비선형 시스템의 불확실성을 보상하는 신경회로망 제어

조현섭, 오명관
청운대학교 디지털방송공학과, 혜전대학 디지털서비스과
e-mail : chohs@chungwoon.ac.kr

Uncertainty-Compensating Neural Network Control for Nonlinear Systems

Hyun-Seob Cho, Myoung Kwan Oh
Dept of Digital Broadcast Engineering Chungwoon University
Dept of Digital Service Hyejeon College

Abstract - We consider the problem of constructing observers for nonlinear systems with unknown inputs. Connectionist networks, also called neural networks, have been broadly applied to solve many different problems since McCulloch and Pitts had shown mathematically their information processing ability in 1943. In this thesis, we present a genetic neuro-control scheme for nonlinear systems. Our method is different from those using supervised learning algorithms, such as the backpropagation (BP) algorithm, that needs training information in each step. The contributions of this thesis are the new approach to constructing neural network architecture and its training.

I. INTRODUCTION

Control refers to a task which is to apply appropriate inputs to a plant so that the plant performs in a desirable way. In practical, many control problems suffer from difficulty due to system nonlinearity, uncertainty, and dynamic property. To cope with this difficulty regarding system dynamics and its environment, the controller has to estimate the unknown information during its operation. When the information pertaining to the unknown features of the plant or its environment is gained, if a control system has an ability to improve its performance in the future based on the obtained past experience, it is called a learning control system. The need for learning capability of control system has made several ways for new control techniques, and neuro-control technique is one of them. A neuro-control system, in general, performs a specific form of adaptive control, with the controller taking the form of multilayer neural network and the adaptable parameters being defined as the adjustable connection weights. BP is popular as a training algorithm for multilayer feedforward neural networks, and also popular neuro-control algorithms. But it has some shortcomings for practical applications. Genetic algorithms have been utilized for many control problems. P. Wang et al. presented a numerical example in which a pH(potential of hydrogen) neutralization process is regulated by a PID controller with its parameters optimized using simple GA. A. Varšek et al. employed GA to derive control rules encoded as decision tables and to optimize the parameters of the induced rules. Some researchers also studied genetically optimized fuzzy logic control. In their study, GAs played a role of optimizing membership functions and fuzzy linguistic rule sets. W. Schiffmann et al. introduced GAs to optimize the BP algorithm for training multilayer neural networks.

Some experimental methodologies for benchmarking of algorithms have been utilized by neural network and machine learning technologies. One of most popular problems is an inverted pendulum problem(also known as pole balancing problem). Michie and Chambers attempted the problem using their boxes paradigm, later

improved by Barto et al.'s ASE/ACE controller. Anderson has applied neural networks to the problem. Jervis applied their controllers to real inverted pendulum. We also applied our method to the problem. Genetic algorithms do not produce complete neural networks, it just utilized for optimization of networks, and then the optimized network is trained by using suitable learning algorithm. We applied reinforcement learning, a powerful machine learning mechanism, particularly Q-learning, to genetically optimized networks.

II. SYSTEMS AND CONTROL

2.1. Nonlinear Systems

In practical control systems, many different types of nonlinearities are found. They may be divided into two classes, depending on whether they are inherent in the system or intentionally inserted into the system, and inherent nonlinearities are unavoidable in control systems.

In a conventional way, nonlinear control problems have been solved by using linearization technique. However, it provides a method which is valid for only a limited range of operation.

An nth-order continuous-time system has the following general form:

$$\begin{aligned}\dot{x}_1(t) &= f_1(x_1(t), x_2(t), \ldots, x_n(t), t)\\ \dot{x}_2(t) &= f_2(x_1(t), x_2(t), \ldots, x_n(t), t)\\ &\vdots\\ \dot{x}_n(t) &= f_n(x_1(t), x_2(t), \ldots, x_n(t), t)\end{aligned} \tag{2.1}$$

Similarly, an nth-order discrete-time system has the following form:

$$\begin{aligned}x_1(k+1) &= f_1(x_1(k), x_2(k), \ldots, x_n(k), k)\\ x_2(k+1) &= f_2(x_1(k), x_2(k), \ldots, x_n(k), k)\\ &\vdots\\ x_n(k+1) &= f_n(x_1(k), x_2(k), \ldots, x_n(k), k)\end{aligned} \tag{2.2}$$

for k=0, 1, 2, ...

Those can be expressed in the vector form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) \tag{2.3}$$

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k)) \tag{2.4}$$

The systems above depend upon both the state $\mathbf{x}$ and time, k or t. These systems are, in general, time varying.

2.2. Dynamics of Neural Networks

The activation function or transfer function, denoted by $F[\cdot]$, maps the unbounded junction value to a bounded output of neuron, and defines the activation level of node. There are three main classes of activation functions which have been developed and used: Binary: The output is hardlimited to binary [0, 1] or bipolar [-1, 1] values.

III. GENETIC ALGORITHMS

Genetic algorithms are a highly parallel mathematical algorithms that transform a set(population) of mathematical objects(typically fixed-length binary character strings), each with an associated fitness value, into a new set(population) of mathematical objects. The transformation performed by GAs consists of naturally occurring genetic operations and the Darwinism of reproduction and survival of the fittest.

The application of GAs to control problem can be classified into two areas: off-line design and on-line adaptation, learning and optimization. Recently, GAs have been applied to the various control areas, e.g., system identification, PID controller design, fuzzy controller design, evolving neuro-controllers, and so on.

3.1. Genetic Reinforcement Learning Control

In the inverted pendulum problem, each real-valued string in the population is decoded to form a network with five input units, five hidden units, and one output units. The network is fully connected and this network configuration is same as that used by Anderson with the AHC algorithm. Since there are 35 links in the network, each string used by the genetic search includes 35+1 real values concatenated together. Before any input is applied to the network, the four state variables are normalized between 0 and 1. A bias unit fixed at 0.5 is also used as a fifth input to the net; a weight from the bias unit to a hidden node(or output node) in effect changes the threshold behavior of that node. The action of the neural network for a particular set of inputs is determined from the activation of the output unit.

A random start state is supplied to the network, which determines the action on the cart. The new state of the system is then calculated and reinforced as a new input to the net. This continues until a failure occurs. Feedback takes the form of an accumulator that determines how long the pole stays up and how long the cart avoids the end of the track; this is used as a relative measure of fitness.

Learning is stopped when a network was found that was able to maintain the system without

generating a failure signal for 120,000 time steps. One potential problem with such a simple evaluation criterion is that a favorable or unfavorable start state may bias the fitness ranking of an individual net. In other words, the evaluation function is noisy. We would like to assign a fitness value to a string based on its ability to perform across all possible start states.

IV. SIMULATION RESULTS

The experiments of most researchers only attempt to learn to balance the pole within the 12-degree range. Additional experiments were carried out with failure signals occurring at two different positions of the pole: 12 degrees and 36 degrees. In the implementation used here, changing the angle at which the failure signal occurs also changes the range of the input variable representing the pole angle. We are therefore learning using a different representation of state space. The 12-degree resiriction means that the inverted pendulum problem has a solution that is approximately linear. At 36 degrees the problem is nonlinear and contains many start states where it is impossible to balance the pole.

One fact that emerged from these tests is that AHC sometimes fails to learn. The genetic algorithm converged to a solution in every experiment, regardless of whether the failure signal occurred at 12 and 36 degrees.

Figure 4.1 and Figure 4.2 shows results for an AHC network and a genetically trained network using a failure signal occurring at 12 degrees during learning. These plots illustrate the tracking control behavior over time. During training, the output is determined probabilistically depending on the activation of the output unit. During testing, the action applied to the system is obtained by deterministically thresholding the activation value of the output unit. If the activation value is greater than 0, then output 1 and push right; if it is less than or equal to 0, then output -1 and push left.

If the pole is vertical and the cart is centered and the velocities are 0, then all state variables will have the normalized value 0. When the system is started in an ideal state, then a successfully trained network will maintain the state variables close to the 0 level. It is not possible to balance the pole and avoid the track terminals from all possible, a perfectly trained network should drive all state variables back to the 0 level representing the ideal state of the system. In Figure 4.1 and 4.2, the cart is at the far right end of the track with the pole learning 32degrees to the left; the 12-degree failure signal is not suitable for these tests. The cart velocity and pole velocity are initialized 0. This initial state constitutes a position from which it is difficult for the system to recover. Both the AHC network and the genetically trained network used to produce these graphs are the best networks obtained for the 12-degree problem. In the case of Figure 4.2, the genetically trained network gets all of the input variables into tolerable ranges fairly quickly, whereas the AHC network takes longer. The AHC network quickly damps pole velocity and reduces oscillation in the pole position, however at that time, the cart almost crashes into the opposite end of the track. The genetically trained network handles problems with starting pole angles beyond 32 degrees, but the AHC network does not. Figure 4.3 and 4.4 show results for an AHC network and a genetically trained network using a failure signal at 36 degrees during learning. These plots indicate that both AHC network and the genetically trained network exploit similar information to determine the output activation levels and that they employ similar control strategies. The networks trained at 36 degrees proved to be more similar across a wider range of start states, but as the difficulty of the initial start states is increased the AHC networks fail sooner than the genetically trained networks. In these plots, the system is started with cart in the same far right position and the pole learning 35 degrees to the left. Cart velocity and pole velocity are initially 0. These plots make it evident that both networks track pole velocity by varying the magnitude of the output value. The correlation between pole velocity and the output activation is not as discernible in the first 50 to 100 time steps because the system is recovering from a difficult initial situation; correlation between the pole velocity and the output activation is much more pronounced as the networks begin to bring the system under control. Also notable is that cart velocity and pole velocity tend to be negatively correlated. Given the input definitions used in our experiments, cart velocity and pole velocity have a similar, but opposite relationship.

V. CONCLUSIONS

In this thesis, we showed genetic algorithms can be used for optimizing neural network topology and connection weights. In addition, we presented the optimized neural network was good

for solving nonlinear control problem. The performance of the proposed system was confirmed by applying it to the inverted-pendulum control problem.

There are several differences in the information used by the AHC and genetic algorithms, at least in the learning component. In both cases, the action network requires input information about the current state in order to produce an output. For the AHC algorithm, the evaluation network also requires state information to learn, since without state information there is no prediction of failure. But the genetic algorithm does not require state information at each time step; it only needs feedback about how long the pole stayed up in order to rank competing sets of weights. The reinforcement back-propagation that occurs in the AHC algorithm means that learning continues even when failures are not occurring. Futhermore, the use of the temporal difference method means that the evaluation network is not also being updated, even failures are not occurring. In our genetic approach, updates to the action network occur only after one or more failures: learning is not continuous. Another difference is that the genetic approach used in the experiments will assign an equal evaluation to two networks that avoid failure for an equal number of time steps. However the AHC algorithm evaluates networks by the trajectory of states by that are experienced. The evaluation associated with any two networks would differ when the AHC algorithm is used, favoring the network that drives the cart-pole through more highly valued states. The restriction of the search to highly valued states may also explain why the performance of AHC networks did not improve when the stricter stopping criterion was used.

Another difference is the lack of success using the AHC algorithm when the failure signal occurs at wider pole bounds. This possibly results from the combination of the incremental learning algorithm used to adjust the weights and the way the AHC evaluation network generalizes. A good prediction of failure is hard to learn when the majority of start states lead to failure. Without a good failure prediction function, a successful control strategy cannot be learned using the reinforcement back-propagation. The genetic algorithm, because it ranks each network based on performance, is able to ignore those cases where the pole cannot be balanced; only the successful cases will obtain the chase to engage in genetic reproduction. For the AHC evaluation network, however, the preponderance of failures may cause all states to overpredict failure. This problem can be corrected either by selectively sampling the space to extract a better balance of success and failure information or by tuning the AHC algorithm to place more emphasis on positive results and less on failure.

Overall, proposed method can be used for nonlinear control problems. Improvement for more nonlinear and complicated problems is the future work.

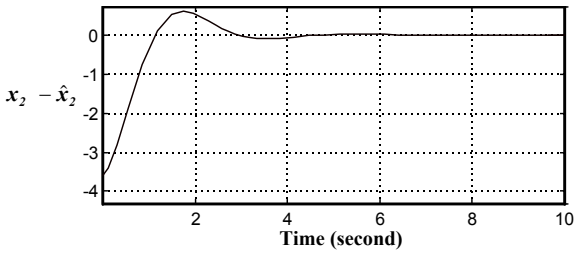

Figure 4.1 Control results by AHC network for 12°problem.

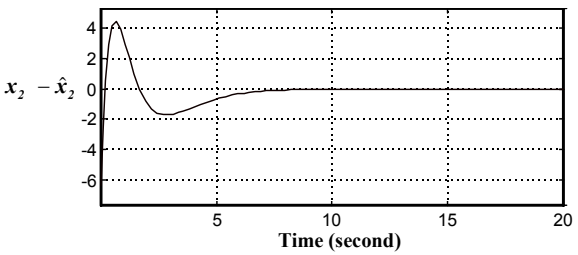

Figure 4.2 Control results by genetically optimized network for 12°problem.

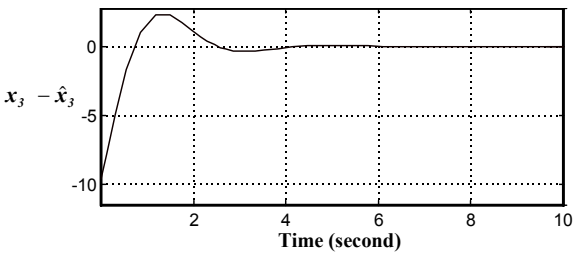

Figure 4.3 Control results by AHC network for 36°problem.

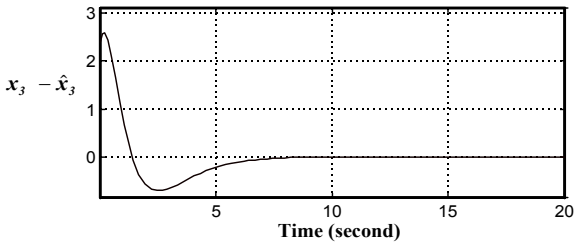

Figure 4.4 Control results by genetically optimized network for 36°problem.

REFERENCES

[1] Simon Haykin, *Neural Networks*, Macmillan, 1994.

[2] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed., Springer-Verlag, 1996.

[3] Gene F. Franklin, J. David Powell, Abbas Emami-Naeini, *Feedback Control of Dynamic Systems*, 3rd ed., Addison-Wesley, 1994.

[4] W. R. Kolk and R. A. Lerman, *Nonlinear System Dynamics*, Van Nostrand Reinhold, 1992.

[5] Katsuhiko Ogata, *Modern Control Engineering*, 2nd Ed., Prentice Hall, 1990.

[6] Leslie M. Hocking, *Optimal Control: An Introduction to the Theory with Applications*, Oxford University Press, 1991.

[7] LiMin Fu, *Neural Networks in Computer Intelligence*, McGraw-Hill, 1994.