Neural Network Approach to Control System Identification with Variable Activation Functions

Michael C. Nechyba and Yangsheng Xu

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Human beings epitomize the concept of "intelligent control." Despite its apparent computational advantage over humans, no machine or computer has come close to achieving the level of sensor-based control which humans are capable of. Thus, there is a clear need to develop computational methods which can abstract human decision-making processes based on sensory feedback. Neural networks offer one such method with their ability to map complex nonlinear functions. In this paper, we examine the potential of an efficient neural network learning architecture to the problems of system identification and control. The cascade two learning architecture dynamically adjusts the size of the network as part of the learning process. As such, it allows different units to have different activation functions, resulting in faster learning, smoother approximations, and fewer required hidden units. We use the methods discussed here towards identifying human control strategy.

I. Introduction

Researchers in the field of robotics and autonomous systems frequently find themselves looking towards human intelligence as a guide for developing "intelligent" machines. Paradoxically, control tasks which seem easy or even trivial for humans, are often extremely difficult or impossible for computers or robots to duplicate. Rule-based systems usually fail to anticipate every eventuality and thus are ill-suited for robots in uncertain and new environments. There is a clear need to develop computational methods which can, in a general framework, abstract the human decision-making process based on sensory feedback.

Modeling and identifying human control processes can be a significant step towards transferring human knowledge and skill in real-time control. This can lead to more intelligent robotic systems, and can bridge the gap between traditional artificial intelligence theory and the development of intelligent machines.

Artificial neural networks have shown great promise in identifying complex nonlinear systems. Thus, neural networks are well suited for generating the complex internal mapping from sensory inputs to control actions, which humans possess. Our goal is to develop a feasible neural network-based method for identifying human control strategy and transferring that control strategy to robotic systems. To this end, we are looking at an efficient and flexible neural network architecture which is capable of modeling nonlinear dynamic systems.

We propose to use the cascade two learning architecture [3], which adjusts the structure of the network as part of the training process. Unlike a fixed network architecture, where hidden unit activation functions are specified before training begins, the incremental addition of new hidden units allows for the hidden units to have variable activation functions. Although sigmoidal activation functions may be best suited for networks with binary outputs, this is not necessarily the case for continuous-valued outputs. Allowing new hidden units with variable activation functions can lead to smoother approximations, as well as faster learning with fewer required computations and hidden units.

In this paper, we first provide background information on this new architecture for neural network learning and a theoretical basis for its use. We then present simulation results for this architecture in identifying both static and dynamic systems, including a nonlinear controller for an inverted pendulum system. Finally, we show some preliminary results in modeling human control strategy.

II. Neural Network Architecture

A. Cascade Architecture

The cascade two learning architecture, developed by Fahlman [4], is very similar to the previously developed cascade correlation algorithm. Both cascade correlation and cascade two combine the following two notions: (1) the *cascade architecture*, in which hidden units are automatically added one at a time to an initially minimal network, and (2) the accompanying learning algorithm, which creates and installs the new hidden units [3][4].

In cascade correlation, the learning algorithm attempts to maximize the magnitude of the correlation between the new hidden unit's output and the residual error signal. This covariance measure tends to overshoot small errors, however, and thus is not suitable for continuous-valued outputs. The cascade two algorithm corrects this problem by attempting to minimize the sum-squared difference between the scaled unit outputs and the residual error [3].

Training proceeds as summarized below. Initially, there are no hidden units in the network, only the input/output connections. These weights are trained first, thereby capturing any linear relationship between the inputs and outputs. With no further appreciable decrease in the error measure (in cascade two, the sum-squared error), the first hidden unit will be added to the network from a pool of *candidate* units. Using the quickprop algorithm [2], these candidate units are trained independently and in parallel with different random initial weights.

After no more appreciable error reduction occurs, the best candidate unit is selected and installed in the network. Once installed, the hidden unit input weights are frozen, while the weights to the output units are retrained. By freezing the input weights for all previous hidden units, each training cycle is equivalent to training a three-layer feedforward neural network with a single hidden unit. This allows for much faster convergence of the weights during training than in a standard backprop network where many hidden unit weights are trained simultaneously.

The process is repeated until the algorithm succeeds in reducing the sum-squared error sufficiently for the training set or the number

of hidden units reaches a specified maximum number. Note that each new hidden unit receives as input connections from all previous units, including all input units as well as previous hidden units. Figure 1 below illustrates how a 2-input, 1-output network grows as 2 hidden units are added.

We believe that the cascade two architecture offers several advantages, particularly relevant for mapping of non-linear continuous-valued functions. First, the algorithm adjusts the architecture of the network automatically, thus obviating the need for *a priori* guessing of the necessary network architecture. Second, the cascade architecture can potentially model higher degrees of nonlinearity with fewer hidden units than might be required in a single or two hidden-layer network. Finally, and perhaps most importantly, the incremental addition of hidden units allows for new hidden units to have variable activation functions.

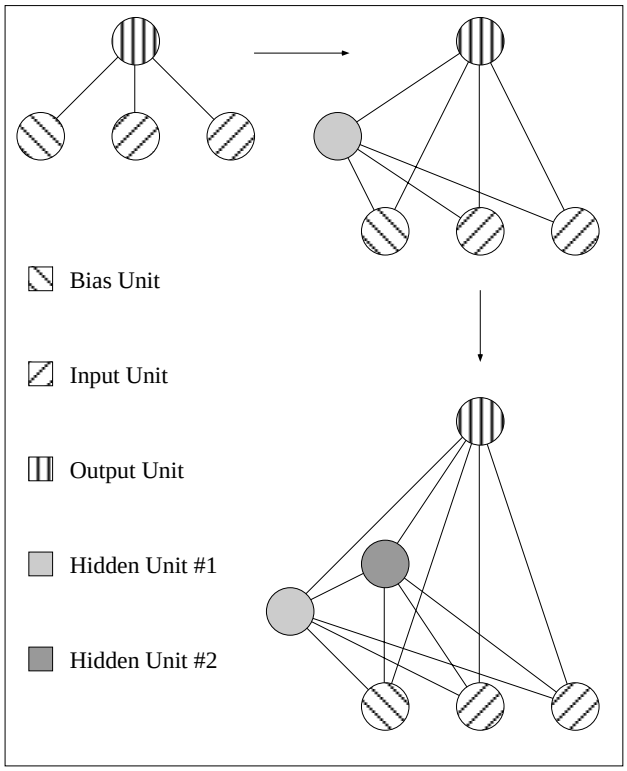

Fig. 1: The cascade two learning architecture adds hidden units one at a time as shown in the above diagram. All connections are feedforward.

In the pool of candidate units, we can assign a different nonlinear activation function to each unit. These functions can include but are not limited to the sigmoid function, sine or cosine functions, and the Gaussian function. Thus, if the function to be approximated has a a strong sinusoidal dependence of some sort, it is more efficient to have one sinusoidal hidden unit rather than several sigmoidal units which have to act together to first approximate a sinusoidal dependence. During candidate training, the algorithm will select for installment whichever candidate unit reduces the sum-squared error of the training data the most. Hence, the unit with the most appropriate activation function at that point during training is selected.

Finally, we note that the cascade two architecture is capable of arbitrary, nonlinear function approximation. Using Kolmogorov's theorem, Kurkova shows in [9] that a feedforward neural network with two hidden layers is sufficient for arbitrary function approximation. In fact, Cybenko and Funahashi have shown separately that a continuous feedforward neural network with a *single* hidden layer and sigmoidal activation functions can approximate nonlinear mappings arbitrarily well [1][5]. Since any multilayer feedforward neural network with full connectivity between consecutive layers is simply a special case of a cascade network with an equal number of hidden units, these function approximation theorems extend trivially for this architecture. Furthermore, Cybenko shows that there is no strict theoretical argument for confining the activation functions exclusively to sigmoidal functions, and shows, for example, that *sine* and *cosine* are complete in the space of *n*-dimensional continuous functions [1].

B. Dynamic System Identification

In general, a dynamic system may be expressed as a finite difference equation of the general form,

$$\bar{y}(k+1) = g(\bar{y}(k), \bar{y}(k-1), \ldots, \bar{y}(k-n), \bar{u}(k), \bar{u}(k-1), \ldots, \bar{u}(k-m)) \quad \text{(Eq. 1)}$$

where $g(\cdot)$ is some arbitrary nonlinear function, $\bar{y}(k)$ is the output vector and $\bar{u}(k)$ is the input vector at time step k. Most neural networks are only capable of static input/output mapping, however. To overcome this problem, Narendra suggests providing a time history of data as input to the neural network [13][14]. Thus, *static* feedforward neural networks have the potential to approximate complex nonlinear mappings of *dynamic* systems for which no analytic model may exists. For example, Figure 2 illustrates how this is done for a SISO system of the form,

$$y(k+1) = f(y(k), y(k-1), y(k-2), u(k), u(k-1)) \quad \text{(Eq. 2)}$$

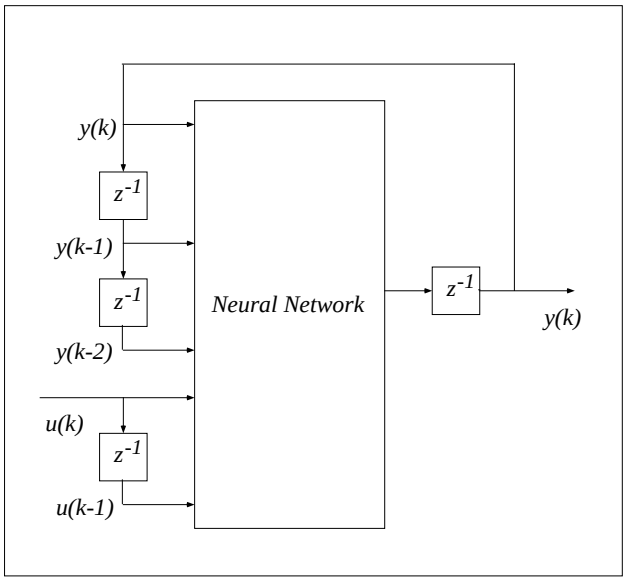

Fig. 2: The diagram illustrates how a dynamic system is mapped onto a static feedforward neural network.

In [14], special cases of (Eq. 1) are classified depending on whether part of the relationship in the equation is linear. Since the cascade architecture begins with direct linear connections between inputs and outputs, such classification is unnecessary here.

III. Control System Modeling

Below, we present simulation results which serve a three-fold purpose. First, the simulations demonstrate the feasibility and advantage of variable activation functions over *a priori* specification

of activation functions. Second, the simulations demonstrate the neural network's ability to model dynamic systems from input/output data vectors. Finally, we show that the neural network can learn a known control strategy for a sample system. Such learning is crucial to identifying components of the human control process.

For all simulations, we allowed a maximum of 250 epochs to train the weights in a pool of eight candidate units. In the case of variable activation types, the pool of candidate units has the following function types: (1) standard symmetric sigmoid, with a (-0.5, 0.5) range, (2) standard zero-mean Gaussian, (3) Bessel function of the first kind of order zero and one, (4) sine, (5) cosine, (6) double frequency sine, and (7) double frequency cosine. Output units are linear, so as to allow the outputs to assume any real value.

Unless otherwise noted, the dashed or dotted line in each figure shows the function to be modeled, while the solid line plots the output from the trained network

A. One-dimensional Function Approximation

Here, we demonstrate the consequences of utilizing different nonlinear activation functions for each hidden unit by modeling a simple, static one-variable function given by,

$$f(x) = \frac{1}{(1+x^2)} \qquad \text{(Eq. 3)}$$

Here, we use 1500 uniformly distributed random data points in the interval $x \in [-4, 4]$ as training data.

We train two different networks, one with all sigmoidal units, and one with variable hidden units. In each case we stop training after six hidden units have been added. The hidden unit types in the network with variable activation functions follow in the order of insertion: (1) Bessel function of order zero, (2) sine, (3) double frequency cosine, (4) Bessel function of order one, (5) sine, and (6) double frequency cosine.

Figure 3 below shows the network output for the variable-unit network, while Figure 4 shows the network output for the sigmoidal network. Figure 5 and Figure 6 show the approximation errors for Figure 3 and Figure 4, respectively.

Table 1 summarizes the relationship between approximation error and activation functions for three different $f(x)$. In each case, the neural networks are trained to a size of six hidden units.

Table 1: Approximation Error for Various f(x)

f(x)	Sigmoidal Units (RMS error)	Variable Units (RMS error)
$0.6\sin(\pi x) + 0.3\sin(3\pi x) + 0.1\sin(5\pi x)$	0.0397	0.0097
$x^3 + 0.3x^2 - 0.4x$	0.0140	0.0069
$\frac{1}{(1+x^2)}$	0.0215	0.0045

B. Non-linear Difference Equation

Below, we compare our network architecture to a standard multilayer feedforward backprop network as described by Narendra in [14]. The difference equation we want to approximate is given by,

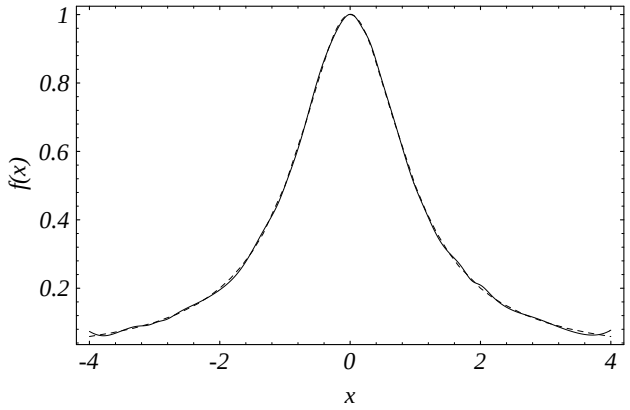

Fig. 3: The network with non-sigmoid units performs better in approximating the function *f(x)*.

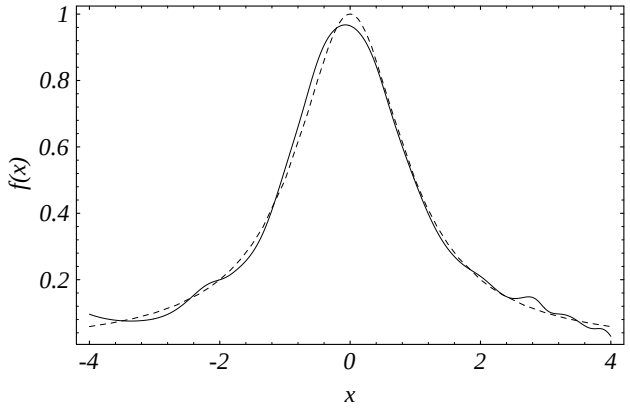

Fig. 4: The sigmoidal network performs less well in approximating *f(x)*.

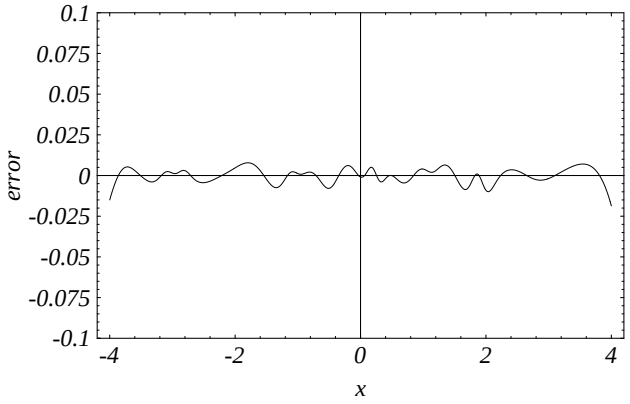

Fig. 5: The approximation error is relatively uniform over the training interval.

$$y(k+1) = \begin{array}{c} 0.3y(k) + 0.6y(k-1) + \\ u(k)^3 + 0.3u(k)^2 - 0.4u(k) \end{array} \qquad \text{(Eq. 4)}$$

We use 1000 uniformly distributed random inputs in the interval $u(k) \in [-1, 1]$ as training data. The resulting cascade network has three hidden units whose activation functions are in order of insertion: (1) double frequency cosine, (2) Bessel function of order one, and (3) sine.

To test the network, we use an input given by,

$$u(k) = \sin\left(\frac{2\pi k}{250}\right) \qquad \text{(Eq. 5)}$$

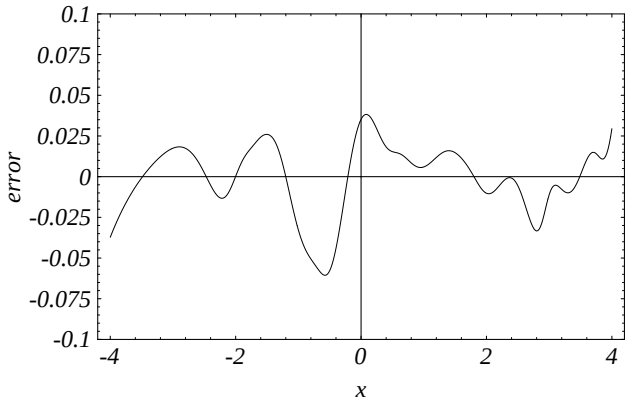

Fig. 6: The approximation error is much larger for the sigmoidal network.

The resulting output is shown in Figure 7 below. This result compares very favorably to the simulation in [14] where a network with two hidden layers of 20 units and 10 units, respectively is used, and the training data set includes over 50,000 data points.

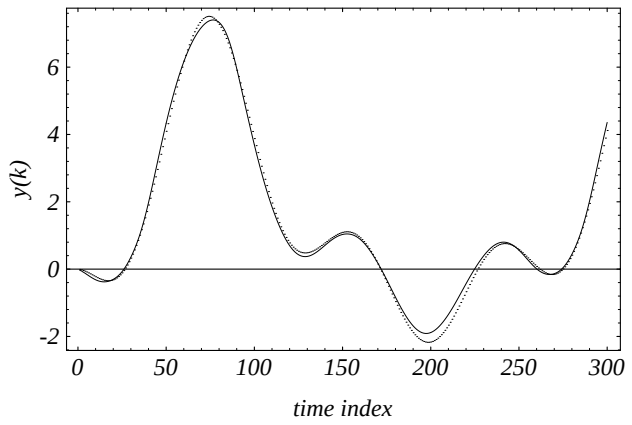

Fig. 7: The 3-hidden unit neural network performs well in tracking the output of the dynamic system.

A network with three sigmoidal hidden units performs significantly worse as is shown in Figure 8.

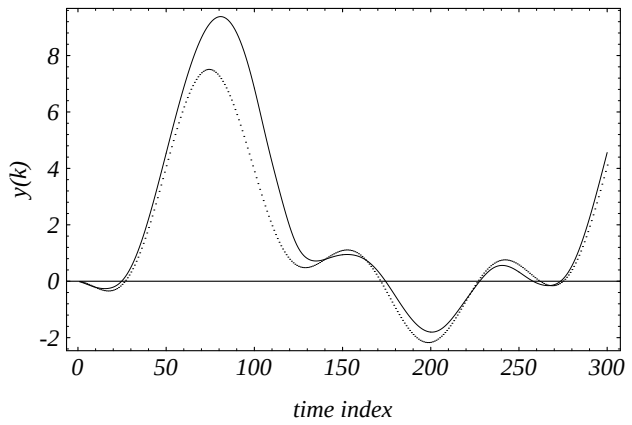

Fig. 8: More sigmoid hidden units are required to match the performance of the neural network in Figure 7.

C. Control Application

Below, we simulate a nonlinear controller for the classic inverted pendulum system. This is a traditional benchmark problem in control since the dynamics of the system are nonlinear and coupled, and the open-loop system is unstable. The dynamics of the system are governed by the following equations [6]:

$$\ddot{\theta} = \frac{3}{4l}(g\sin\theta - \ddot{x}\cos\theta) \quad \text{(Eq. 6)}$$

$$\ddot{x} = \frac{m\left(l\sin\theta\dot{\theta}^2 - \frac{3}{8}g\sin 2\theta\right) - f\dot{x} + u}{M + m\left(1 - \frac{3}{4}\cos^2\theta\right)} \quad \text{(Eq. 7)}$$

We use the following nonlinear control law as teacher to the neural network:

$$h_1 = \frac{3}{4l}g\sin\theta \quad \text{(Eq. 8)}$$

$$h_2 = \frac{3}{4l}\cos\theta \quad \text{(Eq. 9)}$$

$$f_1 = m\left(l\sin\theta\dot{\theta}^2 - \frac{3}{8}g\sin 2\theta\right) - f\dot{x} \quad \text{(Eq. 10)}$$

$$f_2 = M + m\left(1 - \frac{3}{4}\cos^2\theta\right) \quad \text{(Eq. 11)}$$

$$u = \frac{f_2}{h_2}\left[h_1 + k_1(\theta - \theta_d) + k_2\dot{\theta} + c_1(x - x_d) + c_2\dot{x}\right] - f_1 \quad \text{(Eq. 12)}$$

For the simulations, we used the following numeric values: $M = 1$ kg, $m = 0.1$ kg, $l = 1$ m, $f = 5$ kg/s, $g = 9.81$ m/s^2, $k_1 = 25$, $k_2 = 10$, $c_1 = 1$, $c_2 = 2.6$. Also we set $x_d = 0$ m, and $\theta_d = 0$ rad, which are the desired position of the cart and angle of the pendulum respectively. For details on all the parameters see [6].

This system is simulated numerically using Euler's approximation method with a time step of $T = 0.02$ seconds. The neural network takes as input the current and previous x positions, as well as the current and previous θ positions. It is trained to approximate the control law given in (Eq. 12). As training data, we generate 500 uniformly distributed random input/output vectors in the following range:

$$x \in [-1, 1] \quad \text{(m)} \quad \text{(Eq. 13)}$$

$$\theta \in [-0.5, 0.5] \quad \text{(rad)} \quad \text{(Eq. 14)}$$

$$(x_{current} - x_{previous}) \in [-0.04, 0.04] \quad \text{(m)} \quad \text{(Eq. 15)}$$

$$(\theta_{current} - \theta_{previous}) \in [-0.02, 0.02] \quad \text{(rad)} \quad \text{(Eq. 16)}$$

We found that as few as three hidden units were sufficient to model the controller, even for large initial values of θ. Below we compare the performance of a trained neural network with three sinusoidal hidden units to that of the actual control law. The initial conditions for the simulation are,

$$[x, \dot{x}, \theta, \dot{\theta}] = [0, 0, 0.6\,(rad), 0] \quad \text{(Eq. 17)}$$

Note that the initial condition for θ is outside the range of the training data. Figure 9 compares the actual controller and the neural network controller performance. The controller and neural network generate virtually identical results. Figure 10 shows the difference in response between the actual and the neural network controller.

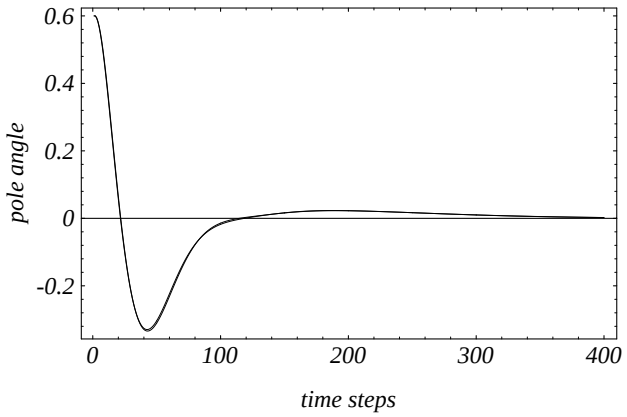

Fig. 9: The angle of the pendulum is controlled almost identically for the nonlinear control law and the neural network controller.

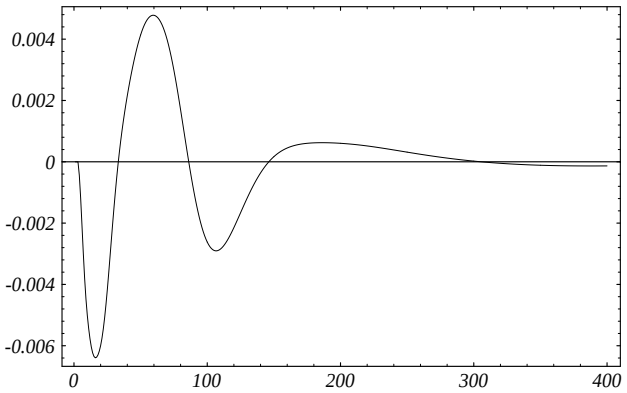

Fig. 10: This figure plots the error between the angle position caused by the nonlinear control law and the angle position caused by the neural network controller.

IV. Modeling Human Control Strategy

A. Experimental Setup

In this section, we show preliminary results in modeling human control strategy. For the experiment, a human subject is shown an inverted pendulum-cart system on a computer screen, and is able to control the horizontal force to be applied to the cart via the horizontal mouse position. The parameters for this cart-pendulum system are equivalent to those given in the previous section. Thus, we have replaced the nonlinear control law with a human being as teacher for the neural network. The system state, as well as the control input provided by the human, are recorded at 100 Hz.

B. Modeling Results

Case 1: After numerous failed attempts at keeping the pendulum from falling for any meaningful period of time, the first human subject successfully controls the system for 23 seconds or 2300 data points. Figure 11 below shows the pendulum angle for the time that the human is able to keep the pendulum from falling. From this data, 750 randomly selected data points are selected to train the network, while another 750 randomly selected data points are used for cross validation.

The neural network to be trained from this data takes six inputs, namely, the past five values of the pendulum angle, as well as the velocity of the cart,

$$\{\theta(k-4), \ldots, \theta(k-1), \theta(k), \dot{x}(k)\} . \qquad \text{(Eq. 18)}$$

As output, the network generates the horizontal force to be applied to the cart in the next time step, $u(k+1)$.

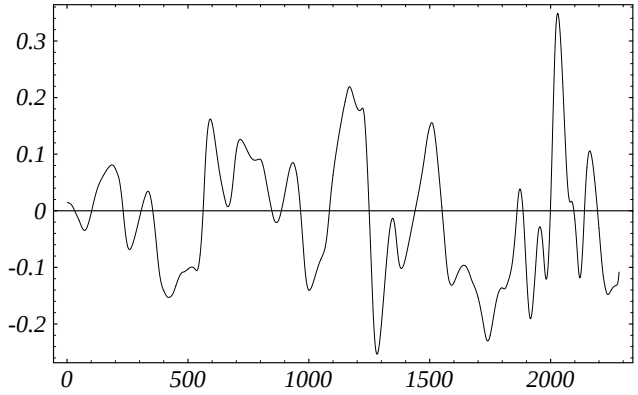

Fig. 11: Data from this run of 23 seconds was used to train the neural network to control the inverted cart-pendulum system. Here, the pendulum angle is shown in radians.

We allow a maximum of 150 epochs to train the weights as each new hidden unit is added. Here, all hidden unit activation function types are one of the sinusoidal functions; we stop training with twelve hidden units. Figure 12 below shows the resulting neural network control of the pendulum-cart system with $\theta_{initial} = 0.2$ for 20 seconds.

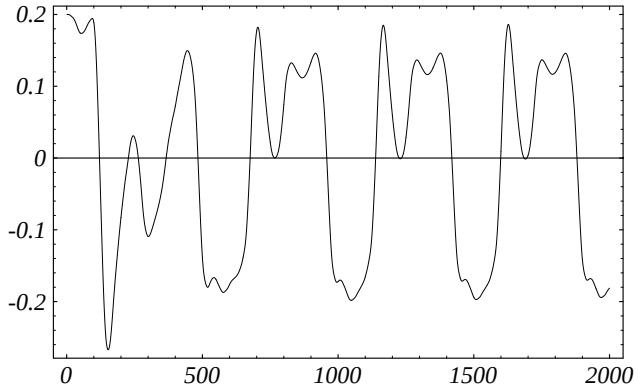

Fig. 12: Neural network control of the inverted pendulum-cart system. Here, the pendulum angle is shown in radians.

By plotting the trajectory of the pendulum in phase space, we see that the trajectory, although not periodic, does exhibit a definite pattern over a long period of time. In Figure 13 below, 200 seconds of the pendulum trajectory in phase space are plotted.

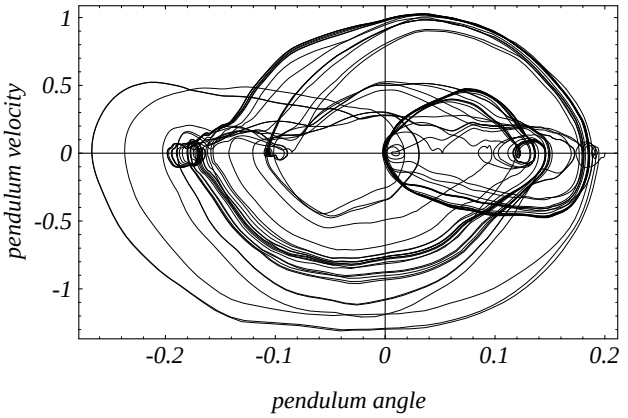

Fig. 13: Pendulum trajectory in phase space.

It was determined experimentally that this neural network controller is stable for $-0.92 < \theta_{initial} < 0.98$.

Case 2: A different human subject is also asked to control the system. This subject shows greater skill than the first subject and has a successful run of approximately 60 seconds. From this data, 1000 randomly selected data points are selected to train the network, while another 1000 randomly selected data points are used for cross validation. This network has the same inputs and outputs, and training proceeds as in *Case 1* above.

After training, an examination of the resulting weights in the network revealed several weights which are much larger than many smaller weights. The largest weight (in magnitude) is, for example, approximately 1201. Thus, all weights less than six in magnitude are set to zero. The resulting controller proves to be remarkably simple, and can be expressed by,

$$u(k+1) = w_1\theta(k) + w_2\theta(k-2) + w_3\theta(k-4) + w_4\dot{x} \quad \text{(Eq. 19)}$$

where $w_1 \approx -350$, $w_2 \approx 1201$, $w_3 \approx -925$, and $w_4 \approx 7.6$. Thus, a traditional, linear feedback controller has been abstracted from training data provided by a human operator. This controller is stable for $-1.04 < \theta_{initial} < 1.05$.

V. Discussion

The simulation and experimental results in the previous two sections bring up some interesting issues. From the simulation results presented in this paper, it is apparent that using variable activation functions can contribute significantly to better nonlinear function approximation. When allowing selection between activation types, sinusoidal units are often preferred by the learning algorithm. This suggests that if the continuous-valued mapping we are trying to teach to a neural network is smooth, sigmoidal units may not be appropriate. In general functional mappings in control applications do vary in a smooth manner with sensory input.

Generalizability of the neural network to points outside the range of the training set must, however, also be addressed in this context. For example, due to the discretization of the dynamic simulation, the inverted pendulum controller actually becomes unstable for certain large initial values of θ. For the nonlinear control law in (Eq. 12), it was determined experimentally that the system becomes unstable for $\theta_{initial} \approx 1.12$ rad. The neural network controller becomes unstable for a slightly lower $\theta_{initial} \approx 1.03$ rad. However, for a three hidden unit network with sigmoidal activation functions, the system becomes unstable only at $\theta_{initial} \approx 1.07$ rad. This suggests that using variable activation functions may trade off some generalizability in favor of better approximation. Further work needs to be done in this direction.

As far as learning control strategy, the cascade two learning algorithm would be significantly more attractive if learning could occur in real time from a continuous stream of data, rather than nicely conditioned batches of data. Here, the main problem is to decide when to add an additional hidden unit, i.e. what the terminating condition should be for learning without adding a new unit. Currently, we are working on allowing the cascade two learning architecture to be used for real-time learning.

Finally, the preliminary results on modeling human control strategy raise two very important problems. First, how can we judge whether or not a neural network has truly learned the human control strategy? As the human is demonstrating his/her control, he/she is prone to making errors, and in general the human performance is stochastic in nature. Second, can we bridge the gap between traditional control theory and neural network control more frequently, as was possible for the second human subject, where a robust linear feedback control law was deduced from the trained neural network? Obviously, further work needs to be done in this direction.

VI. Conclusion

Learning control from humans by example is an important concept for making robots and machines more intelligent. Neural networks are well suited to generate the complex nonlinear mapping of the human control process, which maps sensory inputs to control action outputs. We have presented encouraging results for nonlinear continuous function mapping and dynamic system identification by utilizing a new neural network architecture. We have demonstrated that the cascade network architecture and learning process is well suited for efficiently mapping continuous nonlinear functions. We have also demonstrated that the method allows a neural network to learn both a known nonlinear, coupled control law, as well as unknown nonlinear human control strategy.

Acknowledgments

We would like to thank Scott Fahlman for his advice in using the cascade two learning architecture. Also, we would like to thank Scott Fahlman, Michael Kingsley, David C. Lambert for the use of their neural network simulation software.

References

[1] Cybenko, G., "Approximation by Superposition of a Sigmoidal Function," *Mathematics of Control, Signals, and Systems*, Vol. 2, No. 4, pp. 303-314, 1989.

[2] Fahlman, S. E., "An Empirical Study of Learning Speed in Back-Propagation Networks," Technical Report, CMU-CS-TR-88-162, Carnegie Mellon University, 1988.

[3] Fahlman, S.E., and Boyan, J.A., "The Cascade Two Learning Architecture," Technical Report (forthcoming), CMU-CS-94-100, Carnegie Mellon University, 1994.

[4] Fahlman, S.E., and Lebiere, C., "The Cascade-Correlation Learning Algorithm," Technical Report, CMU-CS-90-100, Carnegie Mellon University, 1990.

[5] Funahashi, K., "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Net.*, Vol. 2, No. 3, pp. 183-192, 1989.

[6] Guez, A., Selinsky, J., "A Trainable Neuromorphic Controller," *Jour. of Robotic Sys.*, Vol 5., No. 4, pp. 363-388, 1988.

[7] Hornik, K., Stinchcombe, M., and White, H., "Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks," *Neural Net.*, Vol. 3, No. 5, pp. 551-560, 1990.

[8] Hunt, K. J., *et. al.*, "Neural Networks for Control Systems - A Survey," *Automatica*, Vol. 28, No. 6, pp. 1083-1112, 1992.

[9] Kurkova, V., "Kolmogorov's Theorem and Multilayer Neural Networks," *Neural Net.*, Vol. 5, No. 3, pp. 501-506, 1992.

[10] Narendra, K. S., "Adaptive Control of Dynamical Systems Using Neural Networks," *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, White D. A., and Sofge D. A., ed., pp. 141-184, 1992.

[11] Narendra, K. S., Parthasarathy, K., "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans. on Neural Networks*, Vol. 1, No. 1, pp. 4-27, 1990.