ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 333-338

Recurrent Neural Networks and Motor Programs

Paul I. Miller

Centre for Cognitive and Computational Neuroscience
University of Stirling, Scotland
p.i.miller@psych.stir.ac.uk

Abstract. It is well known that biological control systems have a hierarchical design, consisting of brain, central pattern generators (CPGs) and actuators. The presence of the CPG means that descending control signals from the brain tend to be of a simple form. In this paper, this general design principle is implemented in a neural network framework. CPGs are implemented as fully recurrent nets, with Distal Learning used to estimate error derivatives for a controller. This ideas are tested on a simulated robotic arm.

1. Introduction

The concept of a *Central Pattern Generator* (CPG) has been very important in understanding biological control systems. CPGs are largely autonomous systems that are capable of producing complex behaviours in the absence of external inputs, although they are usually able to take advantage of any external inputs (say from proprioceptors) to shape their behaviour to suit the task. (Robertson and Pearson, 1985; Cohen et al, 1988). It is the interaction of the basic behaviour of the CPG, and the sensory inputs that produces the final motor output. Neurotransmitters have also been shown to be widely used in modifying the behaviour of simple motor systems; the role of neurotransmitters is often to choose between several discrete modes of behaviour.

One way to interpret the role of CPG's is that of representing prior knowledge for a controller: the presence of a CPG simplifies the control problem. In extreme cases, all the complexity goes into the CPG, and the role of the controller is reduced to a non-specific 'go' signal.

In contrast, most neural net approaches to adaptive control have used controllers with many degrees of freedom, with learning starting from scratch. In this paper, the effect of adding simple CPG circuits to neural network controllers is studied. It is shown that judicious choice of CPG circuits can improve the performance and stability of learning for simple motor tasks.

2. Backpropagation Reconsidered

Consider a network as producing a parameterised function of it's inputs:

$$\mathbf{u} = g(\mathbf{w}, \mathbf{q}).$$

where $\mathbf{q}$ is an input vector, and $\mathbf{w}$ the network weights. Backpropagation is a gradient-descent method that minimises error functions like:

$$J = \sum_j (\mathbf{u} - \mathbf{u}^*)^2;$$

assuming target values $\mathbf{u}^*$ are known. Backpropagation calculates error derivatives $\nabla_{\mathbf{w}} E$; in addition, derivatives like $\nabla_{\mathbf{q}} J$ are calculated as part of the backpropagation procedure. This gives a way of optimising the *inputs* to a net; this is used in Distal Learning (Jordan 1990) to transform derivatives in one space to those in another. Similarly, recurrent backprop can be used to optimise the inputs to a recurrent net.

In Jordan 1990, Distal Learning was used to learn control when the only available targets are in the space of the plant output. A forward plant model is used to transform error derivatives back into the space of the controllers output. It is possible to extend these ideas; the transforming system can be an arbitrary dynamic system, and backpropagating through such a system will have very different effects, depending its dynamics. This provides a very simple mechanism for introducing motor programs into a control system.

3. Recurrent Networks as Controllers

Recurrent neural networks are a very general class of non-linear dynamic systems, and several gradient-descent based procedures have been devised for their optimisation (Williams & Zipser, 1989; Pearlmutter, 1990).

In this paper, we will consider controllers that are fully recurrent, asymmetric networks. Each unit evolves according to

$$T_i\, \dot{x}_i = -x_i + \sigma\left[\sum_j w_{ij} x_j\right] + I_i(t);$$

where σ is taken to be the usual sigmoid function, and $I_i(t)$ is an external input. Time-Dependent Recurrent Learning (Pearlmutter, 1990) was used to train the recurrent nets in this paper.

Following Jordan, a static planar arm is used as the plant, and the tasks involve requiring the endpoint of the arm to touch buttons in the plane at certain times. The simulated arm has five degrees of freedom, two positional, and three joints. Two of the controller's outputs specify the position of the shoulder; the remaining three give the three joint angles. A feedforward network with 20 hidden units was used to approximate the behaviour of the plant;

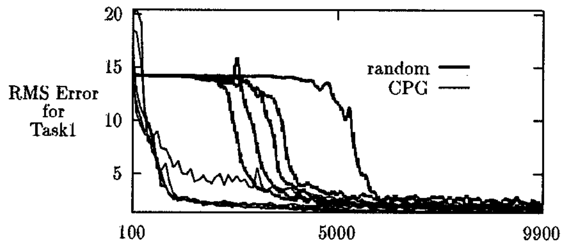

Figure 1: Learning curves for Task1

the network was optimised to a reasonable degree of accuracy before the controller was adapted. Backpropagating through the network transforms target endpoint positions into target for the controller's output.

Given that the targets are only specified for the endpoint, the controller's task is ill-defined. There are two components to this redundancy: firstly, fixing the endpoint of the arm only partly specifies the state of the arm, secondly, targets are only specified at certain points in the trajectory, with the arm unconstrained at all other times. Jordan uses a regulariser that penalises large movements from one time step to the next; the implicit smoothness of the dynamics used here achieves the same effect.

Task 1:

In the first task, the controller is required to press four buttons in the correct order, but at a specific speed chosen randomly from a continuous range, which is specified by one input to the controller.

Task 2:

In this task, the controller is required to press the same buttons as before, but the position of the buttons is continuously varied, and coded by two external inputs, one for each co-ordinate.

Recurrent nets with 14 units were initialised with random weights, and trained on the two tasks. The training times are rather long, suggesting that these are hard tasks for a recurrent net to learn (figs 1 and 2, plots marked 'random').

Because recurrent backpropagation consists of changing the parameters of a non-linear dynamic system, it might be expected that stability would be a problem, and this turns out to be the case. Doya (1993) showed that recurrent nets can go through bifurcations during learning, which is a complication for a gradient descent procedure. If these bifurcations could be avoided, we would expect the stability of the learning to improve. It is precisely because a recurrent net is such a general purpose system that these bifurcations arise: if the network was restricted to producing qualitatively correct behaviour, the learning should improve. This is a possible role for the CPG's described earlier. The approach taken here is represent CPG's as recurrent networks; these nets

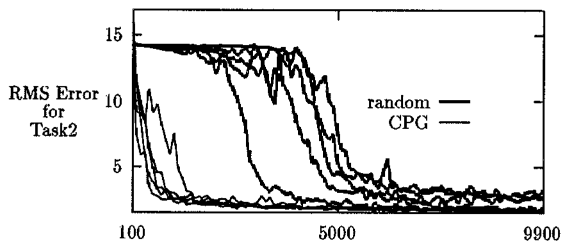

Figure 2: Learning curves for Task 2

are placed in series between the controller and the plant. In just the same way as a plant model was used to transform error derivatives from the space of the plant's outputs to that of the plant's inputs, we can backpropagate through the CPG to give error derivatives for the controller. It is also sensible to allow the controller to observe the state of the CPG, so the activations of all of the CPG's units is given as input to the controller, in addition to any other inputs.

The main problem is finding the best CPG for the task. One possible method is to learn a simpler task, and use the controller for that task as the CPG. If learning the simpler task involves solving a subproblem of the complex task, then a degree of transfer would be expected.

One obvious choice for a transfer task would be to touch the buttons at constant speed, and with constant position. A controller was trained on this task until the performance was judged to be sufficiently good, and the resulting controller used as a CPG for the two tasks. For each task, the controller was a randomly initialised recurrent net coupled to the CPG as described, and the weights of the CPG were fixed during learning. The size of the controller and the CPG net were chosen so that the combined system had the same number of units (and weights) as the randomly initialised controllers used before (ie the controller had six units; the CPG eight).

The resulting learning curves show that there is indeed a transfer effect (figs 1 and 2, plots marked 'CPG'). The learning proceeds much more quickly than with a randomly initialised controller. Although stability of high-dimensional systems is hard to measure, we can show that the stability of has been improved, as the learning rate for the controller can be greatly increased (to around 0.9), and learning still converges.

Are these CPG nets optimal for this task? One way to answer this is to use a genetic algorithm to build CPG nets, and select them on their ability to help the learning of the two tasks. There are a few problems with this: firstly, this will be a rather noisy evaluation function, as it is dependent on the initial weights of the controller. More seriously, as the string codes directly for the values of the weights, a genetic algorithm would be expected to have problems in using crossover effectively. This is because crossover is only an effective operator when

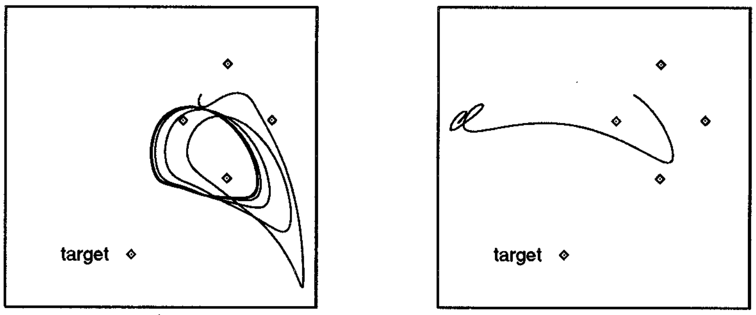

Figure 3: Trajectories for the arm endpoint produced by two evolved CPGs.

the closeness of parameters on the string is related to their degree of coupling, with nearby string elements coding for highly coupled parameters. There are other problems too, such as the fact that all the units in a net can be re-labelled, and the behaviour of the net will be unchanged, but the genetic representation will be radically different. Despite all the theoretical problems, crossover has been found to be an effective operator in these problems (Beer & Gallagher, 1992), and a reasonable crossover rate of 0.6 per string was used in these experiments. The weights and time constants were gray-encoded, with 10 bits allocated to each parameter, and the ranges for the parameters were fairly large: [-16, 16] for the weights, and [0.5, 5.5] for the time constants. The population consisted of 100 individuals, and the GA was run for 100 generations. Each evaluation consisted of 400 runs of recurrent backprop, with the learning rates of the controller set fairly high (0.7), as structures that give stable learning should be selected. Five runs were performed for each of Task 1 and Task 2.

There are several interesting results of this experiment. Firstly, the structures found by the GA are divided between CPG's that are oscillators, and those that cannot produce stable oscillation in the absence of external inputs (they are near oscillators - see fig3). Secondly, the CPG's that are oscillators have been evolved rather quickly - much more quickly than a GA would be expected to evolve oscillators without any learning. This is an example of the Baldwin effect: the presence of learning in the evaluation cycle interacts with the evolution, and increases the speed of the optimisation.

Finally, the CPG's that correspond to stable oscillators are not the best for the task. This can be seen by picking the best oscillator CPG, and the best near oscillator CPG, and testing them with random controllers. The near oscillator CPG consistently outperforms the oscillator. This is very much in line with the work of Schoner et al (1992). Loss of stability is seen to correspond to pattern change, which suggests that systems have to lose stability to have their behaviour modified. There is clearly a tradeoff between stability and controlability.

4. Discussion and Future Research

A method has been discussed for representing simple motor programs with neural networks. This is closely related to the notion of 'shaping' a controller; one important difference is that the CPGs may have a variety of sources, not necessarily the current learning task. For example, knowledge of biological systems could guide the design of a CPG. The presence of such a net can improve the performance and stability of learning, at least for the simple tasks used here. One problem with the approach presented here is that it requires a plant model. For complex plants, forming an accurate model may be unrealistic and unnecessary. One interesting alternative would be to directly learn mappings from control inputs to behavioural consequences, and future work will concentrate on finding CPG circuits that simplify such mappings. For example, a possible role for a CPG net would be to reduce to complexity of such a mapping; say from a dynamic relation to a static function. A nice example of this idea is the work of Taga *et al*(1991). The problem addressed there is bipedal locomotion, and the proposed control system is a relatively simple set of coupled oscillators, together with plausible reflex feedback loops. It is shown that settings of the parameters exist that produce stable walking gaits in the simulated system, and that the speed of locomotion is easily controlled by one input variable. Nowhere in the control system is a model of the plant, but there is a simple mapping from control inputs to behavioural consequences.

References

R. D. Beer & J. C. Gallagher (1992) Evolving dynamical neural networks for adaptive behaviour *Adaptive Behaviour* **1** (1); 91-122

A. H. Cohen, S. Rossignol & S. Grillner (1988) (eds), *Neural Control of Rhythmic Movements in Vertebrates.* New York: Wiley.

K. Doya (1993) Bifurcations of Recurrent Neural Networks in Gradient Descent Learning *submitted to IEEE Transactions on Neural Networks*

M. I. Jordan & D. E. Rumelhart (1990) Forward Models: Supervised Learning with a Distal Teacher. *submitted to Cognitive Science.*

B. A. Pearlmutter (1990) Learning State Space Trajectories in Recurrent Neural Networks. *Neural Computation* **1**(3):263-269.

R. M. Robertson & K. G. Pearson (1985) Neural Networks Controlling Locomotion in Locusts. In A. Selverston (ed), *Model Neural Networks and Behaviour*, 21-35. New York: Plenum Press.

G. Schoner, P. Zanone & JAS. Kelso (1992) Learning as Change of Coordination Dynamics: Theory and Experiment. *Journal of Motor Behaviour* **24**: 29-48

G. Taga, Y. Yamaguci & H. Simizu (1991) Self-organised Control of Bipedal Locomotion by Neural Oscillators in Unpredictable environment *Biological Cybernetics* **65**; 147-159.

R. J. Williams & D. Zipser (1989) A Learning Algorithm for Continuously Running Fully Recurrent Neural Networks. *Neural Computation* **1**(3):270-280.