Reinforcement Learning for Multi-Linked Manipulator Control

Chen K. Tham, Richard W. Prager *
Cambridge University Engineering Department,
Trumpington Street, Cambridge CB2 1PZ, UK.
e-mail : ckt@eng.cam.ac.uk, rwp@eng.cam.ac.uk

Abstract

We present an automatic trajectory planning and obstacle avoidance method for a multi-linked manipulator which uses position and velocity sensor information directly to produce the appropriate continuous-valued torques for each joint. The inputs are fed into a Cerebellar Model Arithmetic Computer (CMAC) (Albus, 1975) and in each state, the expected reward and torques for each joint are learnt through self-experimentation using a combination of the Temporal Difference (TD) technique (Sutton, 1987) and stochastic hillclimbing (Williams, 1988). Actions which cause the manipulator to reach the desired destination are rewarded whereas actions which lead to collisions with either joint limits or obstacles are punished by an amount proportional to the velocity before collision. After training, the manipulator is able to move along collision free paths from different start positions in the workspace to the destination.
Keywords: Reinforcement Learning; Machine Learning; Robotics; Connectionist Models

1 Introduction

The task of controlling a multi-linked manipulator can be broadly divided into three parts : path planning; coordinate transformation from task-oriented coordinates to body coordinates as in inverse kinematics; and the generation of motor commands. Path planning involves finding the best path from a start position to the destination and usually obstacle avoidance, while the generation of motor commands involves trajectory tracking, i.e. applying the appropriate torques at each joint in order to follow some desired position and velocity profile. An accurate dynamical model of the manipulator is normally required.

This paper proposes a method that handles these tasks without using a dynamical model. A simulation of a robot manipulator with two links is used and the objective is to move from a start point to some desired destination. A *reinforcement* signal which indicates when a collision with an obstacle or joint limit has occurred or when the destination has been reached is required. Joint angles and velocities from sensors are used as inputs to the system, but inputs can also come from a range sensor or a vision system. The appropriate torques to each joint are obtained which not only enables the manipulator to reach its destination, but also avoid collisions with obstacles in the workspace.

2 Background

Barto, Sutton and Anderson (Barto et al., 1983) employed reinforcement learning in the control of a dynamical system in the form of a pole on a cart. The pole was free to move in a vertical

*This paper was presented at the 6th International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, in August, 1992. The authors thank Eli Tzirkel-Hancock, Tim Jervis and Chris Dance for their comments and help during the course of this work.

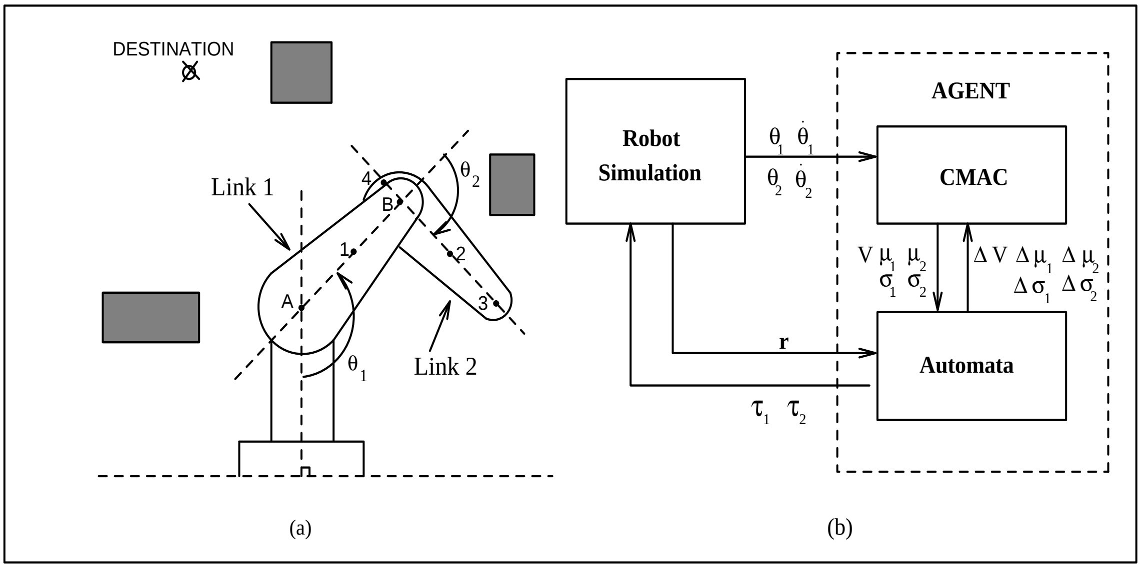


Figure 1: (a) Robot manipulator with obstacles in the workspace. (b) Interaction between the agent and robot.

plane and the cart horizontally in left and right directions. The objective was to balance the pole by applying a horizontal force to the cart. A reinforcement signal indicates when the pole has fallen over and learning was aided by the use of an Adaptive Critic Element (ACE) which tried to predict the reinforcement signal.

Nguyen and Widrow (Nguyen and Widrow, 1989) employed two connectionist networks in the control of a simulated trailer truck backing into a loading dock, one to represent the kinematic model of the trailer truck and another the controller which has to generate the appropriate steering signal at different truck positions. This is a highly non-linear control problem and the controller was able to perform the task successfully even when the cab and trailer were initially 'jack-knifed'.

More recently, Prescott and Mayhew (Prescott and Mayhew, 1991) used reinforcement learning to enable a simulated mobile robot to acquire reflexive obstacle avoidance behaviour. The input to the mobile robot is from a primitive range sensor and after training, continuous-valued forward and angular velocities are obtained which cause it to move at an optimal speed along collision free paths in an environment containing obstacles.

The task we address here contains elements of the above. As in the truck backer-upper, the robot manipulator is a non-linear system; however, it is not necessary to have a two-stage learning process in our method since an emulator of the robot dynamics is not required. We also seek to control the manipulator through continuous-valued torques in a dynamical formulation of the problem, as in the case of pole balancing.

In the discussion that follows, *robot* refers to the two-linked manipulator under consideration; *agent*, the controller which has to learn the correct actions to take; and *state* $\boldsymbol{x}$, the angular positions and velocities of the joints, i.e. $\boldsymbol{x} = [\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2]$.

3 The Robot Simulation

The robot simulation is based on an actual robot and has two revolute joints, each free to rotate through an angle of $\frac{6\pi}{4}$rads, i.e. $\frac{\pi}{4} < \theta_1 < \frac{7\pi}{4}$rads and $-\frac{3\pi}{4} < \theta_2 < \frac{3\pi}{4}$rads, where θ_1 and θ_2 are as shown in Figure 1 (a). The equations of motion are derived from the Lagrangian in terms of the potential and kinetic energies. (refer to Appendix)

A collision occurs when any part of the links hits an obstacle or when the robot attempts to move outside the allowed ranges of θ_1 and θ_2.

4 Cerebellar Model Arithmetic Computer (CMAC)

The joint angular positions and velocities are first fed into CMACs (Albus, 1975) (see Figure 1 (b)) which are used to represent several functions of the inputs described below.

The CMAC is a coarse-coding structure where each region in the input space has a set of overlapping but offset tiles associated with it. Every tile is defined by quantizing functions operating on each input and corresponds to a component of the desired output value. The sum of several of these components makes up the output value. The CMAC performs fast function approximation and has good generalization properties which promote learning since function values in neighbouring states are usually similar.

Each CMAC that we implemented has six resolution elements (in Albus' notation, $Q = 6$) over the input range for each quantizing function, there are four quantizing functions for each input ($K = 4$) and four inputs ($N = 4$), giving a total of $KQ^N = 5184$ elements. This allowed a quantizing resolution of 0.25 rads for angular positions and 1 rad/s for angular velocities for the input ranges that we considered.

5 Reinforcement Learning

Reinforcement learning is a direct learning method where the performance of the system is judged on the basis of a single scalar signal - the *reinforcement*, r. The objective is to determine the appropriate *actions* at each state from a *policy* which would maximise the *return*, i.e. the long term discounted sum of rewards available. The *evaluation function* gives an immediately accessible prediction of expected return at any state.

The learning procedure is an on-line method in which the agent seeks to learn the evaluation function $V(\boldsymbol{x})$ of a policy and adjusts the policy for improved performance in every time step. (Barto et al., 1989)

The TD (temporal difference) error is obtained by :

$$\varepsilon_{t+1} = r_{t+1} + \gamma V_t(\boldsymbol{x}_{t+1}) - V_t(\boldsymbol{x}_t) \tag{1}$$

where $\boldsymbol{x}_t$ is the state of the system at time t and γ is the *discount factor*.

The evaluation function, represented by a CMAC, is updated according to :

$$V_{t+1}(\boldsymbol{x}) = V_t(\boldsymbol{x}) + \alpha\varepsilon_{t+1}c_t(\boldsymbol{x}) \tag{2}$$

$$\begin{array}{llcll} \text{where} & c_t(\boldsymbol{x}) & = & 1 + \lambda c_{t-1}(\boldsymbol{x}) & \text{if } \boldsymbol{x} = \boldsymbol{x}_t \\ & c_t(\boldsymbol{x}) & = & \lambda c_{t-1}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \neq \boldsymbol{x}_t \end{array}$$

α is the learning rate parameter and $c_t(\boldsymbol{x})$, a *stimulus trace* affecting the states through which the system passed in the time steps preceding t. This stimulus trace, which decays at a rate dependent on the parameter λ, can accelerate the learning process as shown by Sutton (Sutton, 1987).

6 Stochastic Hillclimbing

In the task under consideration, the actions in each state are continuous-valued torques whose magnitudes and directions are determined stochastically by *associative stochastic learning automata* (Williams, 1988)[1].

Each torque is chosen from a gaussian probability distribution with two parameters, the mean μ and standard deviation σ, which are represented in state space by CMACs. The probability of applying torque τ in a joint is given by the gaussian probability density function

$$g(\tau, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\tau-\mu)^2}{2\sigma^2}}$$

Williams proposed a method to update the parameters μ and σ in the direction of increasing expected return $E(r \mid \boldsymbol{x})$, in a stochastic hillclimbing manner[2] :

[1] Prescott and Mayhew used a similar approach to obtain the forward and angular velocities for a mobile vehicle, as described above.

[2] β is used here for the learning rate parameter instead of α as in Williams to distinguish it from α in Equation 2.

$$
\begin{aligned}
\Delta\mu &= \beta(r-b_\mu)e^{(\mu)} \quad \text{where } e^{(\mu)} = \frac{\partial \ln g}{\partial \mu} = \frac{(\tau-\mu)}{\sigma^2} \\
\text{and} \quad \Delta\sigma &= \beta(r-b_\sigma)e^{(\sigma)} \quad \text{where } e^{(\sigma)} = \frac{\partial \ln g}{\partial \sigma} = \frac{(\tau-\mu)^2-\sigma^2}{\sigma^3}
\end{aligned}
$$

The terms $e^{(\mu)}$ and $e^{(\sigma)}$ are the *characteristic eligibilities* of parameter updates; $(r-b_\mu)$ and $(r-b_\sigma)$ are the *reinforcement offsets* indicating how good the last action was with respect to *reinforcement baselines* b_μ and b_σ.

The TD error ε_{t+1}, which provides a utility measure of the last action taken with respect to the expected return of all actions performed when the agent is in a particular state (Barto et al., 1989), can be used as the reinforcement offset. As in Equation 2, we shall employ stimulus traces $s_t^{(\mu)}$ and $s_t^{(\sigma)}$ when updating μ and σ respectively. Since the operations involved in updating μ and σ are the same, let

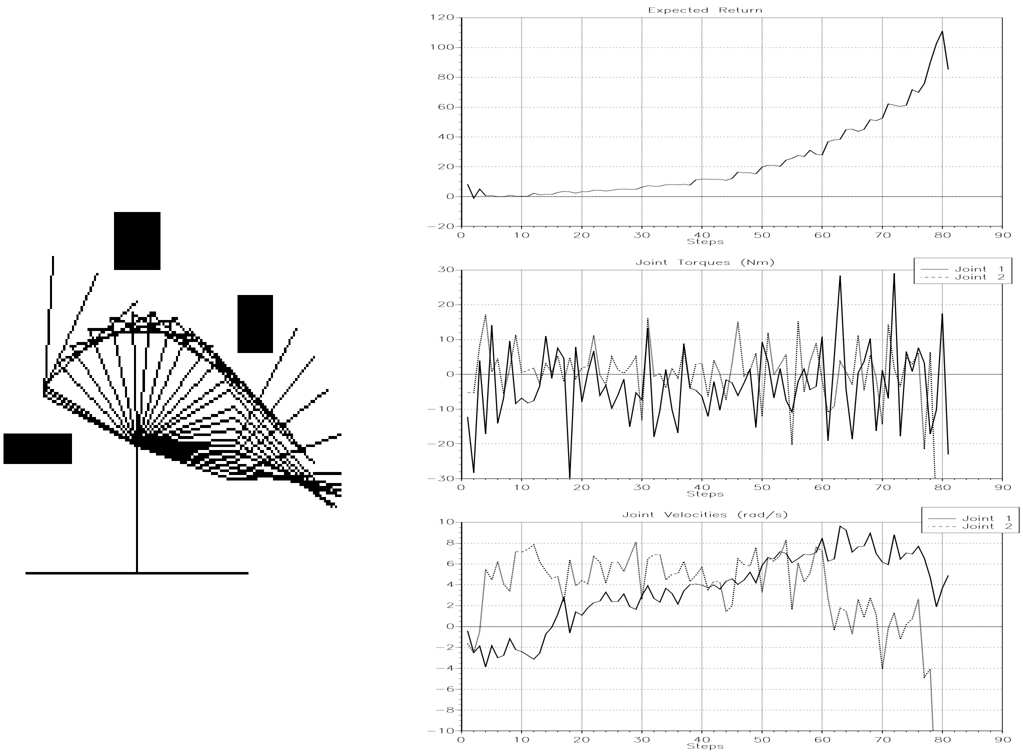
$$
\boldsymbol{p} = \begin{bmatrix} \mu \\ \sigma \end{bmatrix} \quad ; \quad \boldsymbol{e} = \begin{bmatrix} e^{(\mu)} \\ e^{(\sigma)} \end{bmatrix} \quad ; \quad \boldsymbol{s} = \begin{bmatrix} s^{(\mu)} \\ s^{(\sigma)} \end{bmatrix}
$$

Writing $\boldsymbol{p}_t$, $\boldsymbol{e}_t$ and $\boldsymbol{s}_t$ for the terms above evaluated at time t, the policy update rule is :

$$
\boldsymbol{p}_{t+1}(\boldsymbol{x}) = \boldsymbol{p}_t(\boldsymbol{x}) + \beta\varepsilon_{t+1}\boldsymbol{s}_t(\boldsymbol{x}) \tag{3}
$$

$$
\begin{aligned}
\text{where} \quad \boldsymbol{s}_t(\boldsymbol{x}) &= \boldsymbol{e}_t(\boldsymbol{x}) + \lambda \boldsymbol{s}_{t-1}(\boldsymbol{x}) & \text{if } \boldsymbol{x} = \boldsymbol{x}_t \\
\boldsymbol{s}_t(\boldsymbol{x}) &= \lambda \boldsymbol{s}_{t-1}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \neq \boldsymbol{x}_t
\end{aligned}
$$

This update rule allows μ to increase towards τ if $\tau > \mu$ and the return from taking action τ is better than the average action μ, as indicated by positive ε_{t+1} (and vice versa). σ is decreased if $|\tau - \mu| < \sigma$ and ε_{t+1} is positive, leading to convergence towards a locally optimum action. The converse of this causes σ to increase and allows more exploratory behaviour.



(a) (b)

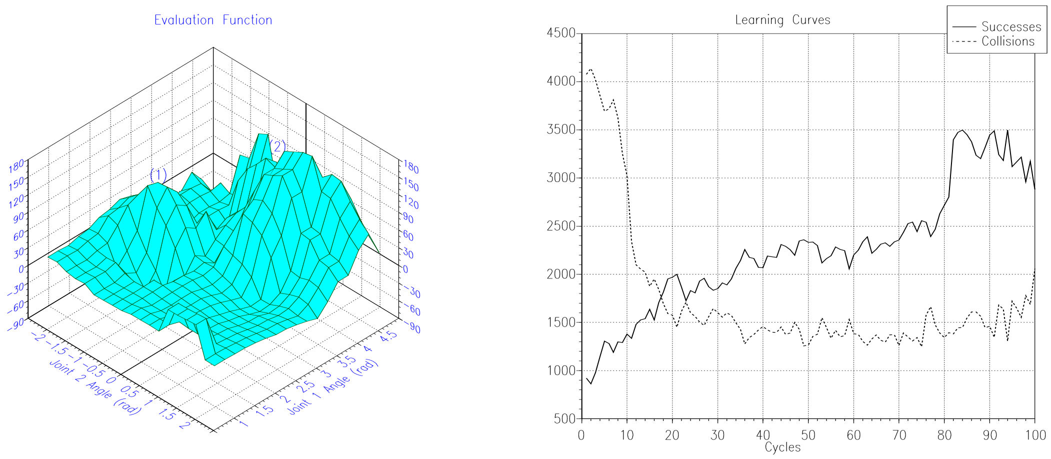
Figure 2: (a) A path taken by the robot to reach the destination. (b) Graphs of expected return, torques τ_1 and τ_2, and velocities $\dot{\theta}_1$ and $\dot{\theta}_2$ along the path.

7 Reinforcement Structure

The learning ability of the agent depends to a large extent on the structure of the reinforcement signal. Since collisions occur more frequently than successes, especially in the early stages of learning, the negative reinforcement that comes with collisions cannot be too large as to overwhelm any positive reinforcement previously acquired when it reached the destination. However, insufficient negative reinforcement decreases the agent's ability to avoid obstacles.

We implemented the following structure, where r is the reinforcement :

1. If the destination is reached, $r = +50$.
2. If a collision occurs, $r = -5\times \mid \dot{\theta} \mid$ of the link involved, subject to the constraint that the maximum negative reinforcement is $r = -50$.
3. If both $\dot{\theta}_1$ and $\dot{\theta}_2$ are greater than 10 rad/s, $r = -10$. This is to indicate that excessively high velocities are undesirable.



(a) (b)

Figure 3: (a) Evaluation function surface at $\dot{\theta}_1 = 0.5$ rad/s, $\dot{\theta}_2 = 0.5$ rad/s. (b) Learning curves.

8 Results

The following values were used for the agent's parameters : $\alpha = 0.1$, $\beta = 0.1$, $\lambda = 0.5$ and $\gamma = 0.95$. We set $V(\boldsymbol{x})$ to the maximum reward attainable and $\sigma = 10$ for all states at the beginning so that the agent starts with exploratory behaviour and does not get stuck at local maxima in the early stages. μ is set to zero throughout so as not to bias the direction of movement one way or the other. Since the stimulus traces decay rapidly, we need only to maintain a queue of 5 previous states and update the function values of these states.

Training is conducted in cycles of 5000 trials where each trial consists of at most 100 steps. If the destination is reached or a collision occurs, the current trial ends and the next one begins with the robot starting from a random position in space.

Figure 2 (a) shows the ability of the robot to move from a start position on the right to the destination after 84 cycles of training. This task requires a downward motion followed by an upward motion of link 1, with link 2 moving appropriately to avoid collisions and finally to reach the destination. The graphs in Figure 2 (b) show the variation in expected return, torques and angular velocities during the 81-step sequence. The torques generated by the agent lead to a steady increase in expected return but are oscillatory. The resulting velocities are within ± 10 rad/s.

There are two possible robot configurations in which the end-effector reaches the destination, i.e. two solutions to the inverse kinematics. These correspond to two peaks in the evaluation function surface in Figure 3(a) as indicated by (1) and (2) in the graph.

Figure 3(b) shows how the performance of the agent improves with training in terms of the increasing number of successes and decreasing number of collisions in each cycle. The best performance is achieved after 84 cycles, after which performance deteriorates slightly.

9 Conclusion

We have demonstrated how reinforcement learning can be used in an *integrated* method for controlling a multi-linked manipulator.

We note that our method does not find a collision free path in every situation; even when a collision free path is found, it may not be the optimal path. As the learning algorithm performs stochastic hillclimbing, the agent is not immune to being stuck at local maxima in some parts of the state space and a sub-optimal policy .

Another point of concern is the oscillatory nature of the torques produced which wastes energy and can cause 'chattering' in a real physical system. This can be reduced by awarding additional reinforcement based on the the minimization of a quadratic measure of performance involving the rate of change of torques (Uno et al., 1987).

Appendix

The following are the equations of motion used in the robot simulation described in Section 3 :

$$\begin{aligned}
\ddot{\theta}_1 &= \frac{m_{22}[\tau_1+n_1\sin(\theta_1)+n_2\sin(\theta_1-\theta_2)+2m_{12}\dot{\theta}_2^2\sin(\theta_2)]-m_{12}\cos(\theta_2)[\tau_2-n_2\sin(\theta_1-\theta_2)]}{2[m_{11}m_{22}-(m_{12}\cos(\theta_2))^2]} \\
\ddot{\theta}_2 &= \frac{m_{12}\cos(\theta_2)[\tau_1+n_1\sin(\theta_1)+n_2\sin(\theta_1-\theta_2)+2m_{12}\theta_2^2\sin(\theta_2)]-m_{11}[\tau_2-n_2\sin(\theta_1-\theta_2)]}{2[(m_{12}\cos(\theta_2))^2-m_{11}m_{22}]}
\end{aligned}$$

where

$$\begin{aligned}
m_{11} &= \tfrac{1}{6}m_1l_1^2 + \tfrac{1}{2}m_2l_1^2 + \tfrac{1}{2}m_3l_1^2 + \tfrac{1}{2}m_4l_1^2 \\
m_{12} &= \tfrac{1}{2}(\tfrac{1}{2}m_2l_1l_2 - m_4l_1l_3 + m_3l_1l_2) \\
m_{22} &= \tfrac{1}{6}m_2l_2^2 + \tfrac{1}{2}m_3l_2^2 + \tfrac{1}{2}m_4l_3^2 \\
n_1 &= l_1g(\tfrac{1}{2}m_1 + m_2 + m_3 + m_4) \\
n_2 &= l_2g(\tfrac{1}{2}m_2 + m_3) - m_4l_3g
\end{aligned}$$

$$\begin{aligned}
m_1 &= 1.6\,\mathrm{kg} & l_1\ (\text{A-B}) &= 0.23\,\mathrm{m} \\
m_2 &= 0.6\,\mathrm{kg} & l_2\ (\text{B-3}) &= 0.25\,\mathrm{m} \\
m_3 &= 0.3\,\mathrm{kg} & l_3\ (\text{B-4}) &= 0.05\,\mathrm{m} \\
m_4 &= 0.5\,\mathrm{kg} & g &= 9.81\,\mathrm{m/s^2}
\end{aligned}$$

References

Albus, J. S. (1975). A new approach to manipulator control : The cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement and Control*, 97(3):220–227.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13(5):835–846.

Barto, A. G., Sutton, R. S., and Watkins, C. J. C. H. (1989). Learning and sequential decision making. Technical Report COINS 89–95, Dept. of Computer and Information Science, University of Massachusetts, Amherst.

Nguyen, D. and Widrow, B. (1989). The truck backer-upper : An example of self-learning in neural networks. In *International Joint Conference on Neural Networks.*

Prescott, T. J. and Mayhew, J. E. W. (1991). Obstacle avoidance through reinforcement learning. In Moody, J., Hanson, S. J., and Lippman, R. P., editors, *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann, San Mateo, CA.

Sutton, R. S. (1987). Learning to predict by the methods of temporal differences. Technical Report TR 87–509.1, GTE Laboratories Inc.

Uno, Y., Kawato, M., and Suzuki, R. (1987). Formation of optimum trajectory in control of arm movement: minimum torque change criterion. Technical Report MBE86-79, Japan IEICE.

Williams, R. J. (1988). Toward a theory of reinforcement-learning connectionist systems. Technical Report NU-CCS-88-3, College of Computer Science, Northeastern University, Boston, MA.