NEURAL NETWORK CONTROL SYSTEMS THAT LEARN TO PERFORM APPROPRIATELY

JOHN A. BULLINARIA & PATRICIA M. RIDDELL
Department of Psychology, The University of Reading, Reading, RG6 6AL, UK
j.bullinaria@physics.org , p.m.riddell@reading.ac.uk

Setting up a neural network with a learning algorithm that determines how it can best operate is an efficient way to formulate control systems for many engineering applications, and is often much more feasible than direct programming. This paper examines three important aspects of this approach: the details of the cost function that is used with the gradient descent learning algorithm, how the resulting system depends on the initial pre-learning connection weights, and how the resulting system depends on the pattern of learning rates chosen for the different components of the system. We explore these issues by explicit simulations of a toy model that is a simplified abstraction of part of the human oculomotor control system. This allows us to compare our system with that produced by human evolution and development. We can then go on to consider how we might improve on the human system and apply what we have learnt to control systems that have no human analogue.

1. Introduction

In recent years, increasingly sophisticated techniques have become available for formulating control systems and this is allowing these systems to become more and more 'intelligent'. Powerful optimization techniques are already commonly used to adjust the parameters in standard control models to satisfy the required constraints such as limits on the acceptable response times, overshoots, and so on. Often, however, the system's requirements are significantly more complex than this. Consider a typical human control system. To see objects clearly at different distances, our oculomotor control system must produce appropriate eye rotations (vergence) and focus changes (accommodation). These responses are correlated and driven by a range of different cues (blur, disparity, looming, parallax, texture, etc.) that vary in accuracy, reliability and availability across different viewing conditions. Transitions between targets at different locations in the visual field need to be as quick as possible whilst minimizing the overshoots and oscillations that can arise from discontinuous requirements and time delayed feedback. The system must also deal with numerous age dependent factors, such as the disparity cue not becoming available until about four months of age, and the ability to accommodate falling steadily between the end of childhood and about sixty years of age. It also needs to adapt on several timescales, initially to compensate for maturational factors such as the growing inter-pupillary distance, later to correct for eye damage (or the use of new spectacles), and also to reduce the strain under constant conditions such as repeated near work. To program such a system would be a formidable task, yet evolution has resulted in an oculomotor control system that learns efficiently to organise itself to perform appropriately. Many real-world engineering control systems have analogous operating requirements of equal, or greater, complexity [1]. It makes good sense to explore human-like solutions for these systems, since they may prove superior to the optimization techniques currently in use.

Several engineering style control models based on empirical human data already provide a good account of the performance of the adult oculomotor control system for unpredictable target sequences [2]. Neural network models loosely based on the human brain have an advantage over these systems in that, rather than being set up by hand to simulate adult performance, they are set up to *learn* to perform the given task as best they can [3, 4]. Moreover, if we require that the adult system be able to adapt, for example to deal better with changing operating conditions or degradation in performance of the plant, it is an important additional advantage to have it stable with respect to learning at the outset of its adaptation. Such neural models are refined by repeated comparisons against human data of: (1) their pattern of learning, (2) their final performance, and (3) their ability to adapt. These comparisons all provide important clues on how to set up our systems to learn most effectively. Naturally, our artificial neural networks are vast over-simplifications of the biological neural systems that are found in the human brain, but nevertheless, these simple models can account for a surprisingly wide range of human abilities. Moreover, they are powerful computational devices which can learn to perform

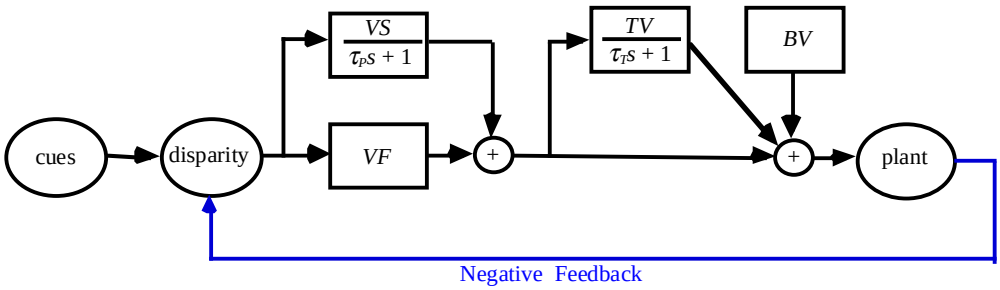

Figure 1: Traditional simplified linear control systems model of the human vergence system.

exceedingly well, and it would be foolish not to consider their use in engineering applications. We can also go on to explore how we might improve on the human system for analogous engineering control systems, and carry what we have learnt over to other control systems that have no human analogue.

In principle, the engineering system building process is straightforward when a corresponding human system exists. The basic neural network architecture is given by known physiology and/or the existing control models. We know what the network is meant to be learning to do, for example, minimizing blur and disparity in the case of oculomotor control. So we 'simply' need to use some form of gradient descent based connection weight learning to minimize an appropriate cost function, and then compare the resulting network performance with empirical developmental and adult data.

In practice, determining some of the crucial details is not so straightforward. The first problem we face is to choose the cost function for learning. For oculomotor control, some function of blur and disparity is a fairly obvious error component, but making an appropriate choice for the regularization (i.e. smoothing) component is not so straightforward, and we will need to have a convenient parameterization of the trade-off between them. The second complication is the need to determine the effect of using different starting points for the learning process, as it is possible that a wrong choice will lead to a far from optimal local minimum of the cost function, or result in the system taking an unacceptably long time to find any minimum at all [5]. A third, related, question is the effect of choosing different learning rates for the various system components, as a bad choice here can have similar consequences to picking a wrong starting point. The starting point and learning rate differences may either be explicit features of the model, or implicit in the choice of the model's parameterization. This paper explores the consequences of these three design choices by explicit simulation of a series of neural network control models that have been appropriately simplified to retain the crucial features, whilst freed of any application specific complications. We shall see that our conclusions will be applicable to neural network control systems more generally.

2. The Neural Network Model

The study to be presented in this paper is based on a toy model derived from an abstraction of the human oculomotor control system. The control systems for vergence and accommodation are actually of a very similar form [2, 4, 6], so for clarity of analysis we shall concentrate on a simplified version of the vergence system. Figure 1 shows the structure of what has become the traditional engineering style linear control model of this system [2, 6, 7]. The system's plant output is the vergence response (i.e. the difference between the directions of the two eyes) which gets fed back with a time lag to the disparity sub-system which calculates the output error relative to the system's input (i.e. the appropriate vergence for the visual target). This error signal is passed through proportional and integral controllers with gains/strengths *VS* and *VF* to activate the plant. A leaky integrator tonic component of strength *TV* reduces the strain under conditions of constant demand, and a bias signal of strength *BV* maintains a convenient resting state.

The neural network modelling approach begins by formulating a natural neural architecture of sufficient generality that the existing traditional control systems models can arise from it as a special case. This guarantees that the neural system will be able to perform at least as well as the existing models, though getting the network to learn how to do this for itself will not necessarily be an easy task. Figure 2 shows the simplest natural neural network extension of the vergence model of Figure 1. Each "neuron" and the plant in the network model are leaky integrators with the same empirically determined time constants τ as the traditional model. From a physiological point of view, the "neurons" represent the action of assemblies of real neurons, and the plant is a good approximation to the human ocular muscles [7]. The system processes information by

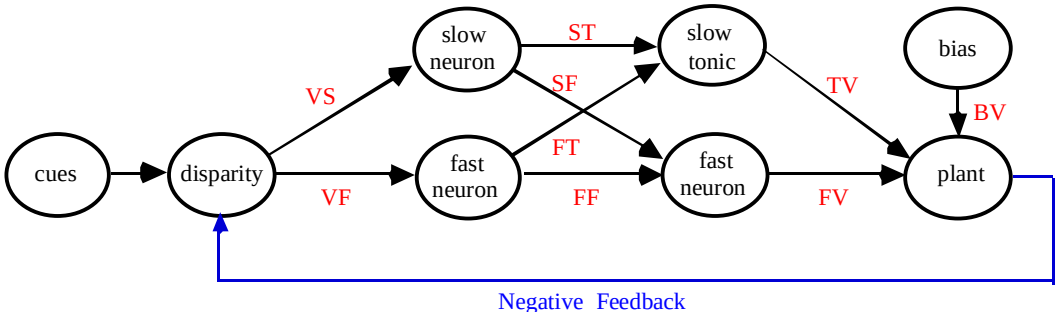

Figure 2: Simplified neural network model of the vergence system with learnable connection weights.

having activation flow between the components via the weighted connections shown. The direct correspondence with the equivalent engineering style control model is easy to see. The first pair of fast and slow neurons correspond to the proportional and integral components of the standard phasic sub-system (τ_P = 5s), the slow tonic and bias correspond to the standard tonic sub-system (τ_T = 20s), and the plant corresponds to the usual first-order control model plant (τ_E = 0.15s). In a complete oculomotor control model, there are thought to be additional cross-links to and from the accommodation system, that connect between the phasic and tonic sub-systems [2, 4, 6], but we shall ignore such complications for the current study. The system's plant output is the vergence response which gets fed back with a time lag of 0.15s to the disparity sub-system which calculates the output error signal that is fed through the network. Obviously, in the real system, this feedback loop will be external to the system, but for the purposes of simulation we have a fully dynamical recurrent neural network.

The traditional control systems models are linear. If our neurons are also linear over their operating ranges, we can use the standard rules of matrix multiplication to simplify our network to a single layer of learnable weights. In other words, we can set four of the nine connection parameters shown to convenient values without loss of generality. We therefore normalise the four weights $VS = VF = FV = TV = 1$, leaving just the middle layer of weights and the bias to be learnt. Comparing the network of Figure 2 with the traditional model of Figure 1, reveals that exact correspondence is achieved by setting the five weights $SF = ST = FF = FT = FV = 1$. Some simple algebra then shows that we have direct equivalence between our chosen neural network parameterisation and the traditional control systems models if our network's remaining free parameters *SF*, *ST*, *FF*, *FT* and *BV* satisfy $ST.FF = SF.FT$.

As noted above, an important feature of our neural network systems is that, rather than setting their free parameters by hand so that their performance matches that of human adults, we allow them to *learn* the parameter values that enable them to perform as best they can. This learning is achieved by carrying out gradient descent on an appropriate cost function. What gets learnt can potentially depend on the learning algorithm, the cost function, and the initial conditions. In Section 3 we deal with the problem of choosing an appropriate cost function. Then in Section 4 we look at the effect of using different starting weights and different learning rates in our models. In the human system there will often also be various maturational factors, such as the quality of the vergence cues changing with age, that affect the learning process [4]. Clearly, if we can always minimize the cost function to a unique global minimum, such details of the learning algorithm, initial conditions and maturational factors will not matter, but in practice they tend to interact with each other, and we can end up in different minima. In fact, one of the original reasons for formulating these models was to understand the causes of abnormal developmental trajectories in children with a view to identifying precursors and designing remedial actions. Naturally, there are analogous considerations for engineering systems. This is why we believe it is so important to investigate these factors, and to explore the significance of the selection of the cost function when compared with these other choices that may potentially lead to even larger variations in the learnt parameters.

A final consideration of crucial importance for control systems is the question of stability and bounded outputs. The stability of traditional linear systems is usually analysed in terms of the locations of the poles in the closed loop transfer function [1, Ch 9]. One major advantage of neural network modelling, and Optimal Control Design in general, is that optimizing a suitable performance index will, under rather mild assumptions, select system variables that will automatically guarantee closed-loop stability [1, Ch 48]. In a sense, when given appropriate training data and cost function, our neural network systems will learn to be stable.

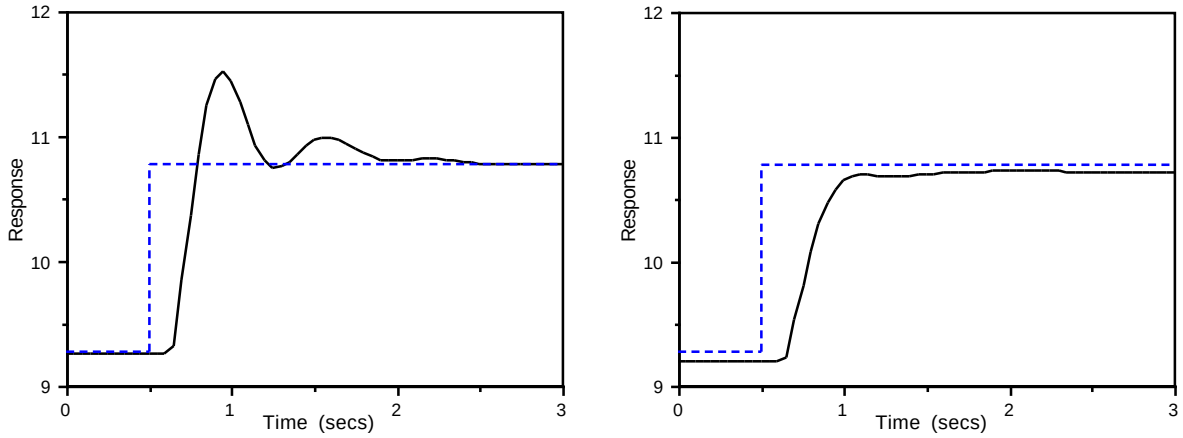

Figure 3: The neural network control system responses for (a) un-regularized and (b) regularized training.

3. Choosing the Cost Function

The standard approach to neural network learning we shall follow involves iteratively updating each free parameter (i.e. connection weight) w to perform gradient descent on some cost function E. We thus have the weight update equation

$$\Delta w = -\eta \frac{\partial E}{\partial w}$$

with learning rate η. Related algorithms, such as learning with momentum or conjugate gradient learning, are well known to be able to speed up the learning process considerably, but for the present purposes we can avoid these complications. A classic application is the standard neural network regularization approach [8] which attempts to recover a function $f(\mathbf{x})$ from a set of data points $\{(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}\}_{i=1}^N$ obtained by random sampling with noise, by minimizing the cost function

$$E[f] = \sum_{i=1}^{N} (f(\mathbf{x}_i) - y_i)^2 + \lambda \Phi[f] \ .$$

This involves a parameterized trade-off between a sum-squared error term that keeps f close to the data, and a regularization term $\Phi[f]$ that enforces some form of smoothness.

The problem of optimizing the performance of our control model might appear to be rather different, but we can actually formulate a similar solution. We have a fully dynamical feedback system with training data parameterized by the time t. The network outputs (vergence responses) $f(x(t))$ are required to match the inputs (vergence cues) $x(t)$ as closely as possible given the network architecture and the constraint that $f(x(t))$ be suitably smooth despite the time lag in the feedback loop and $x(t)$ frequently being discontinuous. Again there is a trade-off between error and smoothness components, so it makes sense to re-use the above regularization cost function with the summation over i replaced by an integral over t,

$$E[f] = \int (f(x(t)) - x(t))^2 \, dt + \lambda \Phi[f] \ .$$

Naturally, we still have to specify the details of $\Phi[f]$. In this paper we shall consider three natural forms of the regularization functional:

a) $\Phi[f] = 0$

b) $\Phi[f] = \int \left| \frac{\partial f(t)}{\partial t} \right|^m dt$

c) $\Phi[f] = \int \omega^{2m} |F(\omega)|^2 \, d\omega$, where $F(\omega) = \int f(t) \, e^{-i\omega t} \, dt$.

In practice, our simulations will involve performing discrete approximations to these integrals over finite ranges, so the simple mathematical relation between the $m = 2$ case *b* and the $m = 1$ case *c* is broken. Clearly, case *a* is identical to the $\lambda = 0$ limits of cases *b* and *c*, but it is worthy of separate consideration since it reveals the problem of output overshoots and oscillation and the need for regularization in the first place. Case *b* attempts to reduce these problems by attaching cost to the velocity of the eyes' movement as has been suggested previously [4]. Case *c* attempts to deal with the output oscillations more directly. Working with the Fourier transform $F(\omega)$ and the power $|F(\omega)|^2$ at frequency ω, allows us to penalize the high frequency components and reduce the oscillations in that way [8, 9]. The best values for the integer powers m, and the question of whether the extra computation involved in case *c* can be

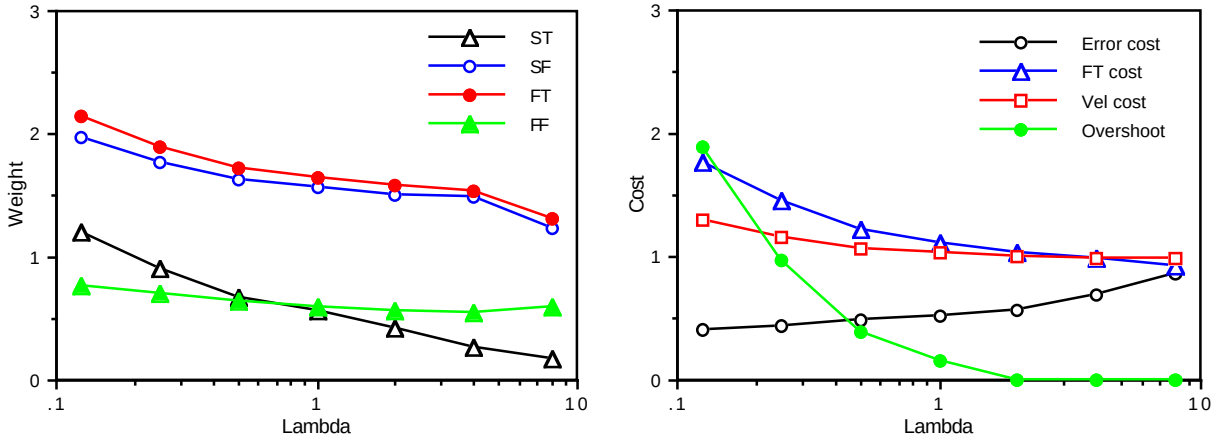

Figure 4: λ dependence of the weights and costs for an L1 error term and Velocity regularization.

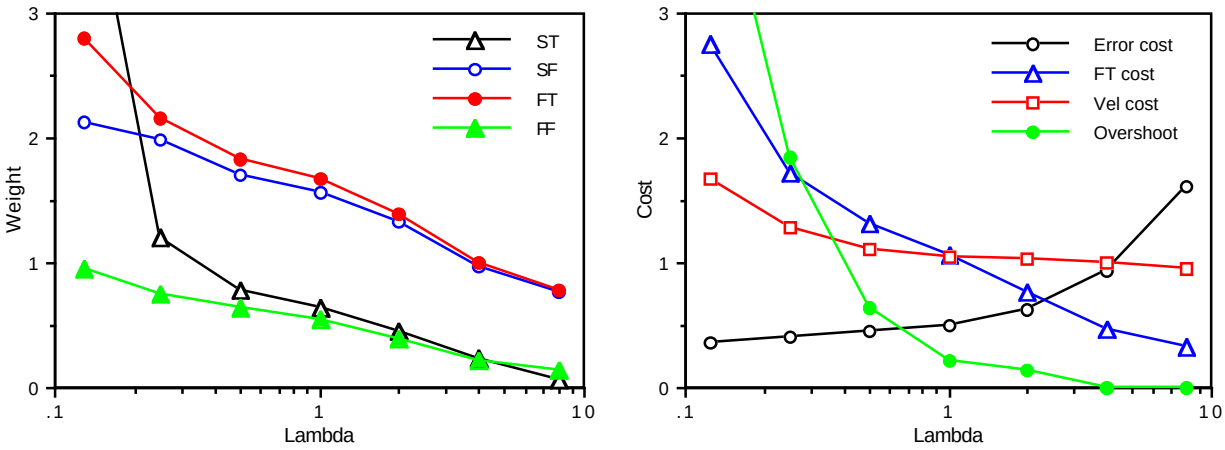

Figure 5: λ dependence of the weights and costs for an L1 error term and Fourier transform regularization.

justified by improvement in the obtained results, are two of the main factors under investigation.

The trade-off between our chosen regularization and the output error is parameterized by λ and will clearly also depend on the error function that is used. For a network with one real valued output *f(t)* the natural class of error terms to consider is

$$\int |f(x(t)) - x(t)|^n \, dt \ .$$

Taking $n = 1$ gives the integral of the standard L1 norm $|f(x(t)) - x(t)|$, and for $n = 2$ we have the traditional sum-squared error term shown above. The remainder of this Section presents explicit simulation results that explore the consequences of the $E[f]$ choices as a function of λ. Then in the following Section we investigate the extent to which their differences are significant compared with those caused by other factors, in particular, differences in the initial weights and learning rates.

Our investigation involved repeatedly training the network to asymptote on random sequences of natural vergence values, with the gradient descent weight update equation being applied in an online fashion after every 80s sub-sequence of the training data. Each training run was started from a set of small initial weights chosen from a flat random distribution in the range [0, 0.02]. The learning rate was chosen to make the training as fast as possible without allowing the random variations between blocks of training data to generate large fluctuations in the weights. The cost functions are rather insensitive to the bias *BV* and consequently its value has little effect on the discussion that follows. It does, however, tend to learn to take on a value related to the mean output level, and so, for convenience, we started each training run with *BV* near that value (namely at 2.0) and thus training had negligible effect on it.

To understand our simulation results we begin by looking at the trained networks' output responses. Figure 3a shows the model's response to a step change of input when trained without regularization (case *a*).

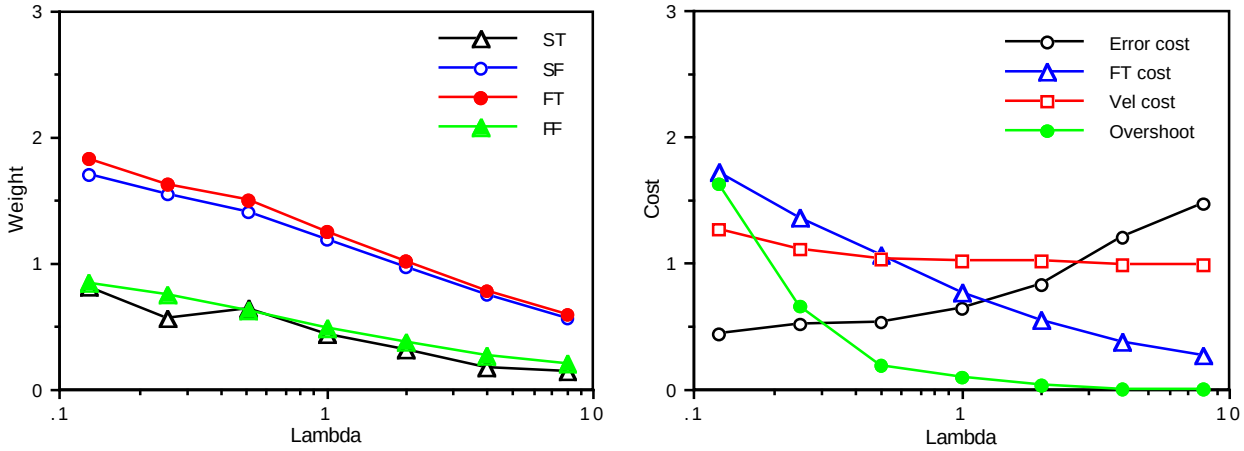

Figure 6: λ dependence for the Sum squared error term and Squared velocity regularization.

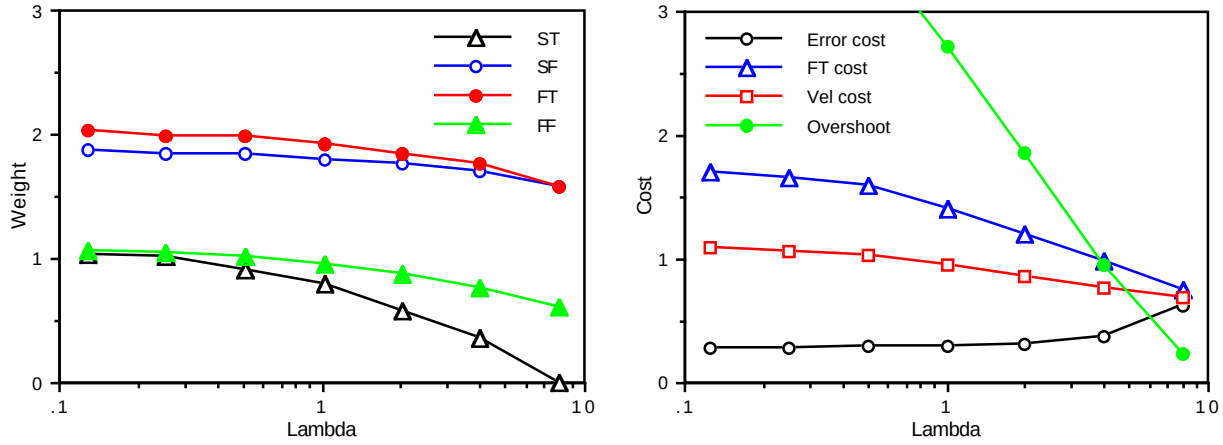

Figure 7: λ dependence for the Sum squared error term and Fourier transform regularization.

There is considerable overshoot and oscillation, which is not observed in humans. Such under-damped responses are likely to be equally unacceptable in many other control systems. Figure 3b shows the smoother and more human-like response to the same input step that is produced by an appropriately regularized model.

Determining the most appropriate regularization will clearly require a systematic study. We do this by comparing the network performance resulting from training with each of the different cost functions discussed above, across a wide range of trade-off parameters λ. To ease the comparison between the different regularization models, in each case we shall plot the L1 ($n = 1$) error and both $m = 1$ regularization costs, irrespective of the cost function actually used to train the model. We shall also plot an overshoot measure defined as the total response change (summed over oscillations) in the direction opposite to the standard step producing it.

Figure 4 shows the effect of λ on the final weights and costs when the model is trained with the L1 error term and simple velocity regularization (case *b*, $n = 1$, $m = 1$). We see that, as expected, there is a clear trade-off between error and over-shoot as we increase λ. Figure 5 shows the equivalent plots for the L1 error term with a simple Fourier transform regularization (case *c*, $n = 1$, $m = 1$). In this case we have to suffer a much larger error in order to remove the overshoot. Figure 6 shows what happens with a sum squared error term and sum squared velocity regularization (case *b*, $n = 2$, $m = 2$). The error required for zero overshoot is larger again. Finally, Figure 7 shows that no better results are obtained using the sum squared error term with simple Fourier transform regularization (case *c*, $n = 2$, $m = 1$). The remaining permutations and values of m and n are found to perform even less well.

Taken together, Figures 4 to 7 show that, whilst an increase in trade-off parameter λ for any cost function results in the expected reduction in each of the velocity cost, Fourier transform cost and overshoot at the expense

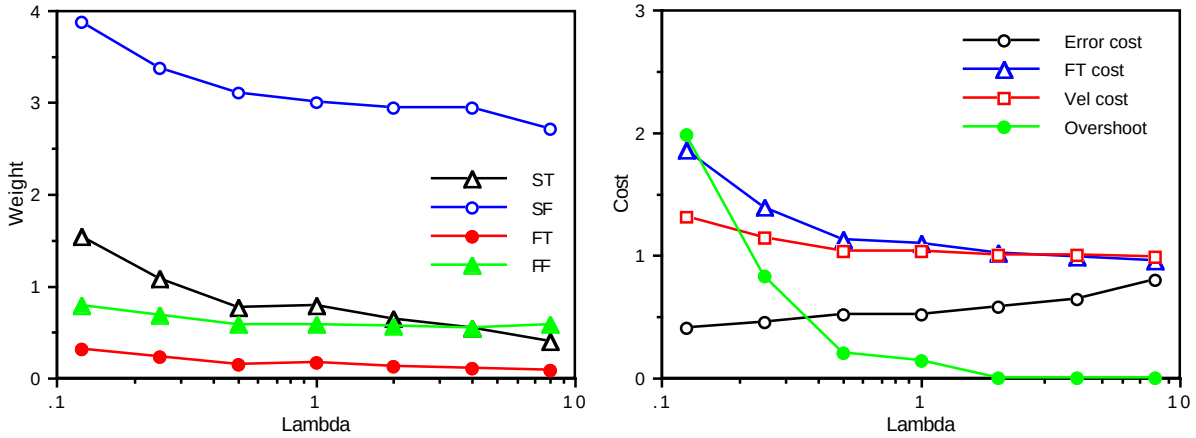

Figure 8: λ dependence for the traditionally parameterised version of the network used for Figure 4.

of increased error, there is considerable variation between cases. Finding a convenient cost function that results in responses with human-like smoothness is not enough to justify a claim that we have found the unique accurate model of human performance, nor that we have found the best performing artificial system.

From an engineering systems building perspective, there is more detailed information to be learnt from our results. First, we can see the exact nature of the error versus overshoot (or regularization cost) trade-off in each case. It is good to see that it is the simplest case (shown in Figure 4) that also gives the best overall results. The choice of λ will clearly depend on the details of the system. The human oculomotor control system has evolved to allow a slight overshoot but no oscillation. Other systems may view the trade-off differently. A second important feature to note is that, as λ increases, some of the weights become relatively small, or even negative. This may make the learning very slow or unstable depending on the learning rates we use. In this case it may be a good idea to help out the learning process by using different learning rates for the different components, or by using a different pattern of starting weights. The consequences of doing this are something we needed to explore anyway.

4. Initial Weights and Learning Rates

Figures 4 to 7 reveal a recurring feature of the trained models in all cost function cases, namely the tendency for $SF \sim FT$ to be several times $ST \sim FF$. This is very different from the structure of the traditional control systems models [2, 6] which, as we noted above, are generally set up such that in our notation $ST.FF = SF.FT$. For comparison, Figure 8 shows what happens when the models of Figure 4 are trained from scratch under this traditional model constraint. We get a radically different pattern of final weights, yet the output response curves and costs are hardly distinguishable. Moreover, if we remove the constraint on the weights and continue the training, we find that the weights are stable, suggesting that we have at least two roughly equivalent cost function minima widely separated in weight space.

This confirms our suspicion that the details of our trained networks depended on their initial conditions [5]. We consequently carried out a systematic study of the effect of starting the training with different initial weights. There are clearly many combinations we had to consider, and we shall present just one representative example that will suffice to reveal the extent of this effect. Figure 9 shows how the learnt weights (for the L1 error and simple velocity regularization case with $\lambda = 2$) change as we vary the initial value of the weight *SF* whilst keeping the initial values of the other weights *ST*, *FT* and *FF* near zero. We can now see that the results from Figures 4 and 8 are just two points on a continuum of stable solutions that are allowed by a trade-off of contributions from the similar *SF* and *FT* pathways.

A related factor that will also affect the final weights is the learning rates. There is no fundamental reason why each weight should learn at the same rate. Actually, in Laplace transform notation [1, Ch 6], there is a natural ambiguity whether the neurons should be represented as

$$\frac{1}{\tau s+1} \quad \text{or} \quad \frac{1}{s+\frac{1}{\tau}} = \frac{\tau}{\tau s+1}\,.$$

Clearly, the multiplying gains/weights can easily be adjusted to compensate for the additional factor of τ. If the gains/weights are fixed by hand this makes no difference, but in our networks that learn by gradient descent, the extra factor scales the weights and hence the learning rates. We clearly need to check whether simple alternative choices of parameterization like this can have

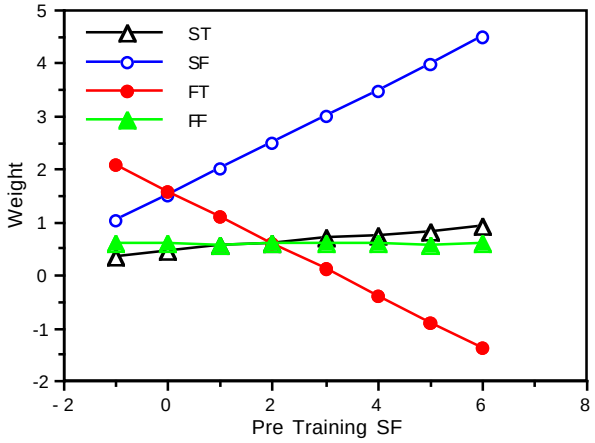

Figure 9: Dependence on the starting value of *SF*.

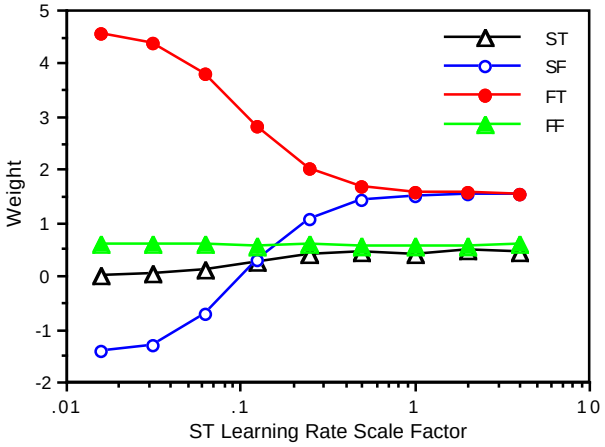

Figure 10: Dependence on the *ST* learning rate.

a significant effect on the trained networks. In fact, we need to check the effect of having different learning rates more generally. Again there are many combinations we had to consider, and we shall present just one representative example to illustrate what can happen. Figure 10 shows how the learnt weights (again for the L1 error and simple velocity regularization case with $\lambda = 2$) change as we vary a scale factor multiplying only the learning rate of the weight *ST*. Throughout, we start the training from random near zero initial weights. We see that a small scale factor results in a smaller final value of the weight *ST*, and this has a big effect on the trade-off between the weights *SF* and *FT*, but little effect on the error or regularization costs.

Having discovered empirically that the initial weights and learning rates do have a big effect on the final network weights, it is natural to wish to understand in an analytical fashion how this can happen, if only to within a certain degree of approximation. In fact, in the approximation that the neurons are purely linear, the fast neurons are infinitely fast, all the leaky integrators are simulated with no loss of accuracy, and the weights are sufficiently stable despite the random online training data, it is not too difficult to study the crucial part of our networks' operation analytically. In this case the transfer function of the central portion of our network can be written in Laplace transformed notation as

$$X(s) = \frac{\tau_P}{\tau_P s+1}.SF.1 + \frac{\tau_P}{\tau_P s+1}.ST.\frac{\tau_T}{\tau_T s+1} + 1.FT.\frac{\tau_T}{\tau_T s+1} + 1.FF.1$$

where τ_P and τ_T are the time constants of the slow phasic and tonic neurons. A little elementary algebra allows this to be written in the simplified form

$$X(s) = \frac{a.s^2 + b.s + c}{(\tau_P s+1)(\tau_T s+1)}$$

in which we have defined the constants

$$a = FF.\tau_P.\tau_T$$
$$b = FF.\tau_P + FF.\tau_T + FT.\tau_P.\tau_T + SF.\tau_P.\tau_T$$
$$c = FF + SF.\tau_P + FT.\tau_T + ST.\tau_P.\tau_T$$

Clearly, if these take on the values that minimize the cost function we can work backwards to determine the under-specified weight values. This gives

$$FF = \frac{a}{\tau_P.\tau_T}$$
$$SF = \frac{-a + b.\tau_P - c.\tau_P^2}{(\tau_T - \tau_P)\tau_P^2} + \frac{\tau_P.\tau_T}{(\tau_T - \tau_P)}ST$$
$$FT = \frac{a - b.\tau_T + c.\tau_T^2}{(\tau_T - \tau_P).\tau_T^2} - \frac{\tau_P.\tau_T}{(\tau_T - \tau_P)}ST$$

If, for example, we allow *ST* to take any value, then the other weights can compensate to restore the optimal network performance. Figure 11 shows the pattern of weights this gives for values of *a*, *b* and *c* determined from the single empirical set of weights corresponding to the $\lambda = 2$ case of Figure 4 (the same L1 error and velocity regularization case used for Figures 9 and 10). Figure 12 shows the actual pattern found in our simulated networks by re-plotting Figures 9 and 10 with the weight *ST* as the independent variable. We see that our approximate analytical treatment does provide a rather good account of our networks' properties.

It is clear then, that empirically the initial weights and learning rates do have a large effect on the learnt weights, and that to a pretty good approximation the distribution of final weights span a whole space of

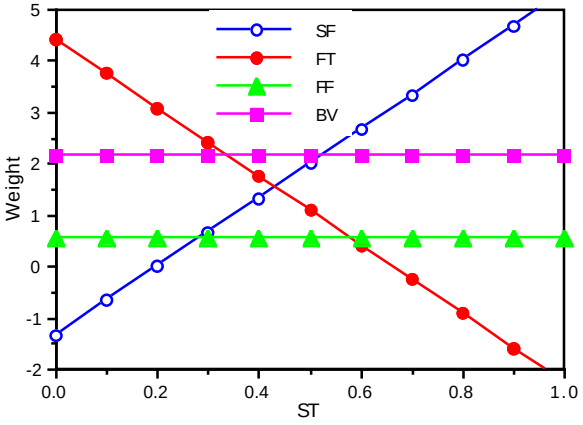

Figure 11: Analytically predicted Figure 12 weights.

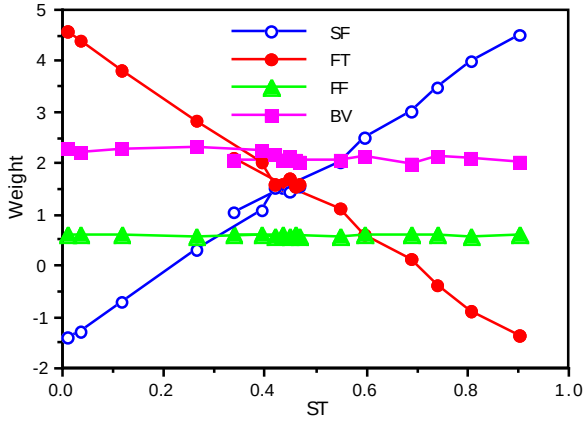

Figure 12: Weights from Figures 9 and 10 re-plotted.

mathematically equivalent network transfer functions. At this stage it is natural to ask whether there is any real need to worry about the possibility of ending up in different local minima depending on the initial weights and learning rates we use.

Perhaps the first question we should consider is whether these effects are merely artefacts of the over simplification of our model that will be absent in more complex models and in real human systems. Certainly, in more realistic models, the mathematical simplicity that allows the main pathway trade-off effect we have observed, namely that between the *SF* and *FT* routes, is likely to be broken. However, we can expect similar trade-off complications to arise in any system (either artificial or human) that is complex enough to allow more than one strategy for carrying out the given task with near optimal efficiency. If the different learnt configurations are always mathematically equivalent, then the differences can probably be ignored. However, there may be important conditions under which the differences reveal themselves and could prove crucial. Consider the human oculomotor control system again. Our preliminary investigations of models of the full accommodation and vergence system [4, 10] indicate that different initial weights and learning rates here *do* lead to different internal configurations that all perform equally well under normal conditions. However, if we interfere with these systems by removing one of the primary input cues (i.e. either blur or disparity) by opening one of the two feedback loops, we can reveal the wide range of underlying differences between the models. Empirical studies on human subjects have shown that the same kinds of individual differences are also observed in the real human system [11]. We can clearly expect a similar situation to arise in artificial engineering systems as well. It seems, therefore, that the kind of initial weight and learning rate dependencies we have observed and studied in our simplified model will continue to be important in more complex systems.

What remains for us to understand are the processes that constrain the systems and limit the ranges of individual differences that could be caused by many factors, including different initial weights and learning rates. The natural assumption is that, in biological systems, evolutionary factors will have provided the constraints and the corresponding limits [10]. There are at least two distinct factors at play here. First, evolution may favour a particular sub-set of weight configurations from the space of possibilities that give the same optimal responses, because they provide advantages over the other choices (such as robustness to noise, or loss of inputs, or changing environments) that are not included in the cost function used by the learning algorithm. Second, it is now well known that learning and evolution interact and that genetic assimilation of learnt behaviour may occur without Lamarckian inheritance [12, 13]. This process can lead to particular distributions for the initial weights and learning rates, with corresponding particular distributions for the learnt weights. This has been demonstrated explicitly in a preliminary study of the evolution by natural selection of populations of simulated simplified control models of the type studied in this paper [14]. It was shown in some detail in that study how appropriate constrained distributions of initial weights and learning rates will evolve naturally.

5. Conclusions

This paper began by presenting a systematic study of the use of different cost functions for the gradient descent training of neural network control systems. It was found that both the L1 and sum-squared error cost, with either velocity or Fourier transform regularization, all gave good, but slightly different, final network performance.

Moreover, we could see exactly how the regularization/ error trade-off parameter λ affected the network's responses in each case. We continued by investigating the degree to which starting the network training from different initial weights, and/or using different learning rates, could result in large differences to the patterns of learnt weights with very little difference to the networks' output responses. In particular, we found that the usual default of starting the training with small random initial weights, and using equal learning rates for all the weights, resulted in networks which had equivalent performance but somewhat different internal structure to that assumed in the traditional engineering style control system models [2, 6, 7]. An approximate analytical treatment of the problem revealed how we can go about understanding such effects.

From the brain modelling point of view, our study indicates that it is no longer so clear that the existing control system models should be considered as good starting points for our neural network models simply because they already provide a good account of adult human responses [4]. Rather, we should start again from known physiology, and if our models learn different structures from the existing control models, we must either find fault with the performance of those traditional models, or think more carefully about our modelling assumptions.

From the engineering systems perspective, our simulation results suggest that, even if we use near optimal values for the regularization parameters λ, m and n, factors other than the details of the cost function can have a more significant influence on the weights that are learnt. Allowing individual weights to have different starting values and/or different learning rates can result in the systems ending up with very different final weights. In humans, evolutionary factors will have placed limits on these variables, as well as the cost function. Other constraints on the weight patterns, such as might arise from innate brain layout, may also have a big effect. Whilst these complications are problematic for modelling human systems, they may actually be advantageous for artificial system building, in that they leave us with additional degrees of freedom with which we may optimize aspects of the systems' performance not included in the learning cost function. Evolution has no doubt constrained the details of the human system in a manner appropriate for its environment, such as improving the speed of learning, robustness, and such like. Efficient engineering system building is likely to come in two analogous stages – first the evolution of proficient learning systems, and then the learning of appropriate performance by those systems under the required operating conditions [10, 14]. We are left with the option of implementing other criteria for the learning systems, which can deviate from those of human evolutionary history, and may allow more effective artificial systems to be built.

References

1. Levine, W.S. (Ed.) (1996). *The Control Handbook*. Boca Raton, FL: CRC Press.
2. Eadie, A.S. & Carlin, P.J. (1995). Evolution of control system models of ocular accommodation, vergence and their interaction. *Medical & Biological Engineering & Computing*, **33**, 517-524.
3. Bullinaria, J.A., Riddell, P.M. & Rushton, S.K. (1999). Regularization in oculomotor adaptation. In *Proceedings of the European Symposium on Artificial Neural Networks*, 159-164. Brussels: D-Facto.
4. Riddell, P.M. & Bullinaria, J.A. (1999). Incorporating developmental factors into models of accommodation and vergence. Technical Report – Submitted for Publication.
5. Kolen, J.F. & Pollack, J.B. (1991). Back propagation is sensitive to initial conditions. *Complex Systems*, **4**, 269-280.
6. Schor, C.M., Alexander, J., Cormack, L. & Stevenson, S. (1992). Negative feedback control model of proximal convergence and accommodation. *Ophthalmic and Physiological Optics*, **12**, 307-318.
7. Krishnan, V.V. & Stark, L. (1983). Model of the disparity vergence system. In C.M. Schor & K.J. Ciuffreda (Eds), *Vergence Eye Movements: Basic and Clinical Aspects*, 349-371. Boston, MA: Butterworths.
8. Girosi, F., Jones, M. & Poggio, T. (1995). Regularization theory and neural network architectures. *Neural Computation*, **7**, 219-269.
9. Duchon, J. (1977). Spline minimizing rotation-invariant semi-norms in Sobelov spaces. In W. Schempp & K. Zeller (Eds), *Constructive Theory of Functions of Several Variables, Lecture Notes in Mathematics*, 571. Berlin: Springer-Verlag.
10. Bullinaria, J.A. & Riddell, P.M. (2000). Learning and evolution of control systems. *Neural Network World*, **10**, 535-544.
11. Horwood, A.M., Turner, J.E., Houston, S.M. & Riddell, P.M. (2000). Variations in accommodation and convergence responses in a naturalistic setting. *Optometry and Vision Science*, under revision.
12. Baldwin, J.M. (1896). A new factor in evolution. *The American Naturalist*, **30**, 441-451.
13. Belew, R.K. & Mitchell, M. (Eds) (1996). *Adaptive Individuals in Evolving Populations.* Reading, MA: Addison-Wesley.
14. Bullinaria, J.A. (2000). Exploring the Baldwin Effect in evolving adaptable control systems. To appear in: *Proceedings of the Sixth Neural Computation and Psychology Workshop.* London: Springer.