EXPERIMENTS WITH SIMPLE HEBBIAN-BASED LEARNING RULES IN PATTERN CLASSIFICATION TASKS

George F. Harpur and Richard W. Prager

CUED/F-INFENG/TR 168

February 1994

Cambridge University Engineering Department
Trumpington Street
Cambridge CB2 1PZ
England

E-mail: `gfh@eng.cam.ac.uk, rwp@eng.cam.ac.uk`

Abstract

This report presents a neural network architecture which performs pattern classification using a simple form of learning based on the Hebb rule. The work was motivated by the desires to decrease computational complexity and to maintain a greater degree of biological plausibility than most other networks designed to perform similar tasks. A method of pre-processing the inputs to provide a distributed representation is described. A scheme for increasing the power of the network using a layer of 'feature detectors' is introduced: these use an unsupervised competitive learning scheme, again based on Hebbian learning. Simulation results from testing the networks on two 'real-world' problems are presented, and compared to those produced by other types of neural network.

1 Introduction

Most of the methods currently used in neural networks for pattern classification are based on the technique of supervised learning using error-correction, the most notable examples being the Widrow-Hoff (or Least Mean Square) rule (Widrow and Hoff, 1960), and back propagation (Rumelhart and Hinton, 1986).

Interest has continued, however, in simple Hebbian learning (Hebb, 1949), or modifications thereof, in which adaptation of network weights is based on the correlation between the activation of a unit and the strengths of its inputs. It has for example been shown that a Hebbian rule with an added decay term to normalise the weights can be used to perform principal component extraction (Oja, 1982).

This report investigates how, with suitable encoding of the data, Hebbian learning may be used for pattern classification using a system of *forced learning*. With the exception of a global learning rate, the learning rule employed requires only signals available locally to each unit, and no special mechanism is required during learning phases. There is no requirement for differentiability, and so simple linear or linear threshold transfer functions in the network units can be used. In many cases the networks can be trained with just one presentation of the input patterns. Despite the simplicity of the method, it is able to compete reasonably well with more complex networks — some experimental results are presented in section 5. It is also possible to draw some parallels between the input representation, structure and learning of the network model, and what is thought to occur in the brain.

2 Learning Rules

Hebbian learning in its simplest form is given by the weight update rule

$$\Delta w_{ij} = \eta \, a_i a_j \tag{1}$$

where Δw_{ij} is the change in the strength of the connection from unit j to unit i, a_i and a_j are the activations of units i and j respectively, and η is the learning rate. In training a network to classify patterns with this rule, it is necessary to have some method of forcing a unit to respond strongly to a particular pattern. Consider a set of data divided into classes $C_1, C_2, \ldots, C_m$. Each data point $\mathbf{x}$ is represented by the vector of inputs $(x_1, x_2, \ldots, x_n)$. A possible network for learning to classify the data is given in figure 1. All units are linear. During training the class inputs $c_1, c_2, \ldots, c_m$ for a point $\mathbf{x}$ are set as follows:

$$\begin{aligned} c_i &= 1 \quad \mathbf{x} \in C_i \\ c_i &= 0 \quad \mathbf{x} \notin C_i \end{aligned}$$

Each of the class inputs is connected to just one corresponding output unit, i.e. c_i connects to o_i only for $i = 1, 2, ..., m$. There is full interconnection from the data inputs $x_1, x_2, \ldots, x_n$ to each of the outputs.

In terms of Pavlovian conditioning, the class inputs represent an unconditioned stimulus (UCS) and the data inputs a conditioned stimulus (CS) — the task of the network is to learn an association between each UCS and CS so that the presence of a CS alone is sufficient to excite the response produced by the corresponding UCS.

The connections from the class inputs (shown using thick lines in figure 1) have non-modifiable weights and are treated differently from the others. One approach is to use the following learning rule for the output units:

$$\Delta w_{ij} = \eta \, u_i a_j \tag{2}$$

where u_i represents the input to unit i from a UCS. For the output units in figure 1 we have that $u_i = c_i$ for $i = 1, 2, ..., m$. This method has the attractive property that it is extremely localised: only information about the inputs to a unit is required — not even the output is used in the learning

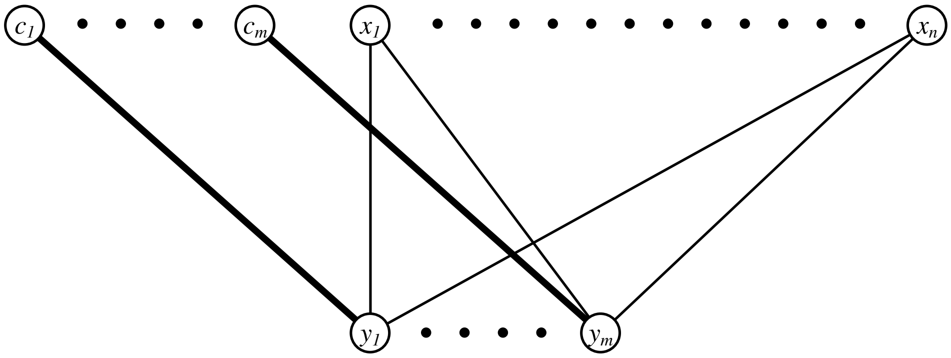

Figure 1: A single-layer network with m class inputs, n data inputs and m outputs.

rule. This corresponds also to physiological evidence — experiments, for example, with the marine snail *Hermissenda crassicornis* suggest that the molecular changes that result in learning occur as a result of local interactions between inputs from a UCS and a CS in the dendritic tree of a neuron, independently of the firing of the neuron itself (Alkon, 1989).

An alternative is to use the standard Hebbian rule (1), but with large non-modifiable connection weights w'_i from UCS (class) inputs. This technique is often referred to as *forced* learning, which is similar in effect to supervised learning, but rather than having an explicit teacher signal, the unit's activation is forced to approximately the correct value during learning by the class input. Provided that $w'_i \gg w_{ij}$ for all i, j, then this method is approximately equivalent to (2).

A problem with both of these rules is that as the networks are presented with more and more training patterns, the weights will grow without limit. A useful way of dealing with this problem is to keep the weights normalised — from a biological point of view this can be thought of as the synapses competing for a share of limited resources. One method of doing this is with Oja's rule (Oja, 1982)

$$\Delta w_{ij} = \eta \, a_i (a_j - a_i w_{ij}) \tag{3}$$

which can be shown to converge to a state in which the weights are normalised provided that η is sufficiently small.

3 Input Representation

For simple networks of the type described above to perform any useful function in pattern classification, the form in which data are presented to the network is of primary importance. It is worth considering what it should mean when a unit (input or otherwise) becomes more strongly activated. In many pattern classification networks, a unit is required to 'mean' two or more different things: for example, below a threshold value it may denote the presence of one class of input, and above the threshold the presence of another class. There is little evidence that biological neurons behave in this way — if a neuron responds to a stimulus, then a more rapid firing of the neuron is typically in response to 'more' of the same stimulus. As a consequence of such units, networks that use them require relatively complex learning rules (typically based on error-correction) and adjustable thresholds (by the use of a variable 'bias' term).

Nevertheless the raw data for many pattern classification problems, if fed directly to a network, would require just this type of behaviour of the input units. To avoid the complications mentioned above we need to use some pre-processing to re-represent the data in a form more suited to processing by a neural network. An example from nature of such pre-processing is the ear, which

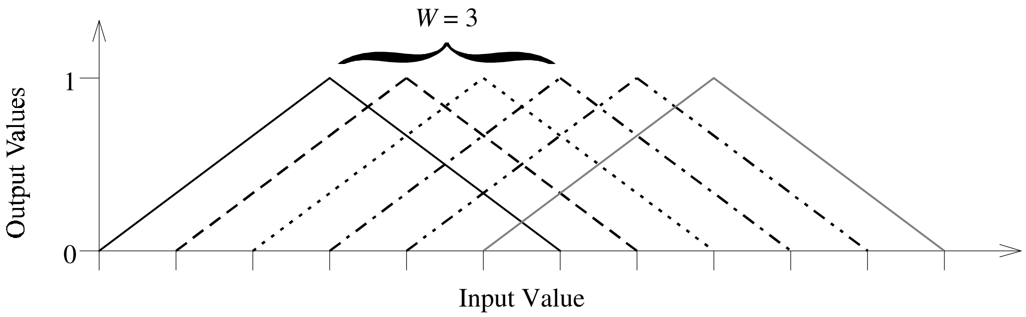

Figure 2: Overlapping receptive fields used to encode a continuous-valued input. Each different style of line represents the output value of a different receptive field.

takes a single continuous input signal, and re-represents it as a large number of signals, each coarsely tuned to a particular frequency component.

Figure 2 depicts a simple scheme for performing such pre-processing on a continuous-valued input signal. A number of 'receptive fields' are scattered evenly across the range of possible values. Each is 'tuned' to a particular value at which it has an output of one, and its response falls away linearly with distance from this value until it reaches zero. The overlap between neighbouring fields can be characterised by W, the width of the field relative to the spacing of the fields. Having a large overlap between adjacent receptive fields provides a *distributed representation* for the data, and does *not* decrease the accuracy with which a data point is specified. The advantage of such a representation is that it increases the network's power to *generalise*: two data points that are reasonably close to each other will produce similar responses from the pre-processor. If the receptive fields were very narrow, even points that were close to each other would result in the activation of completely different units with no overlap in the activation patterns. These factors are discussed at greater length in Hinton, McClelland and Rumelhart (1986).

A final crucial factor in the use of such pre-processing (a fixed non-linear mapping) is that it improves the discriminatory power of the network. A single-layer network is only able to form a linear discriminant function, but by transforming the input into a higher dimensional space, the pre-processing allows the network to form non-linear discriminations (for the case where there are two data inputs, the class boundaries are piecewise linear over the input space — at higher dimensionalities, they are composed of a number of hyperplanes, that number being dependent on the number of receptive fields used).

Providing that an integral value of W is used, the sum of the responses from all active receptive fields will be constant (equal to W) over the input space, except at the edges. Thus, assuming these responses are summed linearly (as they are in the networks presented here), no special emphasis is placed on the values chosen as the centres of the receptive fields. In order to avoid edge effects, it is necessary to place $(W-1)$ receptive fields on either end of each dimension of the input space. Thus in figure 2 only the input values between the peaks of the third and fourth receptive fields are devoid of edge effects.

4 Feature Detection

A problem with the pre-processing scheme just described is that each input value is encoded separately, and so when the network architecture of figure 1 is used, each output unit can only form a linear combination of the (encoded) inputs, that is to say that if it is to respond strongly to a *combination* of two or more inputs, then it must respond moderately to one of them in isolation.

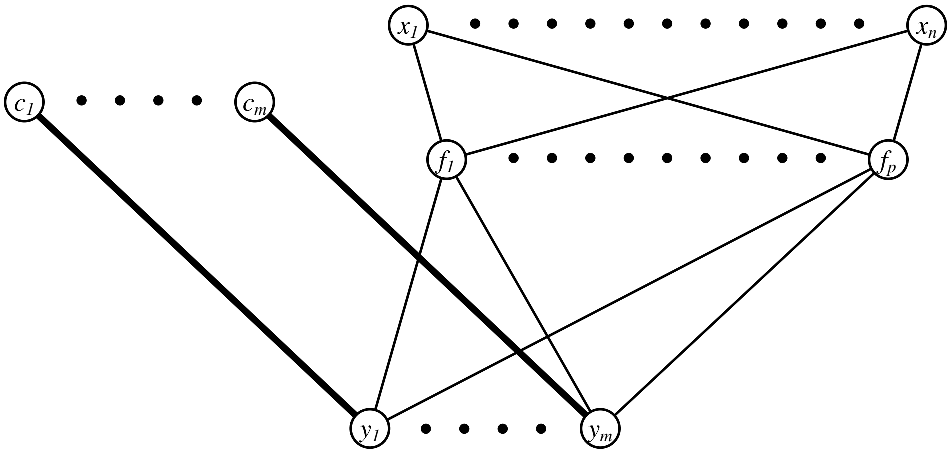

Figure 3: A network incorporating a 'hidden' layer of p feature detectors. The connection weights for this layer are adapted using unsupervised competitive learning.

In many cases this is desirable, but in others it limits the ability of the network to correctly discriminate between classes.

One could envisage a scheme of pre-processing where receptive fields were responsive to uniformly spaced *regions* in the input space as opposed to values of a single input. The problem here is that the number of receptive fields required to cover the input space uniformly grows exponentially with the number of input values. Furthermore, data points from most real-world classification problems only occupy a small fraction of the total input space, so a large proportion of the receptive fields would never be used.

What is required is an adaptive method of responding to important 'features' (i.e. combinations of inputs) in the input patterns. Figure 3 demonstrates one such method. The inputs are encoded using receptive fields as before to give an input vector $(x_1, x_2, \ldots, x_n)$ to the network. These inputs are fully connected to a layer of 'feature detectors' which are activated competitively using a 'winner takes all' scheme, i.e. only the unit with highest excitation is active at any one time. This could be achieved using a system of self-excitation and mutual inhibition within the layer, but in computer simulations it is convenient just to select the unit with the highest activation. The weights from the inputs are initialised with random values and updated using Oja's rule (3) so that the weights are normalised. The effect is that as a unit learns, it becomes more specialised to a particular combination of inputs. If the initial weight values are sufficiently large then an untrained unit (one that has not previously been activated) is likely to respond to a point in the input space that is not already covered by a trained unit, and thus become responsive to a new 'feature'.

The results of this extra processing stage are fully connected to the output units, where forced learning using the class inputs takes place as before. Note that the learning in the layer of feature detectors is unsupervised — the class inputs take no part in it. This corresponds to what is thought to occur in both visual and auditory systems in mammals: early processing extracts 'interesting' features from the data, and it is only at a higher level that any meaning is assigned to them.

5 Experiments

The networks described above have been tested on a number of 'real-world' problems. Some of the results obtained are summarised here.

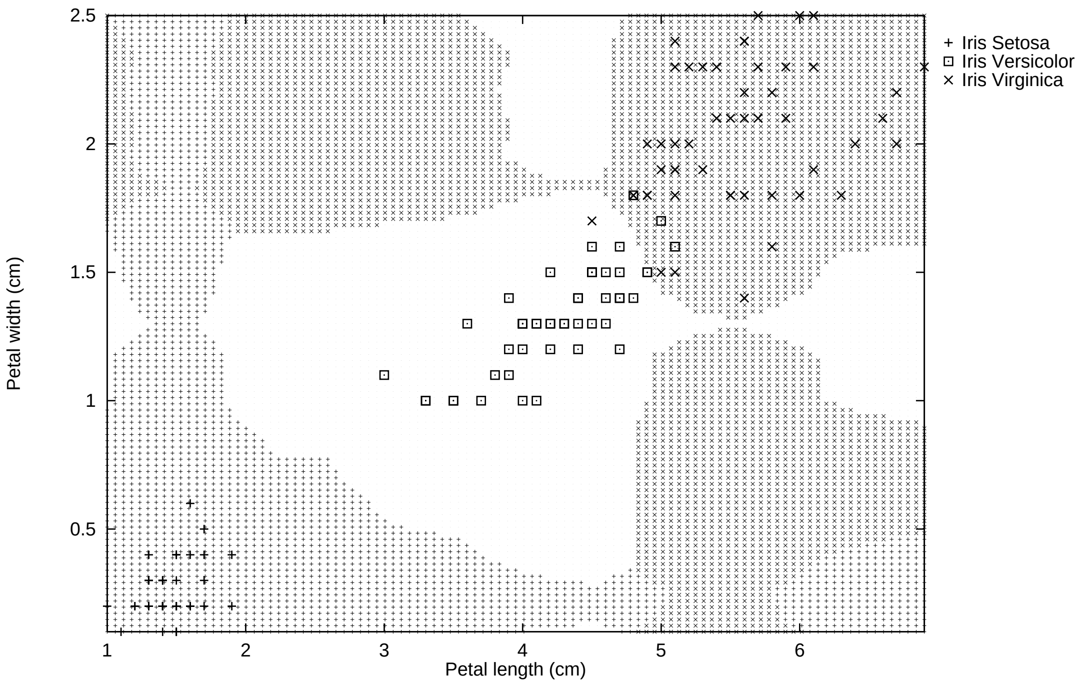

Figure 4: The classifications formed by a single-layer network trained using the iris database. The large symbols indicate the training examples. The small symbols indicate the network output giving the strongest response to test patterns distributed evenly over the entire input plane.

5.1 The Iris Plants Database

The database described here was obtained from Murphy and Aha (1992). It was first reported in Fisher (1936), and has been used in many publications since.

The problem involves classifying the data into one of three types of iris plant (Setosa, Versicolor and Virginica) based on measurements of sepal length, sepal width, petal length and petal width. In order that the results be displayed diagrammatically here, only the last two of the measurements were used for classification. The database contains 150 instances, with 50 of each type of plant. The inputs were pre-processed with 11 evenly-spaced receptive fields in each dimension with a relative width, W, of 4. A single presentation of each data point was made, and the weights modified after each presentation using equation (2) with a learning rate, η, of 0.01. The network was then tested with points distributed evenly across the input plane, and the unit with the largest output was taken as the network's classification. Figure 4 shows the input data (large symbols), and the resulting division of the input plane (small symbols). Only four mis-classifications were made, one of which was unavoidable because two data points from different classes were coincident. The class boundaries away from the training examples (the top left and bottom right corners of figure 4) are not highly significant because the responses of all three outputs are relatively low in these regions.

Figure 5 gives an indication of how one of the three output units responds to the test data. It clearly shows how the response for a combination of inputs is the sum of the responses for each of the inputs individually as discussed in section 4.

5.2 The Peterson & Barney Vowel Database

Tests were carried out with a version of the Peterson and Barney vowel data (Peterson and Barney, 1952) consisting of two repetitions of 10 vowels by 76 people (33 men, 28 women and 15 children). Some of the data is missing and the database used contains 1494 of the possible 1520 vowels. Each data item consists of the four formant frequencies of the associated vowel, and the task was to classify a set of formant frequencies as corresponding to one of the 10 possible vowels.

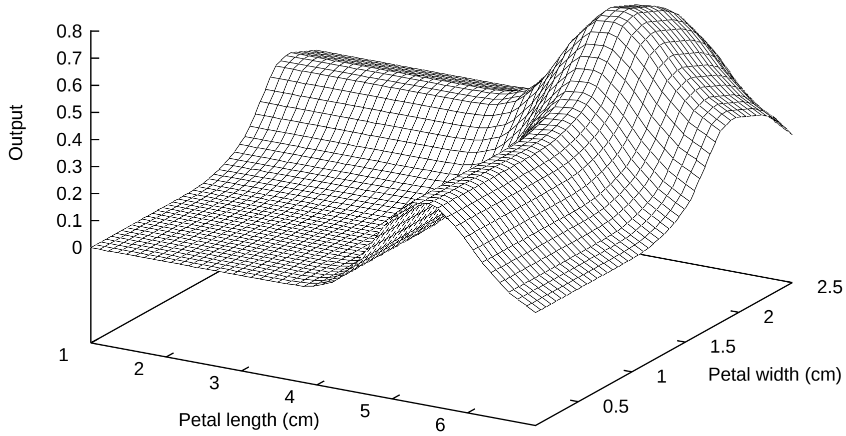

Figure 5: The activation of the output unit trained to respond to the Iris Virginica class in response to test patterns distributed over the input plane.

Two tests were made: the first used all 1494 examples for both training and testing; the second split the database randomly into a 1000 example training set and a test set of 494 examples. The second of these gives an indication of the network's ability to *generalise*, that is to correctly classify unseen data. These tests were carried out using the single-layer network of figure 1, and then with two networks of the type shown in figure 3 incorporating 100 and 500 feature detecting units respectively. For convenience in the latter case, the two layers were trained separately — the feature detectors were presented with examples picked at random (with repetitions) from the pre-processed training set. The weights were adapted using Oja's rule (3) with a learning rate, η, of 0.001. A number of examples equal to four times the number in the training set were presented to allow the weights to normalise. It was found that the number of times each detector was activated over the examples was fairly well spread — after 5976 examples had been presented to 500 feature detectors, the number of times each unit had 'won' (i.e. been activated) had an average of 11.95 and a standard deviation of only 2.03.

In both cases, just one presentation of each example was made to the output units, whose weights were adapted using equation (2). The results are summarised in table 1. Results reported in Prager (1993) and Gosling (1992) on exactly the same problem are included for comparison. The 'conventional' single-layer network refers to a network without pre-processed inputs trained using a relaxation algorithm as described in Duda and Hart (1973). It was trained until no further improvement in the performance was seen. The back propagation network used the standard backprop algorithm with 15 hidden units, a learning rate of 0.9 and a momentum term of 0.1.

The single-layer network achieved 62.1% successful classifications on the second test, and the networks using feature detectors achieved 66.4% and 70.9% success. All of these outperformed the conventional single-layer network, but were themselves outperformed by the back propagation network. This is not surprising, but the remarkable feature is that such results were achieved with just one presentation of the inputs (training cycle). In this case there is in fact no benefit to be had from repeated presentations — because both the activation function and the learning rule are linear, further training using the same examples will increase the responses of the outputs, but the ratio between responses will remain identical.

Network Type	Training cycles	Train & test: 1494	Train: 1000 test: 494
Single-layer	1	72.4	62.1
Two-layer (100 feature detectors)	4, 1	80.6	66.4
Two-layer (500 feature detectors)	4, 1	86.3	70.9
'Conventional' single-layer	4371	–	59.3
Two-layer back propagation	1000	–	85.2

Table 1: Network performance percentages for the Peterson & Barney vowel classification problem. A single training cycle is a complete presentation of all the training examples. The two figures for training cycles for the feature-detecting networks are those for the first and second layers respectively.

6 Conclusions

The networks presented here compete reasonably well with established methods of pattern classification based on error-correction, while having extremely simple learning rules and low computational complexity. The results are achieved without the need for large numbers of presentations of the input patterns.

The networks retain more biologically plausibility than most current neural network classifiers in that they use simple Hebbian-based learning rules and do not require any special mechanism for training. The price to be paid for this is that the data used must be pre-processed into a form which itself has some degree of biological plausibility, and a greater number of inputs are required as a result.

However, while increased numbers of units may be required, the units are basic, use learning rules that can operate solely on data available locally, and do not need any great degree of external control.

References

Alkon, D. L. (1989). Memory storage and neural systems, *Scientific American* pp. 26–34.

Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems, *Annual Eugenics* Part II pp. 179–188.

Gosling, J. P. M. (1992). Using the GANNET evolutionary neural network training system with the Peterson and Barney vowel database, *Technical Report of the GANNET IED Project (GR/F 34848)*, Cambridge University Engineering Department.

Hebb, D. O. (1949). *The Organization of Behaviour*, John Wiley & Sons, New York.

Hinton, G. E., McClelland, J. L. and Rumelhart, D. E. (1986). Distributed representations, *in* D. E. Rumelhart and J. L. McClelland (eds), *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Vol. 1, MIT Press, Cambridge, MA, pp. 77–109.

Murphy, P. M. and Aha, D. W. (1992). *UCI Repository of Machine Learning Databases* [Machine-readable data repository], University of California, Department of Information and Computer Science, Irvine, CA.

Oja, E. (1982). A simplified neuron model as a principal component analyzer, *J. Math. Bio.* **15**: 267–273.

Peterson, G. E. and Barney, H. L. (1952). Control methods used in a study of the vowels, *JASA* **24**: 175–184.

Prager, R. W. (1993). Networks based on Kanerva's sparse distributed memory: Results showing their strengths and limitations and a new algorithm to design the location matching layer, *Proc. IEEE International Conference on Neural Networks* (San Francisco, 1993), Vol. 2, pp. 1040–1045.

Rumelhart, D. E. and Hinton, G. E. Rumelhart, D. E. (1986). Learning internal representations by error propagation, *in* D. E. Rumelhart and J. L. McClelland (eds), *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Vol. 1, MIT Press, Cambridge, MA, pp. 318–362.

Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits, *1960 IRE WESCON Convention Record*, IRE, New York, pp. 96–104.