Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Stochastic Optimal Control with Neural Networks and Application to a Retailer Inventory Problem

Zhongwu Huang, Xiaohua Wang and S. N. Balakrishnan

Abstract—**Overwhelming computational requirements of classical dynamic programming algorithms render them inapplicable to most practical stochastic problems. To overcome this problem a neural network based Dynamic Programming (DP) approach is described in this study. The cost function which is critical in a dynamic programming formulation is approximated by a neural network according to some designed weight-update rule based on Temporal Difference (TD) learning. A Lyapunov based theory is developed to guarantee an upper error bound between the output of the cost neural network and the true cost. We illustrate this approach through a retailer inventory problem.**

I. Introduction

Many important natural and man-made systems are stochastic processes. Take the retailer inventory problem for example. The customers' demand is stochastic in nature and unknown in advance which makes the whole process stochastic. In optimal stochastic process control [1], [2], the objective is to minimize some defined cost function. To be more specific, it is to make a sequence of decisions that make the system perform optimally with respect to some predetermined performance criterion (cost function).

Markov Decision Process (MDP) [3], [4] is a basic modeling framework for stochastic process control. An important property to be used in this paper is that an aperiodic Markov chain can reach to a stationary status at the rate of geometric progression and each state has a steady possibility of occurrence.

Although the concept of Dynamic programming [5], [6] is good for optimization of an MDP, the resulting computational load is sometimes overwhelming. A sensible way of dealing with this difficulty is to generate a compact parametric representation that can approximate the cost function. Bellman and Dregfus [7] used polynomials as compact representations for solving dynamic programming based problems. Similar ideas with using neural networks can be found in Werbos [8] and Barto [9].

Central to the DP algorithms is the idea of how to approximate the cost function. Temporal Difference (TD) learning, originally proposed by Sutton [10], is a method for approximating long-term future cost as a function of current states. Whereas conventional prediction-based learning methods are driven by the error between predicted and actual outcomes, TD methods are driven by the error or difference between temporally successive predictions; with these techniques, learning occurs whenever there is a change in prediction over time. Rigorous analysis of TD methods is however very difficult. Prior work [11], [12] has established convergence of TD learning with a probability of 1 when the cost function is represented as a table where each state has its own entry. Dayn [13] has done some preliminary studies on convergence using linear function approximation. Gordon [14] has proved that TD learning converges for representations called "averagers" on which the TD method is a max-norm contraction mapping. Bertsekas and Tsitsiklis [15], [16] have provided a comprehensive discussion about applying neural networks in dynamic programming based problems with temporal difference learning. They call it Neuro-Dynamic Programming (NDP). Combined with the properties of the Markov chain, they have derived error bounds for the results. Van Roy has continued further with studies on the application of NDP [17], [18]. However their work has been based on a single-layer linear neural network. Carefully chosen basis functions are critical for single-layer neural networks in approximating complicated nonlinear functions. In this paper, we use multilayer neural networks and our approach to stochastic problems originates from our experience with a successful extra control design for robust control in deterministic problems [19]. Inventory and transportation policy determination are common problems in businesses. Some studies have been done in the past [20-22]. We use a retailer inventory problem to illustrate our ideas.

II. Neural Network based dynamic programming

A. Optimal Control of Stochastic Processes

Let us consider a discrete-time system that, at time t_i, takes on a state x_i and evolves according to

$$x_{t+1} = f(x_i, u_i, w_i) \qquad (1)$$

where u_i is a control and w_i is a disturbance. The state, control and disturbance spaces are denoted by X, U and W. For simplicity, we assume these spaces are all finite-dimensional. Each disturbance $w_i \in W$ is independently sampled from some fixed distribution. In the

Zhongwu, Huang, did his PhD from the department of mechanical and aerospace engineering, University of Missouri Rolla, Rolla, Mo, 65401, USA (email: huang@umr.edu)

XiaoHua Wang, is a PhD student with department of mechanical and aerospace engineering, University of Missouri Rolla, Rolla, Mo, 65401, USA (email: wxw98@umr.edu)

S.N. Balakrishnan, Professor, is with department of mechanical and aerospace engineering, University of Missouri Rolla, Mo, 65401, USA (email, bala@umr.edu, Tel: 573-341-4675)

retailer inventory problem that is the focus of this paper, the disturbance is the customer demand. Control u_i depends on the state x_i and the rule by which we select the controls. This rule is called a policy. A stationary policy is a mapping μ : $X \to U$ that generates state-contingent control. In this paper, a policy is stationary if not specified.

Along the transition from x_i to x_j, there is an associated cost noted as $g(x_i,u_i,x_j)$ or $g(x_i,\mu(x_i),x_j)$ with some probability $p_{x_i,x_j}(u_i)$.

So the corresponding Bellman's optimality equation will be:

$$J^*(x_i)=\min_{u_i} \sum_{j=1}^{N} p_{i,j}(u_i)[g(x_i,u_i,x_j)+\alpha J^*(x_j)] \qquad (2)$$

where $\alpha \in [0,1]$ is a discount factor; $p_{t,j}(u_t)$, an easier notation for $p_{x_t,x_j}(u_t)$, is the transition probability from x_t to x_j under control u_t. It could be zero for some x_j s; $x_1,\ldots,x_N$ belong to the set X.

From Bellman's optimality equation, we can get the following two equations on the minimizing path:

$$\frac{\partial J(x_i)}{\partial u}=0 \qquad (3)$$

$$J(x_i)=\sum_{j=1}^{N} p_{i,j}(u_i)[g(x_i,u_i,x_j)+\alpha J(x_j)] \qquad (4)$$

In this study, two neural networks will be used to iteratively approximate these two conditions and get the optimal policy and optimal cost function. One neural network is called the cost neural network with the system states as inputs and cost J as the output. The other is the control neural network with states as inputs with control u being the network outputs. When the outputs of these two neural networks are mutually consistent in their convergence, the optimality condition is satisfied and outputs of these two neural networks are optimal. This is what we call neural network based approximate dynamic programming method. Here we use a policy iteration process as the approximation technique.

B. *Policy Iteration with Neural Networks*

Policy iteration algorithm starts with a proper policy μ_0. Then we perform a policy evaluation step, computing $J^{\mu_0}(x_i)$ as the solution to the equations

$$J(x_i)=\sum_{j=1}^{N} p_{i,j}(\mu_0(x_i))[g(x_i,\mu_0(x_i),x_j)+\alpha J(x_j)], i=1\cdots N \qquad (5)$$

This step is to generate the cost function from the policy μ_0. Then we perform a policy improvement step and compute an improved policy μ_1 based on the cost function just obtained as

$$\mu_1(x_i)=\arg\min_{u} \sum_{j=1}^{N} p_{i,j}(u)[g(x_i,u,x_j)+\alpha J^{\mu_0}(x_j)] \qquad (6)$$

This process is repeated until $J^{\mu_{k+1}}(x_i)=J^{\mu_k}(x_i)$ for all i. What we did here is to use the control and the cost neural network outputs to provide the policy μ and cost function J in the above process.

C. *Cost Function Approximation*

Normally the state space of a large dynamic system is very large and to implement (5) in the above policy iteration will be very difficult. We need another way to synthesize the cost neural network. To help achieve this, we generate some simulations of the underlying stochastic process and use that information to tune the weights of cost neural network directly such that the output of cost neural network is close to the true cost. The idea here is similar to our earlier work in the extra control design for the deterministic problems with uncertainties [19]. In other words, the weight-update rule of cost neural network is designed directly. The weight-update rule for a two layer cost neural network is chosen as

$$W_1(t+1)=W_1(t)+\gamma_1\varphi_1(x_t)[-W_1^T(t)\varphi_1(x_t)+B_1 d_t]^T$$

$$W_2(t+1)=W_2(t)+\gamma_2\varphi_2(W_1^T(t)\varphi_1(x_t))d_t \qquad (7)$$

where B1 is a coefficient matrix, γ_1 and γ_2 are the learning rates, and d_t is called the temporal difference and $d_t=g(x_t,x_{t+1})+\alpha J_t(x_{t+1})-J_t(x_t)$. When the Bellman equation is satisfied, $E[d_t]=0$. W_1 and W_2 are neural network weight matrices for the first layer and second layer respectively. And φ_1, φ_2 are respectively neural network activation function for first layer and second layer.

When the iteration is executed in a simulation, normally the crux of the problem is that the cost NN can't be well represented over a large state space. As a result, there can be big errors for some states when we use the cost NN to calculate the improved controls later. However, if the policy is proper, the state of systems will fall into a small set of frequently visited states in a few steps although with a small probability it may go outside again; a properly designed policy will bring it back quickly. The size of this set is usually manageably small and the objective should be to make the cost NN more accurate in that region. Also, those states are the ones will appear most time in the simulation or in a real life application. Hence, the costs and controls with respect to those states are important and should be carefully treated. From this point of view, we need to use an on-line sampling scheme to acquire these frequently visited states.

Next, we will show that with the weight-update rule in (7), the difference between the output of cost neural network and the true cost is bounded. In this case, it will be more reasonable if the expectation of weight-update rule is considered since the underlying process is stochastic. As mentioned in the introduction, an aperiodic Markov chain can reach a stationary status very fast. Expectation of the weight-update rule is presented in Eq (8) with respect to the steady-state distribution π . The detailed derivation is omitted for brevity.

$$\begin{aligned} W_1(t+1) &= W_1(t) - \gamma_1 \Phi_1^T D \Phi_1 W_1(t) + \gamma_1 \Phi_1^T D \bar{g} B_1^T \\ &\quad + \gamma_1 \Phi_1^T D(\alpha P - I)\Phi_2(0) W_2(t) B_1^T \\ W_2(t+1) &= \mathrm{W}_2(t) + \gamma_2 {\Phi_2}^T(0) D \bar{g} \\ &\quad + \gamma_2 {\Phi_2}^T(0) D(\alpha P - I)\Phi_2(0) W_2(t) \end{aligned} \tag{8}$$

where

$D = diag(\pi(1), \pi(2), ..., \pi(N))$,

$\Phi_2(0) = \begin{bmatrix}\varphi_2(1) & \dots & \varphi_2(N)\end{bmatrix}^T$ with $\varphi_2(i) = \varphi_2(W_1^T(t)\varphi_1(x_i))$

$$\Phi_1 = \begin{bmatrix} \varphi_1^T(x_1) \\ \vdots \\ \varphi_1^T(x_N) \end{bmatrix} \qquad \bar{\mathrm{g}} = \begin{bmatrix} \bar{\mathrm{g}}(1) \\ \vdots \\ \bar{\mathrm{g}}(\mathrm{N}) \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{N} P_{1j}\ \mathrm{g(1,j)} \\ \vdots \\ \sum_{j=1}^{N} P_{Nj}\ \mathrm{g(N,j)} \end{bmatrix}$$

The proof is Lyapunov function based. Let the ideal weights for the true cost J^{μ} corresponding to the policy μ be W_1^* and W_2^* , that is

$$J^{\mu}(x_i) = W_2^{*T} \varphi_2(W_1^{*T} \varphi_1(x_i)) + \varepsilon(x_i) \tag{9}$$

where $\varepsilon(x_i)$ is the neural network functional approximation error. By choosing proper number of neurons for each layer, this approximation error can be very small and bounded. Here we assume $\left\| D^{1/2}\varepsilon \right\| \le \varepsilon_N$.

A Lyapunov function candidate is chosen as the following

$$L_i = \frac{1}{\gamma_1} tr(\tilde{W}_1^T(i)\tilde{W}_1(i)) + \frac{1}{\gamma_2} tr(\tilde{W}_2^T(i)\tilde{W}_2(i)) \tag{10}$$

where $\tilde{W}_1(i) = W_1^* - W_1(i), \tilde{W}_2(i) = W_2^* - W_2(i)$ and γ_1, γ_2 are constants. We can show that

$$\begin{aligned} & L_{i+1} - L_i \\ & = \frac{1}{\gamma_2}[tr(\tilde{W}_2^T(i+1)\tilde{W}_2(i+1) - tr(\tilde{W}_2^T(i)\tilde{W}_2(i))] + \frac{1}{\gamma_1}[tr(\tilde{W}_1^T(i+1)\tilde{W}_1(i+1) - tr(\tilde{W}_1^T(i)\tilde{W}_1(i))] \end{aligned} \tag{11}$$

$$\begin{aligned} & L_{i+1} - L_i \le \\ & -\eta \left\| D^{1/2}\Phi_2(0)\tilde{W}_2(t) \right\|^2)^2 \left\| D^{1/2}\Phi_2(0) \right\|^2 - \rho \left\| D^{1/2}\Phi_1 \tilde{W}_1(t) \right\|_F^{\ 2} + \zeta \\ & -(1-\alpha)[\left\| D^{1/2}\Phi_2(0)\tilde{W}_2(t) \right\| - \frac{1}{1-\alpha} \left\| D^{1/2}(\bar{g} + (\alpha P - I)\Phi_2(0) W_2^*) \right\|]^2 \\ & -\gamma_2[(1+\alpha) \left\| D^{1/2}\Phi_2(0) \right\| \left\| D^{1/2}\Phi_2(0)\tilde{W}_2(t) \right\| - \left\| D^{1/2}\Phi_2(0) \right\| \left\| D^{1/2}(\bar{g} + (\alpha P - I)\Phi_2(0) W_2^*) \right\|]^2 \\ & -\gamma_1[\left\| D^{1/2}\Phi_1 \right\| \left\| D^{1/2}\Phi_1 \tilde{W}_1(t) \right\|_F - \left\| D^{1/2}\Phi_1 \right\| \left\| D^{1/2}\Phi_1 W_1^* - dB_1^T \right\|_F]^2 \\ & -(1+2\gamma_1 \left\| D^{1/2}\Phi_1 \right\|^2)[(1+\alpha)\left\| B_1 \right\| \left\| D^{1/2}\Phi_2(0)\tilde{W}_2(t) \right\| - \left\| D^{1/2}\Phi_1 W_1^* \right\|_F]^2 \\ & -(1+2\gamma_1 \left\| D^{1/2}\Phi_1 \right\|^2)[(1+\alpha)\left\| B_1 \right\| \left\| D^{1/2}\Phi_2(0)\tilde{W}_2(t) \right\| - \left\| D^{1/2}(\bar{g} + (\alpha P - I)\Phi_2(0) W_2^*) \right\| \left\| B_1 \right\|]^2 \end{aligned} \tag{12}$$

where:

$$\eta = 1 - \alpha - 2\gamma_2(1+\alpha)^2 \left\| D^{1/2}\Phi_2(0) \right\|^2 - 3(1+2\gamma_1 \left\| D^{1/2}\Phi_1 \right\|^2)(1+\alpha)^2 \left\| B_1 \right\|^2$$

$$\rho = 1 - 2\gamma_1 \left\| D^{1/2}\Phi_1 \right\|^2$$

$$\begin{aligned} \zeta = & [\frac{1}{1-\alpha} + 2\gamma_2 \left\| D^{1/2}\Phi_2(0) \right\|^2 + 2(1+2\gamma_1 \left\| D^{1/2}\Phi_1 \right\|^2)\left\| B_1 \right\|^2] \left\| D^{1/2}(\bar{g} + (\alpha P - I)\Phi_2(0) W_2^*) \right\|^2 \\ & + 2[1+2\gamma_1 \left\| D^{1/2}\Phi_1 \right\|^2] \left\| B_1 \right\| \left\| D^{1/2}\Phi_1 W_1^* \right\|_F \left\| D^{1/2}(\bar{g} + (\alpha P - I)\Phi_2(0) W_2^*) \right\| \\ & + 2[1+2\gamma_1 \left\| D^{1/2}\Phi_1 \right\|^2] \left\| D^{1/2}\Phi_1 W_1^* \right\|_F^{\ 2} \end{aligned} \tag{13}$$

If $\eta > 0$, $\rho > 0$, then $L_{t+1} < L_t$, when $\left\| D^{1/2}\Phi_2(0)\tilde{W}_2(t) \right\| > \sqrt{\zeta/\eta}$ or $\left\| D^{1/2}\Phi_1 \tilde{W}_1(t) \right\|_F > \sqrt{\zeta/\rho}$.

Whereas $\left\| D^{1/2}\left(J_i - J^{\mu}\right) \right\| \le \left\| D^{1/2}\Phi_2(0)\tilde{W}_2(i) \right\| + \left\| D^{1/2}\Phi_1 \tilde{W}_1(t) \right\|_F \left\| W_2^* \right\| + \varepsilon_N$,

So $\left\| D^{1/2}\Phi_1 \tilde{W}_1(t) \right\|_F$ and $\left\| D^{1/2}\Phi_2(0)\tilde{W}_2(t) \right\|$ are both bounded. And we know that if $\left\| D^{1/2}\left(J_i - J^{\mu}\right) \right\|$ is bounded, the state distribution weighted difference between the output of the cost neural network and the true cost is therefore bounded.

After we obtain the cost neural network, the improved control will be calculated as

$$\mu_1(x_i) = \arg\min_{u} \underset{w}{E}[g(x_i, u, w) + \alpha J^{\mu_0}(f(x_i, u, w))] \tag{14}$$

and used to train the control neural network.

III. Retailer Inventory Applications

A. Retailer Inventory System Model

We use the retailer inventory system form Nahmias and Smith [20]. This problem deals with ordering and positioning retailer inventories in warehouses and stores in order to meet customer demands while minimizing a specified cost.

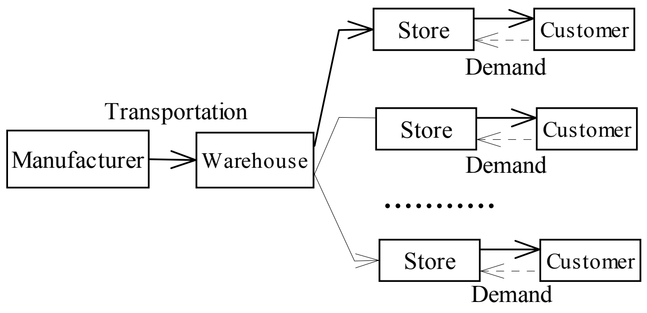

Fig. 1. Model of the retailer inventory system

The working process can be described as follows: First, demands occur at each store from customer requests. If there is enough goods available at that store, these demands can be satisfied. In case of shortages, if the customers are willing to wait, special deliveries are sent directly from a warehouse. If the inventory at the warehouse is still not enough, a shortage cost arises. At the end of the day, stores will place orders to the warehouse when low on inventory. Since transportation is involved, there is usually a delivery delay from the warehouse to the stores. Coupled with the uncertainty of future demands, it creates a need for inventory at the store level. When the warehouse receives orders from the stores, it will fulfill them as much as possible given the current levels of inventory. At the same time, a warehouses needs to place orders to the manufacturers if its own inventory is low. Due to reasons similar to those for stores, this shows the need for an inventory of goods at the warehouse. This retailer inventory management is a stochastic process due to varying customer demands and uncertainty of future demands.

Associated with this model, there are three costs: storage costs at stores and the warehouse, transportation cost, and shortage cost. Cost function is the sum of these costs. Objective of the problem is to minimize the total cost by finding the optimal orders for stores and warehouse. For simplicity, transportation cost won't be considered here.

The parameters of the model are list in Table 1.

Table I

Parameters of Retailer Inventory Model

Number of warehouse	1
Number of store	10
Delay to stores and warehouse	2
Production capacity	100
Warehouse capacity	1000
Store capacity	100
Warehouse and store storage cost	3
Shortage cost	60
Probability of customer waiting	0.8
Cost of special delivery	0
Mean demand	5
Demand STDEV	14

B. System Model

In this study, control (order) will come in first followed by the customers' demands. Hence, we need to define two sets of states. First is the pre-order state denoted by x. The second is the post-order state denoted by y. Each post-order state is given by $y_t=f_2(x_t,u_t)$ for some function f_2. Each pre-order state is given by $x_{t+1}=f_1(y_t,w_t)$ for some function f_1.

The elements of state space x are defined as follows:

$$x=[x_{w,0},x_{w,1},x_{w,2},x_{s,1,0},x_{s,1,1},x_{s,1,2},\ldots,x_{s,10,2}]_{33\times 1} \quad (15)$$

where :

$x_{w,0}$: current inventory at the warehouse

$x_{w,1}$: goods currently being transported that will arrive at the warehouse in 1 day.

$x_{w,2}$: goods currently being transported that will arrive at the warehouse in 2 days.

$x_{s,i,0}$: current inventory at the ith store.

$x_{s,i,1}$: goods currently being transported that will arrive at the ith store in 1 day.

$x_{s,i,2}$: goods currently being transported that will arrive at the ith store in 2 days.

Components of y are defined similarly except the values are those after orders are placed.

Control u is defined as

$$u=[u_0,u_1,\ldots,u_{10}]_{11\times 1} \quad (16)$$

where u_0 denotes the order from warehouse and $u_i, i\in 1 \text{ to } 10$ denotes the order from ith store.

Given the current pre-order state as defined in (15), the control u must obey the following constraints:

1. Each element of the control should be non-negative.

$$u_i\geq 0, \forall i=0,1,\cdots,10 \quad (17)$$

2. The order from warehouse should be less the manufacturer production capacity.

$$u_0\leq C_p \quad (18)$$

3. The total orders from stores should be less than the current inventory at the warehouse.

$$x_{w,0}\leq \sum_{i=1}^{10} u_i \quad (19)$$

4. The total quantity of goods at and on-route to any particular store should be less than the store capacity.

$$u_i \leq C_s - \sum_{k=0}^{2} x_{s,i,k} \qquad \forall i=1,2,\cdots 10 \quad (20)$$

5.The total quantity of goods at and on-route to the warehouse minus the orders from the stores should be larger than the warehouse capacity.

$$u_0\leq C_w+\sum_{l=1}^{10} u_l-\sum_{k=0}^{2} x_{w,k} \quad (21)$$

where C_p=100 is the manufacturer production capacity, C_s=100 is the store capacity, and C_w=1000 is the warehouse capacity.

The demand w is a 10×1 vector defined as

$$w=[w_1,w_2,\ldots,w_{10}]_{10\times 1} \quad (22)$$

where each element is independently sampled from a normal distribution $N(\mu,\sigma)$ with $\mu=5$ and $\sigma=14$. Of course, it will be rounded and set to the nearest non-negative integer. Also we assume possibility of the willingness of customers to wait for special delivery is 0.8.

Now let us begin with a pre-order state $x_t=[x_{w,0},x_{w,1},x_{w,2},x_{s,1,0},x_{s,1,1},x_{s,1,2},\ldots,x_{s,10,2}]$. We choose a designed control $u_t=[u_0,u_1,\ldots,u_{10}]$. The elements of post-order y_t is calculated as

$$\begin{aligned}&y_{w,0}=x_{w,0}-\sum_{l=1}^{10} u_i \;;\; y_{w,2}=x_{w,2}+u_0\\&y_{w,1}=x_{w,1} \;;\; y_{s,i,2}=x_{s,i,2}+u_i\\&y_{s,i,0}=x_{s,i,0} \;;\; y_{s,i,1}=x_{s,i,1}\end{aligned} \quad (23)$$

Now let us assume that the demand is $w=[w_1,w_2,\ldots,w_{10}]$.

First, customer demands are fulfilled by inventories in the stores according to

$$\begin{aligned}&\bar{y}_{s,i,0}=y_{s,i,0}-w_i\\&\bar{\bar{y}}_{s,i,0}=\max(0,\bar{y}_{s,i,0}), \quad \forall i=1,2,\cdots 10\end{aligned} \quad (24)$$

If some $\bar{y}_{s,i,0}$ s are less that zero, a special delivery (SD) will be needed for the corresponding stores. Special deliveries are filled by the warehouse according to

$$SD=\sum_{i=1}^{10} 0.8(\max(0,-\bar{y}_{s,i,0}))\;;\; \bar{y}_{w,0}=\max(0,y_{w,0}-SD) \quad (25)$$

If $y_{w,0}-SD<0$, a shortage cost will incur.

Next, goods progresses to x_{t+1} according to

$$\begin{aligned}&x_{w,0}=\bar{y}_{w,0}+y_{w,1}\,;\quad x_{w,1}=y_{w,2}\\&x_{w,2}=0\,;\\&x_{s,i,0}=\bar{\bar{y}}_{s,i,0}+y_{s,i,1}\,;\\&x_{s,i,1}=y_{s,i,2}\,;\qquad x_{s,i,2}=0\end{aligned}\tag{26}$$

So the cost from i to (i+1) with a demand w_i is

$$g(y_t,w_t)=3y_{w,0}+3\sum_{i=1}^{10}y_{s,i,0}+60\max(0,SD-y_{w,0})\tag{27}$$

where SD is defined in (25).

C. Discussion and Approximation of Cost Function

Initially, we may not have any knowledge about the cost J. But in most cases, we may know something about the policy by experience or common sense, that is, what kind of control or decision to make in some situation. So, an initial policy can be defined through heuristics or other considerations. This policy will be used to train the control neural network first. Then, this trained control neural network will provide controls for the approximation of the cost function J. We use an s-type heuristic policy here [21], that is, at each time step the inventory manager tries to place the order such that current inventory and the goods expected to arrive at the warehouse is equal to the warehouse order-up-to level. Similarly the current inventory and the goods expected to arrive at any store is equal to the store order-up-to level. For this problem, the warehouse and store order-up-to levels are 330 and 23. We choose some initial states and generate the corresponding controls using this heuristic policy. Next, we train the control neural network with these data. The structure of the control neural network is chosen as $N_{22\text{-}8\text{-}8\text{-}11}$. Figure 2 shows the cost per-stage corresponding to this control neural network. The average value is 1176.

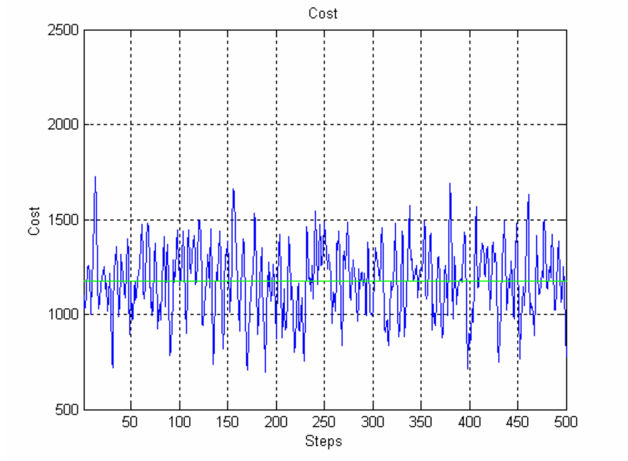

Fig. 2. Cost Per-stage with average 1176

A two-layer neural network is used to approximate the cost function for this retailer inventory problem. The structure of the cost neural network is $N_{22\text{-}20\text{-}1}$. Parameters in the weight update rule are chosen as: $\gamma_1=10^{-4}$, $\gamma_2=10^{-4}$ for the first 3×10^5 steps, $\gamma_1=10^{-5}$, $\gamma_2=10^{-5}$ for the rest steps, $\alpha=0.99$ and $B_1=10^{-2}\times[0.05\ 0.1\ \ldots\ 0.95\ \ 1]_{20\times1}$. We run a single long simulation path of the system (5×10^5 steps) with some random initial states. Fig. 3 shows the output of the cost neural network during the learning process. Note that it converges. This cost neural network can be used anew for the next series of simulations. This sequence was carried out several times to make the cost neural network learn as much as possible.

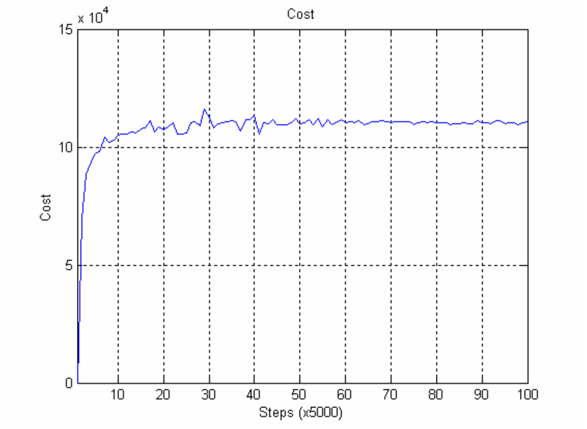

Fig. 3. Output of Cost Neural Network during Learning Process

After the cost neural network stabilized in its outputs, the improved control was calculated according to Eq(14). Figure 4 shows the cost per-stage with the new control neural network. The average value is only 860 which is much lower than 1176 with the initial control neural network as showing in Fig. 5

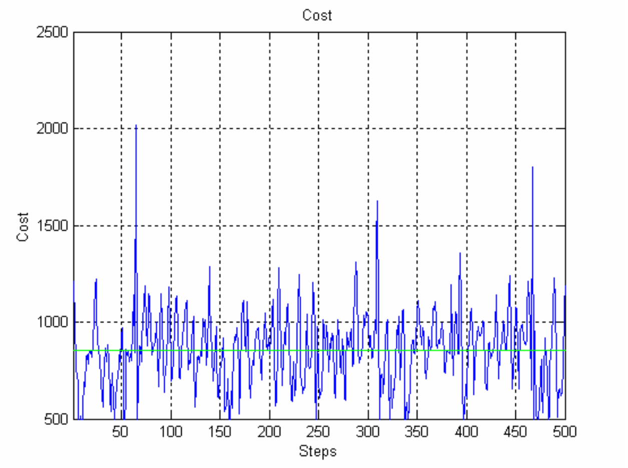

Fig. 4. Cost Per-stage with average 860

The state of the warehouse and one-delay state to the warehouse are shown in Figure 5. Note that any warehouse related-data can be retrieved from these data. Figure 6 shows the states related to store 1. Controls (orders) for warehouse and store 1 are presented in Figure 7. Note that, in addition to obtaining an average order-up to level, these data can be used to help with online decision too if needed.

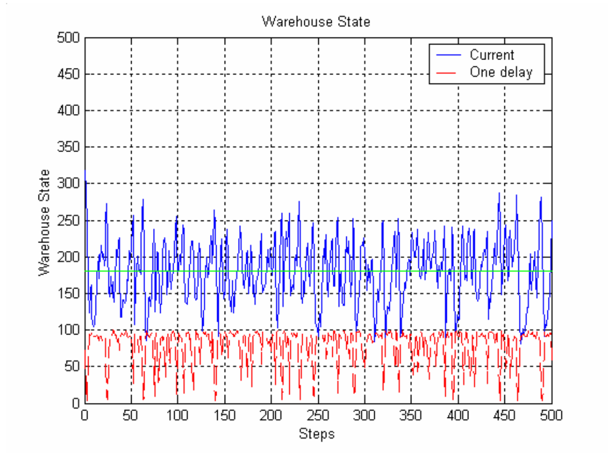

Fig. 5. Warehouse State

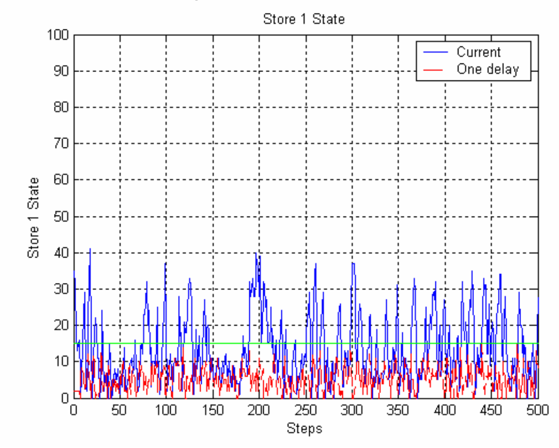

Fig. 6. Store 1 State

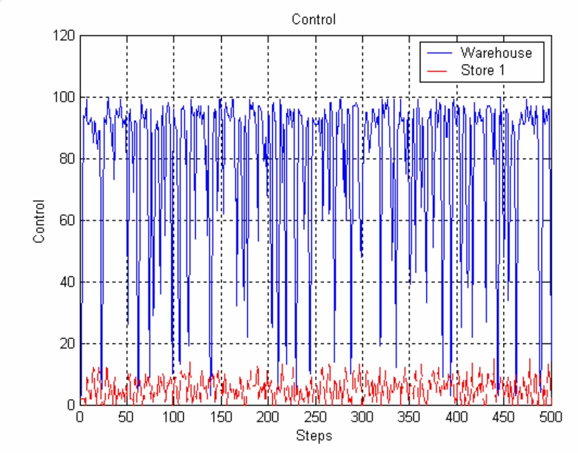

Fig.7. Control History

IV. Conclusions

A neural network based dynamic programming method has been developed for optimal stochastic process control problems. Simulation results from a retailer inventory problem show the effectiveness of this approach. Error bound on the cost accuracy has also been derived in this paper.

V. ACKNOWLEDGEMENT

Grants from the National Science Foundation, ECS 0201076 and 0324428 in support of this study are gratefully acknowledged.

References

[1] A.E. Bryson and Y.Ho, Applied Optimal Control. Hemisphere Publishing Co., 1975, pp. 128-211.
[2] Puterman, Martin L., (1994), Markov Decision Process: Discrete Stochastic Dynamic Programming, John Wiley & Sons Inc., Canada.
[3] R.A. Howard, Dynamic Programming and Markov Process, MIT Press, Cambridge, 1960.
[4] E.A. Feinberg and Adam Shwartz, Handbook of Markov Decision Processes, Kluwer, 2002.
[5] R.E. Bellman, Dynamic Programming, Princeton, NJ: Princeton University Press, 1957.
[6] D. White, Dynamic Programming, San Francisco, CA: Holden-Day, 1969.
[7] Bellman, R.E. and Dreyfus, S.E. (1959), “Function Approximation and Dynamic Programming,” Math., Tables and Other Aids Comp., Vol. 13, pp.247-251.
[8] Werbos, P., “Building And Understanding Adaptive Systems: A Statistical/ Numerical Approach to Factory Automation And Brain Research,” IEEE Transactions on Systems, Man And Cybernetics, Vol. SMC-17, No.1, pp.7-20., 1987.
[9] Barto, A., Sutton, R., and Anderson, C., “Neuron-like Adaptive Elements That Can Solve Difficult Learning Control Problems,” IEEE Transactions On Systems, Man And Cybernetics, Vol. SMC-13, No. 5, pp.834-846, 1983.
[10] R.S. Sutton, “Learning to Predict by the methods of Temporal Difference,” Machine Learning, Vol.3, 1988, pp. 835-846.
[11] T. Jaakkola, S.P. Singh, and M.I. Jordan, “Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems,” Advances in Neural Information Processing Systems 7, Cambridge, Massachusetts, MIT Press, 1995, pp. 345-352..
[12] S.P. Singh & R.S. Sutton, “Reinforcement Learning with Replacing Eligibility Traces,” Machine Learning, Vol.22, 1994.
[13] P.D. Dayan, “The Convergence of TD(λ) for General λ,” Machine Learning, Vol. 8, 1992, pp. 341-362.
[14] G.J. Gordon, “Stable Fuction Approximation in Dynamic Programming,” Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, CA, July 1995.
[15] D.P. Bertsekas and J.N. Tsitsiklis, Neuro-dynamic Programming, Athena Scientific, Belmont, MA,1996.
[16] D.P. Bertsekas, Dynamic Programming and Optimal Control, Vols. I,II, Athena Scientific, Belmont, MA,1995.
[17] D. P. de Farias and B. Van Roy, “On the Existence of Fixed Points for Approximate Value Iteration and Temporal-Difference Learning,’ Journal of Optimization Theory and Applications, Vol. 105, No. 3, June, 2000.
[18] P. Rusmevichientong and B. Van Roy, “A Tractable POMDP for a Class of Sequencing Problems,” Proceedings of the Conference on Uncertainty in Artificial Intelligence, 2001.
[19] Balakrishnan, S.N. and Huang, Zhongwu, "Robust Adaptive Critic Based Neurocontrollers for Helicopters with Unmodeled Uncertainties," AIAA Atmospheric Flight Mechanics Conference, Montreal, Aug. 2001.
[20] Nahmias, S. and Smith, S.A., “Mathematical Models of Inventory Retailer Systems: A review,” Perspectives on Operations Management, Essays in Honor of Elwood S. Buffa, Sarin, R., editor, Kluwer Academic Publishers, Boston, MA, pp.249-278, 1993.
[21] B. Van Roy, D.P. Bertsekas, Y. Lee, J.N. Tsitsiklis, “A Neuro-dynamic Programming Approach To Retailer Inventory Management,” Proceedings of the 36th IEEE Conference on Decision and Control, Vol. 4, 1997, pp.4052-4057.
[22] Shervais, Stephen, Adaptive Critic Design of Control Polices for Multi-echelon Inventory System, Ph.D thesis, Portland State University, 2000.