비선형 시스템의 불확실성을 보상하는 신경회로망 제어

김 성 우*, 홍 선 기, 이 주 장
한국과학기술원 전기 및 전자공학과

Uncertainty-Compensating Neural Network Control for Nonlinear Systems

Sung-Woo Kim, Sun-Gi Hong and Ju-Jang Lee

Department of Electrical Engineering, KAIST

Abstract — **A novel neural network learning and control scheme is presented. The neural network learns uncertainty of the system and compensates it. To train the network, uncertain compensation learning is derived. Efficiency of the proposed learning and control are verified through the simulation study.**

Indexing terms:

Uncertainty compensation, Neural networks, Learning, Control.

I. Problem Statements

Consider a class of single-input nonlinear time-varying systems described by

$$x^{(n)} + f(x, \dot{x}, \cdots, x^{(n-1)}, t) = u, \qquad (1)$$

where x is a scara output of interest of the system, $f(\cdot)$ is an unknown nonlinear function, and u is a scara control input [6]. Define the state vector $\mathbf{x} = [x, \dot{x}, \ldots, x^{(n-1)}]^T$ and desired state vector $\mathbf{x}_d = [x_d, \dot{x}_d, \ldots, x_d^{(n-1)}]^T$. Then the control problem is for the state $\mathbf{x}$ to follow the desired state $\mathbf{x}_d$ in the presence of uncertainty on the function $f(\cdot)$. In other words, if we define the error state vector $\mathbf{x}_e = \mathbf{x}_d - \mathbf{x}$, then the control problem is equivalent to design a controller to make $\mathbf{x}_e \to 0$.

If the uncertain nonlinear function $f(\cdot)$ is known exactly, such a control problem can be easily solved by designing a controller:

$$u^* = f(\mathbf{x}, t) + x_d^{(n)} + \mathbf{k}^T \mathbf{x}_e, \qquad (2)$$

where $\mathbf{k} = [k_1, k_2, \ldots, k_n]^T$ is a constant gain vector. We can make the closed-loop system response stable when k_i's are chosen as the polynomial $h(s) = s^n + k_n s^{(n-1)} + \cdots + k_2 s + k_1$ is a Huriwtz.

However, this controller can not be implemented because the nonlinear function $f(\cdot)$ is not known. Hence, we will design a neural network controller compensating the uncertain nonlinearity.

II. Uncertainty-Compensating Neural Networks

The main objective is to compensate uncertainty using neural networks and to achieve a good control performance. To obtain such a control objective, we design a controller of the form:

$$u = \hat{u} + u_{nn} = \hat{f} + x_d^{(n)} + \mathbf{k}^T \mathbf{x}_e + u_{nn}, \qquad (3)$$

where $\hat{u} = \hat{f} + x_d^{(n)} + \mathbf{k}^T \mathbf{x}_e$, $\hat{f}$ is the estimation of f, and u_{nn} is the neural network controller.

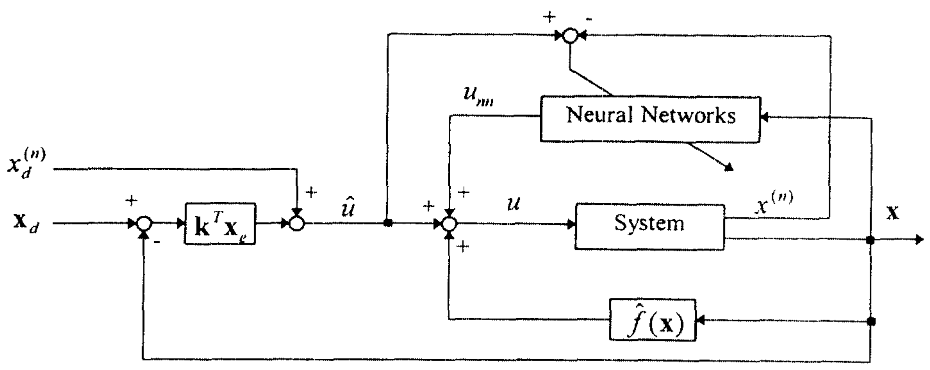


Fig. 1: Uncertainty compensation learning and control

Applying the controller (3) to the system (1), we obtain the closed-loop error dynamics:

$$x_e^{(n)} + \mathbf{k}^T\mathbf{x}_e = \Delta f - u_{nn}, \qquad (4)$$

where $\Delta f = f - \hat{f}$. From this result, we can see that if the neural network controller compensates uncertainty Δf, then the right-hand side of (4) vanishes, so does the error dynamics.

To see how this can be achieved, consider the *learning rule* of neural networks:

$$\frac{dw_j}{dt} = -\eta\frac{\partial E}{\partial w_j}, \qquad (5)$$

where w_j is a typical parameter (weight or bias) of neural networks, η is a constant learning rate, and E is a cost function to be minimized. Since the control objective is to minimize $||u_{nn} - \Delta f||$, it is sufficient to choose

$$E = \frac{1}{2}||u_{nn} - \Delta f||^2. \qquad (6)$$

From this, the learning rule can be rewritten as

$$\begin{aligned}\frac{dw_j}{dt} &= -\eta\left(u_{nn} - \Delta f\right)\frac{\partial u_{nn}}{\partial w_j}\\ &= \eta\left(x_e^{(n)} + \mathbf{k}^T\mathbf{x}_e\right)\frac{\partial u_{nn}}{\partial w_j}, \qquad (7)\end{aligned}$$

where (4) is substituted into (7). We call this *uncertainty compensation learning* (UCL), since the neural network learns and compensates uncertainty by this learning rule (Fig. 1). The UCL structure has the following advantage:

- Teaching signals or desired outputs of neural networks are not necessary. Only preparing the error dynamics is sufficient for learning.
- Error back-propagation through the plant [5] or the plant identification model [3] is not required.
- Control and learning can be conducted simultaneously.

III. Simulations

We apply the proposed neural network learning and control to an uncertain nonlinear time-varying system:

$$\ddot{x} + [0.5 + 0.2\sin(t)]\dot{x} + 3.2\cos(x) = u \qquad (8)$$

which can be thought of as an *inverted pendulum* on the presence of viscous friction. We assume there is no adequate estimation of f, so we only set $\hat{f} = 0$. In this case, the uncertainty becomes the nonlinear function itself, $\Delta f = f = [0.5 + 0.2\sin(t)]\dot{x} + 3.2\cos(x)$.

The following control input is applied to (8):

$$u = (\ddot{x}_d + k_2\dot{x}_e + k_1x_e) + u_{nn} \qquad (9)$$

where k_1 and k_2 are chosen as 100 and 20, respectively.

The neural network controller is trained by the UCL rule:

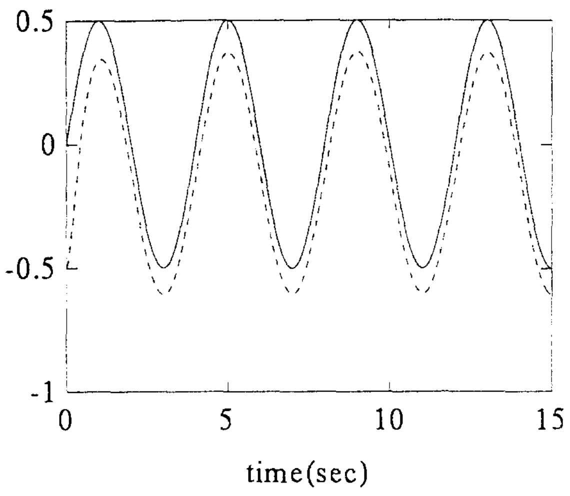
$$\frac{dw_j}{dt} = \eta\left(\ddot{x}_e + k_2\dot{x}_e + k_1 x_e\right)\frac{\partial u_{nn}}{\partial w_j}, \qquad (10)$$

where the learning rate is chosen as $\eta = 0.02$. Neural networks here are layered networks that have 3, 10, 6, and 1 number of nodes in input, first hidden, second hidden, and output layers, respectively. Hidden nodes have nonlinear activation functions, tanh(·), while output nodes have linear activation functions.

Following figures show the simulation results for the desired trajectory $x_d(t) = 0.5\sin(t)$. To show the efficiency of the learning control method, we also plot the response when only $\hat{u} = \ddot{x}_d + 20\dot{x}_e + 100x_e$ is applied. Fig. 2 shows that the controller $\hat{u}$ alone cannot decrease the tracking error. On the contrary, it shows the control performance becomes better and better as the neural network learns (Fig. 3) and compensates the nonlinear function more and more accurately (Fig. 4).

IV. Conclusion

The main idea of neural network learning control presented here is to compensate uncertainty. Neural networks learn uncertainty by the proposed uncertainty compensation learning. The presented learning scheme does not require desired outputs of the neural network, nor back-propagation of the error through the plant. Furthermore, learning is proceeded while controlling the plant.



Desired and actual trajectory (solid and dashed line, respectively).

Fig. 2: Response of the case $\hat{u} = \ddot{x}_d + 20\dot{x}_e + 100x_e$ only is applied.

References

[1] Sung-Woo Kim, Sun-Gi Hong, Tae-Duck Ohm and Ju-Jang LEE, "Neural network identification and control of uncertain systems using supervisory control while learning," in *Proc. IEEE Int. Conf. Neural Networks*, 1994, pp. 2500-2505

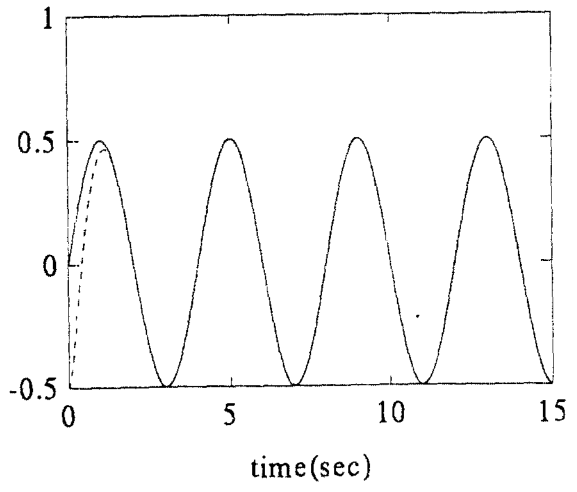
[2] Sung-Woo Kim and Ju-Jang LEE, "Neural network control by learning the inverse dynamics of uncertain robotic systems," *Journal of Control, Automation, and Systems Engineering*, 1995, vol. 1, no. 2, pp. 83-93 (in Korean)

[3] K. S. Narendra, and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, 1990, vol. 1, no. 1, pp. 4-27

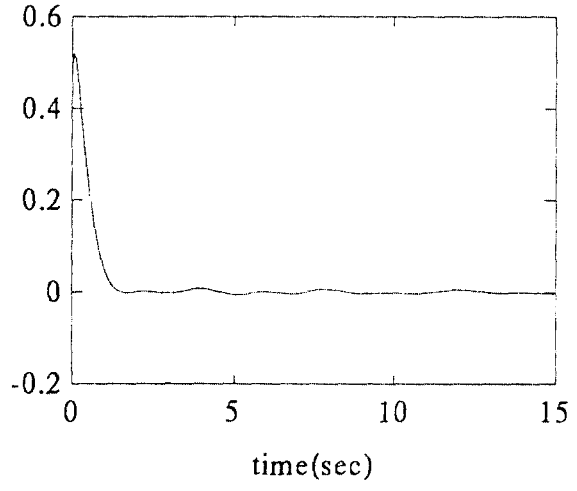
[4] H. Miyamoto, M. Kawato, T. Setoyama, and R. Suzuki, "Feedback-error-learning neural network for trajectory control of a robotic manipulator," *Neural Networks*, 1988, vol. 1, pp. 251-265

[5] D. Psaltis, A. Sideris and A. A. Yamamura, "A multilayered neural network controller," *IEEE Control Systems Magazine*, 1988, pp. 17-21

[6] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*, 1991, Englewood Cliffs, NJ, Prentice Hall



(a) Desired and actual trajectory (solid and dashed line, respectively).



(b) Tracking error ($x_e = x_d - x$)

Fig. 3: Responses of the case $u = \hat{u} + u_{nn}$ is applied.

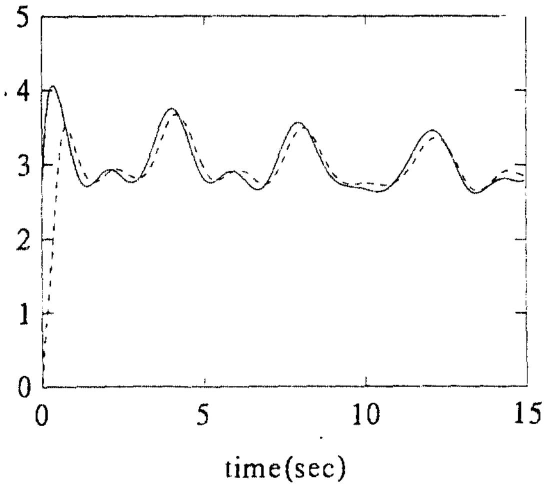


Fig. 4: $f(\mathbf{x}, t)$ and the neural network output (solid and dashed line, respectively).