Controller Scheduling by Neural Networks

Enrique D. Ferreira and Bruce H. Krogh
Department of Electrical and Computer Engineering
Carnegie Mellon University
5000 Forbes Av.
Pittsburgh, PA, 15213-3890 USA
edf/krogh@ece.cmu.edu

Abstract

This paper presents a method for real-time switching among a pool of controllers to achieve desired performance. The algorithm uses a set of neural networks to evaluate the future performance of each controller to make the selection. Results are shown in simulation cases. Theoretical issues are discussed and future lines of research are stated.

1 Introduction

This paper concerns the problem of scheduling a collection of controllers to optimize performance. It is related then to gain scheduling techniques and the recently proposed supervisory control for families of controllers [7]. This paper extends previous work on using neural networks to learn stability regions for nonlinear systems [4] by incorporating performance measures and also implementing the multi-controller scheduling using the neural network monitors.

There are several motivations for this research on the use of neural networks for controller scheduling. Neural networks are very good and robust non-linear function approximators which allow us to attack real-world problems that are essentially non-linear. The ability to adapt to system changes is also important for on-line applications. Controller scheduling, mainly gain scheduling, has been used successfully for many years in practice, but the way it has been designed and applied has been almost always empirical. Neural networks offer the possibility to automate some of these empirical procedures. A final motivation comes from the possible application of the proposed scheme to a fault-tolerant system called SIMPLEX, being developed at the Software Engineering Institute at Carnegie Mellon University [13]. SIMPLEX operates within a real-time multitasking operating system to help control a complex process using a set of controllers. All of the controllers run concurrently as separate tasks, and SIMPLEX monitors the performance of the controllers and the plant to determine which controller should actually send control inputs to the plant. This monitor activity implies security and guaranteed baseline performance, provided the switching rules are correct.

The remainder of the paper is organized as follows. The general problem is formulated in section 2. The proposed neural network architecture and training techniques are described in section 3. Section 4 presents preliminary results for the application of the proposed switching scheme to the inverted pendulum demonstration problem for the SIMPLEX architecture. The concluding section summarizes the results of this paper and our future research plans.

2 Problem Formulation

Consider a state-constrained and control-constrained nonlinear system with state equations

$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}, \mathbf{u}) \\ \mathbf{x} &\in S \end{aligned} \tag{1}$$

where $\mathbf{x} \in R^n$ is the state vector and $\mathbf{u} \in R^m$ the control input vector. The nonlinear function f is assumed continuous and smooth. We also assume that the state $\mathbf{x}$ is observable. The control signal $\mathbf{u}$ to be applied on the system can be generated by

M different state feedback controllers of the form:

$$\mathbf{u}_i = g_i(\mathbf{x}), \quad i = 1, 2, \ldots, M. \tag{2}$$

The goal of the controllers is to take the system to the origin.

In this paper we are interested in designing a strategy to select which controller should be applied to the system at each sampling instant in order to achieve optimal performance in some sense (defined below).

Previous research on this problem includes the work by Morse [7] [8] on families of linear controllers and Balakrishnan and Narendra [2] on adaptive controllers. The use of neural networks in some of these architectures is also assessed in [1], but using them as system model estimators rather than for controller scheduling.

Let us denote $\mathbf{x}_i(t, t_o, \mathbf{x}_o)$ as the trajectory of the system (1) with the initial state $\mathbf{x}_o$ at time t_o resulting from the application of the state feedback control $\mathbf{u}_i$. We consider performance indices for the controllers of the form

$$J_i(\mathbf{x}_o) = \sum_{k=0}^{\infty} \delta^k U(\mathbf{x}_i(kT, 0, \mathbf{x}_o)), \tag{3}$$

where $i = 1, \ldots, M$, $\delta < 1$ is a discount factor and T is the sampling period. $U(.)$ represents the cost function for a particular state.

The proposed approach for selecting the controller at each sampling instant is illustrated in figure 1. A neural network is used to estimate the performance index evaluated at the current state and the controller corresponding to the lowest valued estimate is applied during the next sampling period. Hence, the problem is to construct the neural network estimator for each of the performance indices.

3 Neural Network approach

In this section we describe a neural network architecture and training techniques to estimate an index of performance for a given controller. The proposed approach is based on a modification of a dynamic programming procedure. Dynamic programming has a long history, but its application to real-world problems has been difficult due to the well-known "curse of dimensionality". The introduction of neural networks as approximators in the so-called neuro-dynamic programming [3] has revived the field and some applications have been developed (see e.g., [10]).

The ability of neural networks to classify information and to approximate functions to a specific degree of accuracy is well known. The network architecture and the training methods to be used must be selected with good engineering judgment, and research continues to provide new insights for making these decisions. The selection of the network size and training data is also crucial. Recent results by Rao et al. [11] state a bound on the number of samples needed to get specific probability of error in the approximation depending on the smoothness of the function and the learning parameters for feedforward networks. In the following we describe the network architecture and training methods we have chosen, and give some justifications for these decisions.

3.1 Neural network architecture

A multilayer feedforward network was selected for its capacity as an universal approximator with a size that is small relative to the size of the data set [12]. Even though one hidden layer is sufficient to achieve accurate approximations, the use of two hidden layers plus a linear input-output component gives more robust results, albeit at the cost of longer training [6].

Figure 2 shows the architecture chosen. The relationship between inputs and outputs for this type of network is given by:

$$\mathbf{y} = \mathbf{b}_o + W_o\mathbf{x} + W_3\ g(W_2\ g(W_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2),$$

where $W_i, \mathbf{b}_i$ are the weight matrices and threshold vectors, respectively. The nonlinear function $g(.)$ for the network hidden units is the hyperbolic tangent function. The output units are linear. The approximation is then a summation involving linear and nonlinear functions.

3.2 Training procedure

Neuro-dynamic programming requires a different type of learning procedure. The neural network is used to estimate the future performance of the closed loop system as expressed by (3).

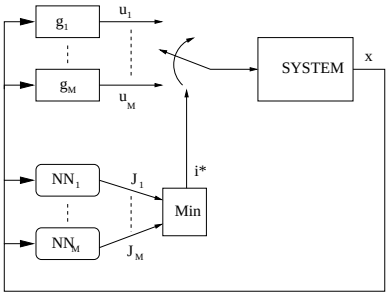

Figure 1: Controller switching scheme used.

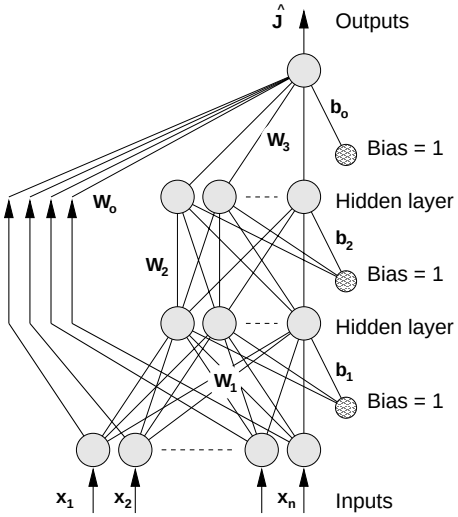

Figure 2: Neural network architecture used.

A Heuristic Dynamic Programming (HDP) algorithm is used [14]. If we denote by $\hat{J}_i(\mathbf{x}_k)$ the network estimate of the future performance for the i^{th} controller at state $\mathbf{x}_k$, the learning procedure computes the desired value of the future performance at $\mathbf{x}_k$ as

$$J_i^*(\mathbf{x}_k) = U(\mathbf{x}_k) + \delta \hat{J}_i(\mathbf{x}_{k+1}) \qquad (4)$$

The discount factor δ is chosen small enough to guarantee convergence of the infinite sum (3). The next step is to choose the network parameters to minimize the square of the error

$$\epsilon_k = J_i^*(\mathbf{x}_k) - \hat{J}_i(\mathbf{x}_k, \mathbf{p})$$

where the parameter vector $\mathbf{p}$ is the set of network coefficients. Figure 3 shows the learning scheme.

The pointwise cost function used is a standard quadratic form:

$$U(\mathbf{x}) = \mathbf{x}^T \; P \; \mathbf{x}, \qquad P > 0$$

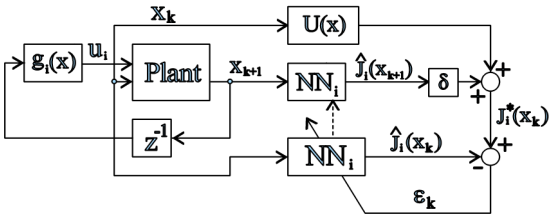

Figure 3: HDP learning scheme.

Therefore, for each trajectory the input-output patterns for the network can be computed and supervised learning can be applied. Additional supervised training may be performed with representative patterns to improve the quality of learning [9].

4 The inverted pendulum

As a example of the method explained above we consider the well known inverted pendulum (IP). We use a physical IP that has served as a standard laboratory example for the SIMPLEX architecture described in the introduction. Here, we present results of the neural network estimation of the performance index for three different controllers based on data from a simulation model using HDP.

The IP model equations used are:

$$\begin{aligned} J_t\ddot{x} + \frac{1}{2}\, ml\, cos\alpha\, \ddot{\alpha} + B_r\dot{x} - \frac{1}{2}\, ml\, sin\alpha\, \dot{\alpha}^2 &= F \\ \frac{1}{2}\, m\, cos\alpha\, \ddot{x} + \frac{1}{3} ml\ddot{\alpha} - \frac{1}{2}\, mg\, sin\alpha &= 0 \end{aligned}$$

with the following constraints:

$$\begin{aligned} |F| &\leq F_{max} \\ |x| &\leq x_{max} \\ |\dot{x}| &\leq v_{max} \end{aligned}$$

where α is the angle of the pendulum, x is the position of the pendulum base, m is the effective mass at the end of the pendulum, l is the pendulum length, J_t is the inertia of the base, B_r is the friction coefficient for the base, and g is the gravitational acceleration.

The parameters take the values:

J_t	0.6650	Kg	g	9.8	m/s^2
m	0.21	Kg	F_{max}	2	N
l	0.61	m	x_{max}	2.0	m
B_r	0.1	Kg/s	v_{max}	3.0	m/s

The main idea is to show how the neural network performs the scheduling. Towards this end, three controllers were design to achieve stabilization of the pendulum around the origin, each one designed to work differently and in such a way that from the combination of the controllers we expect to get better performance than just by the application of any one of them.

The first control algorithm applied is a standard LQR controller for the linearized model of the IP around the origin. The matrices Q and R selected are:

$$R = 1, \quad Q = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The feedback control law obtained is then

$$\begin{aligned} \mathbf{u} &= K\mathbf{x}, \\ K &= [10.0\ 12.60\ 48.33\ 9.09] \\ \mathbf{x} &= [x\ \dot{x}\ \alpha\ \dot{\alpha}]^T \end{aligned}$$

The LQR controller does not have a very large domain of stability since it is based on a linearization of the dynamics. Therefore, a sliding mode (SM) controller is designed to bring α to zero as fast as possible using a nonlinear switching rule and saturating control. The sliding surface is simply defined as

$$s(t) = \left(\frac{d}{dt} + \lambda\right)\alpha$$

Furthermore, to avoid chattering the saturation function of bandwidth ϕ is used instead of a sign function. The SM controller takes the form

$$\mathbf{u} = \hat{u}(\mathbf{x}, \lambda) + \gamma sat\left(\frac{s(t)}{\phi}\right)$$

where $\hat{u}$ is the nominal control action that satisfies the equation $\dot{s}(t) = 0$ and

$$\lambda = 5, \quad \phi = 0.1, \quad \gamma = 20$$

The SM controller is designed to bring the pendulum angle to zero, however, it may do so with a nonzero terminal base speed. Therefore, a third controller is designed to bring the base to rest, although not necessarily at $x = 0$. This controller is just another LQR feedback controller but with the gain for the base position set to zero, so we call it the *velocity feedback* (VF) controller. The control law is given by

$$\mathbf{u} = K\mathbf{x}, \quad K = [0\ 30.92\ 87.63\ 20.40]$$

Given the three controllers described above, three neural networks were trained to approximate the cost-to-go function of each controller. Training was performed off-line in each case using 5000 random trajectories, and then continued on-line during the simulated control experiments using HDP. The discount factor δ and the cost matrix P chosen were

$$\delta = 0.5, \quad P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multilayer feedforward networks were used together with conjugate gradient optimization methods.

Figures 4, 5 and 6 show slices of the approximated performance measure in a 3D plot for each of the controllers with the angle rate and base position set to zero. Note that these relative index measures are intuitive in their shapes for each controller, and the relative values are also intuitive in that the SM controller obtains a better performance for larger values of α, but is not as good as the LQR controller for smaller angles.

The initial condition used in the trajectory simulation is $\alpha = 0.5$. All other state variables equal zero initially. As shown in figure 7, for this initial state: the LQR controller cannot stabilize the system; if the SM controller is applied alone, the angle goes to zero, but the final base speed is nonzero; and the VF controller will not drive the base position to zero, even if it can stabilize the pendulum and bring the base to rest. Figure 8 shows the position trajectory from the same initial condition along with the corresponding switches between controllers. Noise was added to the state measurements in the simulation, leading to the chattering

observed in the controller selection. Note that the controller scheduling obtains an asymptotically stable performance for a situation in which none of the controllers is acceptable by itself.

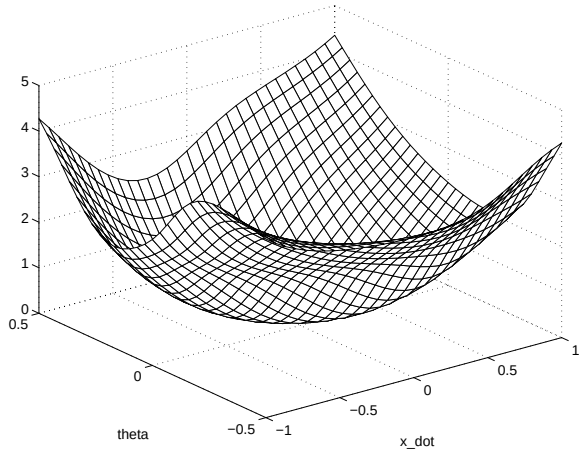

Figure 4: Cut of the neural network estimation for the performance of the LQR controller (LQR).

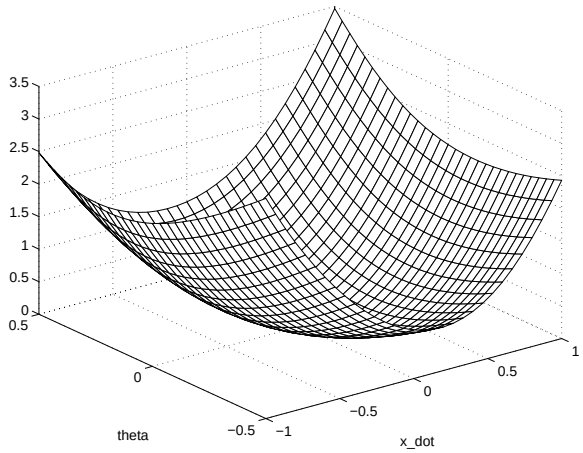

Figure 5: Cut of the neural network estimation for the performance of the sliding controller (SM).

5 Discussion

A technique is developed to perform controller scheduling. Using a multilayer feedforward neural network and neuro-dynamic programming to estimate a performance index for each controller. Further work involves the inclusion of the stability regions into the performance characterization [4]. The algorithm must look also at a smooth transition in control actions. We are also investigating

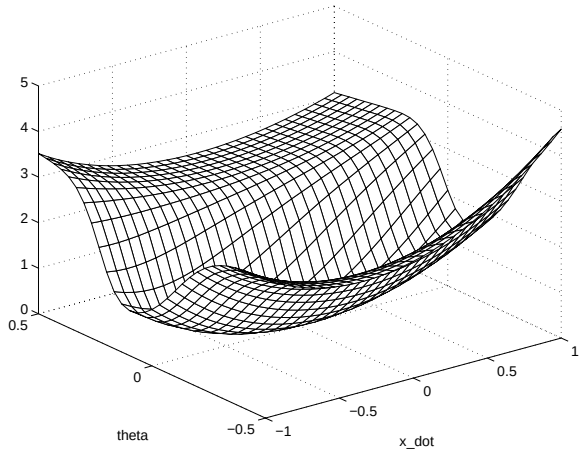

Figure 6: Cut of the neural network estimation for the performance of the velocity feedback controller (VF).

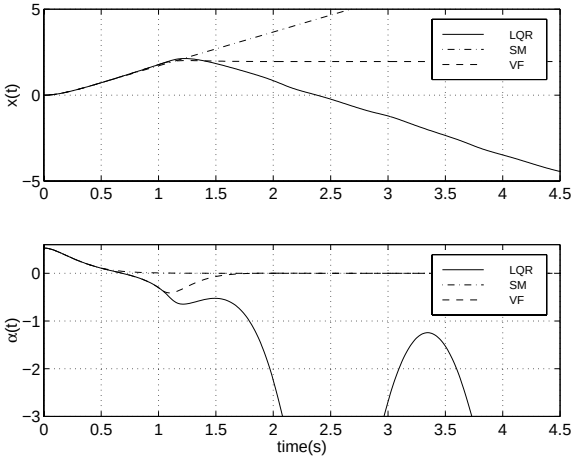

Figure 7: Arbitrary trajectory showing the system being controlled by each designed controller by itself.

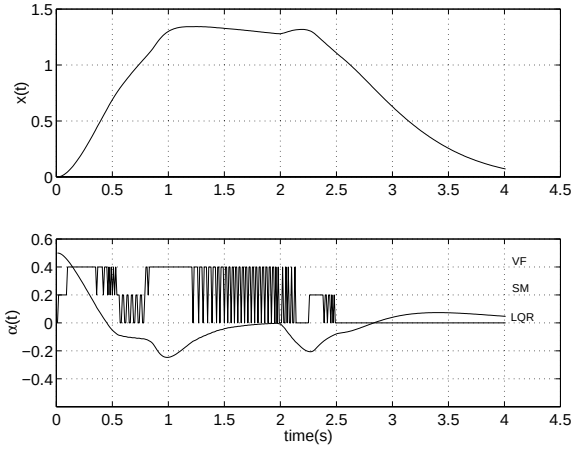

Figure 8: Arbitrary trajectory showing the system behavior under the control switching scheme.

the use of better features as inputs to the network to speed up the learning and have more robust results. Finally, there are several theoretical issues currently under investigation. Recent work by Malmborg et al. [5] establishes a scheduling technique and proves stability for a family of state feedback controllers based on a Lyapunov analysis. We are working on proofs of convergence for the neuro-estimation learning algorithms and stability of the closed-loop system with controller scheduling.

Acknowledgments

This research is been supported by CONICYT-IBD, the Organization of American States, the Software Engineering Institute at Carnegie Mellon University and DARPA, contract number F33615-97-C-1012.

References

[1] J. Balakrishnan and K.S. Narendra. Adaptation and learning using multiple models, switching, and tuning. *IEEE Control Systems Magazine*, 15(3):37–51, Jun 1995.

[2] J. Balakrishnan and K.S. Narendra. Intelligent control using fixed and adaptive models. In *Proceedings of 1995 American Control Conference*, volume 1, pages 597–601, Evanston, IL, USA, Jun 1995.

[3] Dimitri P. Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[4] E.D. Ferreira and B.H. Krogh. Using neural networks to estimate regions of stability. In *Proceedings of 1997 American Control Conference*, Albuquerque, NM, June 1997.

[5] J. Malmborg, B. Berhardsson, and K.J. Aström. A stabilizing switching scheme for multi-controller systems. In *Proceedings of the IFAC World Congress*, volume F, pages 229–234, San Francisco, California, USA, 1996. IFAC'96, Elsevier Science.

[6] P.M. Mills, A.Y. Zomaya, and M.O. Tade. *Neuro-Adaptive Process Control*. John Wiley & Sons Ltd., 1996.

[7] A.S. Morse. Supervisory control of families of linear set-point controllers - part ii: Robustness. In *34th IEEE Conference on Decision and Control*, volume 2, pages 1750–5, New York, USA, Dec 1995.

[8] A.S. Morse. Supervisory control of families of linear set-point controllers - part i: Exact matching. *IEEE Trans. on Automatic Control*, 41(10):1413–31, Oct 1996.

[9] D.A. Pomerleau. Alvinn: an autonomous land vehicle in a neural network. Technical report, Computer Science Dept. Carnegie Mellon University, 1989.

[10] D.V. Prokhorov and D.C. Wunsch II. Advanced adaptive critic design. In *Proceedings of the World Congress on Neural Networks (WCNN-96)*, pages 83–87, San Diego, CA, Sep. 1996. Invited paper.

[11] N.S.V. Rao, V. Protopopescu, R.C. Mann, E.M. Oblow, and S.S. Iyengar. Learning algorithms for feedforward networks based on finite samples. *IEEE Trans. on Neural Networks*, 7(4):926–940, Jul 1996.

[12] R.M. Sanner and J-J. E. Slotine. Gaussian networks for direct adaptive control. *IEEE Trans. on Neural Networks*, 3(6):837–863, 1992.

[13] L. Sha. A software architecture for dependable and evolvable industrial computing systems. In *Proc. IPC'95*, pages 145–156, Detroit, MI, May 1995.

[14] P. Werbos. A menu of designs in reinforcement learning over time. In W.T. Miller III, R.S. Sutton, and P. Werbos, editors, *Neural Networks for control*, chapter 3. MIT Press, 2nd edition, 1991.