalen 2007), but chapter 6 has introduced the formal framework for them. Predictive methods approximate a predicted distribution with a different parameterisation. This has been formalised as minimising the KL divergence. The framework also subsumes standard model compensation methods, which minimise the KL divergence to the predicted distribution.

Predictive transformations give a powerful framework for combining the advantages of two forms of distribution. Since the introduction of predictive linear transformations for noise-robustness, other variants have been proposed, some of which derive from VTLN or combine predicted statistics from with statistics directly from data (Flego and Gales 2009; Xu *et al.* 2009; 2011; Breslin *et al.* 2010).

Section 6.4, based on joint work with Federico Flego, has introduced another, fast, variant, which estimates a transformation to minimise the divergence to the predicted corrupted speech, but applies it to speech recogniser features.

9.2 Asymptotically exact likelihoods

Model compensation methods aim to model the corrupted speech distribution, but usually fall short by definition. With Gaussian speech and noise, the corrupted speech is not Gaussian. Standard model compensation assumes it is, and can never provide ideal compensation. Chapter 7 has introduced a more accurate approximation to the corrupted speech likelihood. Rather than a parameterised density, it uses a sampling method, which approximates the integral over speech, noise and phase factor that the likelihood consists of. Because the probability density has an awkward shape, the integral is first transformed. Then, sequential importance resampling deals with the high dimensionality. As the number of samples tends to infinity, this approximation converges to the real likelihood.

Because the method cannot precompute distributions, it is too slow to embed in a speech recogniser. However, it is possible to find the KL divergence from the real corrupted speech distribution to an approximation up to a constant. The new method essentially gives the point where the KL divergence is 0, so it can be assessed how close compensation methods are to the ideal. The KL divergence for different compensation methods appears to predict their word error rates well. One of these compensation methods is iterative data-driven parallel model combination (IDPMC), which takes impractically long to train but it is feasible to run speech recognition with. A version of IDPMC that uses extended feature vectors comes close to transformed-space sampling in terms of cross-entropy, and improves the word error rate substantially. Given the link between the cross-entropy and the word error rate, this should indicate the best possible performance with these speech and noise models, and this mismatch function.

Using the KL divergence technique, it also becomes possible to examine approximations to the mismatch function. These include assuming the corrupted speech distribution Gaussian, and diagonalising that Gaussian's covariance. One common approximation, assuming the phase factor fixed, has seen particular interest in recent years. This work has introduced model compensation using a phase factor distribution for extended VTS, extended DPMC, and extended IDPMC. This has turned out to improve the cross-entropy more than discrimination. In particular, for VTS compensation setting the phase factor to a fixed value other than its mode appears to counter some effects of the vector Taylor series approximation at different signal-to-noise ratios.

9.3 Future work

This thesis has found improved models for the corrupted speech, assuming the speech and noise Gaussian-distributed. This should give insight into what are viable directions of research in model compensation.

The search for better compensation with diagonal-covariance Gaussians continues. Better approximations to the mismatch function (Xu and Chin 2009*b*; Seltzer *et al.* 2010, and the phase factor distribution in section 8.2 of this work) and using an alternative sampling scheme, the unscented transformation (Hu and Huo 2006; van Dalen and Gales 2009*a*; Li *et al.* 2010) have been investigated. This line of research has two issues. First, if the noise distribution is known, the theoretical bound for Gaussian compensation with full covariance matrices, and therefore also for diagonal covariance matrices, is now known. It is given by extended DPMC, presented in this thesis. Second, in practice, it is necessary to estimate the noise model. This is currently possible with maximum-likelihood estimation for standard VTS. Using such a noise model, extended DPMC hardly beats extended VTS (also presented in this thesis, and reasonably fast), whether with full or diagonal covariances. This indicates that any new practical method for Gaussian compensation will need to improve over extended VTS in terms of accuracy, and in terms of noise model estimation. This may be a tough search for little gain. Noise estimation can absorb the differences between compensation schemes if they model the environment reasonably well. It is telling that (with diagonal covariances) a well-tuned implementation of unscented transformations performs an insignificant 0.02 % worse than a well-tuned implementation of VTS (Li *et al.* 2010). It may therefore be advisable to call off the search for new methods of Gaussian compensation.

However, this thesis has only sketched (in section 5.5.2.2) how to directly estimate a noise model for extended VTS. Implementing this would not only make estimation and compensation with EVTS consistent, but also allow full-covariance noise models. For a known noise distribution, a full-covariance Gaussian has yielded some improvement in accuracy (in section 8.1.1.2). It would also be interesting to estimate a noise model that optimises the likelihood for compensation with full-covariance joint uncertainty decoding or predictive transformations directly.

Non-Gaussian compensation may be harder to find, though it may yield bigger gains. This thesis has upper-bounded the gain (in section 8.2), but with slow meth-

ods. Suitably fast non-Gaussian distributions that are better tailored to the corrupted speech distribution in multiple dimensions might help. Alternatively, but potentially even harder, forms of clean speech and noise distributions that when combined through the mismatch function produce a distribution for the corrupted speech that is easier to approximate would be helpful. This work has only given a theoretical bound for Gaussian speech and noise models. However, the techniques presented in this thesis should be general enough to estimate how far from optimal any new proposal is.

There are alternative directions for research on improving noise-robust speech recognition, though. One is to investigate other noise models, for example, with more temporal structure. Again, this thesis has not given theoretical bounds for that.

A more practically-minded strand of research that may follow from this thesis, not necessarily restricted to noise-robustness, is in predictive transformations. It has already become clear that predictive linear transformations bring advantages to other areas than noise-robustness (Breslin *et al.* 2010). That the framework is so general, and formalised in this thesis, provides opportunity for a wide range of interesting instantiations.

Appendices

Appendix A

Standard techniques

This appendix gives details of a number of well-known equalities and algorithms for reference.

A.1 Known equalities

The following useful equalities are well-known.

A.1.1 *Transforming variables of probability distributions*

If variables $\mathbf{x}$ and $\mathbf{y}$ are deterministically linked, a probability distribution over $\mathbf{x}$, $p(\mathbf{x})$, can be converted into one over $\mathbf{y}$ with (see, for example, Bishop 2006, 11.1.1)

$$p(\mathbf{y}) = p(\mathbf{x}) \left| \frac{\partial \mathbf{x}}{\partial \mathbf{y}} \right| . \tag{A.1}$$

A.1.2 *Matrix identities*

The Woodbury identity relates three matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ (see e.g. Petersen and Pedersen 2008, 3.2.2):

$$\left(\mathbf{A} + \mathbf{C}\mathbf{B}\mathbf{C}^{\mathsf{T}}\right)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{C}\left(\mathbf{B}^{-1} + \mathbf{C}^{\mathsf{T}}\mathbf{A}^{-1}\mathbf{C}\right)^{-1}\mathbf{C}^{\mathsf{T}}\mathbf{A}^{-1}. \tag{A.2}$$

The inverse of a block symmetric matrix is given by (see e.g. Petersen and Pedersen 2008, 9.1.3)

$$\begin{bmatrix} \mathbf{A} & \mathbf{C}^\mathsf{T} \\ \mathbf{C} & \mathbf{B} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{D}^{-1} & -\mathbf{D}^{-1}\mathbf{C}^\mathsf{T}\mathbf{B}^{-1} \\ -\mathbf{B}^{-1}\mathbf{C}\mathbf{D}^{-1} & \mathbf{E}^{-1} \end{bmatrix} \tag{A.3a}$$

$$= \begin{bmatrix} \mathbf{D}^{-1} & -\mathbf{A}^{-1}\mathbf{C}^\mathsf{T}\mathbf{E}^{-1} \\ -\mathbf{E}^{-1}\mathbf{C}\mathbf{A}^{-1} & \mathbf{E}^{-1} \end{bmatrix}, \tag{A.3b}$$

where $\mathbf{D} = \mathbf{A} - \mathbf{C}^\mathsf{T}\mathbf{B}^{-1}\mathbf{C}$ is the Schur complement of the matrix with respect to $\mathbf{B}$, and $\mathbf{E} = \mathbf{B} - \mathbf{C}\mathbf{A}^{-1}\mathbf{C}^\mathsf{T}$ is the Schur complement of the matrix with respect to $\mathbf{A}$.

The determinant of the matrix is

$$\left|\begin{bmatrix} \mathbf{A} & \mathbf{C}^\mathsf{T} \\ \mathbf{C} & \mathbf{B} \end{bmatrix}\right| = |\mathbf{A}| \cdot |\mathbf{E}| = |\mathbf{B}| \cdot |\mathbf{D}| \,. \tag{A.4}$$

A.1.3 *Multi-variate Gaussian factorisation*

It can be useful to decompose the evaluation of a multi-variate Gaussian into factors. An obvious choice of factors would be the actual distribution of one coefficient conditional on all previous ones. Straightforward derivations of this usually (e.g. Bishop 2006) assume that the Gaussian is normalised (so that constant factors can be dropped) and assume the input for the Gaussian is linear in the variable of interest (so that the integral over coefficients is constant). These assumptions, however, can not be made in this work. Below derivation therefore explicitly considers all constants.

Let q an unnormalised Gaussian density with parameters $\mathbf{a}$ and $\mathbf{b}$,

$$q\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}\right) = \exp\left(-\frac{1}{2}\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} - \begin{bmatrix} \mu_\mathbf{a} \\ \mu_\mathbf{b} \end{bmatrix}\right)^\mathsf{T} \begin{bmatrix} \Lambda_\mathbf{a} & \Lambda_{\mathbf{ab}} \\ \Lambda_{\mathbf{ba}} & \Lambda_\mathbf{b} \end{bmatrix} \left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} - \begin{bmatrix} \mu_\mathbf{a} \\ \mu_\mathbf{b} \end{bmatrix}\right)\right), \tag{A.5}$$

where Λ is the precision matrix, the inverse of the covariance matrix Σ:

$$\Lambda = \begin{bmatrix} \Lambda_\mathbf{a} & \Lambda_{\mathbf{ab}} \\ \Lambda_{\mathbf{ba}} & \Lambda_\mathbf{b} \end{bmatrix} = \begin{bmatrix} \Sigma_\mathbf{a} & \Sigma_{\mathbf{ab}} \\ \Sigma_{\mathbf{ba}} & \Sigma_\mathbf{b} \end{bmatrix}^{-1} = \Sigma^{-1}. \tag{A.6}$$

From the expression for the inverse of a symmetric block matrix, given in (A.3), it follows that $\boldsymbol{\Sigma}_a^{-1} = \boldsymbol{\Lambda}_a - \boldsymbol{\Lambda}_{ab}\boldsymbol{\Lambda}_b^{-1}\boldsymbol{\Lambda}_{ba}$, which will be useful in the derivation below.

The density can be decomposed into a factor dependent on $\mathbf{a}$ and one dependent on both $\mathbf{a}$ and $\mathbf{b}$. The steps the derivation follows are (A.7a) expanding the terms; (A.7b) gathering terms containing $\mathbf{b}$; (A.7c) completing the square and compensating for that; and finally (A.7d) simplifying.

$$
\begin{aligned}
q\left(\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix}\right) &= \exp\left(-\frac{1}{2}\left(\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\mu}_a\\ \boldsymbol{\mu}_b\end{bmatrix}\right)^{\mathsf{T}} \begin{bmatrix}\boldsymbol{\Lambda}_a & \boldsymbol{\Lambda}_{ab}\\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_b\end{bmatrix} \left(\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\mu}_a\\ \boldsymbol{\mu}_b\end{bmatrix}\right)\right) \\
&= \exp\Big(-\tfrac{1}{2}(\mathbf{a}-\boldsymbol{\mu}_a)^{\mathsf{T}}\boldsymbol{\Lambda}_a(\mathbf{a}-\boldsymbol{\mu}_a) - (\mathbf{b}-\boldsymbol{\mu}_b)^{\mathsf{T}}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a) \\
&\qquad -\tfrac{1}{2}(\mathbf{b}-\boldsymbol{\mu}_b)^{\mathsf{T}}\boldsymbol{\Lambda}_b(\mathbf{b}-\boldsymbol{\mu}_b)\Big) \\
&= \exp\Big(-\tfrac{1}{2}(\mathbf{a}-\boldsymbol{\mu}_a)^{\mathsf{T}}\boldsymbol{\Lambda}_a(\mathbf{a}-\boldsymbol{\mu}_a) - \mathbf{b}^{\mathsf{T}}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a) + \boldsymbol{\mu}_b^{\mathsf{T}}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a) \\
&\qquad -\tfrac{1}{2}\mathbf{b}^{\mathsf{T}}\boldsymbol{\Lambda}_b\mathbf{b} + \mathbf{b}^{\mathsf{T}}\boldsymbol{\Lambda}_b\boldsymbol{\mu}_b - \tfrac{1}{2}\boldsymbol{\mu}_b^{\mathsf{T}}\boldsymbol{\Lambda}_b\boldsymbol{\mu}_b\Big) \qquad \text{(A.7a)} \\
&= \exp\Big(-\tfrac{1}{2}(\mathbf{a}-\boldsymbol{\mu}_a)^{\mathsf{T}}\boldsymbol{\Lambda}_a(\mathbf{a}-\boldsymbol{\mu}_a) + \boldsymbol{\mu}_b^{\mathsf{T}}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a) - \tfrac{1}{2}\boldsymbol{\mu}_b^{\mathsf{T}}\boldsymbol{\Lambda}_b\boldsymbol{\mu}_b \\
&\qquad -\tfrac{1}{2}\mathbf{b}^{\mathsf{T}}\boldsymbol{\Lambda}_b\mathbf{b} + \mathbf{b}^{\mathsf{T}}\boldsymbol{\Lambda}_b\Big(\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_b^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a)\Big)\Big) \qquad \text{(A.7b)} \\
&= \exp\Big(-\tfrac{1}{2}(\mathbf{a}-\boldsymbol{\mu}_a)^{\mathsf{T}}\boldsymbol{\Lambda}_a(\mathbf{a}-\boldsymbol{\mu}_a) + \boldsymbol{\mu}_b^{\mathsf{T}}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a) - \tfrac{1}{2}\boldsymbol{\mu}_b^{\mathsf{T}}\boldsymbol{\Lambda}_b\boldsymbol{\mu}_b \\
&\quad -\tfrac{1}{2}\Big(\mathbf{b} - \Big(\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_b^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a)\Big)\Big)^{\mathsf{T}}\boldsymbol{\Lambda}_b\Big(\mathbf{b} - \Big(\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_b^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a)\Big)\Big) \\
&\quad +\tfrac{1}{2}\Big(\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_b^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a)\Big)^{\mathsf{T}}\boldsymbol{\Lambda}_b\Big(\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_b^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a)\Big)\Big) \qquad \text{(A.7c)} \\
&= \exp\Big(-\tfrac{1}{2}(\mathbf{a}-\boldsymbol{\mu}_a)^{\mathsf{T}}\Big(\boldsymbol{\Lambda}_a - \boldsymbol{\Lambda}_{ab}\boldsymbol{\Lambda}_b^{-1}\boldsymbol{\Lambda}_{ba}\Big)(\mathbf{a}-\boldsymbol{\mu}_a) \\
&\quad -\tfrac{1}{2}\Big(\mathbf{b}^{\mathsf{T}} - \Big(\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_b^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a)\Big)\Big)^{\mathsf{T}}\boldsymbol{\Lambda}_b\Big(\mathbf{b}^{\mathsf{T}} - \Big(\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_b^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a)\Big)\Big)\Big) \\
&= \exp\Big(-\tfrac{1}{2}(\mathbf{a}-\boldsymbol{\mu}_a)^{\mathsf{T}}\boldsymbol{\Sigma}_a^{-1}(\mathbf{a}-\boldsymbol{\mu}_a)\Big) \\
&\exp\Big(-\tfrac{1}{2}\Big(\mathbf{b}^{\mathsf{T}} - \Big(\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_b^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a)\Big)\Big)^{\mathsf{T}}\boldsymbol{\Lambda}_b\Big(\mathbf{b}^{\mathsf{T}} - \Big(\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_b^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a)\Big)\Big)\Big). \\
&\qquad \text{(A.7d)}
\end{aligned}
$$

A normalised Gaussian can be factorised analogously. Using (A.4), the determinant of the block matrix $\boldsymbol{\Sigma}$ can be decomposed into the determinants of the covariance matrices of the two terms in (A.7): $|\boldsymbol{\Sigma}| = |\boldsymbol{\Sigma}_a| \cdot \left|\boldsymbol{\Lambda}_b^{-1}\right|$. A normalised Gaussian then

can be decomposed into two normalised Gaussians:

$$
\begin{aligned}
&\mathcal{N}\left(\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}_a\\ \boldsymbol{\mu}_b\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_a & \boldsymbol{\Sigma}_{ab}\\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_b\end{bmatrix}\right)\\
&= |2\pi\boldsymbol{\Sigma}|^{-1/2}\, q\left(\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix}\right)\\
&= |2\pi\boldsymbol{\Sigma}_a|^{-1/2}\cdot\left|2\pi\boldsymbol{\Lambda}_b^{-1}\right|^{-1/2}\exp\left(-\tfrac{1}{2}(\mathbf{a}-\boldsymbol{\mu}_a)^\top\boldsymbol{\Sigma}_a^{-1}(\mathbf{a}-\boldsymbol{\mu}_a)\right)\\
&\exp\left(-\tfrac{1}{2}\Big(\mathbf{b}^\top-\Big(\boldsymbol{\mu}_b-\boldsymbol{\Lambda}_b^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a)\Big)\Big)^\top\boldsymbol{\Lambda}_b\Big(\mathbf{b}^\top-\Big(\boldsymbol{\mu}_b-\boldsymbol{\Lambda}_b^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a)\Big)\Big)\right)\\
&= \mathcal{N}(\mathbf{a};\,\boldsymbol{\mu}_a,\,\boldsymbol{\Sigma}_a)\,\mathcal{N}\Big(\mathbf{b};\,\boldsymbol{\mu}_b-\boldsymbol{\Lambda}_b^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a),\,\boldsymbol{\Lambda}_b^{-1}\Big)\\
&= \mathcal{N}(\mathbf{a};\,\boldsymbol{\mu}_a,\,\boldsymbol{\Sigma}_a)\,\mathcal{N}\Big(\mathbf{b};\,\boldsymbol{\mu}_b+\boldsymbol{\Sigma}_{ba}\boldsymbol{\Sigma}_a^{-1}(\mathbf{a}-\boldsymbol{\mu}_a),\,\boldsymbol{\Sigma}_b-\boldsymbol{\Sigma}_{ba}\boldsymbol{\Sigma}_a^{-1}\boldsymbol{\Sigma}_{ab}\Big)\,. \qquad \text{(A.8)}
\end{aligned}
$$

If the density q is a probability distribution and $\mathbf{a}$ and $\mathbf{b}$ are distributed according to it:

$$
\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_a\\ \boldsymbol{\mu}_b\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_a & \boldsymbol{\Sigma}_{ab}\\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_b\end{bmatrix}\right), \qquad \text{(A.9)}
$$

then the two factors in (A.8) are the marginal probability distributions of $\mathbf{a}$ and the distribution of $\mathbf{b}$ conditional on $\mathbf{a}$, so that

$$
\begin{aligned}
\mathbf{a} &\sim \mathcal{N}(\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a)\,; && \text{(A.10a)}\\
\mathbf{b}|\mathbf{a} &\sim \mathcal{N}\Big(\boldsymbol{\mu}_b-\boldsymbol{\Lambda}_b^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a}-\boldsymbol{\mu}_a), \boldsymbol{\Lambda}_b^{-1}\Big) && \text{(A.10b)}\\
&\sim \mathcal{N}\Big(\boldsymbol{\mu}_b+\boldsymbol{\Sigma}_{ba}\boldsymbol{\Sigma}_a^{-1}(\mathbf{a}-\boldsymbol{\mu}_a), \boldsymbol{\Sigma}_b-\boldsymbol{\Sigma}_{ba}\boldsymbol{\Sigma}_a^{-1}\boldsymbol{\Sigma}_{ab}\Big)\,. && \text{(A.10c)}
\end{aligned}
$$

This is a standard result. Note that the distribution of $\mathbf{a}$ is more concisely expressed in terms of the joint's covariance matrix, and the distribution of $\mathbf{b}|\mathbf{a}$ in terms of the precision matrix.

A.2 Kullback-Leibler divergence

An important tool in this work is the Kullback-Leibler (KL) divergence (Kullback and Leibler 1951). It measures the difference between two distributions. If p and q are

distributions over a continuous domain, the KL divergence between them, $\mathcal{KL}(p\|q)$ is defined as

$$\mathcal{KL}(p\|q) = \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x}. \tag{A.11}$$

In this work it is used both as a criterion for optimisation and to assess the accuracy of models. The KL divergence has the following properties.

The expression in (A.11) can be decomposed as

$$\mathcal{KL}(p\|q) = \mathcal{H}(p\|q) - \mathcal{H}(p)\,, \tag{A.12a}$$

where $\mathcal{H}(p\|q)$ is the cross-entropy of p and q,

$$\mathcal{H}(p\|q) = -\int p(\mathbf{x}) \log q(\mathbf{x}) d\mathbf{x}, \tag{A.12b}$$

and $\mathcal{H}(p)$ is the entropy of p,

$$\mathcal{H}(p) = -\int p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x}. \tag{A.12c}$$

The KL divergence is always non-negative, since the cross-entropy is always greater than or equal to the entropy. If and only if the distributions have the same density for each $\mathbf{x}$, the cross-entropy and the entropy are equal, so that the KL divergence becomes 0.

To find a distribution of a particular form that best matches another distribution, minimising the KL divergence is often a useful criterion. Well-known algorithms for training distribution parameters such as *expectation–maximisation*, described in appendix 2.3.2.1, can be seen as minimising a KL divergence. Inference in graphical models is often expressed as minimising a KL divergence as well. The expectation propagation and belief propagation algorithms are examples.

When optimising q or comparing different distributions q, the reference distribution p is often fixed. In that case, the cross-entropy is the KL divergence up to a constant. Optimising the cross-entropy, or comparing merely the cross-entropy, is therefore often a valid alternative for working with the KL divergence.

A.2.1 KL *divergence between Gaussians*

A specific case of interest is the KL divergence between two Gaussians. If two distributions p and q over a d-dimensional space are defined

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x};\, \boldsymbol{\mu}_a,\, \boldsymbol{\Sigma}_a)\,; \qquad q(\mathbf{x}) = \mathcal{N}(\mathbf{x};\, \boldsymbol{\mu}_b,\, \boldsymbol{\Sigma}_b)\,, \tag{A.13a}$$

then the KL divergence between the distributions is

$$\mathcal{KL}(p\|q) = \frac{1}{2}\left(\log\left(\frac{|\boldsymbol{\Sigma}_b|}{|\boldsymbol{\Sigma}_a|}\right) + \operatorname{Tr}\left(\boldsymbol{\Sigma}_b^{-1}\boldsymbol{\Sigma}_a\right) + (\boldsymbol{\mu}_b - \boldsymbol{\mu}_a)^\mathsf{T}\boldsymbol{\Sigma}_b^{-1}(\boldsymbol{\mu}_b - \boldsymbol{\mu}_a) - d\right). \tag{A.13b}$$

A sub-case of this is when the dimensions for both distributions can be partitioned into blocks of dimensions that are mutually independent. The KL divergence then becomes a sum of KL divergences for these blocks. Without loss of generality, assume that the covariance matrices are block-diagonal with

$$\boldsymbol{\Sigma}_a = \begin{bmatrix} \boldsymbol{\Sigma}_{a,1} & 0 \\ 0 & \boldsymbol{\Sigma}_{a,2} \end{bmatrix}; \qquad \boldsymbol{\Sigma}_b = \begin{bmatrix} \boldsymbol{\Sigma}_{b,1} & 0 \\ 0 & \boldsymbol{\Sigma}_{b,2} \end{bmatrix}. \tag{A.14a}$$

Both distributions can then be factorised as distributions over $\mathbf{x}_1$ and $\mathbf{x}_2$, with $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1^\mathsf{T} & \mathbf{x}_2^\mathsf{T} \end{bmatrix}^\mathsf{T}$:

$$p(\mathbf{x}) = p_1(\mathbf{x}_1)p_2(\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_1;\, \boldsymbol{\mu}_{a,1},\, \boldsymbol{\Sigma}_{a,1})\,\mathcal{N}(\mathbf{x}_2;\, \boldsymbol{\mu}_{a,2},\, \boldsymbol{\Sigma}_{a,2})\,; \tag{A.14b}$$

$$q(\mathbf{x}) = q_1(\mathbf{x}_1)q_2(\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_1;\, \boldsymbol{\mu}_{b,1},\, \boldsymbol{\Sigma}_{b,1})\,\mathcal{N}(\mathbf{x}_2;\, \boldsymbol{\mu}_{b,2},\, \boldsymbol{\Sigma}_{b,2}) \tag{A.14c}$$

The KL divergence in (A.13b) then becomes the sum of the parallel divergences between the two factors:

$$\mathcal{KL}(p\|q) = \mathcal{KL}(p_1\|q_1) + \mathcal{KL}(p_2\|q_2)\,. \tag{A.14d}$$

This is true for the KL divergence between any two distributions that display independence between blocks of dimensions. By applying this equation recursively, the KL divergence between any two Gaussians with the same block-diagonal covariance matrices is the sum of the KL divergences of the factors in parallel. The most factorised Gaussians have diagonal covariances, in which case the KL divergence works per dimension. This will be useful both when training parameters of a distribution, and when assessing the distance to the real distribution.

A.2.2 *Between mixtures*

The KL divergence between mixtures does not have a closed form. A number of approximations with different properties (Yu 2006; Hershey and Olsen 2007) are possible. For section 6, it will be useful to minimise the KL divergence between two mixtures of Gaussians. The following will therefore describe a variational approximation to the KL divergence introduced independently by Yu (2006); Hershey and Olsen (2007).[1] This approximation can then be minimised as a proxy for minimising the exact one.

Let p and q be mixture models, with components indexed with m and n respectively:

$$p(\mathbf{x}) = \sum_m \pi^{(m)} p^{(m)}(\mathbf{x}); \qquad q(\mathbf{x}) = \sum_n \omega^{(n)} q^{(n)}(\mathbf{x}), \tag{A.15}$$

with $p^{(m)}(\mathbf{x})$ and $q^{(n)}(\mathbf{x})$ the component distributions, and $\boldsymbol{\pi}$ and $\boldsymbol{\omega}$ the weight vectors, with $\sum_m \pi^{(m)} = \sum_n \omega^{(n)} = 1$.

The KL divergence can be written as the difference between the cross-entropy and the entropy, as in (A.12). The variational approximation that will be presented here finds an upper bound on both the cross-entropy and on the entropy separately. This implies that there is no guarantee for the approximation to the KL divergence to be on either side of the real one. However, in this thesis the approximation is used to minimise $\mathcal{KL}(p\|q)$ with respect to q. Since $\mathcal{H}(p)$ is not a function of q, it suffices to minimise the upper bound on $\mathcal{H}(p\|q)$. The following will therefore present the upper bound on $\mathcal{H}(p\|q)$; the upper bound on $\mathcal{H}(p)$ can be found analogously.

A set of variational parameters $\boldsymbol{\phi}$ is introduced that partition the weight of each component of p up into parts representing the components of q:

$$\sum_n \phi_n^{(m)} = 1, \tag{A.16}$$

with $\phi_n^{(m)} \geq 0$.

[1] The description in Hershey and Olsen (2007) was originally meant for mixtures of Gaussians, which is also what this work will use it for, but the derivation is valid for mixtures of any type of distribution, as Dognin *et al.* (2009) acknowledge.

The following derivation of an upper bound $\mathcal{F}(p, q, \boldsymbol{\phi})$ to the cross-entropy uses Jensen's inequality. It moves a summation outside a logarithm in (A.17a), and because of Jensen's inequality, the result is less than or equal to the logarithm of the sum. In this case, the expressions are negated, so that the result is greater than or equal.

$$
\begin{aligned}
\mathcal{H}(p\|q) &= -\int p(\mathbf{x}) \log q(\mathbf{x}) d\mathbf{x} \\
&= -\int \sum_m \pi^{(m)} p^{(m)}(\mathbf{x}) \log\left(\sum_n \omega^{(n)} q^{(n)}(\mathbf{x})\right) d\mathbf{x} \\
&= -\int \sum_m \pi^{(m)} p^{(m)}(\mathbf{x}) \log\left(\sum_n \phi_n^{(m)} \frac{\omega^{(n)} q^{(n)}(\mathbf{x})}{\phi_n^{(m)}}\right) d\mathbf{x} \\
&\leq -\int \sum_m \pi^{(m)} p^{(m)}(\mathbf{x}) \sum_n \phi_n^{(m)} \log\left(\frac{\omega^{(n)} q^{(n)}(\mathbf{x})}{\phi_n^{(m)}}\right) d\mathbf{x} \qquad \text{(A.17a)} \\
&= -\sum_m \sum_n \pi^{(m)} \phi_n^{(m)} \left(\int p^{(m)}(\mathbf{x}) \log q^{(n)}(\mathbf{x}) d\mathbf{x} + \log \frac{\omega^{(n)}}{\phi_n^{(m)}}\right) \\
&= \sum_m \sum_n \pi^{(m)} \phi_n^{(m)} \left(\mathcal{H}\left(p^{(m)} \middle\| q^{(n)}\right) + \log \frac{\phi_n^{(m)}}{\omega^{(n)}}\right) \\
&\triangleq \mathcal{F}(p, q, \boldsymbol{\phi}). \qquad \text{(A.17b)}
\end{aligned}
$$

Derivatives of this with respect to the variational parameters are

$$
\begin{aligned}
\frac{d\mathcal{F}(p, q, \boldsymbol{\phi})}{d\phi_n^{(m)}} &= \frac{d\pi^{(m)} \phi_n^{(m)} \left(\mathcal{H}\left(p^{(m)} \middle\| q^{(n)}\right) - \log \omega^{(n)} + \log \phi_n^{(m)}\right)}{d\phi_n^{(m)}} \\
&= \pi^{(m)} \left(\mathcal{H}\left(p^{(m)} \middle\| q^{(n)}\right) - \log \omega^{(n)} + \log \phi_n^{(m)} + 1\right); \qquad \text{(A.18a)} \\
\frac{d^2\mathcal{F}(p, q, \boldsymbol{\phi})}{d^2\phi_n^{(m)}} &= \frac{\pi^{(m)}}{\phi_n^{(m)}}. \qquad \text{(A.18b)}
\end{aligned}
$$

On the domain of $\phi_n^{(m)}$, which is $[0, 1]$, its second derivative is non-negative, so that the upper bound is convex.

The upper bound is minimised with respect to the variational parameters $\phi_n^{(m)}$.

The optimisation under the constraints in (A.16) uses Lagrange multipliers:

$$\begin{aligned} 0 &= \frac{\mathrm{d}\mathcal{F}(p, q, \boldsymbol{\phi}) + \lambda\left(\sum_{n'} \phi_{n'}^{(m)} - 1\right)}{\mathrm{d}\phi_n^{(m)}}; \\ \log \phi_n^{(m)} &= \log \omega^{(n)} - \mathcal{H}\left(p^{(m)} \middle\| q^{(n)}\right) - 1 - \frac{\lambda}{\pi^{(m)}}; \\ \phi_n^{(m)} &= \omega^{(n)} \exp\left(-\mathcal{H}\left(p^{(m)} \middle\| q^{(n)}\right) - 1 - \frac{\lambda}{\pi^{(m)}}\right), \end{aligned} \tag{A.19}$$

which, setting λ to satisfy the constraint in (A.16), gives the optimal parameter setting

$$\phi_n^{(m)} := \frac{\omega^{(n)} \exp\left(-\mathcal{H}\left(p^{(m)} \middle\| q^{(n)}\right)\right)}{\sum_{n'} \omega^{(n')} \exp\left(-\mathcal{H}\left(p^{(m)} \middle\| q^{(n')}\right)\right)}. \tag{A.20}$$

Optimising $\boldsymbol{\phi}$ does not change the real cross-entropy; it merely finds a tighter bound. In section 6, this bound is used to optimise the cross-entropy itself with respect to q.

A parameter setting that is of interest results in the *matched-pair bound*. This assumes a one-to-one mapping from components of p to components of q. Assuming that component m in p corresponds to component m in q, the parameters can be set to[2]

$$\phi_n^{(m)} = \begin{cases} 1, & m = n; \\ 0, & m \neq n. \end{cases} \tag{A.21}$$

This reduces the upper bound to

$$\begin{aligned} \mathcal{F}(p, q, \boldsymbol{\phi}) &= \sum_m \pi^{(m)} \left(\mathcal{H}\left(p^{(m)} \middle\| q^{(m)}\right) - \log \omega^{(m)}\right) \\ &= \mathcal{H}\left(\boldsymbol{\pi} \middle\| \boldsymbol{\omega}\right) + \sum_m \pi^{(m)} \mathcal{H}\left(p^{(m)} \middle\| q^{(m)}\right). \end{aligned} \tag{A.22}$$

If additionally the component priors for both mixtures are kept equal, $\boldsymbol{\pi} = \boldsymbol{\omega}$, then optimising the term behind the $+$ sign will tighten this upper bound. In section 6.1.1 this bound is used to find distributions $q^{(m)}$ that approximate $p^{(m)}$, where both derive from the same speech recogniser model set but are parameterised differently.

[2] This is easy to generalise to the case where there is a different deterministic mapping from each component of p to a component of q.

A.3 Expectation–maximisation

Expectation–maximisation aims to maximise $\mathcal{L}(\tilde{p}, q_{\mathcal{U}\mathcal{X}})$ by updating a lower-bound function $\mathcal{F}(\tilde{p}, \rho, q_{\mathcal{U}\mathcal{X}})$ of the likelihood. The lower-bound explicitly uses a distribution over the hidden variables $\rho(\mathcal{U}|\mathcal{X})$. ρ and $q_{\mathcal{U}\mathcal{X}}$ are optimised iteratively. First, ρ must be set to make the lower-bound function equal to the log-likelihood (in the "expectation" step). Then, $q_{\mathcal{U}\mathcal{X}}$ is set to maximise the lower bound (in the "maximisation" step). The generalised EM algorithm replaces this maximisation by an improvement. It is possible to prove, with Jensen's inequality, that the lower bound is indeed lower or equal to the log-likelihood, so that the log-likelihood is guaranteed not to decrease if the lower bound does not decrease. The following details the expectation–maximisation algorithm and its proof.

The statistical model whose parameters are trained will be denoted with $q_{\mathcal{U}\mathcal{X}}(\mathcal{U}, \mathcal{X})$, which is a distribution over the hidden and observed variables. Marginalising out over the hidden variables gives the distribution over the observed variables:

$$q_{\mathcal{X}}(\mathcal{X}) = \int q_{\mathcal{U}\mathcal{X}}(\mathcal{U}, \mathcal{X}) \mathrm{d}\mathcal{U} \tag{A.23a}$$

The log-likelihood for one data point is then

$$\mathcal{L}(\mathcal{X}, q_{\mathcal{U}\mathcal{X}}) \triangleq \log \int q_{\mathcal{U}\mathcal{X}}(\mathcal{U}, \mathcal{X}) \mathrm{d}\mathcal{U}, \tag{A.23b}$$

and for the whole training data

$$\mathcal{L}(\tilde{p}, q_{\mathcal{U}\mathcal{X}}) \triangleq \int \tilde{p}(\mathcal{X}) \mathcal{L}(\mathcal{X}, q_{\mathcal{U}\mathcal{X}}) \mathrm{d}\mathcal{X}. \tag{A.23c}$$

The lower bound that expectation–maximisation optimises is defined for a single data point $\mathcal{X}$ as

$$\mathcal{F}(\mathcal{X}, \rho, q_{\mathcal{U}\mathcal{X}}) \triangleq \int \rho(\mathcal{U}|\mathcal{X}) \log \frac{q_{\mathcal{U}\mathcal{X}}(\mathcal{U}, \mathcal{X})}{\rho(\mathcal{U}|\mathcal{X})} \mathrm{d}\mathcal{U} \tag{A.24a}$$

and for the empirical distribution $\tilde{p}$ representing all training data as

$$\mathcal{F}(\tilde{p}, \rho, q_{\mathcal{U}\mathcal{X}}) \triangleq \int \tilde{p}(\mathcal{X}) \mathcal{F}(\mathcal{X}, \rho, q_{\mathcal{U}\mathcal{X}}) \mathrm{d}\mathcal{X}. \tag{A.24b}$$

The following will optimise the lower bound as a surrogate for optimising the actual log-likelihood. The expectation and the maximisation steps rewrite the lower bound differently to make it possible to optimise its parameters.

A.3.1 *The expectation step: the hidden variables*

Compared to the log-likelihood, the lower bound function takes an extra parameter, ρ. The expectation step sets this distribution so that the lower bound is maximised, and equals the log-likelihood. This uses the posterior distribution of the hidden variables given the data according to the current mode parameters, which will be written $q_{\mathcal{U}|\mathcal{X}}$ with

$$q_{\mathcal{U}|\mathcal{X}}(\mathcal{U}|\mathcal{X}) = \frac{q_{\mathcal{U}\mathcal{X}}(\mathcal{U},\mathcal{X})}{q_{\mathcal{X}}(\mathcal{X})}. \tag{A.25}$$

For a given observation $\mathcal{X}$, the lower bound can be written as a sum of the log-likelihood (independent of the hidden variables) and a KL divergence:

$$\begin{aligned}
\mathcal{F}(\mathcal{X}, \rho, q_{\mathcal{U}\mathcal{X}}) &= \int \rho(\mathcal{U}|\mathcal{X}) \log \frac{q_{\mathcal{U}\mathcal{X}}(\mathcal{U},\mathcal{X})}{\rho(\mathcal{U}|\mathcal{X})} d\mathcal{U} \\
&= \int \rho(\mathcal{U}|\mathcal{X}) \left(\log q_{\mathcal{X}}(\mathcal{X}) + \log \frac{q_{\mathcal{U}|\mathcal{X}}(\mathcal{U}|\mathcal{X})}{\rho(\mathcal{U}|\mathcal{X})} \right) d\mathcal{U} \\
&= \log q_{\mathcal{X}}(\mathcal{X}) + \int \rho(\mathcal{U}|\mathcal{X}) \log \frac{q_{\mathcal{U}|\mathcal{X}}(\mathcal{U}|\mathcal{X})}{\rho(\mathcal{U}|\mathcal{X})} d\mathcal{U} \\
&= \mathcal{L}(\mathcal{X}, q_{\mathcal{U}\mathcal{X}}) - \mathcal{KL}\big(\rho \big\| q_{\mathcal{U}|\mathcal{X}}\big),
\end{aligned} \tag{A.26}$$

The log-likelihood $\mathcal{L}(\mathcal{X}, q_{\mathcal{U}\mathcal{X}})$ does not depend on ρ. To make the lower bound equal to the log-likelihood, the right-hand term needs to be zero. The Kullback-Leibler divergence $\mathcal{KL}(\cdot\|\cdot)$ of two distributions is always non-negative, and zero when the distributions are identical. $\mathcal{F}$ is therefore maximised when ρ is set to the hidden variable posterior $q_{\mathcal{U}|\mathcal{X}}$ for all observations in $\tilde{p}$:

$$\begin{aligned}
\rho^{(k)} &:= \arg\min_{\rho} \mathcal{F}\big(\tilde{p}, \rho, q^{(k)}_{\mathcal{U}\mathcal{X}}\big) \\
&= \arg\min_{\rho} \int \tilde{p}(\mathcal{X}) \mathcal{KL}\big(\rho \big\| q^{(k-1)}_{\mathcal{U}|\mathcal{X}}\big)\, d\mathcal{X} \\
&= q^{(k-1)}_{\mathcal{U}|\mathcal{X}}.
\end{aligned} \tag{A.27}$$

By setting $\rho^{(k)}$ to the hidden variable posterior, the Kullback-Leibler divergence in (A.26) becomes 0, so that

$$\mathcal{L}\big(\tilde{p}, q_{\mathcal{X}}^{(k-1)}\big) = \mathcal{F}\big(\tilde{p}, \rho^{(k)}, q_{\mathcal{U}\mathcal{X}}^{(k-1)}\big). \tag{A.28}$$

This is the first part of the proof of convergence of expectation–maximisation.

A.3.2 *The maximisation step: the model parameters*

The second step of expectation–maximisation, the maximisation step, optimises the parameters of the model, $q_{\mathcal{U}\mathcal{X}}$. Again, like in (A.26), the expression for the lower bound for a single data point is rewritten, this time straightforwardly as a term dependent on $q_{\mathcal{U}\mathcal{X}}$ and one independent:

$$\begin{aligned}\mathcal{F}(\mathcal{X}, \rho, q_{\mathcal{U}\mathcal{X}}) &= \int \rho(\mathcal{U}|\mathcal{X}) \log \frac{q_{\mathcal{U}\mathcal{X}}(\mathcal{U}, \mathcal{X})}{\rho(\mathcal{U}|\mathcal{X})} \mathrm{d}\mathcal{U} \\ &= \int \rho(\mathcal{U}|\mathcal{X}) \log q_{\mathcal{U}\mathcal{X}}(\mathcal{U}, \mathcal{X}) \mathrm{d}\mathcal{U} - \int \rho(\mathcal{U}|\mathcal{X}) \log \rho(\mathcal{U}|\mathcal{X}) \mathrm{d}\mathcal{U}. \end{aligned} \tag{A.29}$$

The right-hand term in this expression is constant when optimising $\mathcal{F}$ with respect to $q_{\mathcal{U}\mathcal{X}}$ in the maximisation step. The new estimate for $q_{\mathcal{U}\mathcal{X}}$ is therefore chosen

$$\begin{aligned} q_{\mathcal{U}\mathcal{X}}^{(k)} &:= \arg\max_{q_{\mathcal{U}\mathcal{X}}} \mathcal{F}(\tilde{p}, \rho^{(k)}, q_{\mathcal{U}\mathcal{X}}) \\ &= \arg\max_{q_{\mathcal{U}\mathcal{X}}} \int \tilde{p}(\mathcal{X}) \mathcal{F}(\mathcal{X}, \rho^{(k)}, q_{\mathcal{U}\mathcal{X}}) \mathrm{d}\mathcal{X} \\ &= \arg\max_{q_{\mathcal{U}\mathcal{X}}} \int \tilde{p}(\mathcal{X}) \int \rho^{(k)}(\mathcal{U}|\mathcal{X}) \log q_{\mathcal{U}\mathcal{X}}(\mathcal{U}, \mathcal{X}) \mathrm{d}\mathcal{U} \mathrm{d}\mathcal{X}. \end{aligned} \tag{A.30}$$

How to perform this maximisation depends on the specific problem. If it is not possible to maximise $\mathcal{F}$, then *generalised* EM can be used. It merely requires that $q_{\mathcal{U}\mathcal{X}}$ does not decrease. In either case, the lower bound is guaranteed to remain equal or increase:

$$\mathcal{F}\big(\tilde{p}, \rho^{(k)}, q_{\mathcal{U}\mathcal{X}}^{(k-1)}\big) \leq \mathcal{F}\big(\tilde{p}, \rho^{(k)}, q_{\mathcal{U}\mathcal{X}}^{(k)}\big). \tag{A.31}$$

This is the second part of the proof of convergence of expectation–maximisation.

A.3.3 *Convergence*

The objective of expectation–maximisation is to increase the log-likelihood at every iteration. It can be proven that the log-likelihood never decreases. The last step of this proof requires Jensen's inequality, which states that for a convex function $\phi(x)$ (for example, the log function), inputs x_i, and non-negative weights π_i, the weighted sum of the function applied to the inputs is never greater than the function applied to the weighted sum of the inputs:

$$\sum_i \pi_i \phi(x_i) \leq \phi\left(\sum_i \pi_i x_i\right). \tag{A.32}$$

The relation between $\mathcal{F}$ and $\mathcal{L}$ is the analogue in the continuous domain. For one observation $\mathcal{X}$, $\mathcal{F}$ was defined in (A.24a). By applying Jensen's inequality, $\mathcal{F}$ turns out to be related to lower bound $\mathcal{L}$ expressed as marginalising out over the hidden variables (as it was in (A.23b)):

$$\begin{aligned}
\mathcal{F}(\mathcal{X}, \rho, q_{\mathcal{U}\mathcal{X}}) &= \int \rho(\mathcal{U}|\mathcal{X}) \log \frac{q_{\mathcal{U}\mathcal{X}}(\mathcal{U}, \mathcal{X})}{\rho(\mathcal{U}|\mathcal{X})} d\mathcal{U} \\
&\leq \log \int \rho(\mathcal{U}|\mathcal{X}) \frac{q_{\mathcal{U}\mathcal{X}}(\mathcal{U}, \mathcal{X})}{\rho(\mathcal{U}|\mathcal{X})} d\mathcal{U} \\
&= \log \int q_{\mathcal{U}\mathcal{X}}(\mathcal{U}, \mathcal{X}) d\mathcal{U} = \mathcal{L}(\mathcal{X}, q_{\mathcal{U}\mathcal{X}}).
\end{aligned} \tag{A.33}$$

This same relation then goes for the log-likelihood and the lower bound over the full training data:

$$\begin{aligned}
\mathcal{F}(\tilde{p}, \rho, q_{\mathcal{U}\mathcal{X}}) &= \int \tilde{p}(\mathcal{X}) \mathcal{F}(\mathcal{X}, \rho, q_{\mathcal{U}\mathcal{X}}) d\mathcal{X} \\
&\leq \int \tilde{p}(\mathcal{X}) \mathcal{L}(\mathcal{X}, q_{\mathcal{U}\mathcal{X}}) d\mathcal{X} = \mathcal{L}(\tilde{p}, q_{\mathcal{U}\mathcal{X}}).
\end{aligned} \tag{A.34}$$

This is the final part of the proof of convergence of expectation–maximisation. Combining (A.28), (A.31), and (A.34):

$$\mathcal{L}\big(\tilde{p}, q_{\mathcal{X}}^{(k-1)}\big) = \mathcal{F}\big(\tilde{p}, \rho^{(k)}, q_{\mathcal{U}\mathcal{X}}^{(k-1)}\big) \leq \mathcal{F}\big(\tilde{p}, \rho^{(k)}, q_{\mathcal{U}\mathcal{X}}^{(k)}\big) \leq \mathcal{L}\big(\tilde{p}, q_{\mathcal{X}}^{(k)}\big). \tag{A.35}$$

This proves that the expectation–maximisation does not decrease the likelihood of the data. Note that the algorithm is not guaranteed to find a global maximum. In practice, it often does find a useful parameter setting.

A.4 Monte Carlo

The previous section considered training data, which naturally consists of a finite number of training samples, as an empirical distribution. Even when parameterised forms of distributions are available, using them directly is not always tractable. In these cases, it is often necessary to produce empirical distributions from parameterised distributions by sampling. Methods that approximate a *target density* with a finite number of samples are called *Monte Carlo* methods.

Many Monte Carlo methods can work with unnormalised densities, which for many applications is a useful feature. Markov chain Monte Carlo, for example, divides the value of the density at two points by each other, so that any normalisation constant cancels out, and can be disregarded. However, the value required in this thesis, the integral of a target density over the whole of a space, is the normalisation constant of the density. The samples themselves are merely a by-product.

This requirement rules out many Monte Carlo methods. One technique, called *importance sampling*, does return an approximation to the normalisation constant. It requires a (normalised) *proposal distribution* that samples can be drawn from and is close in shape to the target density. *Sequential importance sampling* is importance sampling over a multi-dimensional space. In itself, it is just importance sampling that deals explicitly with one dimension at a time. It becomes advantageous once *resampling* is introduced between dimensions. This removes low-weight samples and duplicates high-weight ones, so that the samples focus on the most interesting, high-probability regions of space.

Sequential sampling techniques are often presented as traversing through time. There is no reason, however, why the dimensions should represent time. In this work, dimensions will relate to elements of feature vectors and will be called just "dimensions". This section follows the presentation of Doucet and Johansen (2008). It will discuss implicitly multi-dimensional Monte Carlo and importance sampling. Then, resampling is introduced. Finally, section A.4.5 on page 277 discusses the case where for some dimensions it is possible to draw from the target distribution, and for some

it is not.

A.4.1 *Plain Monte Carlo*

Monte Carlo methods approximate a target distribution with a finite number of samples. Denote the target probability distribution with π. If it is possible to draw L samples $\mathbf{u}^{(l)} \sim \pi$, the Monte Carlo approximation of π is the empirical distribution

$$\tilde{\pi} = \frac{1}{L} \sum_{l} \delta_{\mathbf{u}^{(l)}}, \tag{A.36}$$

where $\delta_{\cdot}$ denotes the Dirac delta. Using the empirical measure in the place of the target distribution, the expectation of any test function ϕ under distribution π can be approximated as

$$\mathcal{E}_{\pi}\{\phi(\mathbf{u})\} = \int \pi(\mathbf{u})\phi(\mathbf{u})d\mathbf{u} \simeq \int \tilde{\pi}(\mathbf{u})\phi(\mathbf{u})d\mathbf{u} = \frac{1}{L} \sum_{l} \phi\left(\mathbf{u}^{(l)}\right). \tag{A.37}$$

This equation is used in section 4.4.1 to estimate empirical means (with $\phi(\mathbf{u}) = \mathbf{u}$) and second moments (with $\phi(\mathbf{u}) = \mathbf{u}\mathbf{u}^{\mathsf{T}}$).

This straightforward Monte Carlo method requires, however, that it is possible to sample directly from the target distribution. When this is not the case, it is often possible to use *importance sampling*, which draws samples from a *proposal distribution* that is close to the target distribution, and then assigns the samples weights to make up for the difference.

A.4.2 *Importance sampling*

If it is impossible to sample from the target distribution, it can still be possible to approximate it by sampling from a proposal distribution ρ similar to the target distribution, and make up for the difference by weighting the samples. This is called *importance sampling*. The weights also give an approximation to the normalisation constant. The process is analogous to the evaluation of a function under a distribution in (A.37). However, L samples $\mathbf{u}^{(l)} \sim \rho$ are drawn from the proposal distribution,

so that the empirical proposal distribution of it is, analogously to (A.36):

$$\tilde{\rho} = \frac{1}{L} \sum_{l} \delta_{\mathbf{u}^{(l)}}. \tag{A.38}$$

That samples are drawn from a proposal distribution and not directly from the target distribution makes it possible to use an unnormalised target density, γ. This is often useful if the normalisation constant of the target density is unknown. It does need to be possible to evaluate γ at any point. Importance sampling finds samples from the distribution at the same time as an approximation to the normalisation constant. γ is a scaled version of π:

$$\pi(\mathbf{u}) = \frac{\gamma(\mathbf{u})}{Z}, \tag{A.39a}$$

where the normalising constant is

$$Z = \int \gamma(\mathbf{u}) d\mathbf{u}. \tag{A.39b}$$

The proposal density needs to cover at least the area that the target distribution covers:

$$\pi(\mathbf{u}) > 0 \Rightarrow \rho(\mathbf{u}) > 0, \tag{A.40}$$

otherwise no samples will be drawn in some regions where π is non-zero.

The key to making up for the difference between proposal and target is the weight function $w(\mathbf{u})$. It gives the ratio between the target density and the proposal distribution:

$$w(\mathbf{u}) = \frac{\gamma(\mathbf{u})}{\rho(\mathbf{u})}. \tag{A.41}$$

Substituting $w(\mathbf{u})\rho(\mathbf{u})$ for $\gamma(\mathbf{u})$ in (A.39a) and (A.39b),

$$\pi(\mathbf{u}) = \frac{w(\mathbf{u})\rho(\mathbf{u})}{Z}; \tag{A.42a}$$

$$Z = \int w(\mathbf{u})\rho(\mathbf{u}) d\mathbf{u}. \tag{A.42b}$$

Now that the target distribution has been expressed in terms of the proposal distribution ρ, the proposal distribution can be replaced by its empirical version $\tilde{\rho}$ in (A.38). This yields the empirical distribution to π, $\tilde{\pi}$, with the samples from ρ weighted by their importance weight:

$$\tilde{\pi} = \frac{1}{L}\sum_{l} \frac{w\big(\mathbf{u}^{(l)}\big)}{\tilde{Z}}\delta_{\mathbf{u}^{(l)}} = \sum_{l} \overline{w}^{(l)}\delta_{\mathbf{u}^{(l)}}, \tag{A.43a}$$

where approximation to the normalisation constant is

$$\tilde{Z} = \int w(\mathbf{u})\tilde{\rho}(\mathbf{u})\mathrm{d}\mathbf{u} = \frac{1}{L}\sum_{l} w\big(\mathbf{u}^{(l)}\big), \tag{A.43b}$$

and the normalised weights $\overline{w}^{(l)}$ are

$$\overline{w}^{(l)} = \frac{w\big(\mathbf{u}^{(l)}\big)}{\sum_{l'} w\big(\mathbf{u}^{(l')}\big)}. \tag{A.43c}$$

$\tilde{\pi}$ is the normalised importance sampling approximation to target distribution γ, and $\tilde{Z}$ is the corresponding approximation to the normalisation constant.

The expectation of a test function $\phi(\mathbf{u})$ under π can be approximated analogously to (A.37), with $\tilde{\pi}$ given by (A.43a):

$$\mathcal{E}_{\pi}\{\phi(\mathbf{u})\} = \int \pi(\mathbf{u})\phi(\mathbf{u})\mathrm{d}\mathbf{u} \simeq \int \tilde{\pi}(\mathbf{u})\phi(\mathbf{u})\mathrm{d}\mathbf{u} = \frac{1}{L}\sum_{l} \overline{w}^{(l)}\phi\big(\mathbf{u}^{(l)}\big). \tag{A.44}$$

A degenerate case of importance sampling is when the proposal distribution is equal to the normalised target distribution $\rho = \pi$. If it is possible to draw samples from π, then using importance sampling is overkill. However, the next section will introduce sequential importance sampling, which samples from one dimension at a time. In that setting, it might be possible to draw from the target distribution for some dimensions, but not for others. In the simple importance sampling case with $\rho = \pi$, the weight function in (A.41) always yields the normalisation constant:

$$w(\mathbf{u}) = \frac{\gamma(\mathbf{u})}{\rho(\mathbf{u})} = Z. \tag{A.45}$$

Substituting this in (A.43a), the approximation $\tilde{\pi}$ of the normalised target distribution π becomes

$$\tilde{\pi} = \frac{1}{L} \sum_{l} \frac{w(\mathbf{u}^{(l)})}{\tilde{Z}} \delta_{\mathbf{u}^{(l)}} = \frac{1}{L} \sum_{l} \delta_{\mathbf{u}^{(l)}}, \tag{A.46}$$

which is exactly the standard Monte Carlo empirical distribution in (A.36).

A.4.3 *Sequential importance sampling*

Sequential importance sampling is an instance of importance sampling that explicitly handles a multi-dimensional sample space. It steps through the dimensions one by one, keeping track of L samples, and extending them with a new dimension at every step. Considering this explicitly is useful because then the set of samples can be adjusted between dimensions. Section A.4.4 will discuss resampling, which drops low-probability samples and multiplies high-probability samples, so that computational effort is focussed on high-probability regions.

Sequential importance sampling is a generalisation of the well-known particle filtering algorithm. It samples from a multi-dimensional distribution dimension by dimension, applying principles similar to those of importance sampling at every step. The distribution must be factored into dimensions. In particle filtering, the dimensions are often time steps, and the factor for each dimension a distribution conditional on previous dimensions. However, for sequential importance sampling the per-dimension target distributions in sequential importance sampling need not be normalised or relate to valid probability distributions, as long as their product is equal to the target distribution.

In section A.4.1 on page 269, the sample space was implicitly multi-dimensional. In this section, the the dimensions of the samples will be explicitly written. The space has d dimensions. Thus, $\mathbf{u} \triangleq \mathbf{u}_{1:d}$. The distributions γ, π, and ρ will be factorised, as will Z and w.

To apply sequential importance sampling, it must be possible to factorise the target

density γ into factors $\gamma_i(\cdot|\cdot)$ for each dimension. $\gamma_i(\cdot|\cdot)$ is therefore defined as

$$\gamma_i(u_i|\mathbf{u}_{1:i-1}) = \frac{\gamma_i(\mathbf{u}_{1:i})}{\gamma_{i-1}(\mathbf{u}_{1:i-1})}. \tag{A.47a}$$

If for every i, $\gamma_i(\mathbf{u}_{1:i})$ is a marginal distribution of $\mathbf{u}_{1:i}$, then (A.47a) is an instantiation of Bayes' rule and $\gamma_i(u_i|\mathbf{u}_{1:i-1})$ is a conditional distribution. However, even though the notation used is $(\cdot|\cdot)$, there is no requirement for the factors to be conditionals or to be normalised. This is a generalisation of particle filtering, and indeed, a strength of sequential importance sampling.

The target density γ can be written as the product of factors γ_i. (A.47b) formulates the factorisation of γ recursively; (A.47c) writes out the recursion:

$$\gamma(\mathbf{u}) = \gamma_d(\mathbf{u}_{1:d}) = \gamma_{d-1}(\mathbf{u}_{1:d-1})\gamma_d(u_d|\mathbf{u}_{1:d-1}) \tag{A.47b}$$

$$= \gamma_1(u_1)\prod_{i=2}^{d}\gamma_i(u_i|\mathbf{u}_{1:i-1})\,. \tag{A.47c}$$

The normalised variant of γ_i will be called π_i and defined analogous to π in the previous section:

$$\pi_i(\mathbf{u}_{1:i}) = \frac{\gamma_i(\mathbf{u}_{1:i})}{Z_i}; \tag{A.48a}$$

$$Z_i = \int \gamma_i(\mathbf{u}_{1:i})\mathrm{d}\mathbf{u}_{1:i}. \tag{A.48b}$$

The proposal distribution ρ is factorised similarly to the target distribution:

$$\rho_i(u_i|\mathbf{u}_{1:i-1}) = \frac{\rho_i(\mathbf{u}_{1:i})}{\rho_{i-1}(\mathbf{u}_{1:i-1})}; \tag{A.49a}$$

$$\rho(\mathbf{u}) = \rho_d(\mathbf{u}_{1:d}) = \rho_{d-1}(\mathbf{u}_{1:d-1})\rho_d(u_d|\mathbf{u}_{1:d-1})$$

$$= \rho_1(u_1)\prod_{i=2}^{d}\rho_i(u_i|\mathbf{u}_{1:i-1})\,. \tag{A.49b}$$

This makes it possible to draw samples $\mathbf{u}_{1:d}^{(l)}$ dimension per dimension. For the first dimension, $u_1^{(l)} \sim \rho_1$. Then, $u_i^{(l)}|\mathbf{u}_{1:i-1}^{(l)} \sim \rho_i$ for dimensions $i = 2, \ldots, d$. Each proposal factor ρ_i approximates target factor γ_i.

Computing the importance weight of a sample can also be done dimension per dimension. Decomposing the weight function in (A.41) recursively, in factors $w_i(\cdot|\cdot)$

related to $\gamma_i(\cdot|\cdot)$ and $\rho_i(\cdot|\cdot)$:

$$w_i(u_i|\mathbf{u}_{1:i-1}) = \frac{w_i(\mathbf{u}_{1:i})}{w_{i-1}(\mathbf{u}_{1:i-1})} = \frac{\gamma_i(\mathbf{u}_{1:i})\rho_{i-1}(\mathbf{u}_{1:i-1})}{\rho_i(\mathbf{u}_{1:i})\gamma_{i-1}(\mathbf{u}_{1:i-1})} = \frac{\gamma_i(u_i|\mathbf{u}_{1:i-1})}{\rho_i(u_i|\mathbf{u}_{1:i-1})}; \tag{A.50a}$$

$$w(\mathbf{u}) = w_d(\mathbf{u}_{1:d}) = w_{d-1}(\mathbf{u}_{1:d-1})w_d(u_d|\mathbf{u}_{1:d-1}) = w_1(u_1)\prod_{i=2}^{d} w_i(u_i|\mathbf{u}_{1:i-1})\,. \tag{A.50b}$$

The empirical distribution of ρ_i then is found from L samples drawn from ρ_i, analogously to the approximation of $\tilde{\rho}$ in (A.38):

$$\tilde{\rho}_i = \frac{1}{L}\sum_{l} \delta_{\mathbf{u}_{1:i}^{(l)}}. \tag{A.51}$$

Using this empirical distribution, the empirical normalisation constant and the normalised weights are

$$\tilde{Z}_i = \int w_i(\mathbf{u}_{1:i})\tilde{\rho}_i(\mathbf{u}_{1:i})\mathrm{d}\mathbf{u}_{1:i} = \frac{1}{L}\sum_{l} w_i\big(\mathbf{u}_{1:i}^{(l)}\big); \tag{A.52a}$$

$$\overline{w}_i^{(l)} = \frac{w_i\big(\mathbf{u}_{1:i}^{(l)}\big)}{\sum_{l'} w_i\big(\mathbf{u}_{1:i}^{(l')}\big)}. \tag{A.52b}$$

The empirical distribution derived from π_i then is, analogously to (A.43a),

$$\tilde{\pi}_i = \sum_{l} \overline{w}_i^{(l)} \delta_{\mathbf{u}_{1:i}^{(l)}}. \tag{A.53}$$

The complete algorithm for sequential importance sampling is described in Algorithm 5.

The final normalisation constant could be approximated as the average of the weights in the last step, using (A.43b). However, in section A.4.4 resampling will be introduced. This at every step removes some samples and duplicates others, and overall weights will not be available. To overcome this problem, the normalising constant Z of γ, defined in (A.39b), can be factorised into terms Z_i/Z_{i-1}, which can be approximated at every step:

$$Z \triangleq Z_d = Z_{d-1}\frac{Z_d}{Z_{d-1}} = Z_1\prod_{i=2}^{d}\frac{Z_i}{Z_{i-1}}. \tag{A.54}$$

procedure SEQUENTIAL-IMPORTANCE-SAMPLING(γ, ρ)
  for dimension $i = 1 \dots d$ **do**
    for sample index $l = 1 \dots L$ **do**
      Sample $u_i^{(l)} \sim \rho_i\big(u_i \big| \mathbf{u}_{1:i-1}^{(l)}\big)$;
      Compute weight $w_i\big(\mathbf{u}_{1:i}^{(l)}\big) = w_{i-1}\big(\mathbf{u}_{1:i-1}^{(l)}\big) \frac{\gamma_i(u_i^{(l)} | \mathbf{u}_{1:i-1}^{(l)})}{\rho_i(u_i^{(l)} | \mathbf{u}_{1:i-1}^{(l)})}$.
  return weighted samples $\Big\{ w_d\big(\mathbf{u}_{1:i}^{(l)}\big), \mathbf{u}_{1:d}^{(l)} \Big\}$.

Algorithm 5 *Sequential importance sampling*

The fraction Z_i / Z_{i-1} can be written in terms of the normalised density for dimension $i-1$ and the proposal distribution and the weight function for dimension i (applying (A.48a), (A.50a), and (A.49a)) as

$$\begin{aligned}
\frac{Z_i}{Z_{i-1}} &= \frac{\int \gamma_i(\mathbf{u}_{1:i}) \mathrm{d}\mathbf{u}_{1:i}}{\int \gamma_{i-1}(\mathbf{u}_{1:i-1}) \mathrm{d}\mathbf{u}_{1:i-1}} = \frac{\int \gamma_{i-1}(\mathbf{u}_{1:i-1}) \gamma_i(u_i | \mathbf{u}_{1:i-1}) \mathrm{d}\mathbf{u}_{1:i}}{\int \gamma_{i-1}(\mathbf{u}_{1:i-1}) \mathrm{d}\mathbf{u}_{1:i-1}} \\
&= \int \pi_{i-1}(\mathbf{u}_{1:i-1}) w_i(u_i | \mathbf{u}_{1:i-1}) \, \rho_i(u_i | \mathbf{u}_{1:i-1}) \mathrm{d}\mathbf{u}_{1:i} \\
&= \int \frac{\pi_{i-1}(\mathbf{u}_{1:i-1})}{\rho_{i-1}(\mathbf{u}_{1:i-1})} w_i(u_i | \mathbf{u}_{1:i-1}) \, \rho_i(\mathbf{u}_{1:i}) \mathrm{d}\mathbf{u}_{1:i}. \qquad \text{(A.55)}
\end{aligned}$$

This can then be approximated at every step i using the empirical distribution $\tilde{\rho}_i$ from (A.51):

$$\frac{\widetilde{Z_i}}{Z_{i-1}} = \frac{1}{L} \sum_{l=1}^{L} \overline{w}_{1:i-1}^{(l)} w_i\big(u_i^{(l)} \big| \mathbf{u}_{1:i-1}^{(l)}\big). \qquad \text{(A.56)}$$

It is straightforward to see that this yields a consistent estimate of Z_i / Z_{i-1}. Note that as long as the samples are not resampled, this computation yields the exact same $\tilde{Z}_d$ as (A.43b).

A.4.4 *Resampling*

A problem with importance sampling is that some samples will be in low-probability regions. As the number of dimensions grows, the number of high-probability samples tends to shrink exponentially. As a measure for this problem, the variance of the sample weights is often used. Sequential importance sampling as presented so far does not do anything to produce lower variances.

A technique that does help to focus on higher-probability regions, and therefore does produce lower-variance weights is *resampling*. Resampling can be applied at every step to find a new empirical measure with unweighted samples from a set of weighted samples.

The unweighted samples can be written as a set of weights and samples: $\{\overline{w}_i^{(l)}, \mathbf{u}_{1:i}^{(l)}\}$. The empirical distribution for dimensions $1 \dots i$ was given in (A.53):

$$\tilde{\pi}_i = \sum_{l=1}^{L} \overline{w}_i^{(l)} \delta_{\mathbf{u}_{1:i}^{(l)}}. \tag{A.57}$$

Resampling aims to find an approximation to this distribution with unweighted samples. The conceptually simplest way of doing this is to draw L samples from $\tilde{\pi}_i$ and construct a new empirical distribution

$$\hat{\pi}_i = \sum_{l=1}^{L} \frac{N_i^{(l)}}{L} \delta_{\mathbf{u}_{1:i}^{(l)}}, \tag{A.58}$$

where $N_i^{(l)}$ is the number of times sample $\mathbf{u}_{1:i}^{(l)}$ was drawn from $\tilde{\pi}_i$, the (integer) number of offspring of sample $\mathbf{u}_{1:i}^{(l)}$. This is called *multinomial resampling*. However, the only requirement to a resampling method is that the expected value of the number of offspring of a sample is proportional to its weight: $\mathcal{E}\{N_i^{(l)}\} = L\overline{w}_i^{(l)}$.

Another way of generating $N_{1:i}^{(l)}$ uses *systematic resampling* (Kitagawa 1996). This uses uniformly distributed $z \sim \text{Unif}\left[0, \frac{1}{L}\right]$. New sample l' then is set equal to original sample l where $\sum_{j=1}^{l-1} \overline{w}_i^{(j)} \leq z + l' < \sum_{j=1}^{l} \overline{w}_i^{(j)}$.

The new empirical distribution can also be described as a list of unweighted samples $\{\frac{1}{L}, \hat{\mathbf{u}}_{1:i}^{(l)}\}$ that contains $N_i^{(l)}$ copies of original sample $\mathbf{u}_{1:i}^{(l)}$. The distribution then is

$$\hat{\pi}_i = \frac{1}{L} \sum_{l=1}^{L} \delta_{\hat{\mathbf{u}}_{1:i}^{(l)}}. \tag{A.59}$$

This makes it straightforward to introduce resampling at every step of the sequential importance sampling algorithm as described in section A.4.3. After drawing samples and computing their weights, the set of samples is resampled to yield equally weighted

procedure SEQUENTIAL-IMPORTANCE-RESAMPLING(γ, ρ)
  for dimension $i = 1 \dots d$ **do**
    for sample index $l = 1 \dots L$ **do**
      Sample $u_i^{(l)} \sim \rho_i\left(u_i \middle| \hat{\mathbf{u}}_{1:i-1}^{(l)}\right)$;
      Compute incremental weight $w_i\left(u_i^{(l)} \middle| \mathbf{u}_{1:i-1}^{(l)}\right) = \frac{\gamma_i(u_i^{(l)}|\mathbf{u}_{1:i-1}^{(l)})}{\rho_i(u_i^{(l)}|\mathbf{u}_{1:i-1}^{(l)})}$.
    Compute $\widetilde{\frac{Z_i}{Z_{i-1}}} = \frac{1}{L}\sum_l w_i\left(u_i^{(l)} \middle| \mathbf{u}_{1:i-1}^{(l)}\right)$.
    $\left\{\hat{\mathbf{u}}_{1:i}^{(l)}\right\} \leftarrow$ RESAMPLE$\left(\left\{w_i\left(u_i^{(l)} \middle| \mathbf{u}_{1:i-1}^{(l)}\right), \mathbf{u}_{1:i}^{(l)}\right\}\right)$.
  Compute $\tilde{Z} = \tilde{Z}_1 \sum_{i=2}^{d} \widetilde{\frac{Z_i}{Z_{i-1}}}$.
  return $\left(\left\{\hat{\mathbf{u}}_{1:d}^{(l)}\right\}, \tilde{Z}\right)$.

Algorithm 6 *Sequential importance resampling*

samples $\left\{\frac{1}{L}, \hat{\mathbf{u}}_{1:i}^{(l)}\right\}$. This set is then used when drawing samples for the next iteration. The complete algorithm for sequential importance resampling is described in Algorithm 6.

A.4.5 *Sampling from the target distribution*

An extension of sequential importance resampling was foreshadowed in (A.46). It concerns the case where for some dimensions it is possible to sample from the normalised target distribution $\pi_i(\mathbf{u}_i|\mathbf{u}_{1:i-1})$. For such a dimension i, the proposal distribution can be set to $\rho_i(\mathbf{u}_i|\mathbf{u}_{1:i-1}) = \pi_i(\mathbf{u}_i|\mathbf{u}_{1:i-1})$. The incremental weight function from (A.50a) then returns the same value, the incremental normalisation constant, for each point:

$$w_i(u_i|\mathbf{u}_{1:i-1}) = \frac{\gamma_i(u_i|\mathbf{u}_{1:i-1})}{\rho_i(u_i|\mathbf{u}_{1:i-1})} = \frac{Z_i}{Z_{i-1}}. \tag{A.60}$$

This is useful because it removes the need to compute the density of the distribution at any point, or to resample the set of samples. In the case where $\gamma_i = \pi_i = \rho_i$, $w_i(u_i|\mathbf{u}_{1:i-1}) = 1$ by definition.

Algorithm 7 on the next page specifies how sequential importance resampling, in algorithm 6, can be extended to deal with dimensions for which it is possible to sample

procedure HYBRID-SEQUENTIAL-IMPORTANCE-RESAMPLING(γ, ρ)
  for dimension $i = 1 \dots d$ **do**
    if $\rho_i\left(u_i \middle| \hat{\mathbf{u}}_{1:i-1}^{(l)}\right) \propto \gamma_i\left(u_i \middle| \hat{\mathbf{u}}_{1:i-1}^{(l)}\right)$ **then**
      for sample index $l = 1 \dots L$ **do**
        Sample $\hat{u}_i^{(l)} \sim \rho_i\left(u_i \middle| \hat{\mathbf{u}}_{1:i-1}^{(l)}\right)$;
      Set $\widetilde{\frac{Z_i}{Z_{i-1}}} = w_i\left(u_i \middle| \mathbf{u}_{1:i-1}\right)$ (for any $\mathbf{u}_{1:i-1}$).
    else
      for sample index $l = 1 \dots L$ **do**
        Sample $u_i^{(l)} \sim \rho_i\left(u_i \middle| \hat{\mathbf{u}}_{1:i-1}^{(l)}\right)$;
        Compute incremental weight $w_i\left(u_i^{(l)} \middle| \mathbf{u}_{1:i-1}^{(l)}\right) = \frac{\gamma_i(u_i^{(l)} | \mathbf{u}_{1:i-1}^{(l)})}{\rho_i(u_i^{(l)} | \mathbf{u}_{1:i-1}^{(l)})}$.
      Compute $\widetilde{\frac{Z_i}{Z_{i-1}}} = \frac{1}{L} \sum_l w_i\left(u_i^{(l)} \middle| \mathbf{u}_{1:i-1}^{(l)}\right)$.
      $\left\{\hat{\mathbf{u}}_{1:i}^{(l)}\right\} \leftarrow$ RESAMPLE$\left(\left\{w_i\left(\mathbf{u}_{1:i}^{(l)}\right)\right\}\right)$.
  Compute $\tilde{Z} = \tilde{Z}_1 \sum_{i=2}^{d} \widetilde{\frac{Z_i}{Z_{i-1}}}$.
  return $\left(\left\{\hat{\mathbf{u}}_{1:d}^{(l)}\right\}, \tilde{Z}\right)$.

Algorithm 7 *Hybrid sequential importance resampling*

from the target distribution, but its density at a point cannot be computed.

Appendix B

Derivation of linear transformations

Section 3.2 has discussed linear transformations for adaptation. Section 6.2 has introduced versions of the same transformations trained on predicted statistics. Their derivations run parallel. The following sections will highlight this by deriving the statistics for both adaptive and predictive transformations of each form side by side. They will discuss, in order, CMLLR, covariance MLLR, and semi-tied covariance matrices, all with their predictive variants.

B.1 CMLLR

The likelihood for CMLLR is (repeated from (3.4b))

$$q^{(m)}(\mathbf{y}|\mathcal{A}) = |\mathbf{A}| \cdot \mathcal{N}\big(\mathbf{A}\mathbf{y} + \mathbf{b};\, \boldsymbol{\mu}_x^{(m)},\, \boldsymbol{\Sigma}_x^{(m)}\big). \tag{B.1}$$

To express the optimisation, it is convenient to write the affine transformation of the observation vectors with one matrix $\boldsymbol{W}$ by appending a 1 to the observation to form

vector ζ:

$$\mathbf{A}\mathbf{y} + \mathbf{b} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix} \triangleq \mathbf{W}\zeta. \tag{B.2}$$

B.1.1 *Adaptive*

The optimisation procedure implements an iterative approximation the maximisation in (3.2). This requires the log-likelihood, which is the logarithm of 3.4b, and its derivative with respect to the transform:

$$\begin{aligned} \log q^{(m)}(\mathbf{y}|\mathcal{A}) &= \log|\mathbf{A}| - \tfrac{1}{2}\log\left|2\pi\boldsymbol{\Sigma}_x^{(m)}\right| \\ &\quad - \tfrac{1}{2}\left(\mathbf{W}\zeta - \boldsymbol{\mu}_x^{(m)}\right)^{\mathsf{T}} \boldsymbol{\Sigma}_x^{(m)^{-1}} \left(\mathbf{W}\zeta - \boldsymbol{\mu}_x^{(m)}\right); \end{aligned} \tag{B.3a}$$

$$\frac{\partial \log q^{(m)}(\mathbf{y}|\mathcal{A})}{\partial \mathbf{W}} = \begin{bmatrix} \mathbf{A}^{-\mathsf{T}} & 0 \end{bmatrix} + \boldsymbol{\Sigma}_x^{(m)^{-1}} \left(\boldsymbol{\mu}_x^{(m)} - \mathbf{W}\zeta\right)\zeta^{\mathsf{T}}. \tag{B.3b}$$

The transformation will be optimised row by row. The derivative of row i of $\mathbf{W}$ is (assuming the covariance matrix is diagonal)

$$\frac{\partial \log q^{(m)}(\mathbf{y}|\mathcal{A})}{\partial \boldsymbol{w}_i} = \begin{bmatrix} \left[\mathbf{A}^{-\mathsf{T}}\right]_i & 0 \end{bmatrix} + \frac{1}{\sigma_{x,ii}^{(m)}} \left(\mu_{x,i}^{(m)} - \boldsymbol{w}_i\zeta\right)\zeta^{\mathsf{T}}, \tag{B.3c}$$

where $\left[\mathbf{A}^{-\mathsf{T}}\right]_i$ is the ith row of the transposed inverse of $\mathbf{A}$.

This expression can be substituted in the maximisation step in (3.2):

$$\begin{aligned} &\frac{\partial \int \tilde{p}(\mathcal{Y}) \sum_m \sum_{t=1}^{T_y} \gamma_t^{(m)} \log q^{(m)}(\mathbf{y}_t|\mathcal{A}) \, d\mathcal{Y}}{\partial \boldsymbol{w}_i} \\ &= \int \tilde{p}(\mathcal{Y}) \sum_m \sum_{t=1}^{T_y} \gamma_t^{(m)} \frac{\partial \log q^{(m)}(\mathbf{y}_t|\mathcal{A})}{\partial \boldsymbol{w}_i} d\mathcal{Y} \\ &= \int \tilde{p}(\mathcal{Y}) \sum_m \sum_{t=1}^{T_y} \gamma_t^{(m)} \left(\begin{bmatrix} \left[\mathbf{A}^{-\mathsf{T}}\right]_i & 0 \end{bmatrix} + \frac{1}{\sigma_{x,ii}^{(m)}} \left(\mu_{x,i}^{(m)} - \boldsymbol{w}_i\zeta_t\right)\zeta_t^{\mathsf{T}} \right) d\mathcal{Y} \\ &= \gamma \begin{bmatrix} \left[\mathbf{A}^{-\mathsf{T}}\right]_i & 0 \end{bmatrix} + \mathbf{k}^{(i)} - \boldsymbol{w}_i \mathbf{G}^{(i)}, \end{aligned} \tag{B.4}$$

where γ, $\mathbf{k}^{(i)}$, and $\mathbf{G}^{(i)}$ are statistics from the adaptation data:

$$\gamma \triangleq \int \tilde{p}(\mathcal{Y}) \sum_m \sum_{t=1}^{T_y} \gamma_t^{(m)} d\mathcal{Y}; \tag{B.5a}$$

$$\mathbf{k}^{(i)} \triangleq \int \tilde{p}(\mathcal{Y}) \sum_m \frac{\mu_{x,i}^{(m)}}{\sigma_{x,ii}^{(m)}} \sum_{t=1}^{T_{\mathcal{Y}}} \gamma_t^{(m)} \zeta_t^{\mathsf{T}} \mathrm{d}\mathcal{Y}; \tag{B.5b}$$

$$\mathbf{G}^{(i)} \triangleq \int \tilde{p}(\mathcal{Y}) \sum_m \frac{1}{\sigma_{x,ii}^{(m)}} \sum_{t=1}^{T_{\mathcal{Y}}} \gamma_t^{(m)} \zeta_t \zeta_t^{\mathsf{T}} \mathrm{d}\mathcal{Y}. \tag{B.5c}$$

To maximise the log-likelihood, (B.4) should be set to zero for all rows of $\boldsymbol{W}$ at once. This is not in general possible, so the algorithm has to resort to updating the rows of $\boldsymbol{W}$ iteratively.

Define $\mathbf{P}$ to be the cofactor matrix of $\mathbf{A}$ with an extra column 0 appended. The first term in (B.4) can be written in terms of $\boldsymbol{w}_i$ and row i of $\mathbf{P}$, $\mathbf{p}_i$:

$$\left[\left[\mathbf{A}^{-\mathsf{T}} \right]_i \, 0 \right] = \boldsymbol{w}_i \cdot |\mathbf{A}|^{-1} = \boldsymbol{w}_i \cdot \left(\mathbf{p}_i \boldsymbol{w}_i^{\mathsf{T}} \right)^{-1} \tag{B.6}$$

The row update is given in Gales (1998*a*):

$$\boldsymbol{w}_i := \left(\eta \mathbf{p}_i + \mathbf{k}^{(i)} \right) \mathbf{G}^{(i)^{-1}}, \tag{B.7a}$$

where η is a solution of the quadratic expression

$$\eta^2 \mathbf{p}_i \mathbf{G}^{(i)^{-1}} \mathbf{p}_i^{\mathsf{T}} + \eta \mathbf{p}_i \mathbf{G}^{(i)^{-1}} \mathbf{k}^{(i)^{\mathsf{T}}} - \gamma = 0. \tag{B.7b}$$

This row update is applied iteratively. It optimises the likelihood given the current settings of the other rows, i.e. given the current setting of $\mathbf{p}_i$. Therefore, the likelihood is guaranteed not to decrease, and the overall process is an instantiation of generalised expectation–maximisation.

B.1.2 *Predictive*

As for adaptive CMLLR, the optimisation is performed row by row. The derivative of the log-likelihood for the output distribution is the same as in (B.3c). This can be substituted into the derivative of the maximand for general predictive linear transformations from (6.18):

$$\frac{\partial \sum_m \gamma^{(m)} \int p^{(m)}(\mathbf{y}) \log q^{(m)}(\mathbf{y}|\mathcal{A}) \mathrm{d}\mathbf{y}}{\partial \boldsymbol{w}_i} \tag{B.8}$$

$$= \sum_m \gamma^{(m)} \int p^{(m)}(\mathbf{y}) \frac{\partial \log q^{(m)}(\mathbf{y}|\mathcal{A})}{\partial \boldsymbol{w}_i} d\mathbf{y} \tag{B.9}$$

$$= \sum_m \gamma^{(m)} \int p^{(m)}(\mathbf{y}) \left(\left[\left[\boldsymbol{A}^{-T}\right]_i \; 0 \right] + \frac{1}{\sigma^{(m)}_{x,ii}} \left(\mu^{(m)}_{x,i} - \boldsymbol{w}_i \zeta\right) \zeta^T \right) d\mathbf{y} \tag{B.10}$$

$$= \sum_m \gamma^{(m)} \left[\left[\boldsymbol{A}^{-T}\right]_i \; 0 \right] + \sum_m \frac{\gamma^{(m)} \mu^{(m)}_{x,i}}{\sigma^{(m)}_{x,ii}} \int p^{(m)}(\mathbf{y}) \zeta^T d\mathbf{y}$$
$$- \sum_m \frac{\gamma^{(m)}}{\sigma^{(m)}_{x,ii}} \int p^{(m)}(\mathbf{y}) \boldsymbol{w}_i \zeta \zeta^T d\mathbf{y} \tag{B.11}$$

$$= \gamma \left[\left[\boldsymbol{A}^{-T}\right]_i \; 0 \right] + \mathbf{k}^{(i)} - \boldsymbol{w}_i \mathbf{G}^{(i)}, \tag{B.12}$$

where γ, $\mathbf{k}^{(i)}$, and $\mathbf{G}^{(i)}$ are predicted statistics:

$$\gamma \triangleq \sum_m \gamma^{(m)}; \tag{B.13a}$$

$$\mathbf{k}^{(i)} \triangleq \sum_m \frac{\gamma^{(m)} \mu^{(m)}_{x,i}}{\sigma^{(m)}_{x,ii}} \int p^{(m)}(\mathbf{y}) \zeta^T d\mathbf{y} = \sum_m \frac{\gamma^{(m)} \mu^{(m)}_{x,i}}{\sigma^{(m)}_{x,ii}} \mathcal{E}_{p^{(m)}} \left\{ \zeta^T \right\}; \tag{B.13b}$$

$$\mathbf{G}^{(i)} \triangleq \sum_m \frac{\gamma^{(m)}}{\sigma^{(m)}_{x,ii}} \int p^{(m)}(\mathbf{y}) \zeta \zeta^T d\mathbf{y} = \sum_m \frac{\gamma^{(m)}}{\sigma^{(m)}_{x,ii}} \mathcal{E}_{p^{(m)}} \left\{ \zeta \zeta^T \right\}, \tag{B.13c}$$

where $\mathcal{E}_{p^{(m)}}\{\cdot\}$ is the expectation under the predicted distribution $p^{(m)}$ for component m.

B.2 Covariance MLLR

The likelihood expression for covariance MLLR is (repeated from (3.6b))

$$q^{(m)}(\mathbf{y}|\mathcal{A}) = |\boldsymbol{A}| \cdot \mathcal{N}\left(\boldsymbol{A}\mathbf{y}; \boldsymbol{A}\boldsymbol{\mu}^{(m)}_x, \boldsymbol{\Sigma}^{(m)}_x\right). \tag{B.14}$$

B.2.1 *Adaptive*

The optimisation of the transformation requires the log-likelihood and its derivative with respect to $\mathbf{A}$:

$$\begin{aligned}\log \mathrm{q}^{(m)}(\mathbf{y}|\mathcal{A}) &= \log|\mathbf{A}| - \tfrac{1}{2}\log\left|2\pi\boldsymbol{\Sigma}_x^{(m)}\right| \\ &\quad - \tfrac{1}{2}\Big(\mathbf{A}\big(\mathbf{y}-\boldsymbol{\mu}_x^{(m)}\big)\Big)^{\mathsf{T}}\boldsymbol{\Sigma}_x^{(m)^{-1}}\Big(\mathbf{A}\big(\mathbf{y}-\boldsymbol{\mu}_x^{(m)}\big)\Big); \qquad \text{(B.15a)}\\ \frac{\partial \log \mathrm{q}^{(m)}(\mathbf{y}|\mathcal{A})}{\partial \mathbf{A}} &= \mathbf{A}^{-\mathsf{T}} - \boldsymbol{\Sigma}_x^{(m)^{-1}}\mathbf{A}\big(\mathbf{y}-\boldsymbol{\mu}_x^{(m)}\big)\big(\mathbf{y}-\boldsymbol{\mu}_x^{(m)}\big)^{\mathsf{T}}. \qquad \text{(B.15b)}\end{aligned}$$

The transformation will be optimised row by row. The derivative of row i of $\mathbf{A}$ is (assuming the covariance matrix $\boldsymbol{\Sigma}_x^{(m)}$ is diagonal)

$$\frac{\partial \log \mathrm{q}^{(m)}(\mathbf{y}|\mathcal{A})}{\partial \mathbf{a}_i} = \Big[\mathbf{A}^{-\mathsf{T}}\Big]_i - \frac{1}{\sigma_{x,ii}^{(m)}}\mathbf{a}_i\big(\mathbf{y}-\boldsymbol{\mu}_x^{(m)}\big)\big(\mathbf{y}-\boldsymbol{\mu}_x^{(m)}\big)^{\mathsf{T}}. \qquad \text{(B.15c)}$$

This expression can be substituted into the maximand in (3.2):

$$\begin{aligned}&\frac{\partial \int \tilde{\mathrm{p}}(\mathcal{Y})\sum_m \sum_{t=1}^{T_{\mathcal{Y}}} \gamma_t^{(m)} \log \mathrm{q}^{(m)}(\mathbf{y}_t|\mathcal{A})\,\mathrm{d}\mathcal{Y}}{\partial \mathbf{a}_i}\\ &= \int \tilde{\mathrm{p}}(\mathcal{Y}) \sum_m \sum_{t=1}^{T_{\mathcal{Y}}} \gamma_t^{(m)} \frac{\partial \log \mathrm{q}^{(m)}(\mathbf{y}_t|\mathcal{A})}{\partial \mathbf{a}_i}\mathrm{d}\mathcal{Y}\\ &= \int \tilde{\mathrm{p}}(\mathcal{Y}) \sum_m \sum_{t=1}^{T_{\mathcal{Y}}} \gamma_t^{(m)} \left(\Big[\mathbf{A}^{-\mathsf{T}}\Big]_i - \frac{1}{\sigma_{x,ii}^{(m)}}\mathbf{a}_i\big(\mathbf{y}_t-\boldsymbol{\mu}_x^{(m)}\big)\big(\mathbf{y}_t-\boldsymbol{\mu}_x^{(m)}\big)^{\mathsf{T}}\right)\mathrm{d}\mathcal{Y}\\ &= \gamma\Big[\mathbf{A}^{-\mathsf{T}}\Big]_i - \mathbf{a}_i\mathbf{G}^{(i)}, \qquad \text{(B.16)}\end{aligned}$$

where γ and $\mathbf{G}^{(i)}$ are statistics from the adaptation data:

$$\gamma \triangleq \int \tilde{\mathrm{p}}(\mathcal{Y}) \sum_m \sum_{t=1}^{T_{\mathcal{Y}}} \gamma_t^{(m)}\mathrm{d}\mathcal{Y}; \qquad \text{(B.17a)}$$

$$\mathbf{G}^{(i)} \triangleq \int \tilde{\mathrm{p}}(\mathcal{Y}) \sum_m \frac{1}{\sigma_{x,ii}^{(m)}} \sum_{t=1}^{T_{\mathcal{Y}}} \gamma_t^{(m)}\big(\mathbf{y}_t-\boldsymbol{\mu}_x^{(m)}\big)\big(\mathbf{y}_t-\boldsymbol{\mu}_x^{(m)}\big)^{\mathsf{T}}\mathrm{d}\mathcal{Y}. \qquad \text{(B.17b)}$$

For covariance MLLR, γ is the same as for CMLLR (in (3.5a)); $\mathbf{G}^{(i)}$ is similar but uses $\mathbf{y}_t - \boldsymbol{\mu}_x^{(m)}$ instead of $\mathbf{y}_t$.

To maximise the log-likelihood, (B.4) should be set to zero for all rows of $\mathbf{A}$ at once. Just like for CMLLR, this is not in general possible, so the algorithm has to resort

to updating the rows of $\mathbf{A}$ iteratively. Each row is set to optimise the likelihood giving the current settings of the other rows. The details of the algorithm are in Gales (1998*a*). The likelihood is guaranteed not to decrease, and the overall process is an instantiation of generalised expectation–maximisation, where the maximisation step is an iteration over the rows.

B.2.2 *Predictive*

The optimisation works row by row. The partial derivative of the log-likelihood was given in (B.15c). This can be substituted into the derivative of the maximand for general predictive linear transformations from (6.18):

$$\frac{\partial \sum_m \gamma^{(m)} \int p^{(m)}(\mathbf{y}) \log q^{(m)}(\mathbf{y}|\mathcal{A}) \mathrm{d}\mathbf{y}}{\partial \mathbf{a}_i} \tag{B.18}$$

$$= \sum_m \gamma^{(m)} \int p^{(m)}(\mathbf{y}) \frac{\partial \log q^{(m)}(\mathbf{y}|\mathcal{A})}{\partial \mathbf{a}_i} \mathrm{d}\mathbf{y} \tag{B.19}$$

$$= \sum_m \gamma^{(m)} \int p^{(m)}(\mathbf{y}) \left(\left[\mathbf{A}^{-\mathsf{T}} \right]_i - \frac{1}{\sigma^{(m)}_{x,ii}} \mathbf{a}_i \left(\mathbf{y} - \boldsymbol{\mu}^{(m)}_x \right) \left(\mathbf{y} - \boldsymbol{\mu}^{(m)}_x \right)^{\mathsf{T}} \right) \mathrm{d}\mathbf{y} \tag{B.20}$$

$$= \gamma \left[\mathbf{A}^{-\mathsf{T}} \right]_i - \mathbf{a}_i \mathbf{G}^{(i)}, \tag{B.21}$$

where γ and $\mathbf{G}^{(i)}$ are predicted statistics:

$$\gamma \triangleq \sum_m \gamma^{(m)}; \tag{B.22a}$$

$$\begin{aligned} \mathbf{G}^{(i)} &\triangleq \sum_m \frac{\gamma^{(m)}}{\sigma^{(m)}_{x,ii}} \int p^{(m)}(\mathbf{y}) \left(\mathbf{y} - \boldsymbol{\mu}^{(m)}_x \right) \left(\mathbf{y} - \boldsymbol{\mu}^{(m)}_x \right)^{\mathsf{T}} \mathrm{d}\mathbf{y} \\ &= \sum_m \frac{\gamma^{(m)}}{\sigma^{(m)}_{x,ii}} \mathcal{E}_{p^{(m)}} \left\{ \left(\mathbf{y} - \boldsymbol{\mu}^{(m)}_x \right) \left(\mathbf{y} - \boldsymbol{\mu}^{(m)}_x \right)^{\mathsf{T}} \right\}. \end{aligned} \tag{B.22b}$$

B.3 Semi-tied covariance matrices

The expression for the likelihood for semi-tied covariance matrices is (repeated from (3.12b))

$$q^{(m)}(\boldsymbol{x}) = |\boldsymbol{A}| \cdot \mathcal{N}\left(\boldsymbol{A}\boldsymbol{x}; \boldsymbol{A}\boldsymbol{\mu}^{(m)}_x, \widetilde{\boldsymbol{\Sigma}}^{(m)}_{x,\mathrm{diag}} \right). \tag{B.23}$$

B.3.1 *From data*

The log-likelihood expression and its derivative both with respect to the component covariances and to the rows of the transformation matrix (the optimisation again is row-wise, similar to (B.15c)) are necessary:

$$\begin{aligned}\log \mathrm{q}^{(\mathrm{m})}(\mathbf{x}) &= \log|\boldsymbol{A}| - \tfrac{1}{2}\log\left|2\pi\widetilde{\boldsymbol{\Sigma}}^{(\mathrm{m})}_{\mathrm{x,diag}}\right| \\ &\quad - \tfrac{1}{2}\Big(\boldsymbol{A}\big(\mathbf{x}-\boldsymbol{\mu}^{(\mathrm{m})}_{\mathrm{x}}\big)\Big)^{\mathsf{T}}\big[\widetilde{\boldsymbol{\Sigma}}^{(\mathrm{m})}_{\mathrm{x,diag}}\big]^{-1}\Big(\boldsymbol{A}\big(\mathbf{x}-\boldsymbol{\mu}^{(\mathrm{m})}_{\mathrm{x}}\big)\Big); \qquad \text{(B.24a)}\end{aligned}$$

$$\frac{\partial \log \mathrm{q}^{(\mathrm{m})}(\mathbf{x})}{\partial \widetilde{\boldsymbol{\Sigma}}^{(\mathrm{m})}_{\mathrm{x,diag}}} = -\tfrac{1}{2}\big[\widetilde{\boldsymbol{\Sigma}}^{(\mathrm{m})}_{\mathrm{x,diag}}\big]^{-1}\Big(\mathbf{I} - \boldsymbol{A}\big(\mathbf{x}-\boldsymbol{\mu}^{(\mathrm{m})}_{\mathrm{x}}\big)\big(\mathbf{x}-\boldsymbol{\mu}^{(\mathrm{m})}_{\mathrm{x}}\big)^{\mathsf{T}}\boldsymbol{A}^{\mathsf{T}}\big[\widetilde{\boldsymbol{\Sigma}}^{(\mathrm{m})}_{\mathrm{x,diag}}\big]^{-1}\Big); \qquad \text{(B.24b)}$$

$$\frac{\partial \log \mathrm{q}^{(\mathrm{m})}(\mathbf{x})}{\partial \boldsymbol{A}} = \boldsymbol{A}^{-\mathsf{T}} - \boldsymbol{\Sigma}^{(\mathrm{m})-1}_{\mathrm{x}}\boldsymbol{A}\big(\mathbf{x}-\boldsymbol{\mu}^{(\mathrm{m})}_{\mathrm{x}}\big)\big(\mathbf{x}-\boldsymbol{\mu}^{(\mathrm{m})}_{\mathrm{x}}\big)^{\mathsf{T}}; \qquad \text{(B.24c)}$$

$$\frac{\partial \log \mathrm{q}^{(\mathrm{m})}(\mathbf{x})}{\partial \boldsymbol{a}_{\mathrm{i}}} = \Big[\boldsymbol{A}^{-\mathsf{T}}\Big]_{\mathrm{i}} - \frac{1}{\widetilde{\sigma}^{(\mathrm{m})}_{\mathrm{x,ii}}}\boldsymbol{a}_{\mathrm{i}}\big(\mathbf{x}-\boldsymbol{\mu}^{(\mathrm{m})}_{\mathrm{x}}\big)\big(\mathbf{x}-\boldsymbol{\mu}^{(\mathrm{m})}_{\mathrm{x}}\big)^{\mathsf{T}}. \qquad \text{(B.24d)}$$

These expressions can be substituted into the maximand in (2.32). For the optimisation of $\widetilde{\boldsymbol{\Sigma}}^{(\mathrm{m})}_{\mathrm{x,diag}}$,

$$2\frac{\partial \int \tilde{\mathrm{p}}(\mathcal{X}) \sum_{\mathrm{m}} \sum_{\mathrm{t}=1}^{\mathrm{T}_{\mathcal{X}}} \gamma^{(\mathrm{m})}_{\mathrm{t}} \log \mathrm{q}^{(\mathrm{m})}(\mathbf{x}_{\mathrm{t}})\,\mathrm{d}\mathcal{X}}{\partial \widetilde{\boldsymbol{\Sigma}}^{(\mathrm{m})}_{\mathrm{x,diag}}} \qquad \text{(B.25)}$$

$$= \int \tilde{\mathrm{p}}(\mathcal{X}) \sum_{\mathrm{t}=1}^{\mathrm{T}_{\mathcal{X}}} \gamma^{(\mathrm{m})}_{\mathrm{t}} \big[\widetilde{\boldsymbol{\Sigma}}^{(\mathrm{m})}_{\mathrm{x,diag}}\big]^{-1}\Big(\boldsymbol{A}\big(\mathbf{x}_{\mathrm{t}}-\boldsymbol{\mu}^{(\mathrm{m})}_{\mathrm{x}}\big)\big(\mathbf{x}_{\mathrm{t}}-\boldsymbol{\mu}^{(\mathrm{m})}_{\mathrm{x}}\big)^{\mathsf{T}}\boldsymbol{A}^{\mathsf{T}}\big[\widetilde{\boldsymbol{\Sigma}}^{(\mathrm{m})}_{\mathrm{x,diag}}\big]^{-1} - \mathbf{I}\Big)\,\mathrm{d}\mathcal{X} \qquad \text{(B.26)}$$

$$= \gamma^{(\mathrm{m})}\big[\widetilde{\boldsymbol{\Sigma}}^{(\mathrm{m})}_{\mathrm{x,diag}}\big]^{-1}\Big(\boldsymbol{A}\boldsymbol{W}^{(\mathrm{m})}\boldsymbol{A}^{\mathsf{T}}\big[\widetilde{\boldsymbol{\Sigma}}^{(\mathrm{m})}_{\mathrm{x,diag}}\big]^{-1} - \mathbf{I}\Big). \qquad \text{(B.27)}$$

Here, $\gamma^{(\mathrm{m})}_{\mathrm{t}}$ and $\gamma^{(\mathrm{m})}$ are found from the training data as in (2.31), and $\boldsymbol{W}^{(\mathrm{m})}$ contains statistics from training observations. Only the diagonal elements of $\widetilde{\boldsymbol{\Sigma}}^{(\mathrm{m})}_{\mathrm{x,diag}}$ are estimated. However since $\boldsymbol{W}^{(\mathrm{m})}$, the empirical full covariance for one component m, is transformed by $\boldsymbol{A}$ (which changes at every iteration), it must be full:

$$\boldsymbol{W}^{(\mathrm{m})} \triangleq \frac{1}{\gamma^{(\mathrm{m})}}\int \tilde{\mathrm{p}}(\mathcal{X}) \sum_{\mathrm{t}=1}^{\mathrm{T}_{\mathcal{X}}} \gamma^{(\mathrm{m})}_{\mathrm{t}}\big(\mathbf{x}_{\mathrm{t}}-\boldsymbol{\mu}^{(\mathrm{m})}_{\mathrm{x}}\big)\big(\mathbf{x}_{\mathrm{t}}-\boldsymbol{\mu}^{(\mathrm{m})}_{\mathrm{x}}\big)^{\mathsf{T}}\mathrm{d}\mathcal{X}. \qquad \text{(B.28)}$$

The optimisation with respect to $\mathbf{A}$ is very similar to the one for covariance MLLR in (B.16):

$$\frac{\partial \int \tilde{p}(\mathcal{X}) \sum_m \sum_{t=1}^{T_\mathcal{X}} \gamma_t^{(m)} \log q^{(m)}(\mathbf{x}_t) d\mathcal{X}}{\partial \mathbf{a}_i} \tag{B.29}$$

$$= \int \tilde{p}(\mathcal{X}) \sum_m \sum_{t=1}^{T_\mathcal{X}} \gamma_t^{(m)} \frac{\partial \log q^{(m)}(\mathbf{x}_t)}{\partial \mathbf{a}_i} d\mathcal{X} \tag{B.30}$$

$$= \int \tilde{p}(\mathcal{X}) \sum_m \sum_{t=1}^{T_\mathcal{X}} \gamma_t^{(m)} \left(\left[\mathbf{A}^{-T} \right]_i - \frac{1}{\widetilde{\sigma}_{x,ii}^{(m)}} \mathbf{a}_i \left(\mathbf{x}_t - \boldsymbol{\mu}_x^{(m)} \right) \left(\mathbf{x}_t - \boldsymbol{\mu}_x^{(m)} \right)^T \right) d\mathcal{X} \tag{B.31}$$

$$= \gamma \left[\mathbf{A}^{-T} \right]_i - \mathbf{a}_i \mathbf{G}^{(i)}. \tag{B.32}$$

Estimating $\mathbf{A}$ requires two types of statistics, γ and $\mathbf{G}^{(i)}$. γ, the total occupancy, is found in a similar way as in (B.17a), but from the training data. Since the procedure for estimating the covariance transformation is the same as for covariance MLLR, the statistics are also basically the same. The difference is that for covariance MLLR, the covariance $\boldsymbol{\Sigma}_x^{(m)}$ of the components is fixed, whereas for semi-tied covariance matrices, the covariances $\widetilde{\boldsymbol{\Sigma}}_{x,\text{diag}}^{(m)}$ get re-estimated every iteration. Fixed $\sigma_{x,ii}^{(m)}$ in (B.17b) therefore is replaced by the diagonal elements of $\widetilde{\boldsymbol{\Sigma}}_{x,\text{diag}}^{(m)}$, $\widetilde{\sigma}_{x,ii}^{(m)}$. $\mathbf{G}^{(i)}$ is rewritten in terms of the part that does not change with every iteration, $\boldsymbol{W}^{(m)}$ in (B.28):

$$\gamma \triangleq \int \tilde{p}(\mathcal{X}) \sum_m \sum_{t=1}^{T_\mathcal{X}} \gamma_t^{(m)} d\mathcal{X}; \tag{B.33a}$$

$$\begin{aligned} \mathbf{G}^{(i)} &\triangleq \int \tilde{p}(\mathcal{X}) \sum_m \frac{1}{\widetilde{\sigma}_{x,ii}^{(m)}} \sum_{t=1}^{T_\mathcal{X}} \gamma_t^{(m)} \left(\mathbf{x}_t - \boldsymbol{\mu}_x^{(m)} \right) \left(\mathbf{x}_t - \boldsymbol{\mu}_x^{(m)} \right)^T d\mathcal{X} \\ &= \sum_m \frac{\gamma^{(m)}}{\widetilde{\sigma}_{x,ii}^{(m)}} \boldsymbol{W}^{(m)}. \end{aligned} \tag{B.33b}$$

B.3.2 *Predictive*

Unlike non-predictive semi-tied covariance matrices, the predictive variant defines a distribution the corrupted speech, similar to (B.23) (repeated from (6.24)):

$$q^{(m)}(\mathbf{y}) = |\mathbf{A}| \cdot \mathcal{N}\left(\mathbf{A}\mathbf{y}; \mathbf{A}\boldsymbol{\mu}_y^{(m)}, \widetilde{\boldsymbol{\Sigma}}_{y,\text{diag}}^{(m)} \right) \tag{B.34}$$

The optimisation of the means, the covariances, and the transformation need the derivatives of the log-likelihood with respect to them. They are similar to (B.24) (the derivative with respect to the means was not given there):

$$\begin{aligned}\log q^{(m)}(\mathbf{y}) &= \log|\mathbf{A}| - \tfrac{1}{2}\log\left|2\pi\widetilde{\boldsymbol{\Sigma}}^{(m)}_{y,\text{diag}}\right| \\ &\quad - \tfrac{1}{2}\Big(\mathbf{A}\big(\mathbf{y}-\boldsymbol{\mu}_y^{(m)}\big)\Big)^{\mathsf{T}}\big[\widetilde{\boldsymbol{\Sigma}}^{(m)}_{y,\text{diag}}\big]^{-1}\Big(\mathbf{A}\big(\mathbf{y}-\boldsymbol{\mu}_y^{(m)}\big)\Big); \qquad \text{(B.35a)}\end{aligned}$$

$$\frac{\partial \log q^{(m)}(\mathbf{y})}{\partial \boldsymbol{\mu}_y^{(m)}} = 2\big[\widetilde{\boldsymbol{\Sigma}}^{(m)}_{y,\text{diag}}\big]^{-1}\mathbf{A}\big(\mathbf{y}-\boldsymbol{\mu}_y^{(m)}\big); \qquad \text{(B.35b)}$$

$$\frac{\partial \log q^{(m)}(\mathbf{y})}{\partial \widetilde{\boldsymbol{\Sigma}}^{(m)}_{y,\text{diag}}} = -\tfrac{1}{2}\big[\widetilde{\boldsymbol{\Sigma}}^{(m)}_{y,\text{diag}}\big]^{-1}\Big(\mathbf{I} - \mathbf{A}\big(\mathbf{y}-\boldsymbol{\mu}_y^{(m)}\big)\big(\mathbf{y}-\boldsymbol{\mu}_y^{(m)}\big)^{\mathsf{T}}\mathbf{A}^{\mathsf{T}}\big[\widetilde{\boldsymbol{\Sigma}}^{(m)}_{y,\text{diag}}\big]^{-1}\Big); \qquad \text{(B.35c)}$$

$$\frac{\partial \log q^{(m)}(\mathbf{y})}{\partial \mathbf{a}_i} = \Big[\mathbf{A}^{-\mathsf{T}}\Big]_i - \frac{1}{\widetilde{\sigma}^{(m)}_{y,ii}}\mathbf{a}_i\big(\mathbf{y}-\boldsymbol{\mu}_y^{(m)}\big)\big(\mathbf{y}-\boldsymbol{\mu}_y^{(m)}\big)^{\mathsf{T}}. \qquad \text{(B.35d)}$$

These can be substituted into the derivative of the maximand for general predictive linear transformations from (6.18).

$$\begin{aligned}&\frac{\partial \sum_m \gamma^{(m)} \int p^{(m)}(\mathbf{y}) \log q^{(m)}(\mathbf{y}|\mathcal{A})\,d\mathbf{y}}{\partial \boldsymbol{\mu}_y^{(m)}} \\ &= 2\sum_m \gamma^{(m)}\big[\widetilde{\boldsymbol{\Sigma}}^{(m)}_{y,\text{diag}}\big]^{-1}\mathbf{A}\int p^{(m)}(\mathbf{y})\big(\mathbf{y}-\boldsymbol{\mu}_y^{(m)}\big)\,d\mathbf{y}. \qquad \text{(B.36)}\end{aligned}$$

To minimise the KL divergence with respect to the means, they are unsurprisingly set to the expected value under the predicted distribution for component m:

$$\boldsymbol{\mu}_y^{(m)} := \int p^{(m)}(\mathbf{y})\mathbf{y}\,d\mathbf{y} = \mathcal{E}_{p^{(m)}}\{\mathbf{y}\}. \qquad \text{(B.37)}$$

Since this expression does not depend on the other variables that are estimated, setting the means is a one-shot process.

The derivative with respect to the covariance of component m is

$$\begin{aligned}&2\frac{\partial \sum_m \gamma^{(m)} \int p^{(m)}(\mathbf{y}) \log q^{(m)}(\mathbf{y})\,d\mathbf{y}}{\partial \widetilde{\boldsymbol{\Sigma}}^{(m)}_{y,\text{diag}}} \\ &= \gamma^{(m)}\big[\widetilde{\boldsymbol{\Sigma}}^{(m)}_{y,\text{diag}}\big]^{-1}\int p^{(m)}(\mathbf{y})\Big(\mathbf{A}\big(\mathbf{y}-\boldsymbol{\mu}_y^{(m)}\big)\big(\mathbf{y}-\boldsymbol{\mu}_y^{(m)}\big)^{\mathsf{T}}\mathbf{A}^{\mathsf{T}}\big[\widetilde{\boldsymbol{\Sigma}}^{(m)}_{y,\text{diag}}\big]^{-1} - \mathbf{I}\Big)\,d\mathbf{y} \\ &= \gamma^{(m)}\big[\widetilde{\boldsymbol{\Sigma}}^{(m)}_{y,\text{diag}}\big]^{-1}\Big(\mathbf{A}\mathbf{W}^{(m)}\mathbf{A}^{\mathsf{T}}\big[\widetilde{\boldsymbol{\Sigma}}^{(m)}_{y,\text{diag}}\big]^{-1} - \mathbf{I}\Big), \qquad \text{(B.38)}\end{aligned}$$

where the predicted covariance in the original feature space for component m is

$$\begin{aligned}\boldsymbol{W}^{(m)} &\triangleq \int p^{(m)}(\boldsymbol{y})\left(\boldsymbol{y}-\boldsymbol{\mu}_y^{(m)}\right)\left(\boldsymbol{y}-\boldsymbol{\mu}_y^{(m)}\right)^{\mathsf{T}} d\boldsymbol{y} \\ &= \mathcal{E}_{p^{(m)}}\left\{\left(\boldsymbol{y}-\boldsymbol{\mu}_y^{(m)}\right)\left(\boldsymbol{y}-\boldsymbol{\mu}_y^{(m)}\right)^{\mathsf{T}}\right\}. \end{aligned} \tag{B.39}$$

This is the equivalent of $\boldsymbol{W}^{(m)}$ for standard semi-tied covariance matrices, but estimated on the predicted distribution rather than an empirical one. Though the component covariance $\widetilde{\boldsymbol{\Sigma}}_{y,\text{diag}}^{(m)}$ is constrained to be diagonal, the statistics $\boldsymbol{W}^{(m)}$ have to be full, because they are transformed by $\boldsymbol{A}$. By setting the bracketed expression in (B.27) to zero, the optimum setting for the diagonal elements of $\widetilde{\boldsymbol{\Sigma}}_{y,\text{diag}}^{(m)}$ is found with

$$\widetilde{\boldsymbol{\Sigma}}_{y,\text{diag}}^{(m)} := \text{diag}\left(\boldsymbol{A}\boldsymbol{W}^{(m)}\boldsymbol{A}^{\mathsf{T}}\right). \tag{B.40}$$

Unlike the expression for the means in (B.37), this expression depends on $\boldsymbol{A}$, which in turn depends on $\widetilde{\boldsymbol{\Sigma}}_{y,\text{diag}}^{(m)}$. Therefore, the procedure will be iterative.

The optimisation of $\boldsymbol{A}$ is performed row-by-row. Again, taking the derivative of the KL divergence,

$$\begin{aligned}&\frac{\partial \sum_m \gamma^{(m)} \int p^{(m)}(\boldsymbol{y}) \log q^{(m)}(\boldsymbol{y}) d\boldsymbol{y}}{\partial \boldsymbol{a}_i} \\ &= \sum_m \gamma^{(m)} \int p^{(m)}(\boldsymbol{y}) \left(\left[\boldsymbol{A}^{-\mathsf{T}}\right]_i - \frac{1}{\widetilde{\sigma}_{y,ii}^{(m)}} \boldsymbol{a}_i \left(\boldsymbol{y}-\boldsymbol{\mu}_y^{(m)}\right)\left(\boldsymbol{y}-\boldsymbol{\mu}_y^{(m)}\right)^{\mathsf{T}} \right) d\boldsymbol{y} \\ &= \gamma \left[\boldsymbol{A}^{-\mathsf{T}}\right]_i - \boldsymbol{a}_i \boldsymbol{G}^{(i)}, \end{aligned} \tag{B.41}$$

where γ and $\boldsymbol{G}^{(i)}$ are the predicted statistics:

$$\gamma \triangleq \sum_m \gamma^{(m)}; \tag{B.42a}$$

$$\begin{aligned}\boldsymbol{G}^{(i)} &\triangleq \sum_m \frac{\gamma^{(m)}}{\widetilde{\sigma}_{y,ii}^{(m)}} \int p^{(m)}(\boldsymbol{y})\left(\boldsymbol{y}-\boldsymbol{\mu}_y^{(m)}\right)\left(\boldsymbol{y}-\boldsymbol{\mu}_y^{(m)}\right)^{\mathsf{T}} d\boldsymbol{y} \\ &= \sum_m \frac{\gamma^{(m)}}{\widetilde{\sigma}_{y,ii}^{(m)}} \boldsymbol{W}^{(m)}. \end{aligned} \tag{B.42b}$$

As in standard semi-tied covariance matrices (in (B.33b)), $\boldsymbol{G}^{(i)}$ can be expressed in terms of $\boldsymbol{W}^{(m)}$, which do not change with every iteration.

Appendix C

Mismatch function

This section gives derivations relating to the mismatch function that would have confused the main text. Section C.1 gives the Jacobians with respect to the sources, which are required for VTS compensation. Section C.2 derives the mismatch function and its derivatives for different powers of the spectrum.

C.1 Jacobians

The mismatch function relates log-spectral coefficients of the observation, speech, and noise with (repeated from (4.9))

$$\exp\left(y_i\right) = \exp\left(x_i + h_i\right) + \exp\left(n_i\right) + 2\alpha_i \exp\tfrac{1}{2}\left(x_i + h_i + n_i\right). \qquad \text{(C.1)}$$

The per-coefficient derivatives of this are

$$\frac{dy_i}{dx_i} = \frac{\exp(x_i + h_i) + \alpha_i \exp\left(\frac{1}{2}(x_i + h_i + n_i)\right)}{\exp(x_i + h_i) + \exp(n_i) + 2\alpha_i \exp\left(\frac{1}{2}(x_i + h_i + n_i)\right)}; \tag{C.2a}$$

$$\frac{dy_i}{dh_i} = \frac{dy_i}{dx_i}; \tag{C.2b}$$

$$\begin{aligned}\frac{dy_i}{dn_i} &= \frac{\exp(n_i) + \alpha_i \exp\left(\frac{1}{2}(x_i + h_i + n_i)\right)}{\exp(x_i + h_i) + \exp(n_i) + 2\alpha_i \exp\left(\frac{1}{2}(x_i + h_i + n_i)\right)} \\ &= 1 - \frac{dy_i}{dx_i};\end{aligned} \tag{C.2c}$$

$$\frac{dy_i}{d\alpha_i} = \frac{\beta \exp\left(\frac{1}{2}(x_i + h_i + n_i)\right)}{\exp(x_i + h_i) + \exp(n_i) + 2\alpha_i \exp\left(\frac{1}{2}(x_i + h_i + n_i)\right)}. \tag{C.2d}$$

The Jacobians of the vector mismatch function in the log-spectral domain are

$$\mathbf{J}_x^{\log} = \frac{\partial \mathbf{y}^{\log}}{\partial \mathbf{x}^{\log}}; \qquad \mathbf{J}_n^{\log} = \frac{\partial \mathbf{y}^{\log}}{\partial \mathbf{n}^{\log}}; \qquad \mathbf{J}_h^{\log} = \frac{\partial \mathbf{y}^{\log}}{\partial \mathbf{h}^{\log}}; \qquad \mathbf{J}_\alpha^{\log} = \frac{\partial \mathbf{y}^{\log}}{\partial \boldsymbol{\alpha}}. \tag{C.3}$$

Since the mismatch function applies per dimension in the log-spectral domain, the Jacobians of the vector mismatch function in this domain are diagonal. They have as diagonal entries the derivatives given in (C.2):

$$j_{x,i}^{\log} = \frac{dy_i}{dx_i}; \qquad j_{n,i}^{\log} = \frac{dy_i}{dn_i}; \qquad j_{h,i}^{\log} = \frac{dy_i}{dh_i}; \qquad j_{\alpha,i}^{\log} = \frac{dy_i}{d\alpha_i}. \tag{C.4}$$

Cepstral features are related to log-spectral through the DCT matrix $\mathbf{C}$. The cepstral-domain can be found through the chain rule.

$$\mathbf{J}_x = \frac{\partial \mathbf{y}^s}{\partial \mathbf{x}^s} = \frac{\partial \mathbf{y}^s}{\partial \mathbf{y}^{\log}} \frac{\partial \mathbf{y}^{\log}}{\partial \mathbf{x}^{\log}} \frac{\partial \mathbf{x}^{\log}}{\partial \mathbf{x}^s} = \mathbf{C}\mathbf{J}_x^{\log}\mathbf{C}^{-1}, \tag{C.5a}$$

and analogously

$$\mathbf{J}_n = \frac{\partial \mathbf{y}^s}{\partial \mathbf{n}^s} = \mathbf{C}\mathbf{J}_n^{\log}\mathbf{C}^{-1}; \quad \mathbf{J}_h = \frac{\partial \mathbf{y}^s}{\partial \mathbf{h}^s} = \mathbf{C}\mathbf{J}_h^{\log}\mathbf{C}^{-1}; \quad \mathbf{J}_\alpha = \frac{\partial \mathbf{y}^s}{\partial \boldsymbol{\alpha}} = \mathbf{C}\mathbf{J}_\alpha^{\log}. \tag{C.5b}$$

C.2 Mismatch function for other spectral powers

The mismatch in the log-spectral domain was given in (4.9). It assumed that the features y_i, x_i, n_i used the power spectrum. This section will write the power β applied

to the spectral coefficients explicitly as $y_i^{(\beta)}, x_i^{(\beta)}, n_i^{(\beta)}$, so that (4.9) becomes:

$$\exp\left(y_i^{(2)}\right) = \exp\left(x_i^{(2)} + h_i^{(2)}\right) + \exp\left(n_i^{(2)}\right) + 2\alpha_i \exp\Big(\tfrac{1}{2}\left(x_i^{(2)} + h_i^{(2)} + n_i^{(2)}\right)\Big)\,. \tag{C.6}$$

The expression for the mismatch relating vectors in domains with different powers than 2 derives from this using an assumption about the mel-filtered spectrum. The assumption is the same that was used to approximate the convolutional noise in (4.7), namely that all spectral coefficients in one mel-bin are equal. In that case, a mel-filtered spectral coefficient is a weighted sum of spectral coefficients to the power of β (see (4.8a)), is equal to the power of the sum:

$$\bar{Y}_i^{(\beta)} = \sum_k w_{ik}|Y[k]|^\beta \simeq \Big(\sum_k w_{ik}|Y[k]|\Big)^\beta. \tag{C.7}$$

The log-spectral coefficients are found by taking the logarithm of this, so that

$$y_i^{(\beta)} = \log\left(\bar{Y}_i^{(\beta)}\right) \simeq \beta \log\Big(\sum_k w_{ik}|Y[k]|\Big). \tag{C.8}$$

This assumption can be applied to all feature vectors. It causes coefficients acquired from the βth-power domain to be assumed related to those using a power of 2 by

$$y_i^{(\beta)} = \tfrac{\beta}{2} y_i^{(2)}; \qquad x_i^{(\beta)} = \tfrac{\beta}{2} x_i^{(2)}; \qquad n_i^{(\beta)} = \tfrac{\beta}{2} n_i^{(2)}; \qquad h_i^{(\beta)} = \tfrac{\beta}{2} h_i^{(2)}. \tag{C.9}$$

For the log-magnitude-spectrum ($\beta = 1$), for example, coefficients y_i, x_i, n_i, are smaller by a factor of 2. Therefore, (C.6) can be generalised to any power β by making up for the power:

$$\begin{aligned}\exp\left(\tfrac{2}{\beta} y_i^{(\beta)}\right) = \exp\Big(\tfrac{2}{\beta}\left(x_i^{(\beta)} + h_i^{(\beta)}\right)\Big) + \exp\left(\tfrac{2}{\beta} n_i^{(\beta)}\right) \\ + 2\alpha_i \exp\Big(\tfrac{1}{\beta}\left(x_i^{(\beta)} + h_i^{(\beta)} + n_i^{(\beta)}\right)\Big)\,,\end{aligned} \tag{C.10a}$$

or

$$\begin{aligned}y_i^{(\beta)} = \tfrac{\beta}{2} \log\Bigg(\exp\Big(\tfrac{2}{\beta}\left(x_i^{(\beta)} + h_i^{(\beta)}\right)\Big) + \exp\left(\tfrac{2}{\beta} n_i^{(\beta)}\right) \\ + 2\alpha_i \exp\Big(\tfrac{1}{\beta}\left(x_i^{(\beta)} + h_i^{(\beta)} + n_i^{(\beta)}\right)\Big)\Bigg).\end{aligned} \tag{C.10b}$$

Derivatives of this are

$$\frac{dy_i^{(\beta)}}{dx_i^{(\beta)}} = \frac{\exp\left(\frac{2}{\beta}\left(x_i^{(\beta)} + h_i^{(\beta)}\right)\right) + \alpha_i \exp\left(\frac{1}{\beta}\left(x_i^{(\beta)} + h_i^{(\beta)} + n_i^{(\beta)}\right)\right)}{\exp\left(\frac{2}{\beta}\left(x_i^{(\beta)} + h_i^{(\beta)}\right)\right) + \exp\left(\frac{2}{\beta} n_i^{(\beta)}\right) + 2\alpha_i \exp\left(\frac{1}{\beta}\left(x_i^{(\beta)} + h_i^{(\beta)} + n_i^{(\beta)}\right)\right)}; \tag{C.11a}$$

$$\frac{dy_i^{(\beta)}}{dh_i^{(\beta)}} = \frac{dy_i^{(\beta)}}{dx_i^{(\beta)}}; \tag{C.11b}$$

$$\begin{aligned}\frac{dy_i^{(\beta)}}{dn_i^{(\beta)}} &= \frac{\exp\left(\frac{2}{\beta} n_i^{(\beta)}\right) + \alpha_i \exp\left(\frac{1}{\beta}\left(x_i^{(\beta)} + h_i^{(\beta)} + n_i^{(\beta)}\right)\right)}{\exp\left(\frac{2}{\beta}\left(x_i^{(\beta)} + h_i^{(\beta)}\right)\right) + \exp\left(\frac{2}{\beta} n_i^{(\beta)}\right) + 2\alpha_i \exp\left(\frac{1}{\beta}\left(x_i^{(\beta)} + h_i^{(\beta)} + n_i^{(\beta)}\right)\right)} \\ &= 1 - \frac{dy_i^{(\beta)}}{dx_i^{(\beta)}}; \end{aligned} \tag{C.11c}$$

$$\frac{dy_i^{(\beta)}}{d\alpha_i^{(\beta)}} = \frac{\beta \exp\left(\frac{1}{\beta}\left(x_i^{(\beta)} + h_i^{(\beta)} + n_i^{(\beta)}\right)\right)}{\exp\left(\frac{2}{\beta}\left(x_i^{(\beta)} + h_i^{(\beta)}\right)\right) + \exp\left(\frac{2}{\beta} n_i^{(\beta)}\right) + 2\alpha_i \exp\left(\frac{1}{\beta}\left(x_i^{(\beta)} + h_i^{(\beta)} + n_i^{(\beta)}\right)\right)}. \tag{C.11d}$$

Some implementations of VTS compensation (e.g. Liao 2007) have used magnitude-spectrum features ($\beta = 1$), but assumed the mismatch function was simply

$$\exp\left(y_i^{(1)}\right) = \exp\left(x_i^{(1)} + h_i^{(1)}\right) + \exp\left(n_i^{(1)}\right). \tag{C.12}$$

It is interesting to see the effect of these assumptions. By converting this back to power-spectral features,

$$\exp\left(\tfrac{1}{2} y_i^{(2)}\right) = \exp\left(\tfrac{1}{2} x_i^{(2)} + \tfrac{1}{2} h_i^{(2)}\right) + \exp\left(\tfrac{1}{2} n_i^{(2)}\right); \tag{C.13a}$$

$$\begin{aligned}\exp\left(y_i^{(2)}\right) &= \left(\exp\left(\tfrac{1}{2} x_i^{(2)} + \tfrac{1}{2} h_i^{(2)}\right) + \exp\left(\tfrac{1}{2} n_i^{(2)}\right)\right)^2 \\ &= \exp\left(x_i^{(2)}\right) + \exp\left(n_i^{(2)}\right) + 2\exp\left(\tfrac{1}{2} x_i^{(2)} + \tfrac{1}{2} h_i^{(2)} + \tfrac{1}{2} n_i^{(2)}\right). \end{aligned} \tag{C.13b}$$

This is exactly equivalent to the real mismatch function, in (C.6), with $\alpha = 1$. This means that performing VTS compensation with vectors in the magnitude domain and ignoring the phase term, as in (C.12) (e.g. Liao 2007), is essentially equivalent to assuming $\alpha = 1$ on log-power-spectral features. Also, when the noise model is ML-estimated, with the same mismatch function used for decoding, then the noise model parameters will subsume much of the difference between model and reality.

Appendix D

Derivation of model-space Algonquin

The Algonquin algorithm, when applied to the model, compensates each Gaussian separately for each observation. It is not clear from the original presentation that this happens.

The original presentation replaces the likelihood calculation for each component by a computation of the component's "soft information score" (Kristjansson and Frey 2002; Kristjansson 2002). It derives this form from the feature enhancement version. The end result can be derived more directly, which the following will do before showing the equivalence to the original presentation. As in the original, this will assume that there is only one component per state (though the generalisation is straightforward).

Consider a speech recogniser, which aims to find the most likely state sequence Θ from an observation sequence $\mathbf{Y}$ with

$$\begin{aligned} P(\Theta|\mathbf{Y}) &= \frac{p(\mathbf{Y}|\Theta)\,P(\Theta)}{p(\mathbf{Y})} \\ &\propto p(\mathbf{Y}|\Theta)\,P(\Theta) = P(\theta_0)\prod_t P(\mathbf{y}_t|\theta_t)\,P(\theta_t|\theta_{t-1})\,. \end{aligned} \tag{D.1}$$

The normalisation constant $p(\mathbf{Y})$ is normally ignored, because it does not change depending on the state sequence Θ. An obvious way of approximating the component likelihood $p(\mathbf{y}_t|\theta_t)$ would be to replace it by the Algonquin approximation for that component m and observation $\mathbf{y}_t$,

$$p(\mathbf{y}_t|\theta_t) \simeq \sum_{m \in \Omega^{(\theta)}} \pi_\theta^{(m)} q_{y_t}^{(m)}(\mathbf{y}_t). \tag{D.2}$$

That this approximation to $p(\mathbf{y}_t|\theta_t)$ is not necessarily normalised is not a problem for the Viterbi algorithm. A relevant question is, however, whether the likelihood approximations for different components can be compared, when they are compensated differently.

The original presentation of Algonquin for model compensation (Kristjansson 2002; Kristjansson and Frey 2002) derives this form (up to a constant factor) through a number of manipulations. This relates the form of models compensation to the one for feature enhancement. The normalisation constant in (D.1) is not ignored, but approximated with a mixture of Gaussians:

$$p(\mathbf{Y}) \simeq \prod_t p(\mathbf{y}_t)\,. \tag{D.3}$$

This decouples the distribution of $\mathbf{y}$ from the state sequence. This is substituted in in the expression that speech recognisers aim to maximise, $P(\Theta|\mathbf{Y})$. It is then rewritten by applying Bayes' rule, the approximation in (D.3), and again Bayes' rule:

$$\begin{aligned} P(\Theta|\mathbf{Y}) = \frac{p(\mathbf{Y}|\Theta)}{p(\mathbf{Y})} P(\Theta) &\simeq P(\theta_0) \prod_t \frac{p(\mathbf{y}_t|\theta_t)}{p(\mathbf{y}_t)} P(\theta_t|\theta_{t-1}) \\ &= P(\theta_0) \prod_t \frac{P(\theta_t|\mathbf{y}_t)\, p(\mathbf{y}_t)}{p(\mathbf{y}_t)\, P(\theta_t)} P(\theta_t|\theta_{t-1}) \\ &= P(\theta_0) \prod_t \frac{P(\theta_t|\mathbf{y}_t)}{P(\theta_t)} P(\theta_t|\theta_{t-1})\,. \end{aligned} \tag{D.4}$$

The expression that now replaces the likelihood computation for component $\theta_t = m$ (again, assuming the state-conditional distribution Gaussian) in the speech recogniser, $P(m|\mathbf{y}_t)\,/P(m)$, is then called the "soft information score". Its numerator,

$P(m|\mathbf{y}_t)$, is approximated by its variational approximation[1]

$$P(m|\mathbf{y}_t) \simeq q_{y_t}(m) = \frac{q_{y_t}^{(m)}(\mathbf{y}_t)P(m)}{\sum_{m'} q_{y_t}^{(m')}(\mathbf{y}_t)P(m')}. \tag{D.5}$$

The soft information score therefore is approximated

$$\frac{P(m|\mathbf{y}_t)}{P(m)} \simeq \frac{q_{y_t}(m)}{P(m)} = \frac{q_{y_t}^{(m)}(\mathbf{y}_t)}{\sum_{m'} q_{y_t}^{(m')}(\mathbf{y}_t)P(m')}. \tag{D.6}$$

The normalisation term for one frame is the same across all components. Therefore, it does not have an effect on decoding, so that the likelihood is in essence replaced by unnormalised distribution $q_{y_t}(\mathbf{y}_t)$, as in (D.2). The question thus remains whether the unnormalised Algonquin approximations to the likelihood are comparable between difference components.

[1] See Kristjansson (2002), equations (10.10), (10.15), and (8.18) for more details.

Appendix E

The likelihood for piecewise linear approximation

This section follows the transformation of the expression for the likelihood presented in Myrvoll and Nakamura (2004). The explanation of the idea behind it is in section 4.5.2. Section E.1 discusses the single-dimensional case as in the original paper. Section E.2 generalises it to more dimensions.

E.1 Single-dimensional

The interaction between the log-spectral coefficients of the speech x, the noise n, and the observation y is assumed to be

$$\exp(y) = \exp(x) + \exp(n)\,. \tag{E.1}$$

y is set to its observed value, y_t.

The substitute variable introduced to replace the integration over x and n is defined

$$u = 1 - \exp(x - y_t)\,, \tag{E.2a}$$

so that

$$\begin{aligned} n &= \log(\exp(y_t) - \exp(x)) = y_t + \log(1 - \exp(x - y_t)) \\ &= y_t + \log(u)\,; \qquad \text{(E.2b)} \\ x &= y_t + \log(1 - u)\,. \qquad \text{(E.2c)} \end{aligned}$$

Two useful derivatives for transforming the integral are the following. The derivative of n with respect to y while keeping x fixed is

$$\frac{\partial n(x,y)}{\partial y} = \frac{\exp(y)}{\exp(y) - \exp(x)} = \frac{1}{1 - \exp(x - y)} = \frac{1}{u}. \qquad \text{(E.3a)}$$

The notation $n(x,y)$ is used to indicate the value of n that the setting of the other two variables (x,y) implies. Similarly, the derivative of x with respect to u while keeping y_t fixed is

$$\frac{\partial x(u,y_t)}{\partial u} = \frac{-1}{1-u}. \qquad \text{(E.3b)}$$

As explained in section 4.5.2 (and see also section A.1.1), the transformation of the integral in the likelihood expression uses the absolute values of the two derivatives.

$$\begin{aligned} p(y_t) &= \int_{-\infty}^{y_t} p(y_t|x)\, p(x) dx \\ &= \int_{-\infty}^{y_t} \left| \frac{\partial n(x,y)}{\partial y}\bigg|_{y_t} \right| p(n(x,y_t))\, p(x) dx \\ &= \int_0^1 \left| \frac{\partial n(x,y)}{\partial y}\bigg|_{y_t} \right| p(n(u,y_t)) \left| \frac{\partial x(u,y_t)}{\partial u} \right| p(x(u,y_t)) du \\ &= \int_0^1 \frac{1}{u} \mathcal{N}\Big(y_t + \log(u)\,;\, \mu_n,\, \sigma_n^2\Big)\, \frac{1}{1-u} \mathcal{N}\Big(y_t + \log(1-u)\,;\, \mu_x,\, \sigma_x^2\Big) du. \end{aligned} \qquad \text{(E.4)}$$

Rewriting only the left-hand term of the integrand, noting that $\frac{1}{u} = \exp(-\log(u))$,

$$\begin{aligned}
&\frac{1}{u}\mathcal{N}\Big(\log(u) + y_t;\, \mu_n,\, \sigma_n^2\Big) \\
&\quad = \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{(\log(u) + y_t - \mu_n)^2}{2\sigma_n^2} - \log(u)\right) \\
&\quad = \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{(\log(u) + y_t - \mu_n + \sigma_n^2)^2}{2\sigma_n^2} + \tfrac{1}{2}\sigma_n^2 + y_t - \mu_n\right) \\
&\quad = \exp\Big(\tfrac{1}{2}\sigma_n^2 + y_t - \mu_n\Big)\,\mathcal{N}\Big(\log(u);\, \mu_n - \sigma_n^2 - y_t,\, \sigma_n^2\Big)\,. \qquad \text{(E.5)}
\end{aligned}$$

The right-hand side of the integrand can be rewritten in a similar way, so that the likelihood expression becomes

$$\begin{aligned}
p(y_t) = \exp\Big(\tfrac{1}{2}\sigma_n^2 + \tfrac{1}{2}\sigma_x^2 - \mu_n - \mu_x + 2y_t\Big) \\
\int_0^1 \mathcal{N}\Big(\log(u);\, \mu_n - \sigma_n^2 - y_t,\, \sigma_n^2\Big)\,\mathcal{N}\Big(\log(1-u);\, \mu_x - \sigma_x^2 - y_t,\, \sigma_x^2\Big)\,du. \qquad \text{(E.6)}
\end{aligned}$$

By approximating $\log(u)$ and $\log(1-u)$ with a piecewise linear function (Myrvoll and Nakamura 2004), the integral can be written as a sum of integrals over part of a Gaussian and a constant factor.

E.2 Multi-dimensional

That the derivation above can use scalars crucially relies on two assumptions. The assumption that the ith coordinate of the clean speech only influences the ith coordinate of the corrupted speech is only valid in the log-spectral domain. The assumption that the coordinates of both the clean speech and the corrupted speech are uncorrelated is marginally valid in the cepstral domain, and invalid in the log-spectral domain. The following generalises the derivation above to a vector of MFCCs. MFCCs are related to log-spectral coefficients by a linear transformation. As long as the distributions in the log-spectral domain are not assumed uncorrelated, therefore, a derivation in the log-spectral domain can be used.

The relation of the clean speech, noise, and corrupted speech for every dimension is the same as the single-dimensional case in (E.1), so that for vectors:

$$\mathbf{exp}(\mathbf{y}) = \mathbf{exp}(\mathbf{x}) + \mathbf{exp}(\mathbf{n})\,. \tag{E.7}$$

Again, $\mathbf{y}$ is set to its observed value, $\mathbf{y}_t$.

The coefficients of the substitute variable $\mathbf{u}$ are defined as in (E.2), so that in vector notation,

$$\mathbf{u} = 1 - \mathbf{exp}(\mathbf{x} - \mathbf{y}_t)\,, \tag{E.8}$$

so that

$$\mathbf{n} = \mathbf{y}_t + \mathbf{log}(\mathbf{u})\,; \tag{E.9}$$

$$\mathbf{x} = \mathbf{y}_t + \mathbf{log}(1 - \mathbf{u})\,, \tag{E.10}$$

where $\mathbf{1}$ is a vector with all entries set to 1.

The absolutes of the derivatives that the transformation of the feature space results in in one-dimensional space generalise to determinants of partial derivatives. Since the relationships between speech, noise, substitute variable, and observation are element-by-element in log-spectral space, the partial derivatives are diagonal. The generalisations of the derivatives in (E.3) therefore is (note that $u \in [0, 1]$)

$$\left|\frac{\partial \mathbf{n}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{y}}\right| = \left|\prod_i \frac{\partial n(x_i, y_i)}{\partial y_i}\right| = \left|\prod_i \frac{1}{1 - \exp(x_i - y_i)}\right| = \prod_i \frac{1}{u_i}; \tag{E.11}$$

$$\left|\frac{\partial \mathbf{x}(\mathbf{u}, \mathbf{y}_t)}{\partial \mathbf{u}}\right| = \left|\prod_i \frac{\partial x(u_i, y_{t,i})}{\partial u_i}\right| = \prod_i \frac{1}{1 - u_i}. \tag{E.12}$$

The additive noise and the clean speech are distributed as

$$\mathbf{n} \sim \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)\,; \qquad \mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)\,. \tag{E.13}$$

The likelihood of $\mathbf{y}_t$ generalises (E.4):

$$\begin{aligned}
p(\mathbf{y}_t) &= \int p(\mathbf{y}_t|\mathbf{x})\, p(\mathbf{x}) d\mathbf{x} \\
&= \int \left| \frac{\partial \mathbf{n}(\mathbf{x},\mathbf{y})}{\partial \mathbf{y}} \Big|_{\mathbf{y}_t} \right| p(\mathbf{n}(\mathbf{x},\mathbf{y}))\, p(\mathbf{x}) d\mathbf{x} \\
&= \int_{[0,1]^d} \left| \frac{\partial \mathbf{n}(\mathbf{x},\mathbf{y})}{\partial \mathbf{y}} \Big|_{\mathbf{y}_t} \right| p(\mathbf{n}(\mathbf{u},\mathbf{y})) \left| \frac{\partial \mathbf{x}(\mathbf{u},\mathbf{y}_t)}{\partial \mathbf{u}} \right| p(\mathbf{x}(\mathbf{u},\mathbf{y})) d\mathbf{u} \\
&= \int_{[0,1]^d} \Big(\prod_i \frac{1}{u_i} \Big) \mathcal{N}(\mathbf{y}_t + \mathbf{log}(\mathbf{u})\,;\, \boldsymbol{\mu}_n,\, \boldsymbol{\Sigma}_n) \\
&\qquad \Big(\prod_i \frac{1}{1-u_i} \Big) \mathcal{N}(\mathbf{y}_t + \mathbf{log}(\mathbf{1}-\mathbf{u})\,;\, \boldsymbol{\mu}_x,\, \boldsymbol{\Sigma}_x)\; d\mathbf{u}.
\end{aligned} \tag{E.14}$$

Noting that

$$\prod_i \frac{1}{u_i} = \exp\Big(-\sum_i \log(u_i) \Big) = \exp\big(-\mathbf{log}(\mathbf{u})^\mathsf{T} \mathbf{1} \big); \tag{E.15}$$

$$\begin{aligned}
&\mathcal{N}(\mathbf{log}(\mathbf{u}) + \mathbf{y}_t;\, \boldsymbol{\mu}_n,\, \boldsymbol{\Sigma}_n) = \\
&\qquad |2\pi\boldsymbol{\Sigma}_n|^{-\frac{1}{2}} \exp\Big(-\tfrac{1}{2} (\mathbf{log}(\mathbf{u}) + \mathbf{y}_t - \boldsymbol{\mu}_n)^\mathsf{T} \boldsymbol{\Sigma}_n^{-1} (\mathbf{log}(\mathbf{u}) + \mathbf{y}_t - \boldsymbol{\mu}_n) \Big),
\end{aligned} \tag{E.16}$$

the left term in (E.14) becomes (generalising (E.5))

$$\begin{aligned}
&\Big(\prod_i \frac{1}{u_i} \Big) \mathcal{N}(\mathbf{log}(\mathbf{u}) + \mathbf{y}_t;\, \boldsymbol{\mu}_n,\, \boldsymbol{\Sigma}_n) \\
&= |2\pi\boldsymbol{\Sigma}_n|^{-\frac{1}{2}} \exp\Big(-\tfrac{1}{2} (\mathbf{log}(\mathbf{u}) + \mathbf{y}_t - \boldsymbol{\mu}_n)^\mathsf{T} \boldsymbol{\Sigma}_n^{-1} (\mathbf{log}(\mathbf{u}) + \mathbf{y}_t - \boldsymbol{\mu}_n) - \mathbf{log}(\mathbf{u})^\mathsf{T} \mathbf{1} \Big) \\
&= |2\pi\boldsymbol{\Sigma}_n|^{-\frac{1}{2}} \exp\Big(-\tfrac{1}{2} (\mathbf{log}(\mathbf{u}) + \mathbf{y}_t - \boldsymbol{\mu}_n + \boldsymbol{\Sigma}_n \mathbf{1})^\mathsf{T} \boldsymbol{\Sigma}_n^{-1} (\mathbf{log}(\mathbf{u}) + \mathbf{y}_t - \boldsymbol{\mu}_n + \boldsymbol{\Sigma}_n \mathbf{1}) \\
&\qquad\qquad + \tfrac{1}{2} \mathbf{1}^\mathsf{T} \boldsymbol{\Sigma}_n \mathbf{1} + \mathbf{1}^\mathsf{T} \mathbf{y}_t - \mathbf{1}^\mathsf{T} \boldsymbol{\mu}_n \Big) \\
&= \mathcal{N}(\mathbf{log}(\mathbf{u})\,;\, \boldsymbol{\mu}_n - \mathbf{y}_t - \boldsymbol{\Sigma}_n \mathbf{1},\, \boldsymbol{\Sigma}_n) \exp\Big(\tfrac{1}{2} \mathbf{1}^\mathsf{T} \boldsymbol{\Sigma}_n \mathbf{1} + \mathbf{1}^\mathsf{T} \mathbf{y}_t - \mathbf{1}^\mathsf{T} \boldsymbol{\mu}_n \Big).
\end{aligned} \tag{E.17}$$

Applying the same process to the right term, the likelihood of $\mathbf{y}_t$ becomes

$$\begin{aligned}
&p(\mathbf{y}_t) = \exp\Big(\tfrac{1}{2} \mathbf{1}^\mathsf{T} \boldsymbol{\Sigma}_n \mathbf{1} + \tfrac{1}{2} \mathbf{1}^\mathsf{T} \boldsymbol{\Sigma}_x \mathbf{1} - \mathbf{1}^\mathsf{T} \boldsymbol{\mu}_n - \mathbf{1}^\mathsf{T} \boldsymbol{\mu}_x + 2 \cdot \mathbf{1}^\mathsf{T} \mathbf{y}_t \Big) \\
&\int_{[0,1]^d} \mathcal{N}(\mathbf{log}(\mathbf{u})\,;\, \boldsymbol{\mu}_n - \boldsymbol{\Sigma}_n \mathbf{1} - \mathbf{y}_t,\, \boldsymbol{\Sigma}_n)\, \mathcal{N}(\mathbf{log}(\mathbf{1}-\mathbf{u})\,;\, \boldsymbol{\mu}_x - \boldsymbol{\Sigma}_x \mathbf{1} - \mathbf{y}_t,\, \boldsymbol{\Sigma}_x) d\mathbf{u}.
\end{aligned} \tag{E.18}$$

In the single-dimensional case, the integral is approximated with 8 line segments. In the multi-dimensional case, the approximation would use 8^d hyperplanes. Since $\mathbf{u}$ has as many dimensions as there are filter bank coefficients, a piecewise linear approximation is infeasible.

Appendix F

The likelihood for transformed-space sampling

To approximate the integral in the expression for the likelihood of the observation, this work uses sequential importance resampling. A number of transformations of the integral are required, some of the details of which are in this appendix.

The detailed derivation of the transformation of the single-dimensional version of the integral is in section F.1. The generalisation of this transformation to the multi-dimensional case is in section F.2. One of the two factorisations of the multi-dimensional integrand that this work presents is detailed in section F.3. The form of the proposal distribution that approximates the single-dimensional integrand and the factors of the multi-dimensional integrand is in section F.4.

F.1 Transforming the single-dimensional integral

In section 7.3 on page 189, one half of a one-dimensional version of the corrupted speech likelihood is rewritten to (repeated from (7.14)):

$$p(y_t, x \leq n) = \int p(\alpha) \int_0^\infty \left| \frac{\partial x(u, y_t, \alpha)}{\partial u} \right| \cdot \left| \frac{\partial n(x, y, \alpha)}{\partial y} \Big|_{y_t, x(u, y_t, \alpha)} \right| \cdot p(x(u, y_t, \alpha)) \cdot p(n(u, y_t, \alpha)) \, du \, d\alpha, \tag{F.1a}$$

where (repeated from (7.11))

$$u = n - x. \tag{F.1b}$$

Because the derivations of the Jacobians and of $x(u, y_t, \alpha)$ and $n(u, y_t, \alpha)$ are long, they are given here.

The mismatch function is (repeated from (7.10))

$$\exp\left(y_t^{\log}\right) = \exp\left(x^{\log}\right) + \exp\left(n^{\log}\right) + 2\alpha \exp\left(\tfrac{1}{2}x^{\log} + \tfrac{1}{2}n^{\log}\right). \tag{F.2}$$

To express n as a function of x, y_t, α, (F.2) can be rewritten to

$$\exp(n) + 2\alpha \exp\left(\tfrac{1}{2}x\right) \exp\left(\tfrac{1}{2}n\right) = \exp(y_t) - \exp(x); \tag{F.3a}$$

$$\left(\exp\left(\tfrac{1}{2}n\right) + \alpha \exp\left(\tfrac{1}{2}x\right)\right)^2 = \exp(y_t) - \exp(x) + \left(\alpha \exp\left(\tfrac{1}{2}x\right)\right)^2. \tag{F.3b}$$

This is where it becomes useful that the computation is restricted to the region where $x \leq n$ so that n has only one solution. Since $-1 \leq \alpha$ (as shown in section 4.2.1.1), the squared expression on the left-hand side, $\exp\left(\frac{1}{2}n\right) + \alpha \exp\left(\frac{1}{2}x\right)$, is always non-negative. Therefore,

$$\exp\left(\tfrac{1}{2}n\right) = -\alpha \exp\left(\tfrac{1}{2}x\right) + \sqrt{\exp(y_t) - \exp(x) + \alpha^2 \exp(x)}; \tag{F.3c}$$

$$n = 2 \log\left(-\alpha \exp\left(\tfrac{1}{2}x\right) + \sqrt{\exp(y_t) + \exp(x)\,(\alpha^2 - 1)}\right). \tag{F.3d}$$

To express n as a function of u, y_t, α, (F.2) can be rewritten with $x = n - u$ from (F.1b):

$$\begin{aligned}\exp(y_t) &= \exp(n-u) + \exp(n) + 2\alpha\exp\left(\tfrac{1}{2}n - \tfrac{1}{2}u + \tfrac{1}{2}n\right)\\ &= \exp(n)\left(1 + \exp(-u) + 2\alpha\exp\left(-\tfrac{1}{2}u\right)\right); \qquad \text{(F.4a)}\\ \exp(n) &= \frac{\exp(y_t)}{1 + \exp(-u) + 2\alpha\exp\left(-\frac{1}{2}u\right)}; \qquad \text{(F.4b)}\\ n &= y_t - \log\left(1 + \exp(-u) + 2\alpha\exp\left(-\tfrac{1}{2}u\right)\right). \qquad \text{(F.4c)}\end{aligned}$$

Similarly, x can be expressed as a function of u, y_t, α by rewriting (F.2) with $n = u + x$ from (F.1b):

$$\begin{aligned}\exp(y_t) &= \exp(x) + \exp(u+x) + 2\alpha\exp\left(\tfrac{1}{2}x + \tfrac{1}{2}u + \tfrac{1}{2}x\right)\\ &= \exp(x)\left(1 + \exp(u) + 2\alpha\exp\left(\tfrac{1}{2}u\right)\right); \qquad \text{(F.5a)}\\ \exp(y_t - x) &= 1 + \exp(u) + 2\alpha\exp\left(\tfrac{1}{2}u\right); \qquad \text{(F.5b)}\\ x &= y_t - \log\left(1 + \exp(u) + 2\alpha\exp\left(\tfrac{1}{2}u\right)\right). \qquad \text{(F.5c)}\end{aligned}$$

Because u was chosen to relate x and n symmetrically, (F.4c) and (F.5c) are the same except that u is replaced by $-u$.

An equality that will come in useful derives from (F.5b):

$$\begin{aligned}\sqrt{\exp(y_t) + \exp(x)\left(\alpha^2 - 1\right)} &= \exp\left(\tfrac{1}{2}x\right)\sqrt{\exp(y_t - x) + \left(\alpha^2 - 1\right)}\\ &= \exp\left(\tfrac{1}{2}x\right)\sqrt{\exp(u) + 2\alpha\exp\left(\tfrac{1}{2}u\right) + \alpha^2}\\ &= \exp\left(\tfrac{1}{2}x\right)\left(\exp\left(\tfrac{1}{2}u\right) + \alpha\right). \qquad \text{(F.6)}\end{aligned}$$

The Jacobians in (7.14) are derivatives of (F.5c) and (F.3d):

$$\frac{\partial x(u, y_t, \alpha)}{\partial u} = -\frac{\exp(u) + \alpha\exp\left(\frac{1}{2}u\right)}{1 + \exp(u) + 2\alpha\exp\left(\frac{1}{2}u\right)} = -\frac{\exp\left(\frac{1}{2}u\right)\left(\exp\left(\frac{1}{2}u\right) + \alpha\right)}{1 + \exp(u) + 2\alpha\exp\left(\frac{1}{2}u\right)}; \qquad \text{(F.7a)}$$

$$
\begin{aligned}
&\frac{\partial n(x, y, \alpha)}{\partial y} \\
&= \frac{2}{\sqrt{\exp(y) + \exp(x)\,(\alpha^2 - 1)} - \alpha \exp\left(\frac{1}{2}x\right)} \cdot \frac{\exp(y)}{2\sqrt{\exp(y) + \exp(x)\,(\alpha^2 - 1)}} \\
&= \frac{\exp(y)}{\left(\exp\left(\frac{1}{2}x\right)\left(\exp\left(\frac{1}{2}u\right) + \alpha\right) - \alpha \exp\left(\frac{1}{2}x\right)\right) \exp\left(\frac{1}{2}x\right)\left(\exp\left(\frac{1}{2}u\right) + \alpha\right)} \\
&= \frac{\exp(y - x)}{\exp\left(\frac{1}{2}u\right)\left(\exp\left(\frac{1}{2}u\right) + \alpha\right)} = \frac{1 + \exp(u) + 2\alpha \exp\left(\frac{1}{2}u\right)}{\exp\left(\frac{1}{2}u\right)\left(\exp\left(\frac{1}{2}u\right) + \alpha\right)}.
\end{aligned} \tag{F.7b}
$$

When these are multiplied, as in the integral in (F.1a), they drop out against each other, except for the negation:

$$
\frac{\partial x(u, y_t, \alpha)}{\partial u} \left.\frac{\partial n(x, y, \alpha)}{\partial y}\right|_{y_t} = -1. \tag{F.7c}
$$

This does not seem to be an intrinsic property of the process.

F.2 Transforming the multi-dimensional integral

The transformation of the integral that returns the likelihood of the corrupted-speech observation is more laborious for multiple dimensions than for a single dimension. The derivation uses three steps. First, the integral is split into separate dimensions. Then, each of the integrals for one dimension is rewritten similarly to appendix F.1. Finally, the dimensions are collated.

The full expression for the likelihood of observation $\mathbf{y}_t$ is (repeated from (7.1))

$$
p(\mathbf{y}_t) = \int\int\int p(\mathbf{y}_t|\mathbf{x}, \mathbf{n}, \boldsymbol{\alpha})\, p(\mathbf{x})\, p(\mathbf{n})\, p(\boldsymbol{\alpha}) \mathrm{d}\mathbf{x} \mathrm{d}\mathbf{n} \mathrm{d}\boldsymbol{\alpha}. \tag{F.8}
$$

Just like in the single-dimensional case, the integration over the clean speech and the additive noise will be rewritten as an integral over a substitute variable. For each dimension, this substitution is the same as the one in (7.11). In multiple dimensions, the substitute variable $\mathbf{u}$ also relates the speech $\mathbf{x}$ and the noise $\mathbf{n}$ symmetrically:

$$
\mathbf{u} = \mathbf{x} - \mathbf{n}. \tag{F.9}
$$

However, the transformation of the integral will work one dimension at a time. Per dimension, the derivation will be split in two regions which use symmetric derivations, like in section 7.3.1. Again, the derivation will be explicitly given only for

$x_i \leq n_i$, with $n_i < x_i$ completely analogous. By formulating (F.8) recursively, it can be transformed one scalar at a time. The following marginalises out one variable at a time, starting with α_i:

$$\begin{aligned} &p(\mathbf{y}_{t,i:d}, x_i \leq n_i | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}) \\ &\quad = \int p(\alpha_i | \boldsymbol{\alpha}_{1:i-1})\, p(\mathbf{y}_{t,i:d}, x_i \leq n_i | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i})\, d\alpha_i, \end{aligned} \tag{F.10a}$$

where, marginalising out x_i,

$$\begin{aligned} &p(\mathbf{y}_{t,i:d}, x_i \leq n_i | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) \\ &\quad = \int p(x_i | \mathbf{x}_{1:i-1})\, p(\mathbf{y}_{t,i:d}, x_i \leq n_i | \mathbf{x}_{1:i}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i})\, dx_i, \end{aligned} \tag{F.10b}$$

where, with the restriction $x_i \leq n_i$ subsumed in the range of the integration over n_i,

$$\begin{aligned} &p(\mathbf{y}_{t,i:d}, x_i \leq n_i | \mathbf{x}_{1:i}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) \\ &\quad = \int_{x_i}^{\infty} p(n_i | \mathbf{n}_{1:i-1})\, p(\mathbf{y}_{t,i:d} | \mathbf{x}_{1:i}, \mathbf{n}_{1:i}, \boldsymbol{\alpha}_{1:i})\, dn_i. \end{aligned} \tag{F.10c}$$

The integrals in (F.10c) and in (F.10b) can then be re-expressed as one integral over the substitute variable.

First, the integral over n_i in (F.10c) can be written without the integral. This is because given the clean speech, additive noise, and phase factor for one dimension, the corrupted speech for that dimension is deterministic:

$$p(y_{t,i} | x_i, n_i, \alpha_i) = \delta_{f(x_i, n_i, \alpha_i)}(y_{t,i})\,. \tag{F.11}$$

The variable of the Dirac delta in (F.11) can be transformed using the Jacobian (see (A.1) in appendix A.1.1):

$$\begin{aligned} &p(\mathbf{y}_{t,i:d}, x_i \leq n_i | \mathbf{x}_{1:i}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) \\ &\quad = \int_{x_i}^{\infty} p(n_i | \mathbf{n}_{1:i-1})\, p(y_{t,i} | x_i, n_i, \alpha_i)\, p(\mathbf{y}_{t,i+1:d} | \mathbf{x}_{1:i}, \mathbf{n}_{1:i}, \boldsymbol{\alpha}_{1:i})\, dn_i \\ &\quad = \int_{x_i}^{\infty} p(n_i | \mathbf{n}_{1:i-1})\, \delta_{f(x_i, n_i, \alpha_i)}(y_{t,i})\, p(\mathbf{y}_{t,i+1:d} | \mathbf{x}_{1:i}, \mathbf{n}_{1:i}, \boldsymbol{\alpha}_{1:i})\, dn_i \end{aligned}$$

$$
\begin{aligned}
&= \int_{x_i}^{\infty} p(n_i|\mathbf{n}_{1:i-1}) \cdot \left| \frac{dn(x_i, \alpha_i, y_i)}{dy_i}\bigg|_{y_{t,i}} \right| \cdot \delta_{n(x_i,\alpha_i,y_{t,i})}(n_i) \\
&\qquad p(\mathbf{y}_{t,i+1:d}|\mathbf{x}_{1:i}, \mathbf{n}_{1:i}, \boldsymbol{\alpha}_{1:i})\, dn_i \\
&= \left| \frac{dn(x_i, \alpha_i, y_i)}{dy_i}\bigg|_{y_{t,i}} \right| \cdot 1(x_i \le n_i)\, p(n(x_i, \alpha_i, y_{t,i})|\mathbf{n}_{1:i-1}) \\
&\qquad p\big(\mathbf{y}_{t,i+1:d}\big|\mathbf{x}_{1:i}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}, n_i = n(x_i, \alpha_i, y_{t,i})\big). \qquad \text{(F.12)}
\end{aligned}
$$

The next step is to substitute this result into (F.10b), and then replace the variable of the integral from x_i to u_i. The Jacobians that result from this are exactly the same as the ones in section 7.3.1 on page 190, in (7.14). Since the product of their absolutes is therefore again 1, they drop out.

$$
\begin{aligned}
&p(\mathbf{y}_{t,i:d}, x_i \le n_i|\mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) \\
&= \int p(x_i|\mathbf{x}_{1:i-1}) \left| \frac{dn(x_i, \alpha_i, y_i)}{dy_i}\bigg|_{y_{t,i}} \right| 1(x_i \le n_i)\, p(n(x_i, \alpha_i, y_{t,i})|\mathbf{n}_{1:i-1}) \\
&\qquad p\big(\mathbf{y}_{t,i+1:d}\big|\mathbf{x}_{1:i}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}, n_i = n(x_i, \alpha_i, y_{t,i})\big)\, dx_i \\
&= \int_0^{\infty} \left| \frac{dx(u_i, \alpha_i, y_i)}{du_i} \right| \cdot \left| \frac{dn(x_i, \alpha_i, y_i)}{dy_i}\bigg|_{y_{t,i}} \right| \\
&\qquad p(x(u_i, \alpha_i, y_{t,i})|\mathbf{x}_{1:i-1})\, p(n(u_i, \alpha_i, y_{t,i})|\mathbf{n}_{1:i-1}) \\
&\qquad p\big(\mathbf{y}_{t,i+1:d}\big|\mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}, x_i = x(u_i, \alpha_i, y_{t,i}), n_i = n(u_i, \alpha_i, y_{t,i})\big) \\
&\qquad du_i \\
&= \int_0^{\infty} p(x(u_i, \alpha_i, y_{t,i})|\mathbf{x}_{1:i-1})\, p(n(u_i, \alpha_i, y_{t,i})|\mathbf{n}_{1:i-1}) \\
&\qquad p\big(\mathbf{y}_{t,i+1:d}\big|\mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}, x_i = x(u_i, \alpha_i, y_{t,i}), n_i = n(u_i, \alpha_i, y_{t,i})\big) \\
&\qquad du_i, \qquad \text{(F.13)}
\end{aligned}
$$

Substituting this into (F.10a) gives one half of the likelihood:

$$\begin{aligned}
&p(\mathbf{y}_{t,i:d}, x_i \leq n_i | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}) \\
&= \int p(\alpha_i | \boldsymbol{\alpha}_{1:i-1}) \int_0^\infty p(x(u_i, \alpha_i, y_{t,i}) | \mathbf{x}_{1:i-1})\, p(n(u_i, \alpha_i, y_{t,i}) | \mathbf{n}_{1:i-1}) \\
&\qquad p\big(\mathbf{y}_{t,i+1:d} \big| \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}, x_i = x(u_i, \alpha_i, y_{t,i}), n_i = n(u_i, \alpha_i, y_{t,i})\big) \\
&\qquad du_i\, d\alpha_i. \qquad \text{(F.14)}
\end{aligned}$$

This gives half the likelihood, because it is constrained to $x_i \leq n_i$. The other part, for $n_i < x_i$, has the exact same derivation with x_i and n_i swapped, and u_i replaced by $-u_i$. This is exactly the same as the single-dimensional case in appendix F.1. The full likelihood, expressed recursively, then combines integrals over $u_i \in [0, \infty)$ and $u_i \in (-\infty, 0)$:

$$\begin{aligned}
&p(\mathbf{y}_{t,i:d} | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}) \\
&= p(\mathbf{y}_{t,i:d}, x_i \leq n_i | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}) \\
&\quad + p(\mathbf{y}_{t,i:d}, n_i < x_i | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}) \\
&= \int p(\alpha_i | \boldsymbol{\alpha}_{1:i-1}) \int p(x_i(u_i, \alpha_i, y_{t,i}) | \mathbf{x}_{1:i-1})\, p(n(u_i, \alpha_i, y_{t,i}) | \mathbf{n}_{1:i-1}) \\
&\qquad p\big(\mathbf{y}_{t,i+1:d} \big| \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}, x_i = x(u_i, \alpha_i, y_{t,i}), n_i = n(u_i, \alpha_i, y_{t,i})\big) \\
&\qquad du_i\, d\alpha_i. \qquad \text{(F.15)}
\end{aligned}$$

This recursive formulation is straightforward to unroll to

$$\begin{aligned}
p(\mathbf{y}_t) &= p(\mathbf{y}_{t,1:d}) \\
&= \int \left[\prod_{i=1}^{d} p(\alpha_i | \boldsymbol{\alpha}_{1:i-1})\right] \int \left[\prod_{i=1}^{d} p(x(u_i, \alpha_i, y_{t,i}) | \mathbf{x}_{1:i-1})\right] \\
&\qquad \left[\prod_{i=1}^{d} p(n(u_i, \alpha_i, y_{t,i}) | \mathbf{n}_{1:i-1})\right] \mathbf{du}\, \mathbf{d}\boldsymbol{\alpha} \\
&= \int p(\boldsymbol{\alpha}) \int p(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t))\, p(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t)) \mathbf{du}\, \mathbf{d}\boldsymbol{\alpha}, \qquad \text{(F.16)}
\end{aligned}$$

where, analogously to the one-dimensional case, $p(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t))$ denotes the value of the prior of $\mathbf{x}$ evaluated at the value of $\mathbf{x}$ implied by the values of $(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t)$, and similar

for $p(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t))$. To find these values for $\mathbf{x}$ and $\mathbf{n}$, the relations in (F.5c) and (F.4c) apply per dimension:

$$\mathbf{x} = \mathbf{y}_t - \mathbf{log}\big(\mathbf{1} + \mathbf{exp}(\mathbf{u}) + 2\boldsymbol{\alpha} \circ \mathbf{exp}\big(\tfrac{1}{2}\mathbf{u}\big)\big)\,; \tag{F.17a}$$

$$\mathbf{n} = \mathbf{y}_t - \mathbf{log}\big(\mathbf{1} + \mathbf{exp}(-\mathbf{u}) + 2\boldsymbol{\alpha} \circ \mathbf{exp}\big(-\tfrac{1}{2}\mathbf{u}\big)\big)\,. \tag{F.17b}$$

In section 7.3.2, the full integrand is called $\gamma(\mathbf{u}, \boldsymbol{\alpha})$, and the integral is approximated with sequential importance sampling. Note that this derivation holds for any form of priors for the speech and noise $p(\mathbf{x})$ and $p(\mathbf{n})$.

F.3 Postponed factorisation of the integrand

This section presents a factorisation of the integrand $\gamma(\mathbf{u}|\boldsymbol{\alpha})$. It should result in factors γ_i so that (repeated from (7.33))

$$\gamma(\mathbf{u}|\boldsymbol{\alpha}) = \mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t);\, \boldsymbol{\mu}_x,\, \boldsymbol{\Sigma}_x)\, \mathcal{N}(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t);\, \boldsymbol{\mu}_n,\, \boldsymbol{\Sigma}_n)\,. \tag{F.18}$$

The two Gaussians on the right-hand side have the same structure. The factorisation here will only explicitly consider the term deriving from the speech prior; the one deriving from the noise prior factorises analogously. A multi-variate Gaussian relates all elements in its input vector through the inverse covariance matrix, the precision matrix $\boldsymbol{\Lambda}_x$. The derivation writes these explicitly. The elements of $\boldsymbol{\Lambda}_x = \boldsymbol{\Sigma}_x^{-1}$ are denoted with $\lambda_{x,ij}$.

$$\begin{aligned}
&\mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t);\, \boldsymbol{\mu}_x,\, \boldsymbol{\Sigma}_x) \\
&= |2\pi\boldsymbol{\Sigma}_x|^{-\frac{1}{2}} \exp\Big(-\tfrac{1}{2}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t) - \boldsymbol{\mu}_x)^\mathsf{T} \boldsymbol{\Lambda}_x (\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t) - \boldsymbol{\mu}_x)\Big) \\
&= |2\pi\boldsymbol{\Sigma}_x|^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} (x(u_i, \alpha_i, y_{t,i}) - \mu_{x,i})\, \lambda_{x,ij} (x(u_j, \alpha_j, y_{t,j}) - \mu_{x,j})\right) \\
&= |2\pi\boldsymbol{\Sigma}_x|^{-\frac{1}{2}} \exp\left(\sum_{i=1}^{d} \Bigg[-\tfrac{1}{2}\lambda_{x,ii}(x(u_i, \alpha_i, y_{t,i}) - \mu_{x,i})^2 \right. \\
&\qquad \left. -(x(u_i, \alpha_i, y_{t,i}) - \mu_{x,i}) \sum_{j=1}^{i-1} \lambda_{x,ij} (x(u_j, \alpha_j, y_{t,j}) - \mu_{x,j}) \Bigg]\right)
\end{aligned}$$

$$= |2\pi\boldsymbol{\Sigma}_x|^{-\frac{1}{2}} \prod_{i=1}^{d} \exp\Big(- \tfrac{1}{2}\lambda_{x,ii}(x(u_i, \alpha_i, y_{t,i}) - \mu_{x,i})^2 - (x(u_i, \alpha_i, y_{t,i}) - \mu_{x,i})\, \nu_{x,i}\Big), \quad \text{(F.19a)}$$

where the term containing coordinates of lower dimensions $\mathbf{u}_{1:i-1}$ is

$$\nu_{x,i} = \sum_{j=1}^{i-1} \lambda_{x,ij}(x(u_j, \alpha_j, y_{t,j}) - \mu_{x,j})\,. \quad \text{(F.19b)}$$

When drawing u_i for dimension i, the coordinates of lower dimensions $\mathbf{u}_{1:i-1}$ are known.

After applying the same factorisation to the noise term, the complete integrand in (F.18) can be written

$$\begin{aligned}\gamma(\mathbf{u}|\boldsymbol{\alpha}) &= \mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t);\, \boldsymbol{\mu}_x,\, \boldsymbol{\Sigma}_x)\, \mathcal{N}(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t);\, \boldsymbol{\mu}_n,\, \boldsymbol{\Sigma}_n) \\ &= |2\pi\boldsymbol{\Sigma}_x|^{-\frac{1}{2}} |2\pi\boldsymbol{\Sigma}_n|^{-\frac{1}{2}} \prod_{i=1}^{d} \exp\Big(\\ &\quad - \tfrac{1}{2}\lambda_{x,ii}(x(u_i, \alpha_i, y_{t,i}) - \mu_{x,i})^2 - (x(u_i, \alpha_i, y_{t,i}) - \mu_{x,i})\, \nu_{x,i} \\ &\quad - \tfrac{1}{2}\lambda_{n,ii}(n(u_i, \alpha_i, y_{t,i}) - \mu_{n,i})^2 - (n(u_i, \alpha_i, y_{t,i}) - \mu_{n,i})\, \nu_{n,i}\Big),\end{aligned} \quad \text{(F.20)}$$

where $\nu_{n,i}$ is defined analogously to (F.19b). The factors are then defined as (it is arbitrary which factor takes the constant determiners)

$$\begin{aligned}&\gamma_1(u_1|\alpha_1) \\ &\quad = |2\pi\boldsymbol{\Sigma}_y|^{-\frac{1}{2}} |2\pi\boldsymbol{\Sigma}_x|^{-\frac{1}{2}} \exp\Big(- \tfrac{1}{2}\lambda_{n,11}(n(u_1, \alpha_1, y_{t,1}) - \mu_{n,1})^2 \\ &\qquad - \tfrac{1}{2}\lambda_{x,11}(x(u_1, \alpha_1, y_{t,1}) - \mu_{x,1})^2 \Big);\end{aligned} \quad \text{(F.21a)}$$

$$\begin{aligned}&\gamma_i(u_i|\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) \\ &\quad = \exp\Big(- \tfrac{1}{2}\lambda_{x,ii}(x(u_i, \alpha_i, y_{t,i}) - \mu_{x,i})^2 - (x(u_i, \alpha_i, y_{t,i}) - \mu_{x,i})\, \nu_{x,i} \\ &\qquad - \tfrac{1}{2}\lambda_{n,ii}(n(u_i, \alpha_i, y_{t,i}) - \mu_{n,i})^2 - (n(u_i, \alpha_i, y_{t,i}) - \mu_{n,i})\, \nu_{n,i}\Big),\end{aligned} \quad \text{(F.21b)}$$

To find a proposal distribution for the resulting density, it can be rewritten so that it is easily related to the one-dimensional γ in section 7.3.1, for which good proposal distributions were discussed in section 7.3.1.2. Again, the following rewrites only part of the term related to the speech prior; the noise term works completely analogously. Since for importance sampling it is the shape rather than the height of the density that the proposal distribution needs to match, the following disregards constant factors (note the use of $\propto$), which are additive within the $\exp(\cdot)$. A technique sometimes called "completing the square" helps find the shape of the term related to the clean speech in (F.21). This derivation is similar to the derivation of the parameters of a conditional Gaussian distribution (see e.g. Bishop 2006). Taking one factor from (F.19a),

$$\begin{aligned}
&\exp\Big(-\tfrac{1}{2}\lambda_{x,ii}(x(u_i,\alpha_i,y_{t,i})-\mu_{x,i})^2-(x(u_i,\alpha_i,y_{t,i})-\mu_{x,i})\,\nu_{x,i}\Big)\\
&\propto \exp\Big(-\tfrac{1}{2}\lambda_{x,ii}(x(u_i,\alpha_i,y_{t,i}))^2+\lambda_{x,ii}\mu_{x,i}x(u_i,\alpha_i,y_{t,i})-\nu_{x,i}x(u_i,\alpha_i,y_{t,i})\Big)\\
&= \exp\left(-\tfrac{1}{2}\lambda_{x,ii}(x(u_i,\alpha_i,y_{t,i}))^2+\lambda_{x,ii}\left(\mu_{x,i}-\frac{\nu_{x,i}}{\lambda_{x,ii}}\right)x(u_i,\alpha_i,y_{t,i})\right)\\
&\propto \exp\left(-\tfrac{1}{2}\lambda_{x,ii}\left[x(u_i,\alpha_i,y_{t,i})-\left(\mu_{x,i}-\frac{\nu_{x,i}}{\lambda_{x,ii}}\right)\right]^2\right)\\
&\propto \mathcal{N}\left(x(u_i,\alpha_i,y_{t,i});\,\mu_{x,i}-\frac{\nu_{x,i}}{\lambda_{x,ii}},\,\lambda_{x,ii}^{-1}\right).
\end{aligned} \tag{F.22}$$

By rewriting the additive noise term in the same manner, the factors in (F.21) turn out to be proportional to two Gaussian distributions that are functions of $x(u_i,\alpha_i,y_{t,i})$ and $n(u_i,\alpha_i,y_{t,i})$:

$$\begin{aligned}
\gamma_i(u_i|\mathbf{u}_{1:i-1},\boldsymbol{\alpha}_{1:i-1}) \propto\ &\mathcal{N}\left(x(u_i,\alpha_i,y_{t,i});\,\mu_{x,i}-\frac{\nu_{x,i}}{\lambda_{x,ii}},\,\lambda_{x,ii}^{-1}\right)\\
&\cdot\mathcal{N}\left(n(u_i,\alpha_i,y_{t,i});\,\mu_{n,i}-\frac{\nu_{n,i}}{\lambda_{n,ii}},\,\lambda_{n,ii}^{-1}\right).
\end{aligned} \tag{F.23}$$

This expression has the same shape as the one-dimensional integrand in (7.19) in section 7.3.1.

F.4 Terms of the proposal distribution

Finding the proposal distribution uses $u(x, y, \alpha)$, the value for u that follows from fixing the other variables. Where this is necessary, $u > 0$. This is equivalent to $x \leq n$, which is the area that this expression for n is valid for (repeated in (F.3d)):

$$n = 2\log\left(-\alpha\exp\left(\tfrac{1}{2}x\right) + \sqrt{\exp(y_t) + \exp(x)\left(\alpha^2 - 1\right)}\right), \tag{F.24}$$

so that u can be found with

$$u = n - x = 2\log\left(-\alpha + \sqrt{\exp(y_t - x) + \alpha^2 - 1}\right). \tag{F.25}$$

The mirror image of this expression is $u(n, y, \alpha)$, which fixes n rather than x, and is required only for $u < 0$. This expression can be found by rewriting (F.4b):

$$\exp(n) = \frac{\exp(y_t)}{1 + \exp(-u) + 2\alpha\exp\left(-\frac{1}{2}u\right)}; \tag{F.26a}$$

$$\exp(-u) + 2\alpha\exp\left(-\tfrac{1}{2}u\right) + 1 = \exp(y_t - n); \tag{F.26b}$$

$$\left(\exp\left(-\tfrac{1}{2}u\right) + \alpha\right)^2 - \alpha^2 + 1 = \exp(y_t - n); \tag{F.26c}$$

$$\left(\exp\left(-\tfrac{1}{2}u\right) + \alpha\right)^2 = \exp(y_t - n) + \alpha^2 - 1. \tag{F.26d}$$

From $u < 0$, it follows that $\exp\left(-\frac{1}{2}u\right) \geq 1$ and $\exp\left(-\frac{1}{2}u\right) + \alpha \geq 0$, so that

$$\exp\left(-\tfrac{1}{2}u\right) + \alpha = \sqrt{\exp(y_t - n) + \alpha^2 - 1}; \tag{F.26e}$$

$$u = -2\log\left(-\alpha + \sqrt{\exp(y_t - n) + \alpha^2 - 1}\right). \tag{F.26f}$$

Bibliography

Alex Acero (1990). *Acoustical and Environmental Robustness in Automatic Speech Recognition*. Ph.D. thesis, Carnegie Mellon University.

Alex Acero, Li Deng, Trausti Kristjansson, and Jerry Zhang (2000). "HMM adaptation using vector Taylor series for noisy speech recognition." In *Proceedings of ICSLP*. vol. 3, pp. 229–232.

Cyril Allauzen, Michael Riley, and Johan Schalkwyk (2009). "A Generalized Composition Algorithm for Weighted Finite-State Transducers." In *Proceedings of Interspeech*. pp. 1203–1206.

Tasos Anastasakos, John McDonough, Richard Schwartz, and John Makhoul (1996). "A Compact Model for Speaker-Adaptive Training." In *Proceedings of ICSLP*. pp. 1137–1140.

Jon A. Arrowood and Mark A. Clements (2002). "Using Observation Uncertainty in HMM Decoding." In *Proceedings of ICSLP*. pp. 1561–1564.

Scott Axelrod, Ramesh Gopinath, and Peder Olsen (2002). "Modeling With A Subspace Constraint On Inverse Covariance Matrices." In *Proceedings of ICSLP*. pp. 2177–2180.

L. E. Baum, T. Petrie, G. Soules, and N. Weiss (1970). "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains." *Annals of Mathematical Statistics* 41 (1), pp. 164–171.

Jeff A. Bilmes (1998). "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models." Tech. rep., U. C. Berkeley.

Chistopher M. Bishop (2006). *Pattern Recognition and Machine Learning*. Springer.

S. F. Boll (1979). "Suppression of acoustic noise in speech using spectral subtraction." *IEEE Transactions on Acoustics, Speech and Signal Processing* 27, pp. 113–120.

C. Breslin, K. K. Chin, M. J. F. Gales, K. Knill, and H. Xu (2010). "Prior Information for Rapid Speaker Adaptation." In *Proceedings of Interspeech*. pp. 1644–1647.

Stanley Chen and Joshua Goodman (1998). "An empirical study of smoothing techniques for language modeling." Tech. rep., Harvard University.

Beverly Collins and Inger Mees (1999). *The Phonetics of English and Dutch*. Brill, Leiden.

Justin Dauwels, Sascha Korl, and Hans-Andrea Loeliger (2006). "Particle Methods as Message Passing." In *Proceedings of 2006 IEEE International Symposium on Information Theory*. pp. 2052–2056.

Steven B. Davis and Paul Mermelstein (1980). "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences." *IEEE Transactions on Acoustics, Speech and Signal Processing* 28 (4), pp. 357–366.

Ángel de la Torre, Dominique Fohr, and Jean-Paul Haton (2002). "Statistical Adaptation of Acoustic Models to Noise Conditions for Robust Speech Recognition." In *Proceedings of ICSLP*. pp. 1437–1440.

A. P. Dempster, N. M. Laird, and D. B. Rubin (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society* 39 (1), pp. 1–38.

Li Deng, Jasha Droppo, and Alex Acero (2004). "Enhancement of log mel power spectra of speech using a phase-sensitive model of the acoustic environment and sequential estimation of the corrupting noise." *IEEE Transactions on Speech and Audio Processing* 12 (2), pp. 133–143.

V. V. Digalakis, D. Rtischev, and L. G. Neumeyer (1995). "Speaker Adaptation Using Constrained Estimation of Gaussian Mixtures." *IEEE Transactions on Speech and Audio Processing* 3 (5), pp. 357–366.

Simon Dobrišek, Janez Žibert, and France Mihelič (2010). "Towards the Optimal Minimization of a Pronunciation Dictionary Model." In *Text, Speech and Dialogue*, Springer, Berlin / Heidelberg, *Lecture Notes in Computer Science*, vol. 6231, pp. 267–274.

Pierre L. Dognin, John R. Hershey, Vaibhava Goel, and Peder A. Olsen (2009). "Restructuring Exponential Family Mixture Models." In *Proceedings of Interspeech*.

A. Doucet and A. M. Johansen (2008). "A tutorial on particle filtering and smoothing: fifteen years later." Tech. rep., Department of Statistics, University of British Columbia. URL: ⟨http://www.cs.ubc.ca/~arnaud/doucet_johansen_tutorialPF.pdf⟩.

Jasha Droppo, Alex Acero, and Li Deng (2002). "Uncertainty Decoding with SPLICE for Noise Robust Speech Recognition." In *Proceedings of ICASSP*. pp. 829–832.

Yariv Ephraim (1990). "A mimimum mean square error approach for speech enhancement." In *Proceedings of ICASSP*. pp. 829–832.

Yariv Ephraim (1992). "Statistical-model-based speech enhancement systems." *Proceedings of the IEEE* 80 (10), pp. 1526–1555.

Friedrich Faubel and Dietrich Klakow (2010). "Estimating Noise from Noisy Speech Features with a Monte Carlo Variant of the Expectation Maximization Algorithm." In *Proceedings of Interspeech*. pp. 2046–2049.

F. Flego and M. J. F. Gales (2009). "Incremental Predictive and Adaptive Noise Compensation." In *Proceedings of ICASSP*. pp. 3837–3840.

Brendan J. Frey, Li Deng, Alex Acero, and Trausti Kristjansson (2001*a*). "ALGONQUIN: Iterating Laplace's Method to Remove Multiple Types of Acoustic Distortion for Robust Speech Recognition." In *Proceedings of Eurospeech*. pp. 901–904.

Brendan J. Frey, Trausti T. Kristjansson, Li Deng, and Alex Acero (2001*b*). "ALGONQUIN: Learning Dynamic Noise Models From Noisy Speech for Robust Speech Recognition." In *Proceedings of NIPS*.

Brendan J. Frey and David J. C. MacKay (1997). "A Revolution: Belief Propagation in Graphs With Cycles." In *Proceedings of Neural Information Processing Systems*. MIT Press, pp. 479–485.

K. Fukunaga (1972). *Introduction to Statistical Pattern Recognition*. Academic Press.

Sadaoki Furui (1986). "Speaker-Independent Isolated Word Recognition Using Dynamic Features of Speech Spectrum." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34 (1), pp. 52–59.

M. J. F. Gales (1996). "The Generation and Use of Regression Class Trees for MLLR Adaptation." Tech. Rep. CUED/F-INFENG/TR.263, Cambridge University Engineering Department.

M. J. F. Gales (1998*a*). "Maximum Likelihood Linear Transformations for HMM-based Speech Recognition." *Computer Speech and Language* 12 (2), pp. 75–98.

M. J. F. Gales (1998*b*). "Predictive Model-Based Compensation Schemes for Robust Speech Recognition." *Speech Communication* 25 (1-3), pp. 49–74.

M. J. F. Gales (1999). "Semi-Tied Covariance Matrices for Hidden Markov Models." *IEEE Transactions on Speech and Audio Processing* 7 (3), pp. 272–281.

M. J. F. Gales (2000). "Cluster adaptive training of hidden Markov models." *IEEE Transactions on Speech and Audio Processing* 8 (4), pp. 417–428.

M. J. F. Gales (2002). "Maximum likelihood multiple subspace projections for hidden Markov models." *IEEE Transactions on Speech and Audio Processing* 10 (2), pp. 37–47.

M. J. F. Gales (2011). "Model-Based Approaches to Handling Uncertainty." In D. Kolossa and R. Haeb-Umbach (eds.), *Robust Speech Recognition of Uncertain Data*, Springer Verlag.

M. J. F. Gales and F. Flego (2010). "Discriminative classifiers with adaptive kernels for noise robust speech recognition." *Computer Speech and Language* 24 (4), pp. 648–662.

M. J. F. Gales and R. C. van Dalen (2007). "Predictive Linear Transforms for Noise Robust Speech Recognition." In *Proceedings of ASRU*. pp. 59–64.

M. J. F. Gales and P. C. Woodland (1996). "Mean and Variance Adaptation within the MLLR Framework." *Computer Speech and Language* 10, pp. 249–264.

Mark J. F. Gales (1995). *Model-Based Techniques for Noise Robust Speech Recognition*. Ph.D. thesis, Cambridge University.

Jean-Luc Gauvain and Chin-Hui Lee (1994). "Maximum *a Posteriori* Estimation for Multivariate Gaussian Mixture Observations of Markov Chains." *IEEE Transactions on Speech and Audio Processing* 2 (2), pp. 291 – 298.

S. Geman and D. Geman (1984). "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images." *IEEETransactions on Pattern Analysis and Machine Intelligence* 6, pp. 721–741.

Arnab Ghoshal, Daniel Povey, Mohit Agarwal, Pinar Akyazi, Lukáš Burget, Kai Feng, Ondřej Glembek, Nagendra Goel, Martin Karafiát, Ariya Rastrow, Richard C. Rose, Petr Schwarz, and Samuel Thomas (2010). "A Novel Estimation of Feature-space MLLR for Full Covariance Models." In *Proceedings of ICASSP*.

R. A. Gopinath, M. J. F. Gales, P. S. Gopalakrishnan, S. Balakrishnan-Aiyer, and M. A. Picheny (1995). "Robust speech recognition in noise - performance of the IBM continuous speech recognizer on the ARPA noise spoke task." In *Proceedings of the ARPA Workshop on Spoken Language System Technology*. pp. 127–130.

Ramesh A. Gopinath, Bhuvana Ramabhadran, and Satya Dharanipragada (1998). "Factor Analysis Invariant to Linear Transformations of Data." In *Proceedings of ICSLP*.

Reinhold Haeb-Umbach (2001). "Automatic Generation of Phonetic Regression Class Trees for MLLR Adaptation." *IEEE Transactions on Speech and Audio Processing* 9 (3), pp. 299–302.

John Harris (1994). *English sound structure*. Blackwell, Oxford.

John R. Hershey, Peder Olsen, and Steven J. Rennie (2010). "Signal Interaction and the Devil Function." In *Proceedings of Interspeech*. pp. 334–337.

John R. Hershey and Peder A. Olsen (2007). "Approximating the Kullback Leibler Divergence Between Gaussian Mixture Models." In *Proceedings of ICASSP*.

Hans-Günter Hirsch and David Pearce (2000). "The AURORA experimental framework for the performance evaluation of speech recognition systems under noise conditions." In *Proceedings of ASR*. pp. 181–188.

J. Holmes and N. Sedgwick (1986). "Noise compensation for speech recognition using probabilistic models." In *Proceedings of ICASSP*. pp. 741–744.

Yu Hu and Qiang Huo (2006). "An HMM Compensation Approach Using Unscented Transformation [*sic*] for Noisy Speech Recognition." In *Proceedings of ISCSLP*. pp. 346–357.

Simon J. Julier and Jeffrey K. Uhlmann (2004). "Unscented Filtering and Nonlinear Estimation." *Proceedings of the IEEE* 92 (3), pp. 401–422.

J. Junqua and Y. Anglade (1990). "Acoustic and perceptual studies of Lombard speech: application to isolated-word automatic speech recognition." In *Proceedings of ICASSP*. pp. 841–844.

O. Kalinli, M. L. Seltzer, and A. Acero (2009). "Noise Adaptive Training Using a Vector Taylor Series Approach for Noise Robust Automatic Speech Recognition." In *Proceedings of ICASSP*. pp. 3825–3828.

O. Kalinli, M. L. Seltzer, J. Droppo, and A. Acero (2010). "Noise Adaptive Training for Robust Automatic Speech Recognition." *IEEE Transactions on Audio, Speech, and Language Processing* 18 (8), pp. 1889–1901.

D. K. Kim and M. J. F. Gales (2010). "Noisy Constrained Maximum Likelihood Linear Regression for Noise-Robust Speech Recognition." *IEEE Transactions on Audio, Speech, and Language Processing* 19 (2), pp. 315–325.

Do Yeong Kim, Chong Kwan Un, and Nam Soo Kim (1998). "Speech recognition in noisy environments using first-order vector Taylor series." *Speech Communication* 24, pp. 39–49.

G. Kitagawa (1996). "Monte Carlo filter and smoother for non-Gaussian non-linear state space models." *Journal of Computational and Graphical Statistics* 5, pp. 1–25.

D. Klatt (1976). "A digital filter bank for spectral matching." In *Proceedings of ICASSP*. pp. 573–576.

Trausti Kristjansson, Brendan Frey, Li Deng, and Alex Acero (2001). "Joint Estimation of Noise and Channel Distortion in a Generalized EM Framework." In *Proceedings of ASRU*.

Trausti T. Kristjansson and Brendan J. Frey (2002). "Accounting for uncertainity [*sic*] in observations: a new paradigm for robust automatic speech recognition." In *Proceedings of ICASSP*. pp. 61–64.

Trausti Thor Kristjansson (2002). *Speech Recognition in Adverse Environments: a Probabilistic Approach*. Ph.D. thesis, University of Waterloo.

S. Kullback and R. A. Leibler (1951). "On Information and Sufficiency." *Annals of Mathematical Statistics* 22 (1), pp. 79–86.

Nagendra Kumar (1997). *Investigation of Silicon-Auditory Models and Generalization of Linear Discriminant Analysis for Improved Speech Recognition*. Ph.D. thesis, John Hopkins University.

C. J. Leggetter (1995). *Improved Acoustic Modelling for HMMs using Linear Transformations*. Ph.D. thesis, Cambridge University.

C. J. Leggetter and P. C. Woodland (1995). "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models." *Computer Speech and Language* 9 (2), pp. 171–185.

Volker Leutnant and Reinhold Haeb-Umbach (2009*a*). "An analytic derivation of a phase-sensitive observation model for noise robust speech recognition." In *Proceedings of Interspeech*. pp. 2395–2398.

Volker Leutnant and Reinhold Haeb-Umbach (2009*b*). "An analytic derivation of a phase-sensitive observation model for noise robust speech recognition." Tech. rep., Universität Paderborn, Faculty of Electrical Engineering and Information Technology.

Jinyu Li, Li Deng, Dong Yu, Yifan Gong, and Alex Acero (2007). "High-Performance HMM Adaptation with Joint Compensation of Additive and Convolutive Distortions via Vector Taylor Series." In *Proceedings of ASRU*. pp. 65–70.

Jinyu Li, Dong Yu, Li Deng, Yifan Gong, and Alex Acero (2009). "A unified framework of HMM adaptation with joint compensation of additive and convolutive distortions." *Computer Speech and Language* 23, pp. 389–405.

Jinyu Li, Dong Yu, Yifan Gong, and L. Deng (2010). "Unscented Transform with Online Distortion Estimation for HMM Adaptation." In *Proceedings of Interspeech*. pp. 1660–1663.

H. Liao and M. J. F. Gales (2005). "Uncertainty Decoding for Noise Robust Speech Recognition." In *Proceedings of Interspeech*. pp. 3129–3132.

H. Liao and M. J. F. Gales (2006). "Joint Uncertainty Decoding for Robust Large Vocabulary Speech Recognition." Tech. Rep. CUED/F-INFENG/TR.552, Cambridge University Engineering Department.

H. Liao and M. J. F. Gales (2007). "Adaptive Training with Joint Uncertainty Decoding for Robust Recognition of Noisy Data." In *Proceedings of ICASSP*. vol. IV, pp. 389–392.

Hank Liao (2007). *Uncertainty Decoding for Noise Robust Speech Recognition*. Ph.D. thesis, Cambridge University.

Richard P. Lippmann (1997). "Speech recognition by machines and humans." *Speech Communication* 22 (1), pp. 1–15.

Tomoko Matsui and Sadaoki Furui (1998). "N-Best-based unsupervised speaker adaptation for speech recognition." *Computer Speech and Language* 12 (1), pp. 41–50.

Thomas Minka (2005). "Divergence measures and message passing." Tech. Rep. MSR-TR-2005-173, Microsoft Research. URL: ⟨ftp://ftp.research.microsoft.com/pub/tr/TR-2005-173.pdf⟩.

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley (2008). "Speech recognition with weighted finite-state transducers." In Larry Rabiner and Fred Juang (eds.), *Handbook on Speech Processing and Speech Communication, Part E: Speech recognition*, Springer-Verlag, Heidelberg, Germany.

Pedro J. Moreno (1996). *Speech Recognition in Noisy Environments*. Ph.D. thesis, Carnegie Mellon University.

Kevin Murphy (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, UC Berkeley, Computer Science Division.

Tor André Myrvoll and Satoshi Nakamura (2003). "Optimal filtering of noisy ceptral [*sic*] coefficients for robust ASR." In *Proceedings of ASRU*. pp. 381–386.

Tor André Myrvoll and Satoshi Nakamura (2004). "Minimum mean square error filtering of noisy cepstral coefficients with applications to ASR." In *ICASSP*. pp. 977–980.

L. Neumeyer, A. Sankar, and V. Digalakis (1995). "A Comparative Study of Speaker Adaptation Techniques." In *Proceedings of Eurospeech*. pp. 1127–1130.

Leonardo Neumeyer and Mitchel Weintraub (1994). "Probabilistic Optimum Filtering for Robust Speech Recognition." In *Proceedings of ICASSP*. vol. 1, pp. 417–420.

Peder A. Olsen and Ramesh A. Gopinath (2004). "Modeling inverse covariance matrices by basis expansion." *IEEE Transactions on Speech and Audio Processing* 12 (1), pp. 37–46.

Tasuku Oonishi, Paul R. Dixon, Koji Iwano, and Sadaoki Furui (2009). "Generalization Of Specialized On-The-Fly Composition." In *Proceedings of ICASSP*. pp. 4317–4320.

Judea Pearl (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

K. B. Petersen and M. S. Pedersen (2008). "The Matrix Cookbook." URL: ⟨http://http://matrixcookbook.com/⟩.

P. Price, W. M. Fisher, J. Bernstein, and D. S. Pallett (1988). "The DARPA 1000-word Resource Management database for continuous speech recognition." In *Proceedings of ICASSP*. vol. 1, pp. 651–654.

R. C. Rose, E. M. Hofstetter, and D. A. Reynolds (1994). "Integrated models of signal and background with application to speaker identification in noise." *IEEE Transactions on Speech and Audio Processing* 2 (2), pp. 245–257.

A-V. I. Rosti and M. J. F. Gales (2004). "Factor analysed hidden Markov models for speech recognition." *Computer Speech and Language* 18 (2), pp. 181–200.

S. Sagayama, Y. Yamaguchi, S. Takahashi, and J. Takahashi (1997). "Jacobian approach to fast acoustic model adaptation." In *Proceedings of ICASSP*. vol. 2, pp. 835 – 838.

George Saon, Mukund Padmanabhan, Ramesh Gopinath, and Scott Chen (2000). "Maximum Likelihood Discriminant Feature Spaces." In *Proceedings of ICASSP*. pp. 1129–1132.

Lawrence Saul and Mazin Rahim (2000). "Maximum Likelihood and Minimum Classification Error Factor Analysis for Automatic Speech Recognition." *IEEE Transactions on Speech and Audio Processing* 8, pp. 115–125.

Mike Seltzer, Alex Acero, and Kaustubh Kalgaonkar (2010). "Acoustic Model Adaptation via Linear Spline Interpolation for Robust Speech Recognition." In *Proceedings of ICASSP*. pp. 4550–4553.

Yusuke Shinohara and Masami Akamine (2009). "Bayesian feature enhancement using a mixture of unscented transformations for uncertainty decoding of noisy speech." In *Proceedings of ICASSP*. pp. 4569–4572.

K. C. Sim and M. J. F. Gales (2004). "Basis superposition precision matrix modelling for large vocabulary continuous speech recognition." In *Proceedings of ICASSP*. vol. I, pp. 801–804.

K. C. Sim and M. J. F. Gales (2005). "Adaptation of Precision Matrix Models on Large Vocabulary Continuous Speech Recognition." In *Proceedings of ICASSP*. vol. I, pp. 97–100.

Veronique Stouten, Hugo Van hamme, and Patrick Wambacq (2004*a*). "Accounting for the Uncertainty of Speech Estimates in the Context of Model-Based Feature Enhancement." In *Proceedings of ICSLP*. pp. 105–108.

Veronique Stouten, Hugo Van hamme, and Patrick Wambacq (2004*b*). "Joint Removal of Additive and Convolutional Noise with Model-Based Feature Enhancement." In *Proceedings of ICASSP*. pp. 949–952.

Veronique Stouten, Hugo Van hamme, and Patrick Wambacq (2005). "Effect of phase-sensitive environment model and higher order VTS on noisy speech feature enhancement." In *Proceedings of ICASSP*. pp. 433–436.

Yee Whye Teh (2006). "A Hierarchical Bayesian Language Model Based On Pitman-Yor Processes." In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sydney, Australia, pp. 985–992.

R. C. van Dalen, F. Flego, and M. J. F. Gales (2009). "Transforming Features to Compensate Speech Recogniser Models for Noise." In *Proceedings of Interspeech*. pp. 2499–2502.

R. C. van Dalen and M. J. F. Gales (2008). "Covariance Modelling for Noise Robust Speech Recognition." In *Proceedings of Interspeech*. pp. 2000–2003.

R. C. van Dalen and M. J. F. Gales (2009*a*). "Extended VTS for noise-robust speech recognition." Tech. Rep. CUED/F-INFENG/TR.636, Cambridge University Engineering Department.

R. C. van Dalen and M. J. F. Gales (2009*b*). "Extended VTS for Noise-Robust Speech Recognition." In *Proceedings of ICASSP*. pp. 3829–3832.

R. C. van Dalen and M. J. F. Gales (2010*a*). "Asymptotically Exact Noise-Corrupted Speech Likelihoods." In *Proceedings of Interspeech*. pp. 709–712.

R. C. van Dalen and M. J. F. Gales (2010*b*). "A Theoretical Bound for Noise-Robust Speech Recognition." Tech. Rep. CUED/F-INFENG/TR.648, Cambridge University Engineering Department.

Rogier C. van Dalen (2007). *Optimal Feature Spaces for Noise-Robust Speech Recognition*. Master's thesis, University of Cambridge.

Rogier C. van Dalen and Mark J. F. Gales (2011). "Extended VTS for Noise-Robust Speech Recognition." *IEEE Transactions on Audio, Speech, and Language Processing* 19 (4), pp. 733–743.

Vincent Vanhoucke and Ananth Sankar (2004). "Mixtures of Inverse Covariances." *IEEE Transactions on Speech and Audio Processing* 13, pp. 250–264.

A. Varga and H. J. M. Steeneken (1993). "Assessment for automatic speech recognition II: NOISEX-92: A database and an experiment to study the effect of additive noise on speech recognition systems." *Speech Communication* 12 (3), pp. 247–251.

A. P. Varga and R. K. Moore (1990). "Hidden Markov model decomposition of speech and noise." In *Proceedings of ICASSP*. pp. 845–848.

A. J. Viterbi (1982). "Error bounds for convolutional codes and asymptotically optimum decoding algorithm." *IEEE Transactions on Information Theory* 13, pp. 260–269.

Shinji Watanabe, Yasuhiro Minami, Atsushi Nakamura, and Naonori Ueda (2004). "Variational Bayesian estimation and clustering for speech recognition." *IEEE Transactions on Speech and Audio Processing* 12, pp. 365–381.

Pascal Wiggers, Leon Rothkrantz, and Rob van de Lisdonk (2010). "Design and Implementation of a Bayesian Network Speech Recognizer." In Petr Sojka, Aleš Horák, Ivan Kopecek, and Karel Pala (eds.), *Text, Speech and Dialogue*, Springer, Berlin/Heidelberg, *Lecture Notes in Computer Science*, vol. 6231, pp. 447–454.

Haitian Xu and K. K. Chin (2009*a*). "Comparison of estimation techniques in joint uncertainty decoding for noise robust speech recognition." In *Proceedings of Interspeech*. pp. 2403–2406.

Haitian Xu and K. K. Chin (2009*b*). "Joint uncertainty decoding with the second order approximation for noise robust speech recognition." In *Proceedings of ICASSP*. pp. 3841–3844.

Haitian Xu, M. J. F. Gales, and K. K. Chin (2011). "Joint Uncertainty Decoding With Predictive Methods for Noise Robust Speech Recognition." *IEEE Transactions on Audio, Speech, and Language Processing* 19 (6), pp. 1665–1676.

Haitian Xu, Mark J. F. Gales, and K. K. Chin (2009). "Improving Joint Uncertainty Decoding Performance by Predictive Methods for Noise Robust Speech Recognition." In *Proceedings of ASRU*. pp. 222–227.

Haitian Xu, Luca Rigazio, and David Kryze (2006). "Vector Taylor series based joint uncertainty decoding." In *Proceedings of Interspeech*. pp. 1125–1128.

Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying (Andrew) Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland (2006). "The HTK book (for HTK Version 3.4)." URL: ⟨http://htk.eng.cam.ac.uk/docs/docs.shtml⟩.

Kai Yu (2006). *Adaptive Training for Large Vocabulary Continuous Speech Recognition*. Ph.D. thesis, Cambridge University.

Kai Yu and Mark J. F. Gales (2007). "Bayesian Adaptive Inference and Adaptive Training." *IEEE Transactions on Audio, Speech, and Language Processing* 15 (6), pp. 1932–1943.