Computers chem. Engng, Vol. 16, No. 4, pp. 241–251, 1992
Printed in Great Britain.

0098-1354/92 $5.00 + 0.00

PROCESS CONTROL VIA ARTIFICIAL NEURAL NETWORKS AND REINFORCEMENT LEARNING

J. C. HOSKINS and D. M. HIMMELBLAU
Department of Chemical Engineering, The University of Texas at Austin, Austin, TX 78712-1062, U.S.A.

(Received 21 January 1991; final revision received 12 August 1991; received for publication 10 December 1991)

Abstract—In the training of artificial neural networks, reinforcement learning substitutes a qualitative binary target of "success" or "failure" for the quantitative error criterion of supervised learning. By this method of learning control can be established for the special case of operation in which no objective function exists. If no model whatsoever exists of the dynamics of a chemical process, it still may be possible to train an artificial neural network to control the process much as a human being would learn by trial and error.

We describe a network architecture for process control, and explain quantitatively how the weights on the connections in the network can be adjusted to yield the desired control action. An example of a nonlinear CSTR is used to illustrate the performance of the proposed net and compare it with that of a tuned PID controller. The net can be trained to meet almost any criteria selected for control as long as the criteria can be expressed in the form of inequality constraints, but requires extensive, and perhaps excessively long, training on a serial computer to do so.

1. INTRODUCTION

Artificial neural networks (ANN for short) have been investigated as tools for control by authors in chemical engineering, robotics, neuroscience, electrical engineering and so on. These studies have posed the control problem in the same form as classical control but with ANN modeling the process and/or the controller. We discuss here the use of ANN for control for a special niche of problems, those for which no objective function exists except in the form of the evaluation of the success or failure of control actions. Then the goal is to develop a function that maps the current state of a process into suitable control actions through experience (Anderson, 1988).

How can a control problem have no objective function? At one extreme, no quantitative measure of merit can be formulated for one reason or another. At the other extreme, too many noncompatible quantitative criteria exist that cannot be aggregated. For example, most books on control list 5 or 10 desirable criteria all of which cannot be combined in a single scalar (or even vector) objective function, but can be posed as inequality constraints (continual reduction of variance, no oscillation, minimum or no overshoot and so on).

One technique used in such circumstances to adjust the weights in an ANN is reinforcement learning. No gradients can be computed so that the degree of information available for control is substantially degraded relative to that ordinarily available for designing control strategies. Reinforcement learning is quite analogous to the way a human being learns about a complex process the results of which cannot be evaluated quantitatively, i.e. learning how to ride a bicycle vs learning how various factors affect the yield of a chemical reaction. Reinforcement learning is not an efficient way of learning and is not recommended for developing control strategies in general, but under some circumstances may prove to be the only fallback technique that works. In so far as we can determine, only one preliminary study (Hoskins and Himmelblau, 1988) has focused on the application of reinforcement learning to chemical engineering problems.

While reinforcement learning does not involve a quantitative objection function, it does involve a "critic" that deems which control actions are acceptable (success) and which are not (failure). A long sequence of actions take place before a failure is declared resulting in a difficult problem of deciding which actions in the sequence contributed to the failure. Barton *et al.* (1983) proposed the AHC (Adaptive Heuristic Critic) algorithm which generates successful action sequences by developing two functions: (a) an action function that maps current stages into control actions; and (b) an evaluation function that maps current states into an evaluation of those states so as to yield a reinforcement signal. Other versions of reinforcement learning exist. A brief review of the history and strategies of

reinforcement learning can be found in Section 1.9 of Miller *et al.* (1990), and a more detailed review exists in the tutorial of Williams (1989).

In applying ANN with the weights adjusted by reinforcement learning, you develop a control strategy by trial and error. The strategy meets the constraints, and thus is a feasible strategy, but is not unique and is not necessarily optimal since optimal cannot be defined directly without an objective function being specified. A dynamic model of the process and controller do not have to be developed either from theoretical principles or via empirical models because the ANN uses historical, or current, data to develop empirically a functional representation of a satisfactory control system. The advantage of using an ANN for nonlinear systems instead of a regression function is that the form of the function does not have to be determined *a priori*. A disadvantage often is that the number of weights in the ANN is large, and consequently a large number of data sets have to be available to train the ANN, and the training time is long.

We first describe the ANN network used to accomplish the evaluation and action functions, and then discuss an application to control of a nonlinear CSTR.

2. ANN CONTROLLER

The ANN controller described in this section differs from traditional PID or adaptive controllers in that is assumes no predefined control law, but learns a control law from operating experience. In addition, the ANN controller implementation can be highly parallel, thus exhibiting the potential to operate in real time. We will describe a particular ANN controller (which we will call the Anderson ANN controller) to demonstrate the potential of this approach to special cases of chemical engineering process control.

In implementing an ANN controller, the most important phase is the learning procedure. An extension of the AHC algorithm (Barto *et al.*, 1983) which was developed by Anderson (1986, 1987, 1988) that incorporates multilayer networks was used to train the ANN controller used in this work. The Anderson ANN controller illustrated in Fig. 1 consists of two ANNs called, respectively, the *evaluation network* and

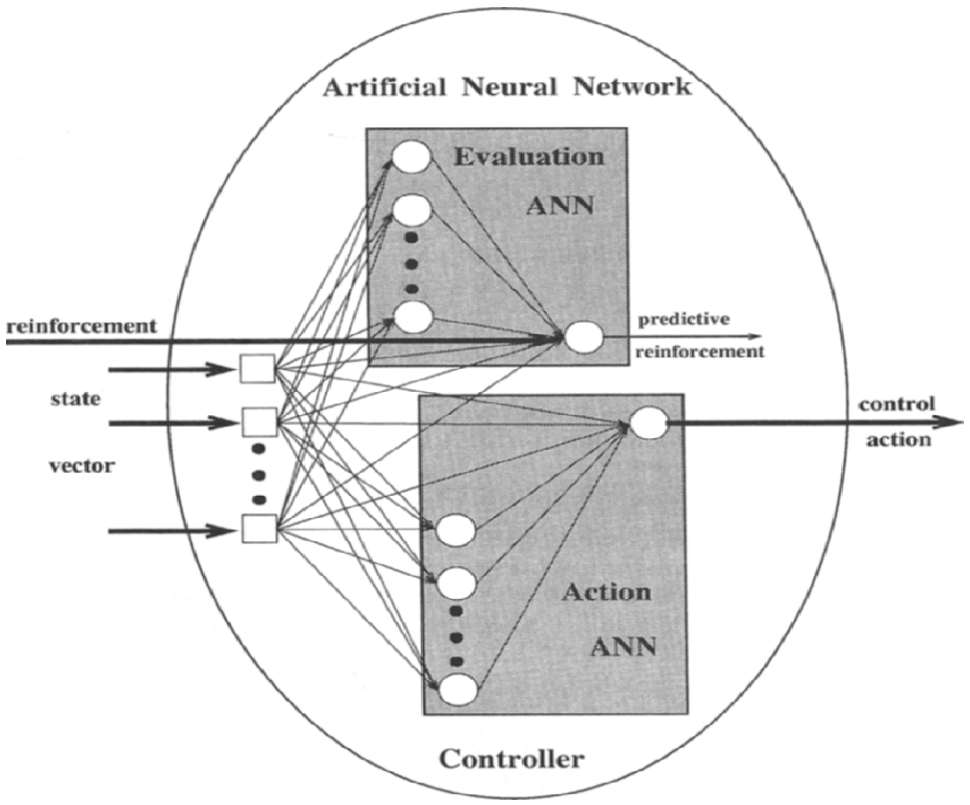


Fig. 1. Anderson ANN controller.

the *action network*. The purpose of the multilayer evaluation network is to learn an *evaluation function* that maps the given state of the process into an evaluation of that state (i.e. was the action that led to this state good or bad?). The purpose of the multilayer action network is to generate the control action to apply for a given state of the process to be controlled.

The networks in the Anderson ANN controller receive the inputs from the process being controlled. To avoid having too many layers and hidden nodes in the network, and to avoid the difficulties of integrating the learning procedures for both networks, the action and evaluation networks do not share hidden nodes. Below we describe the evaluation and action network structures and the learning procedures which incorporate the AHC and back-propagation methods.

2.1. Evaluation network

The function of the evaluation network is to develop an implicit mapping from the external reinforcement input plus the sensor measurements to a scalar predicted reinforcement output as shown in Fig. 2. From these inputs, the evaluation network computes an improved reinforcement signal that is an evaluation of whether the current state is a desired state or not, thus providing an evaluation of the action leading to the current state. Then the weights in the evaluation network are changed in proportion to magnitude of the input values while the improved reinforcement signal provides the direction (i.e. positive or negative) of change. These steps are explained in detail below.

2.1.1. Evaluation network output. The output of the evaluation network (see Fig. 2) is computed in the following manner [after Anderson (1986)]. First the outputs of the hidden layer are calculated in a feedforward fashion. The output y_j at time $t+1$ of hidden node j (note that the value of y_j at time $t+1$ depends on the values of the inputs at both $t+1$ and t) is calculated using the values of its weights **a**, at time t and input vector **x** at time $t+1$ as follows:

$$y_j[t+1] = g\left(\sum_i a_{ij}[t]x_i[t+1]\right), \tag{1}$$

where g is the sigmoid logistic function in equation (1). The value v at $t+1$ is calculated using the values of the weights **b** on the input connections at time t and input vector **x** at time $t+1$ in combination with the values of the weights **c** on the connections from the hidden nodes at time t and the output vector **y** of the hidden nodes at time $t+1$ as follows:

$$v[t+1, t] = \sum_i b_i[t]x_i[t+1] + \sum_i c_i[t]y_i[t+1]. \tag{2}$$

The improved reinforcement $\hat{r}$ is calculated using the output of the evaluation network via a method developed by Sutton (1984) for tasks involving

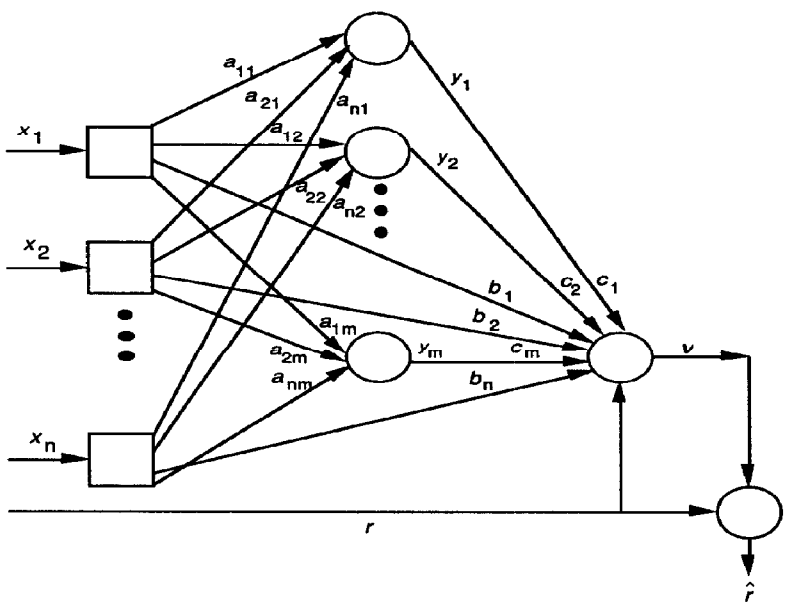


Fig. 2. Evaluation network receiving the process input and external reinforcement.

distinct *trials*, where a trial consists of the following steps:

(1) setting the state of the problem to a *start state*;
(2) letting the learning system and the process interact; until
(3) a *failure state* is encountered, signaled by a particular external reinforcement value.

Following Sutton and Anderson, $\hat{r}$, for trial-based tasks is defined to be:

$$\hat{r} = \begin{cases} 0, & \text{if state at time } t+1 \text{ is a start state,} \\ r[t+1] - v[t], & \text{if state at time } t+1 \text{ is a failure (goal) state,} \\ r[t+1] + \gamma v[t+1] - v[t], & \text{otherwise,} \end{cases} \quad (3)$$

where r is the external reinforcement value [see equations (18) and (19) in Section 4 for the calculation of r], γ is a decay factor with values $0 \leqslant \gamma < 1$ and v is calculated using equation (2).

2.1.2. Evaluation network learning. The weights of the evaluation network are updated using a combination of the reinforcement learning procedure and the backpropagation learning procedure equations. The improved reinforcement $\hat{r}$ assumes the role of the error leading to a new definition of δ:

$$\delta_j[t-1] = \hat{r}[t], \quad (4)$$

thus it follows that the equation which adjusts the values of the weights becomes:

$$b_j[t+1] = b_j[t] + \beta \hat{r}[t+1] x_j[t] \quad (5)$$

and

$$c_j[t+1] = c_j[t] + \beta \hat{r}[t+1] y_j[t], \quad (6)$$

where β corresponds to the learning rate η and $\hat{r}$ corresponds to the error signal such that if $\hat{r}$ is positive (implying a positive change in state evaluations), the weights on the connections entering the node are altered to increase the output of the network v; otherwise the weights are adjusted to decrease the output. Next the error (i.e. $\hat{r}$) is backpropagated from the output node to the hidden nodes. The update expression for the weights on the connections to the hidden nodes is:

$$a_{ij}[t+1] = a_{ij}[t] + \beta_h \hat{r}[t+1] y_i[t] \times (1 - y_i[t]) \operatorname{sgn}(c_i[t]) x_j[t], \quad (7)$$

which includes a modification by Sutton (1985) whereby β_h is decreased by using the sign of the weight which makes the learning procedure less sensitive to the value of the learning rate parameter.

In conclusion, a positive (negative) change in state evaluations [i.e. a positive (negative) $\hat{r}$] results in an increase (decrease) in weight values proportional to the corresponding positive (negative) input values on the preceding time steps. In this way, the evaluation of the preceding state is altered, effectively shifting evaluations to earlier states.

2.2. Action network

The action network receives both an improved reinforcement signal $\hat{r}$ from the evaluation network and the state vector of the process to be controlled. Using these inputs the action network computes a control action based on the action probabilities (i.e. the weights) and the current state of the process.

2.2.1. Action network output. The output of the action network (see Fig. 3) is computed in the following manner after Anderson (1986). First, the outputs of the hidden layer are calculated in a feedforward fashion as in the evaluation network. The output z_j of hidden node j at time $t+1$ is calculated using the values of its weights $\mathbf{d}$ at time t and input vector $\mathbf{x}$ at time $t+1$ as follows:

$$z_j[t+1, t] = g\left(\sum_i d_{ij}[t] x_i[t+1]\right), \quad (8)$$

where g is again the sigmoid logistic function in equation (1). The input for the output node is a combination of the input vector from the process $\mathbf{x}$ and the output vector from the hidden nodes $\mathbf{z}$. The output of the action network, the control action CA, is calculated based on the action probability as follows:

$$\mathrm{CA}[t] = \begin{cases} \Delta F, & \text{if } q[t] = 1, \\ -\Delta F, & \text{if } q[t] = 0, \end{cases} \quad (9)$$

where ΔF represents the change in the manipulated variable and

$$q[t] = \begin{cases} 1, & \text{if } p[t] > 0, \\ 0, & \text{otherwise,} \end{cases} \quad (10)$$

where p is calculated using the values of the weights $\mathbf{e}$ on the input connections at time t and input vector $\mathbf{x}$ at time $t+1$ in combination with the values of the weights $\mathbf{f}$ on the connections from the hidden nodes at time t and the output vector $\mathbf{z}$ of the hidden nodes at time $t+1$ as follows:

$$p[t+1] = \sum_i e_i[t] x_i[t+1] + \sum_i f_i[t] z_i[t+1] + \eta(t), \quad (11)$$

where $\eta(t)$ is a sequence of random variables selected from the mean zero Gaussian distribution with a variance of unity.

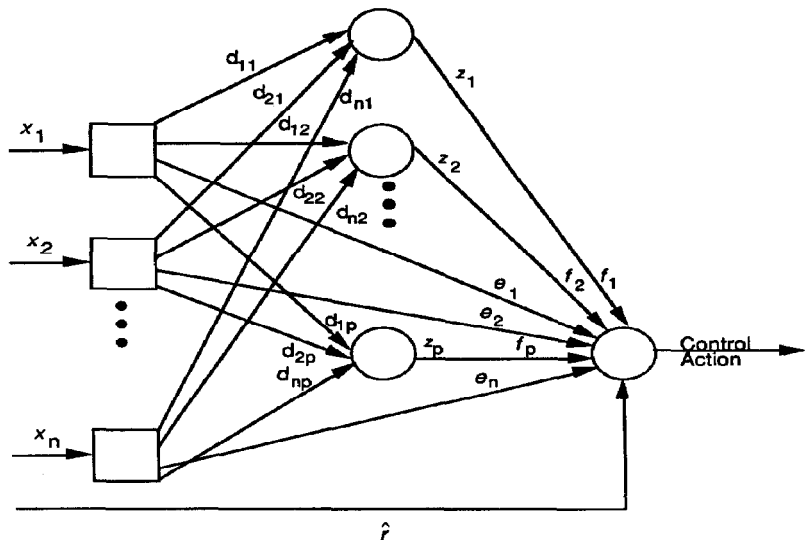


Fig. 3. Action network receiving the process input and the improved reinforcement from the evaluation network.

2.2.2. Action network learning procedure. The weights of the action network are updated using a combination of the reinforcement learning procedure and the backpropagation learning procedure equations; however, the comparison is not quite as straightforward as for the evaluation network. Following Anderson, the role of the error is played by the product of $\hat{r}$ and the difference between the previous action and its expected value. The sign of this product is an indication of whether the action probability should be increased or decreased. We can now write the expression for the error of output node j in the hidden layer as follows:

$$\delta_j[t-1] = \hat{r}[t]\,(q_j[t-1] - \mathrm{E}\{q_j[t-1]\,|\,e, f, z\}), \tag{12}$$

where E is the expectation of the arguments in the brackets. The value of $q_j[t-1] - \mathrm{E}\{q_j[t-1]|e,f,z\}$ can be viewed as a measure of the difference between action $q_j[t-1]$ and the action that is usually taken for the given values of $z_j[t-1]$ with weights $e_i[t-1]$ and $f_i[t-1]$. Thus, a frequently occurring action has more of an impact on the adjustment of weights than do other actions. Since $q_j \in \{0, 1\}$, the expected value of q_j is equal to the probability that q_j is unity, i.e.

$$\mathrm{E}\{q_j[t]|e,f,z\} = \Pr\{q_j[t] = 1\}. \tag{13}$$

For the binary-valued output of the single node in the output layer, $\Pr\{q_j[t] = 1\}$ is Ψ which results in weight update equations for the output node of:

$$e_j[t+1] = e_j[t] + \rho\hat{r}[t+1](q[t] - p[t])x_j[t] \tag{14}$$

and

$$f_j[t+1] = f_j[t] + \rho\hat{r}[t+1](q[t] - p[t])z_j[t] \tag{15}$$

where ρ is the learning rate parameter for the output node. The weight update equation for the hidden nodes is:

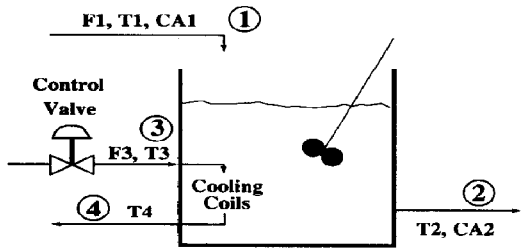
$$d_{ij}[t+1] = \rho_m d_{ij}[t] + \rho_h\hat{r}[t+1]z_i[t] \times (1 - z_i[t])\mathrm{sgn}(f_i[t])(q[t] - p[t])x_j[t], \tag{16}$$

with ρ_h the learning rate and ρ_m the momentum for the hidden nodes for the action network.

One should note that, if the difference in the errors backpropagated in equations (4) and (12) are disregarded, the learning procedures applied to the hidden units of the two networks are identical.

3. CONTINUOUS-STIRRED-TANK REACTOR SYSTEM

The task of controlling the temperature of continuous-stirred-tank reactor (CSTR) with cooling was chosen as an example of reinforcement learning for control because it is a simple well-understood system that has been used many times to demonstrate classical chemical engineering control techniques, and thus the ANN results could easily be evaluated. Certainly a CSTR would not be controlled this way—it is only used as an example—but a poorly-mixed vessel might be. Rather than use an actual CSTR, a C language program was written to simulate the CSTR process model using Franks' simulation routines (Franks, 1972) as a template. The simulated CSTR model represents a well-mixed tank reactor with cooling or



Continuous Stirred Tank Reactor with cooling

Reactor Streams 1 & 2

F1	Inlet flow rate
T1	Inlet temperature
CA1	Inlet concentration of reactant A
T2	Outlet temperature
CA2	Outlet concentration of reactant A

Coolant Streams 3 & 4

F3	Inlet flow rate, ft³/min
T3	Inlet temperature
T4	Inlet temperature

Fig. 4. Schematic diagram of the CSTR system.

heating coils and a PID controller circuit. Figure 4 illustrates the process configuration. The equations representing the dynamic behavior of the process are (see Fig. 4 for notation):

1. Energy balance for the reactor:

$$\frac{dT}{dt} = \frac{q_i}{V}(T_i - T) - \frac{\Delta H_R R_A}{C_P \rho} - \frac{Q}{C_P \rho}.$$

2. Energy balance for the internal coils (based on the arithmetic average driving temperature):

$$Q = UA\left[\frac{2q_c(\rho C_P)_c (T - T_{ci})}{2q_c(\rho C_P)_c + UA}\right].$$

3. Mass balance for the reactant:

$$\frac{dC_A}{dt} = \frac{q_i}{V}(C_{Ai} - C_A) - R_A.$$

4. Reaction kinetics (second-order kinetics were chosen to add simple but nontrivial nonlinearities to the system):

$$2A \rightarrow 2B, \quad R_A = A \exp[-E/RT]C_A^2.$$

For the experiments in the next section, a CSTR with a constant volume of 2.88 m³ was used. The cooling coils had a heat transfer area of 46.5 m² and a heat transfer coefficient of 34 kJ m^{-2} min^{-1} K^{-1} that is assumed to be independent of coolant flowrate and temperature. There was a linear valve on the coolant stream with a range of 0.0–1.7 m³ min^{-1}. The heat capacity used for all streams was 75 kJ kg mol^{-1} K^{-1} and the density used for all streams was 5.58 kg mol m^{-3}. For the second-order reaction, the frequency factor was 8×10^{15} kg mol m^{-3} min^{-1} and the activation energy was 9.33×10^4 kJ kg mol^{-1}. The values of the stream parameters used are given in the next section.

4. LEARNING TO CONTROL THE TEMPERATURE IN A CSTR WITH COOLING

In this section, we present the results of two experiments (i.e. simulations) that demonstrate the use of reinforcement learning to control the temperature of the simulated CSTR described in the previous section. Only the input and output data from the simulator are involved in training the ANN. None of the information about the equations in Section 3 is used. The CSTR simulator generates as output the CSTR's state, a performance evaluation is made, and a control action is input into the simulator. The goal is for an ANN to learn/develop a strategy (i.e. control law) so that it can apply a sequence of fixed magnitude +/− adjustments to the linear control valve regulating the volumetric flowrate F_3 through the cooling coils such that temperature, T_2 is maintained within a specified tolerance with respect to a set point temperature T_{sp}.

The objective of these experiments was to allow the ANN to develop a strategy to control the reactor temperature T_2 to within a specified tolerance of the set point temperature T_{sp} under the following conditions: (1) the initial state of the process is a start-up state (see Table 1); and (2) the set point temperature is not the steady-state temperature so that without control the reactor temperature tends towards the steady-state temperature thus introducing a deviation from the desired state. This experiment is similar to the cart-pole system explored by Anderson (1986, 1987, 1988) which represents a class of classic inherently unstable multiple-output, dynamic systems. The cart-pole task involves a pole hinged to the top of a wheeled cart that travels along a track constrained to move within the vertical plane.

Table 1. Initial start-up parameter values for the CSTR system in Fig. 4

Stream	Parameter	Description	Value
Reactor 1	F1	Inlet flowrate	0.50 m³ min^{-1}
	T1	Inlet temperature	88°C
	CA1	Inlet concentration	16 kg mol m^{-3}
Reactor 2	T2	Outlet temperature	16°C
	CA2	Outlet concentration	56 kg mol m^{-3}
Coolant 3	F3	Inlet flowrate	0.28 m³ min^{-1}
	T3	Inlet temperature	16°C
Coolant 4	T4	Outlet temperature	16°C

The goal is to apply a sequence of left/right forces of fixed magnitude to the cart such that the pole is balanced and the cart does not hit the end of the track. However, there is an important difference between the cart-pole task and the CSTR temperature control task. In the cart-pole task, the specified tolerance was based on the allowable +/− angle of the pole from center. If the pole went passed this specified angle, there was no chance to recover and the pole would fall to the cart resulting in a failure state. For the CSTR temperature control task, if the reactor temperature falls outside the specified tolerance, the outcome does not necessarily indicate failure. Thus, the reinforcement (i.e. the failure signal) for this experiment cannot be defined as follows (as in the cart-pole example):

$$r = \begin{cases} 0 & \text{if } T_{sp} - \xi \leqslant T \leqslant T_{sp} + \xi, \\ -1 & \text{otherwise,} \end{cases} \tag{17}$$

where ξ is the allowable deviation of the reactor temperature T_2 from the set point temperature T_{sp}. This specification of the reinforcement would be appropriate if the CSTR system could be made to always operate within the specified tolerance under the appropriate control. However, if the CSTR temperature T_2 is outside the specified tolerance, the outcome is not deemed a failure state just an undesirable state. A failure state occurs if the reactor temperature does not move back to within the specified tolerance in a reasonable period of time. Thus, the failure criteria is more complicated for the CSTR system. Subsections 4.1 and 4.2 treat types of reinforcement criteria appropriate for the CSTR system.

The experimental procedure for using each criterion was the same except for the scheme used to calculate the external reinforcement r as is explained in the next two subsections. Each experiment consisted of a number of *runs* that differed only in the seed values for the pseudo-random number generator used to initialize the initial values of the connection weights in the ANN to random values and to provide random values for equation (11) which provides the control action. Each run consisted of a number of *trials*, each starting with the CSTR system set to a specified initial state, and ending with the appearance of the failure signal or by successfully controlling the system for a specified time. The values of the parameters used in all the control simulations are $\rho = 1.0$, $\rho_h = 0.2$, $\rho_m = 0.0$, $\beta = 0.2$, $\beta_n = 0.05$, $\beta_m = 0.0$ and $\gamma = 0.9$.

The steps in the learning procedure were as follows:

1. At the start of every run, all connection weights were initialized to random values between −0.1 and 0.1.
2. At time step t, the state vector was retrieved for the process being controlled.
3. Equations (8–11) were used to generate a control action. For the experiments in the next two subsections, the F in equation (9) corresponded to a turn of the control valve resulting in an increased flow of 0.5 $ft^3 min^{-1}$. Equations (1) and (2) were then calculated.
4. The external reinforcement resulting from the control action was obtained. The predictive reinforcement $\hat{r}$ was then calculated using equation (3).
5. The connection weights were adjusted in the evaluation network using equations (4–7) and the action network using equations (12–16).
6. If $r[t] = 0$ (i.e. the external reinforcement returned was a success), a check was made to see if the process had been controlled for a reasonable amount of time (we used 12,000 time steps which corresponded to 48 min of process simulation time); if so the run was terminated; otherwise t was incremented and the procedure returned to Step 2. If $r[t] = -1$ (i.e. the external reinforcement returned was a failure), t was incremented, and the procedure returned to Step 2.

The average (over 10 runs) number of steps to failure for each trial was calculated and is shown in Figs 5 and 7 in Subsections 4.1 and 4.2.

4.1. Goal-directed learning

The reinforcement learning systems discussed previously in Section 2 are goal directed learning systems. The learning in these systems is directed by

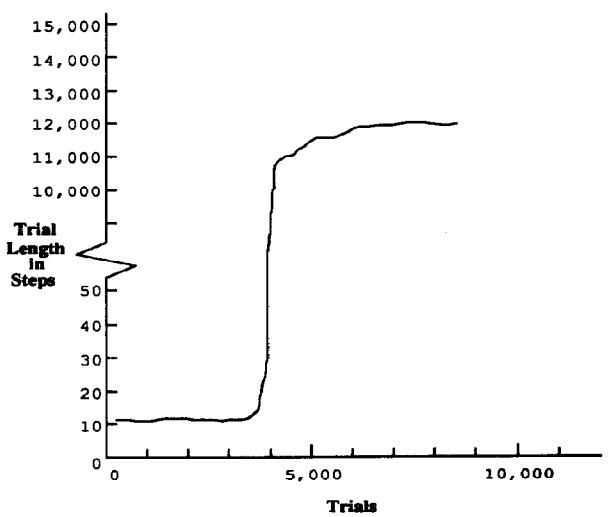


Fig. 5. Learning curve of steps per trial vs number of trials.

a single scalar external reinforcement (i.e. the reinforcement originates from the process to be controlled) that produces a success or failure signal. The goal is for the system to learn to produce control actions such that the process being controlled avoids failure states. An important aspect of reinforcement learning systems that has not received much attention in the literature is the problem associated with choosing a proper criterion on which to base the calculation of the external reinforcement. To provide a basis for discussion, consider how a reinforcement learning system works for control problems. Typically, the reinforcement learning system is initially presented with the system in a nonfailure state. The simple goal is for the learning system to learn appropriate sequences of actions (i.e. develop a strategy) to avoid the failure states inherent in the process being controlled. However, it is not always immediately evident what the criteria should be to decide what is or is not a failure state. Let's consider the task of controlling the temperature of the CSTR described in the previous section. Even though our central goal is to control the temperature to within a specified tolerance, it is not practical for a failure state to be characterized by the temperature being outside the specified tolerance since there are many "valid" states outside this range. For example, there may be a disturbance that pushes the temperature outside this range or the system may be in a start-up condition in which the reactor temperature is outside this range. Although these may not be desirable states, they are not failure states.

A first attempt to extend the external reinforcement parameter value past the cart-pole reinforcement scheme was as follows:

$$r = \begin{cases} 0 & \text{if } T_{sp} - \xi \leqslant T_2 \leqslant T_{sp} + \xi \text{ or } |T(t) - T_{sp}| > |T_2(t+1) - T_{sp}|, \\ -1 & \text{otherwise.} \end{cases} \tag{18}$$

This reinforcement scheme now responds with a success signal ($r = 0$) if the reactor temperature T_2 is within the specified tolerance, or if T_2 is outside the specified tolerance, but T_2 is moving towards the set point T_{sp}. If neither of these two criteria are satisfied, the reinforcement scheme returns a failure signal. This reinforcement scheme requires that historical information be maintained (at least for the previous time step) for T_2.

Figure 5 shows the learning results. The Anderson ANN controller solved the temperature control task achieving 48 min of simulated time in which T_2 remained within its tolerance limits when a value of $\xi = 0.5$ was used. Each run consisted of a number of trials, each starting with the CSTR system in the state determined by the parameters settings listed in Table 1. The run was terminated if no failure occurred within 48 min of simulated time. An average of 4000 failures occurred before a successful strategy or control law was learned.

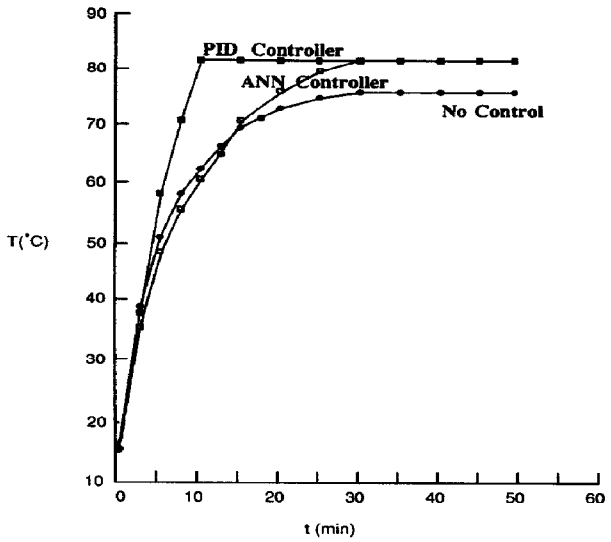


Fig. 6. Temperature vs time during reactor start-up.

The results of several CSTR temperature control experiments are shown in Fig. 6. The curve through the small dark circles is the dynamic response of the system experiencing no control. The curve through the dark squares shows the dynamic response of the CSTR system under PID control. The curve through the open squares illustrates the dynamic response of the CSTR system when controlled by the Anderson ANN controller.

To analyze the control law developed by the Anderson ANN controller due to the goal-directed learning using the criterion of equation (18), we examined the dynamic response of the CSTR system in three phases. The first phase we will discuss is the simulation period from 0 to 15 min. The simulation begins with the CSTR process in an undesirable state (i.e. the start-up state where the reactor temperature is outside the specified tolerance). In this early part of the simulation, the reactor temperature tends towards the steady-state temperature regardless of whether any control

exists or not (see the curve for the system with no control). Hence, it is difficult for the system to experience a failure even when opening the valve to increase cooling. The ANN controller experiences no need to learn better actions since the reinforcement responds with a success even when the actions are counterproductive. Thus, little learning takes place in this phase, and the ANN controller produces actions that are even worse that with no control. The second phase is the simulation period from 15 to 30 min. Now the equilibrium driving force slows down and the ANN must learn actions that will push the reactor temperature towards the set point. In the third phase, the system reaches the desired state with control, and it is easy for the ANN controller to keep the temperature at the set point.

4.2. *Subgoal-directed learning*

In the previous subsection, we saw an example of goal-directed learning seemingly solving the goals of controlling the CSTR process, but, falling short of a standard PID controller in so far as the time taken to reach the set point. Thus, from a practical viewpoint, we not only have the goal of forcing the reactor temperature to reach and stay within the specified tolerance, or at least moving towards that state, but in addition, if the reactor temperature is outside the allowed set point error, we want the reactor temperature to get to the desired state within a reasonable time. These goals lead to the concept of specifying a feasible region for the trajectory of the controlled variable(s), a region formed by transforming any desired objective functions into inequality constraints. Success then involves satisfying the inequality constraints. It would be perfectly possible to add equality constraints that had to be satisfied within some tolerance by formulating them as pairs of inequality constraints, i.e. if $h(\mathbf{x}) = 0$, let $h(\mathbf{x}) > 0$ and $h(\mathbf{x}) < 0$ simultaneously. Also, in general, qualitative constraints might be included in determining success such as having the right color, or adding fuzzy statements such as "not too hot".

In the CSTR example, we added a constraint on the time to reach the set point. If the reactor temperature was outside the desired range for longer than t_{delay}, a failure occurred. We than had a specification for r, the reinforcement learning criteria, as follows:

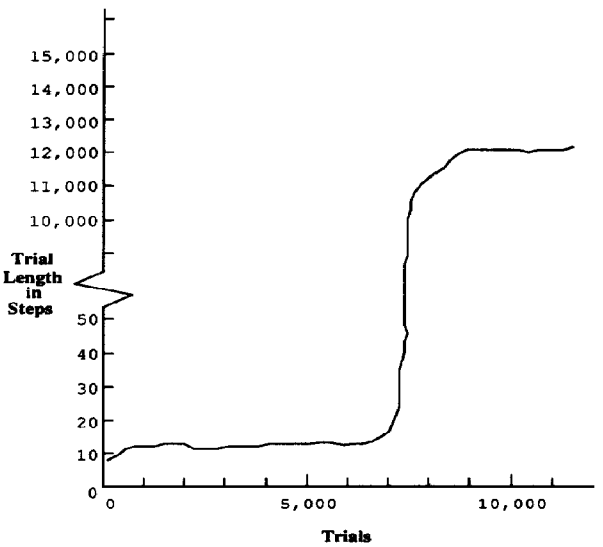


Fig. 7. Learning curve of steps per trial vs number of trials.

where t_{delay} is the maximum time allowed to reach the set point.

Figure 7 shows the learning results. The Anderson ANN controller solved the temperature control task achieving 48 min of simulated time when a value of $\xi = 0.5$ and $t_{\text{delay}} = 10$ min was used. Note, that the run was terminated if no failure occurred within 48 min of simulated time. An average of 7200 failures occurred before a successful strategy was learned. Note that it is more difficult to learn a strategy containing a time component.

The results of several CSTR temperature control experiments are shown in Fig. 8. The curve through the small dark circles is the dynamic response of the system experiencing no control. The curve through the dark squares shows the dynamic response of the CSTR system under tuned PID control. The curve through the open squares illustrates the dynamic response of the CSTR system when controlled by the Anderson ANN controller.

To analyze the control strategy developed by the Anderson ANN controller trained via the goal-directed learning according to equation (19), we again examine the dynamic response of the CSTR system in three phases. The first phase we will discuss is the simulation period from 0 to 5 min. This phase is similar to the first phase in the previous experiment. Little learning took place in this part of the simulation and thus the actions do not help to get to the set

$$r = \begin{cases} 0 & \text{if } T_{\text{sp}} - \xi \leqslant T \leqslant T_{\text{sp}} + \xi \text{ or } |T(k) - T_{\text{sp}}| > |T(k+1) - T_{\text{sp}}|, \\ -1 & \text{if } |T(k) - T_{\text{sp}}| < |T(k+1) - T_{\text{sp}}| \text{ or } T_{\text{sp}} + \xi \leqslant T \leqslant T_{\text{sp}} - \xi \text{ for more than time } t_{\text{delay}}, \end{cases} \tag{19}$$

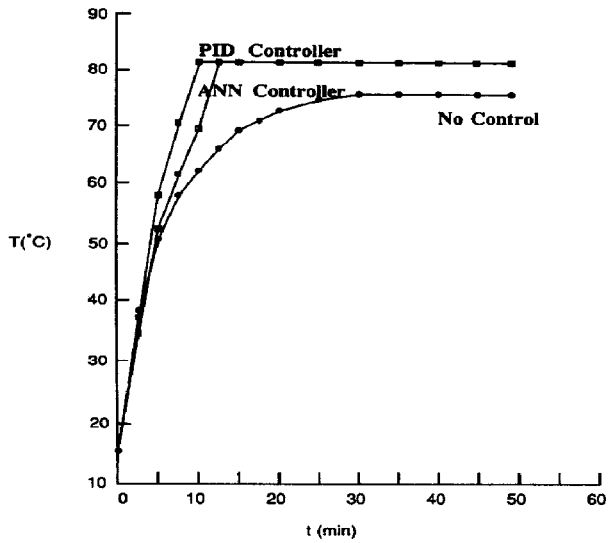


Fig. 8. Temperature vs time during reactor start-up.

point any quicker than the system with no control. The second phase is the simulation period from 5 to 10 min. In this phase, there were two contradictory signals which affected the learning. Before reaching t_{delay} the natural drift towards the set point temperature generates a success response from the external reinforcement regardless of the action. However, upon reaching t_{delay}, if the reactor temperature is not equal to the set point temperature, a failure occurs. This internal conflict between the subgoals extends the learning time, but the overall objective to get to the set point in a faster time than the previous experiment is accomplished as is shown in Fig. 8. In the third phase, the system reaches a new steady-state with control, and it is easy for the ANN controller to keep the temperature at the set point.

A few other starting temperatures, step changes from the steady state, and values of t_{delay} were examined to determine what influence these variations had on the response of a net. It was concluded that small deviations ($\pm 20\%$) in the starting temperature could be accommodated by a net, but large deviations ($\pm 50\%$ or more) required a net trained for the new starting temperature. A reduction in t_{delay} increased the training time; an increase reduced the training time. Small deviations from the steady state seem to be accommodated by any net that was trained for 40 min in the steady state, but we did not explore this situation. We did not try different set point temperatures, but probably the same remark as made about starting temperatures applies. The ANN learns a trajectory shape, and the starting and ending points are reasonably robust to changes.

The knowledge required for reinforcement learning can be seen to be quite minimal. When you develop a control algorithm by existing techniques, you have to decide what an appropriate process model will be (perhaps by experimentation) and estimate the coefficients in the model. You also have to pick the type of controller to mesh with the model, or the process. This phase of development can take considerable effort. When no objective function is available other than the concept of success or failure, the modeling and controller design is even more complex. One can extract knowledge about a process via ordinary modeling or an ANN (which is really an empirical model); the knowledge picked up about the responses of the process will be approximately the same no matter what the modeling technique is if the modeling is carried out to the same degree of accuracy. But the preliminary knowledge that a human has to have about a process for the development of a control strategy is definitely more than a human has to have to implement an ANN for control. We are not suggesting that reinforcement learning should be used in general, or than an ANN should be used in general, for control, but simply that the combination can be one way to treat problems in which a quantitative model and objective function are absent.

Another point to keep in mind is that an ANN "learns" a region for the input–output response of the CSTR. However, "learn" does not mean it is easy to extrapolate, scale up, or transfer the "learning" to other regions of operation as you can with a model composed of differential equations. The control action can only be regarded as an empirical approximation of some complex physical principles embodied in an unusual form, but certainly no more unusual form than a computer code to the nonexpert. Consequently, convergence, stability, etc. have yet to be (may never be) demonstrated theoretically, but currently must be examined by simulation. Generalization (prediction from new starting points) can only take place by having sufficient data sets (in our case trials) to cover the domain of interest in quite the same as required in nonlinear regression for models having error in all of the variables.

5. CONCLUSIONS

This study demonstrates an ANN approach to the control of chemical engineering processes for systems without objective functions. The control system is comprised of an ANN controller that relies on reinforcement learning not only to model the process

but to develop a strategy to control the process. One of the difficult subproblems of using reinforcement rather than supervised learning for training is that the criteria on which to base the reinforcement are qualitative. To apply reinforcement learning for intelligent control an engineer must apply his or her engineering knowledge to specify subgoals or additional appropriate criteria, and quantitative mathematical statements are not essential (although desirable).

The two examples described here demonstrate the effect that the criteria can have on the effectiveness of the reinforcement signal. In the first experiment, control was obtained with fewer trials, but for most practical applications the time to get to the set point is more important. In the second experiment, we added a time constraint which improved the performance (with respect to reaching the control state sooner) at the expense of adding more trials for learning. The delay time could be made shorter such that the trajectory would closely approximate the behavior of the PID controller, albeit again at the expense of increasing the time to learn the strategy to control the process.

REFERENCES

Anderson C. W., Learning and problem solving and multilayered connectionist systems. Doctoral Dissertation, Department of Computer and Information Science, University of Massachusetts, Amherst, MA (1986).

Anderson C. W., Strategy learning with multilayer connectionist representations. *Proc. Fourth Int. Workshop on Machine Learning*, University of California, Irvine (1987).

Anderson C. W., Learning of control—an inverted pendulum with connectionist networks. *Proc. Am. Control Conf.*, Atlanta, GA (1988).

Barto A. G. and P. Anandan, Pattern recognizing stochastic learning automata, *IEEE Trans. Systems, Man. Cybernet.* **15**, 360–374 (1985).

Barto A. G., R. S. Sutton and C. W. Anderson, Neuronlike elements that can solve difficult learning control problems. *IEEE Trans. Systems, Man. Cybernet.* **13**, 835–846 (1983).

Franks R. G. E., *Modeling and Simulation in Chemical Engineering*. Wiley, New York (1972).

Hoskins J. C. and D. M. Himmelblau, Automatic chemical process control using reinforcement learning in artificial neural networks. Presented at *First Annual Meeting of the International Neural Network Society*, Boston, MA (1988).

Miller T. W., R. S. Sutton and P. J. Werbos, *Neural Networks for Control*, MIT Cambridge, MA (1990).

Sutton R. S. (1985).

Williams R., *Reinforcement Learning* (Tutorial No. 14). IEEE, New York (1989).