CONTROL BY A POLICY-AND-EXPERIENCE-DRIVEN NEURAL NETWORK

MASARU ISHIDA

Research Laboratory of Resources Utilization, Tokyo Institute of Technology, 4259 Nagatsuta, Midori-ku, Yokohama 227

Key Words: Process Control, Neural Network, Self-Tuning, Global Policy Learning, Local Experience Learning

Introduction

A new scheme to control processes, based on a layed neural network, was proposed. Adaptive and self-tuning features were achieved by modifying the weight for connections in the network through learning. As the learning sets of data, the relation gotten by experience and a global control policy were used simultaneously. The applicability of this scheme was confirmed by simulation of level control.

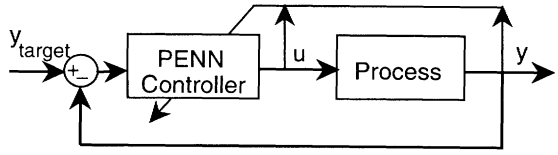
* Received November 8, 1991. Correspondence concerning this article should be addressed to M. Ishida.

1. Proposed Control System and Its Self-Tuning Mechanism

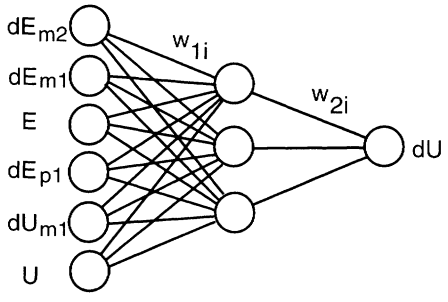
1.1 System structure

Figure 1(a) shows the connection of the controller and the given process. The controller gives a new value for the control variable u based on the present and previous values of u, the controlled variable y and the target value y_{target}. **Figure 1(b)** shows a neural network to compose the controller for a single-input and single-output process.

The network is composed of layers. The input layer consists of six units. Four of them are related to the process output variable, y, and the other two to the control variable, u.



(a) Control system structure



(b) Structure of the neural network

Fig. 1. Control system structure and neural network for the controller

The control variables, u_j, are assumed to take zero or positive values and they are normalized by $U_j = u_j/u_{\max}$, where $u_{\max}$ is the maximum allowable value for u. The controlled variables y_j are converted to errors by

$$e_j = y_{\text{target}} - y_j \,.$$

Then they are normalized by their maximum value, $e_{\max}$, as follows:

$$E_j = 0.5 + 0.5 \times e_j/e_{\max} \,.$$

Hence, U_j and E_j take values between zero and unity.

In Fig. 1(b), U_j and E_j at the present time are inputted. For the values sampled at the previous times (denoted by subscripts m1 or m2) or predicted for the data at the next sampling time (denoted by subscripts p1), their differences to those at the present time are taken, say $dE_{\text{m1}} = E_j - E_{j-1}$, $dE_{\text{m2}} = E_j - E_{j-2}$ and $dE_{\text{p1}} = E_{j+1} - E_j$. The reason for such selection will be clarified later. The value of dE_{p1} is given by the desired path of E against time t.

For the hidden layer, there are three units in Fig. 1(b), and from the output layer, dU is given. Hence the next value of the control variable is obtained by the relation, $U_{j+1} = U_j + dU_j$.

1.2 Learning mechanism

To utilize the above net, the learning mechanism that gets the best values for weight w_{ji} is very important. We have noticed from our experience that some processes have a specific feature in control. This is that the trajectory of the vector (dE_{m2}, dE_{m1}, E, E_{p1}, U_{m1}, U, dU) passes through some difinite routes and does not cover the whole space of the seven-dimensional domain. This does not mean that the relation between dE_{m2}, dE_{m1}, E, dE_{p1}, U_{m1}, U and

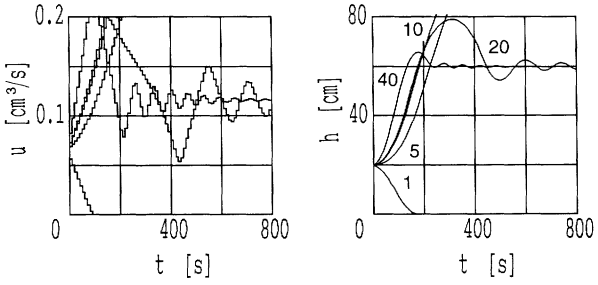


Fig. 2. Change of water level from 20 cm to 60 cm based on local learning through experience

dU which was gotten each time when a new control variable was given is useless. It does mean that such information through previous experience is local and limited. Hence more global information should be given simultaneously.

In this study, learning by error back-propagation[1] is done by the following two procedures: (1) global learning based on control policy and (2) local learning based on previous experience. The first learning mechanism uses an appropriate number of rules and gives information which covers the whole control space. Hence we may say that the former learning gives the control policy. For this purpose we do not need any detailed information. On the other hand, detailed information of the process features can be gotten from the previous experience or the previous runs. This latter learning is done by the observed relation between the process inputs and process outputs. Hence this controller may be called a policy-and-experience-driven neural network (PENN) controller.

2. Experiment by Simulation

The above control scheme was examined by simulation of water-level control when the water level is changed from 20 cm to 60 cm. It is slightly nonlinear but the time constant is calculated as 216 s. For a step change of the feed rate u, the level h becomes 53.2 cm at 400 s and 58.6 cm at 800 s.

Figure 2 shows that the simulated result when the control was done only through experience. At $t = 0$, the initial values of E and U, $dE_{\text{m2}} = dE_{\text{m1}} = dU_{\text{m1}} = 0.5$ (*i.e.*, zero) and the value of dE_{p1} are given, and dU is obtained by the network. For the new U calculated from U and dU, the process proceeds for a sampling period dt, which is set as 10 s, and the new level h is obtained. Note that dE_{p1} is calculated to satisfy the desired change of height against time. In this study, the following simple relation is used:

$$E_{j+1} = E_j/(1 + dt/T_I)$$

A small value of T_I results in rapid change of height and T_I is set at 10 s in this study. At $t = dt$, the same procedure is repeated.

Figure 2 shows the results for the 1st, 5th, 10th, 20th and 40th runs. For the first run, the network had very poor information of the process. Hence the network did not understand what dE_{p1} means, resulting in a decrease of water level. But by learning the relation between dE_{p1} and dU, h began to increase from the second run. As shown in Fig. 2, however, it increased too much, much higher than the target level of 60 cm. As leaning continued, control became better and in the 40th run very satisfactory control was achieved.

It is remarkable that the characteristic feature of the process was grasped by self-learning, *i.e.*, learning from previous experience. But this control scheme does not converge and often gives quite different or peculiar results. This uncertainty is the scheme's greatest disadvantage.

Figure 3 shows also the 1st, 5th, 10th, 20th and 40th runs when only global learning, based on the following five rules, was performed:

(1) For $dE_{m2}=dE_{m1}=dE_{p1}=dU_{m1}=0$, $E=0$, $U=$ U, then dU$=0$

(2) For $dE_{m2}=dE_{m1}=0$, $E=1$, $dE_{p1}=1/(1+dt/T_I)-1$, $dU_{m1}=0$, $U=U$, then $dU=1$

(3) For $dE_{m2}=dE_{m1}=-1$, $E=0$, $dE_{p1}=0$, $U=U$, $dU_{m1}=0$, then dU$=-1$.

(4) For $dE_{m2}=dE_{m1}=0$, $E=-1$, $dE_{p1}=-1/(1+dt/T_I)-(-1)$, $dU_{m1}=0$, $U=U$, then $dU=-1$

(5) For $dE_{m2}=dE_{m1}=1$, $E=0$, $dE_{p1}=0$, $U=U$, $dU_{m1}=0$, then $dU=1$

where these expressions use the range of -1 through 1 instead of 0 through 1 so as to be understood easily.

For each sampling period, only the above five rules were taught, while previous experience was not taught.

It was astonishing that the first run gave good control. But it was not maintained; oscillation began to be observed and was intensified.

The above five rules gave the global information for control or the control policy. The network learned the above policy but did not grasp the detailed features of the process. It is found from this experiment that abnormal behavior can be avoided by the learning of global policy.

Figure 4 shows the case by the PENN controller based on both learning mechanisms. Before the first run, the above five rules were taught once in advance. Then global learning of one of the above five rules and experience learning were performed for each sampling period. The sequence of the choice of rules for global learning was: 1,2,1,3,1,4,1,5, indicating that the first rule for the steady state was taught frequently.

For the first run, the water level changed in a similar way to that shown in Fig. 3. This means that the effect

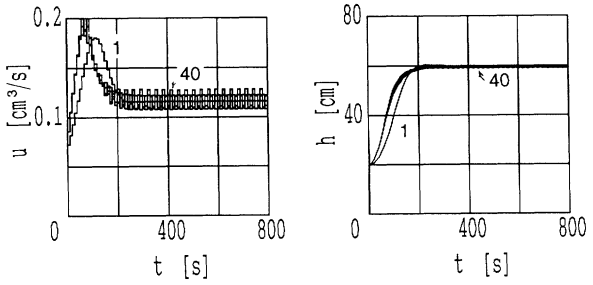


Fig. 3. Change of water level from 20 cm to 60 cm based on global learning of control policy

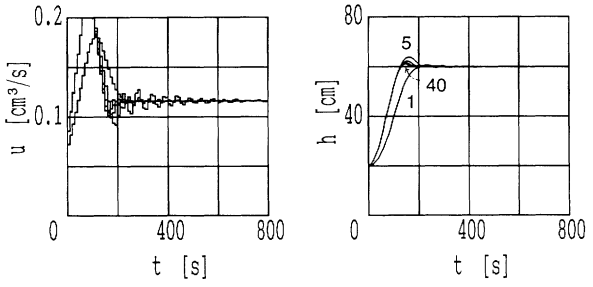


Fig. 4. Change of water level from 20 cm to 60 cm by the proposed policy-and-experience-driven control

of the global learning was significant. But the net began to learn the features of the process through experience and gave quite satisfactory control. It is to be noted that the variables for the input layer of the neural network were selected to make both global policy learning and local experience learning possible.

In Fig. 3 it is observed that further learning gave a rather worse result. For PENN control, however, further learning, say 300 runs, gave gradually better control. For example, the overshoot observed for h was 1.2 cm for the 40th run but diminished to 0.9 cm for the 300th run. This is quite an important feature of the PENN control, since this controller may follow a gradual change in process characteristics by continuing learning. This controller has no parameters to be tuned in advance, a fact that supports the adaptability of this controller.

In this prompt paper the basic feature of the PENN controller has been demonstrated. When we want to simplify the controller, we may omit dE_{m2}, U, and/or dU_{m1} units from the input layer. When we want to apply it to a more complicated process, we may add the appropriate units for input, hidden and output layers. The effect of noise in the controlled variable y, the time lag of the process, interaction between variables, and comparison of various methods using neural networks are also important and will be discussed later.

Conclusion

The use of both local experience and global policy in control based on a neural network was proposed.

This controller has no specific parameters to be tuned in advance and hence is quite adaptive. The potential advantage of the controller was demonstrated for a case study of water-level control.

Literature Cited

1) Rumelhart, D. E., *et al.*: "Parallel Distributed Processing" vol. 1, MIT Press (1986).