On-line EM reinforcement learning

Junichiro Yoshimoto[†1], Shin Ishii [†1†3], Masa-aki Sato [†2†3]

†1 Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma, Nara 630-0101, Japan
{juniti-y, ishii}@is.aist-nara.ac.jp

†2 ATR Human Information Processing Research Laboratories
masaaki@hip.atr.co.jp

†3 CREST, Japan Science and Technology Corporation
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan

Abstract

In this article, we propose a new reinforcement learning (RL) method for a system having continuous state and action spaces. Our RL method has an architecture like the actor-critic model. The critic tries to approximate the Q-function, which is the expected future return for the current state-action pair. The actor tries to approximate a stochastic soft-max policy defined by the Q-function. The soft-max policy is more likely to select an action that has a higher Q-function value. The on-line EM algorithm is used to train the critic and the actor. We apply this method to two control problems. Computer simulations show that our method is able to acquire fairly good control in the two tasks after a few learning trials.

1. Introduction

Reinforcement learning (RL) is a kind of machine learning framework, which automatically acquires an optimal control based on actual experiences and rewards. RL methods have been successfully applied to various Markov decision problems that have finite state/action spaces. On the other hand, many tasks in the real world have continuous state and action spaces such as human or robot motion control problems. These tasks are much more difficult than the former for the following reasons. A table lookup representation of the utility function can be used for finite state/action problems but this is not possible for continuous problems. Therefore, a good function approximator and a fast learning algorithm are crucial in learning the utility function. In addition, it is difficult to calculate the optimal action that maximizes the utility function even if the function approximation is precise.

In the previous article [1], we proposed an RL method based on the on-line EM algorithm [2], which can be applied to continuous state/action problems. The method employed an actor-critic-like architecture, and the critic approximated the Q-function. The actor's training was based on the gradient of the Q-function. In our new learning scheme, on the other hand, the actor is trained to approximate a soft-max policy that is dependent on the critic's Q-function. The soft-max policy is more likely to select an action that has a higher Q-function value. The actor's learning uses the modified on-line EM algorithm.

To test the method's performance, we apply it to automatic control problems of an inverted pendulum [1, 5] and an acrobot [6]. Our computer simulations show that the method is able to acquire fairly good control in these tasks after a few learning trials.

2. NGnet & On-line EM algorithm

The NGnet, which transforms an N-dimensional input vector x into a D-dimensional output vector y, is defined by

$$y = \sum_{i=1}^{M} \left(\frac{G_i(x)}{\sum_{j=1}^{M} G_j(x)} \right) \tilde{W}_i \tilde{x} \tag{1a}$$

$$G_i(x) \equiv (2\pi)^{-N/2} |\Sigma_i|^{-1/2} \exp\left[-\frac{1}{2}(x-\mu_i)' \Sigma_i^{-1} (x-\mu_i) \right], \tag{1b}$$

where $\tilde{W}_i \equiv (W_i, b_i)$ and $\tilde{x}' \equiv (x', 1)$. M denotes the number of units. The prime ($'$) denotes a vector transpose and $|\cdot|$ denotes a matrix determinant. $G_i(x)$ is an N-dimensional full-covariance Gaussian function. $\tilde{W}_i$ is a linear regression matrix.

The NGnet can be interpreted as a stochastic model in which a pair of an input and an output, (x, y), is a stochastic event. For each event, a unit index i is assumed to be selected and is regarded as a hidden variable. A triblet (x, y, i) is called a complete event. The stochastic model is defined by the probability distribution for a complete event:

$$P(x,y,i|\theta) = M^{-1} G_i(x)(2\pi)^{-D/2}\sigma_i^{-D} \exp\left[-\frac{1}{2\sigma_i^2}(y-\tilde{W}_i\tilde{x})^2\right], \tag{2}$$

where $\theta \equiv \{\mu_i, \Sigma_i, \sigma_i, \tilde{W}_i \mid i = 1, ..., M\}$ is a set of model parameters. From this distribution, we can easily prove that the expected value of the output y for a given input x, $E[y|x] \equiv \int yP(y|x)dy$, is identical to the output of the NGnet (1a). Namely, the probability distribution (2) provides a stochastic model for the NGnet. Then, the stochastic model (2) is called the stochastic NGnet.

From a training data set $\{(x(t), y(t)) \mid t = 1, ...t_{max}\}$, the model parameter of the stochastic NGnet (2) can be determined by the EM algorithm [7], which repeats the following E- and M-steps.

In an E(Expectation)-step, by using the present parameter $\bar{\theta}$, the posterior probability that the i-th unit is selected for the t-th datum, $P(i|x(t), y(t), \bar{\theta})$, is calculated by

$$P(i|x(t), y(t), \bar{\theta}) = P(x(t), y(t), i|\bar{\theta}) \Big/ \sum_{j=1}^{M} P(x(t), y(t), j|\bar{\theta}) \,. \tag{3}$$

By using the posterior probability (3), the expected log-likelihood for the set of complete events, $L(\theta|\bar{\theta})$, is defined by

$$L(\theta|\bar{\theta}) = \sum_{t=1}^{t_{max}} \sum_{i=1}^{M} P(i|x(t), y(t), \bar{\theta}) \log P(x(t), y(t), i|\theta). \tag{4}$$

In an M(Maximization) step, the expected log-likelihood is maximized with respect to θ. According to the stationary condition $\partial L/\partial\theta = 0$, the new model parameter can be determined by using the weighted means: $\langle 1\rangle_i(t_{max})$, $\langle x\rangle_i(t_{max})$, $\langle xx'\rangle_i(t_{max})$, $\langle |y|^2\rangle_i(t_{max})$ and $\langle y\tilde{x}'\rangle_i(t_{max})$, where

$$\langle f(x,y)\rangle_i(t_{max}) \equiv \frac{1}{t_{max}} \sum_{t=1}^{t_{max}} f(x(t), y(t)) P(i|x(t), y(t), \bar{\theta}). \tag{5}$$

The EM algorithm introduced above is based on batch learning [8], namely, the parameters are updated after reviewing all of the observed data. Here, we explain the on-line EM algorithm [2]. Hereafter, let $\theta(t)$ be the estimated model parameter after the t-th datum $(x(t), y(t))$.

In an E-step, the posterior probability for the t-th datum $P_i(t) \equiv P(i|x(t), y(t), \theta(t-1))$ is calculated by using the current parameter $\theta(t-1)$ according to equation (3). In an M-step, the weighted mean (5) is replaced by the discounted mean:

$$\langle\langle f(x,y)\rangle\rangle_i(t) \equiv \eta(t) \sum_{\tau=1}^{t} \left(\prod_{s=\tau+1}^{t} \lambda(s)\right) f(x(t), y(t) P_i(t), \tag{6}$$

where $\lambda(s)$ $(0 \le \lambda(s) \le 1)$ is a time-dependent discount factor. $\eta(t) \equiv \left[\sum_{\tau=1}^{t}\left(\prod_{s=\tau+1}^{t}\lambda(s)\right)\right]^{-1}$ is a normalization coefficient that plays a role like a learning rate. The discounted means: $\langle\langle 1\rangle\rangle_i(t)$, $\langle\langle x\rangle\rangle_i(t)$, $\langle\langle |y|^2\rangle\rangle_i(t)$ and $\langle\langle y\tilde{x}'\rangle\rangle_i(t)$, can be calculated by using the step-wise equation:

$$\langle\langle f(x,y)\rangle\rangle_i(t) = \langle\langle f(x,y)\rangle\rangle_i(t-1) + \eta(t)\left[f(x(t), y(t))P_i(t) - \langle\langle f(x,y)\rangle\rangle_i(t-1)\right] \tag{7a}$$

$$\eta(t) = \left[1 + \lambda(t)/\eta(t-1)\right]^{-1}. \tag{7b}$$

After that, the new model parameter $\theta(t)$ is obtained as follows:

$$\mu_i(t) = \langle\langle x\rangle\rangle_i(t)/\langle\langle 1\rangle\rangle_i(t) \tag{8a}$$

$$\tilde{\Lambda}_i(t) = \frac{1}{1-\eta(t)}\left[\tilde{\Lambda}_i(t-1) - \frac{P_i(t)\tilde{\Lambda}_i(t-1)\tilde{x}(t)\tilde{x}'(t)\tilde{\Lambda}_i(t-1)}{(1/\eta(t)-1) + P_i(t)\tilde{x}'(t)\tilde{\Lambda}_i(t-1)\tilde{x}(t)}\right] \tag{8b}$$

$$\tilde{W}_i(t) = \tilde{W}_i(t-1) + \eta(t)P_i(t)\left(y(t) - \tilde{W}_i(t-1)\tilde{x}(t)\right)\tilde{x}'(t)\tilde{\Lambda}_i(t) \tag{8c}$$

$$\sigma_i^2(t) = \left(\langle\langle |y|^2\rangle\rangle_i(t) - \mathrm{Tr}\left(\tilde{W}_i(t)\langle\langle \tilde{x}y'\rangle\rangle_i(t)\right)\right) \Big/ \left(D\langle\langle 1\rangle\rangle_i(t)\right), \tag{8d}$$

where $\mathrm{Tr}(\cdot)$ denotes a matrix trace. $\tilde{\Lambda}_i(t) \equiv [\langle\langle \tilde{x}\tilde{x}'\rangle\rangle_i(t)]^{-1}$ is used to calculate $\Sigma_i^{-1}(t)$ by using the following relation:

$$\tilde{\Lambda}_i(t)\langle\langle 1\rangle\rangle_i(t) = \begin{pmatrix} \Sigma_i^{-1}(t) & -\Sigma_i^{-1}(t)\mu_i(t) \\ -\mu_i'(t)\Sigma_i^{-1}(t) & 1 + \mu_i'(t)\Sigma_i^{-1}(t)\mu_i(t) \end{pmatrix}. \tag{9}$$

If the discount factor $\lambda(t)$ converges to 1 over time, the on-line EM algorithm becomes a stochastic approximation method for finding the maximum likelihood estimator [2].

We also use dynamic unit manipulation mechanisms in order to efficiently allocate the units according to the input-output distribution of data [2]. If a given datum is too distant from all of the present units, a new unit is produced to account for it. A unit that has rarely been used to account for the data is deleted.

3. On-line EM reinforcement learning

In this section, we propose a new RL method based on the on-line EM algorithm. We consider optimal control problems for deterministic nonlinear dynamical systems that have continuous state and action spaces. The learning system is assumed to be able to observe the system state and to receive a reward corresponding to the state and the control at every time step.

3.1. Actor-critic architecture

Our learning method employs an architecture like the actor-critic model [3]. The actor yields a control signal for the current state. The critic predicts the expected return, which is the accumulation of rewards over time. However, our learning scheme differs from that used in the original actor-critic model as explained below.

For the observed current state $x_c(t)$, the actor yields a control signal (action) $u(t)$ based on a policy $\Omega(\cdot)$, i.e., $u(t) = \Omega(x_c(t))$. Using $u(t)$, the state $x_c(t)$ is changed into the next state $x_c(t+1)$ based on the system dynamics. After that, the learning system is given a reward $r(x_c(t), u(t))$. The goal of the learning system is to find the policy $\Omega(\cdot)$ that maximizes the discounted expected return defined by

$$V(x_c) \equiv \sum_{t=0}^{\infty} \gamma^t r\left(x_c(t), \Omega\left(x_c(t)\right)\right)\Bigg|_{x_c(0)=x_c}, \tag{10}$$

where $\gamma (0 < \gamma < 1)$ is a discount factor. $V(x_c)$, which is called the value function, is dependent on the current policy $\Omega(\cdot)$. In addition, the Q-function is defined by

$$Q(x_c, u) = r(x_c, u) + \gamma V(x_c(t+1)), \tag{11}$$

where $x_c(t) = x_c$ and $u(t) = u$ are assumed. $Q(x_c, u)$ indicates the expected return for the current state-action pair (x_c, u), when the policy $\Omega(\cdot)$ is used for the subsequent states.

3.2. Critic learning

From (10) and (11), the value function $V(\cdot)$ has the following relation to the Q-function.

$$V(x_c) = Q(x_c, \Omega(x_c)). \tag{12}$$

From (10), (11) and (12), the Q-function should satisfy the following consistency condition:

$$Q\left(x_c(t), u(t)\right) = r\left(x_c(t), u(t)\right) + \gamma Q\left(x_c(t+1), \Omega\left(x_c(t+1)\right)\right). \tag{13}$$

The critic approximates the Q-function based on (13). The critic is represented by the NGnet (1) and trained using the on-line EM algorithm.

3.3. Actor learning

According to our learning scheme, the actor approximates the soft-max policy that is dependent on the current Q-function. The soft-max policy π is defined by the conditional probability of an action u for a given state x_c:

$$\pi(u|x_c) = \frac{\exp\left[Q(x_c, u)/T\right]}{\int \exp\left[Q(x_c, u)/T\right] du}, \tag{14}$$

where T $(T > 0)$ is the temperature parameter. The soft-max policy is more likely to select an action that has a higher Q-function value. In the low temperature limit $(T \to 0)$, the soft-max policy always selects the optimal action that has the highest Q-function value. Namely, $E[u|x_c] \xrightarrow{T \to 0} \arg\max_u Q(x_c, u)$, where $E[u|x_c]$ is the expected value of u for a given state x_c.

The conditional probability (14) can be derived from the following joint probability distribution for a state-action pair (x_c, u):

$$P_{Q,\rho}(x_c, u) = \exp\left[Q(x_c, u)/T\right] \rho(x_c) / Z_{Q,\rho}, \tag{15}$$

where $\rho(x_c)$ is an unknown distribution density of a state x_c. $Z_{Q,\rho} \equiv \iint \exp\left[Q(x_c,u)/T\right]\rho(x_c)dx_c du$ is a normalization factor. In our learning scheme, the actor approximates the probability distribution (15). Let $P(x_c,u|\theta)$ be the probability distribution realized by the stochastic NGnet corresponding to the stochastic actor. Here, x and y in equation (2) are replaced by x_c and u, respectively. The distance between the stochastic NGnet $P(x_c,u|\theta)$ and the probability distribution $P_{Q,\rho}(x_c,u)$ is given by the following KL-divergence:

$$\begin{aligned} KL(\theta) &\equiv \int P_{Q,\rho}(x_c,u)\log\left[\frac{P_{Q,\rho}(x_c,u)}{P(x_c,u|\theta)}\right]dx_c du \\ &= -\int P_{Q,\rho}(x_c,u)\log P(x_c,u|\theta)dx_c du + (\theta\textit{-independent term}). \end{aligned} \tag{16}$$

The criterion of the learning is to minimize $KL(\theta)$, namely, to maximize the following objective function with respect to the actor's model parameter θ.

$$J(\theta) \equiv \int P_{Q,\rho}(x_c,u)\log P(x_c,u|\theta)dx_c du. \tag{17}$$

$J(\theta)$ becomes maximum when $P = P_{Q,\rho}$, i.e., P represents the soft-max policy (14).

The stochastic NGnet $P(x_c,u|\theta)$ has a hidden variable i. The maximization of (17) can be achieved by defining the following expected objective function:

$$JC(\theta|\bar{\theta}) \equiv \int P_{Q,\rho}(x_c,u)\sum_{i=1}^{M} P(i|x_c,u,\bar{\theta})\log P(x_c,u,i|\theta)dx_c du, \tag{18}$$

where $\bar{\theta}$ is the current model parameter of the stochastic NGnet. We can prove that $JC(\theta|\bar{\theta}) \geq JC(\bar{\theta}|\bar{\theta})$ implies $J(\theta) \geq J(\bar{\theta})$. In order to maximize $J(\theta)$ with respect to θ, we maximize $JC(\theta|\bar{\theta})$ with respect to θ by using the modified EM algorithm.

In the actor learning phase, the stochastic NGnet is trained to approximate the soft-max policy from the state-action trajectory $\mathcal{X} \equiv \{(x_c(t),u(t)) \mid t=1,...,t_{max}\}$. $\mathcal{X}$ is assumed to be drawn independently according to a distribution $\rho(x_c)$ and the current fixed stochastic actor with parameter θ_0. Namely, $(x_c,u) \sim \rho(x_c)P(u|x_c,\theta_0)$. In this case, the expectation of any function $f(x_c,u)$ can be approximated from its empirical mean:

$$\int f(x_c,u)\rho(x_c)P(u|x_c,\theta_0)dx_c du \approx \frac{1}{t_{max}}\sum_{t=1}^{t_{max}} f(x_c(t),u(t)). \tag{19}$$

The approximation becomes exact as $t_{max} \to \infty$. By using (15), (18) and (19), $JC(\theta|\bar{\theta})$ is estimated from the observation $\mathcal{X}$ and the current Q-function as follows:

$$JC(\theta|\bar{\theta},\mathcal{X}) \approx \frac{1}{t_{max}}\frac{1}{Z_{Q,\rho}}\sum_{t=1}^{t_{max}}\sum_{i=1}^{M}\bar{h}(i|x_c(t),u(t),\bar{\theta})\log P(x_c(t),u(t),i|\theta) \tag{20a}$$

$$\bar{h}(i|x_c(t),u(t),\bar{\theta}) \equiv \frac{P(i|x_c(t),u(t),\bar{\theta})}{P(u(t)|x_c(t),\theta_0)}\exp\left[Q(x_c,u)/T\right]. \tag{20b}$$

Note that equation (20a) is similar to equation (4). In addition, the actual value of $Z_{Q,\rho}$ is not important for the maximization of $JC(\theta|\bar{\theta},\mathcal{X})$. Since equation (20a) has a similar form to (4), the M-step can be exactly solved.

Accordingly, the EM algorithm for the stochastic actor is defined as follows. In an E-step, $\bar{h}(i|x_c(t),u(t),\bar{\theta})$ is calculated by (20b). This can be done by using the previous model parameter $\bar{\theta}$, the current Q-function, and the fixed actor model θ_0 that produces the observed trajectory $\mathcal{X}$. The solution for $\partial JC(\theta|\bar{\theta},\mathcal{X})/\partial\theta = 0$ is obtained in an M-step. We derive the modified M-step equations by replacing the weighted mean (5) with the following equation:

$$\langle f(x_c,u)\rangle_i(t_{max}) \equiv \frac{1}{t_{max}}\sum_{t=1}^{t_{max}} f(x_c(t),u(t))\bar{h}(i|x_c(t),u(t),\bar{\theta}). \tag{21}$$

$J(\theta)$ can be maximized by repeating the above E- and M-steps. Similarly, the on-line EM algorithm for the stochastic actor is derived. $\bar{h}(i|x_c(t),u(t),\theta(t-1))$ is calculated in the modified E-step. The step-wise equation (8a) is then replaced by

$$\langle\langle f(x_c,u)\rangle\rangle_i(t) = \langle\langle f(x_c,u)\rangle\rangle_i(t-1) + \eta(t)\left[f(x_c(t),u(t))\bar{h}(i|x_c(t),u(t),\theta(t-1)) - \langle\langle f(x_c,u)\rangle\rangle_i(t-1)\right]. \tag{22}$$

In the modified M-step, the weighted means are updated by (22) and the new model parameter is determined by (8).

3.4. Actor-critic learning

The learning process for the actor-critic architecture process is as follows.

1. Control and critic-learning phase
   - 1-1. For the current state $x_c(t)$, the current actor outputs a stochastic action $u(t)$. The stochastic action is generated by using the stochastic NGnet in the following way. A unit i is selected with the conditional probability $P(i|x_c)$ for state x_c. After that, an action $u(t)$ is generated with the conditional probability $P(u|x_c, i)$ for state x_c and the selected i.
   - 1-2. Using $u(t)$, the system changes its state to $x_c(t+1)$ according to the system dynamics. The learning system observes reward $r(x_c(t), u(t))$.
   - 1-3. The on-line EM algorithm is used to train the critic. The input to the critic NGnet is the state-action pair, i.e., $(x_c(t), u(t))$. The target output is the right-hand side of (13). Note that the second right-hand side term can be defined for either the stochastic policy or the deterministic policy. We use the deterministic policy, which is defined by the expected output of the stochastic actor, because we use the deterministic actor after the training. Therefore, the target output for the critic NGnet is calculated using the current critic and the current deterministic actor, both evaluated at $t+1$.

   The learning system repeats this process for a certain period of time (t_{max}), using a fixed actor. In this period, the critic NGnet is modified to approximate the Q-function for the fixed actor, and the state-action trajectory $\mathcal{X} \equiv \{ (x_c(t), u(t)) \mid t = 1, 2, ..., t_{max}\}$ is saved.
2. Actor-learning phase

   The actor NGnet is trained from the saved trajectory $\mathcal{X}$ by using the modified on-line EM algorithm described in Section 3.3.

4. Experiment

In order to examine the performance of the above learning method, we consider two control tasks. The first task is to swing up and stabilize a single pendulum with a limited torque controller [1, 5]. The state of the system is denoted by $x_c \equiv (q, \dot{q})$, where q is the angle of the pendulum from the upright position and $\dot{q}$ is the angular velocity. The torque u is assumed to be limited to the range $|u| \le u_{max}$, where u_{max} is not large enough to bring the pendulum up without swinging.

The following process is conducted during a single learning episode. After the pendulum is released from the vicinity of the upright position, the learning system observes the system state and yields a control signal according to the stochastic actor at every 0.01 second. A reward $r\left(x_c(t), u(t)\right)$ is given by $\tilde{r}\left(x_c(t+1)\right)$, which is defined by

$$\tilde{r}(x_c) = \exp\left(-q^2/\nu_1 - \dot{q}^2/\nu_2\right), \tag{23}$$

where ν_1 and ν_2 are constants. The reward $\tilde{r}(x_c) \in [0, 1]$ encourages the pendulum to stay in a high position. This control process is conducted for 7 seconds. The learning process for a single episode has been described in Section 3.4. The reinforcement learning proceeds by repeating these episodes.

After 17 learning episodes, the system is able to stabilize the pendulum at the upright position from almost all initial states. For example, the system is able to swing-up and stabilize the pendulum at the upright position when the system initially stays still at the bottom. Figure 1 shows a stroboscopic time-series of this control process.

The second task is to balance an acrobot near an upright position [6]. The acrobot is the two-link underactuated robot depicted in figure 2. The second joint exerts torque while the first joint does not. This task is very difficult because the system has a nonlinear and unstable dynamics. After 18 learning episodes, however, our method is able to make the acrobot stand straight up when the initial state is close to the upright position. Figures 3 and 4 show a typical control process after the learning. Figure 3 shows a stroboscopic time-series of the acrobot state and figure 4 shows the time sequence of the state and the control signal. The dashed (dotted) line denotes the angle of the first (second) joint and the solid line denotes the control signal. The system gradually suppresses the oscillation of the two joints and finally makes them stand straight up at the upright position.

These two tasks were also examined in our previous studies [1, 6], where the actor's deterministic NGnet was trained according to the on-line EM algorithm by using the gradient of the critic NGnet. In the previous method, good control for the inverted pendulum (for the balancing acrobot) was acquired after 40 (37) episodes. The new learning method is thus more efficient than the previous method.

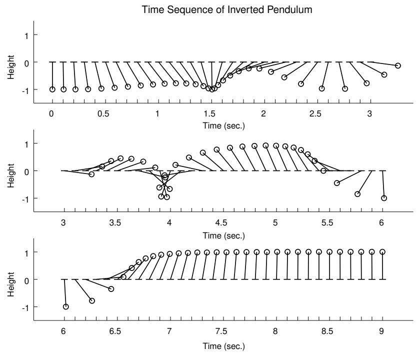

Figure 1: Control for inverted pendulum

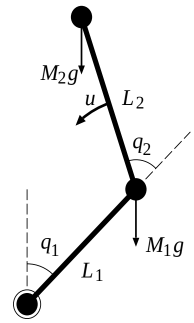

Figure 2: The acrobot

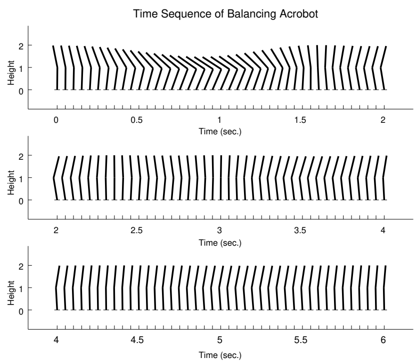

Figure 3: Control for balancing acrobot

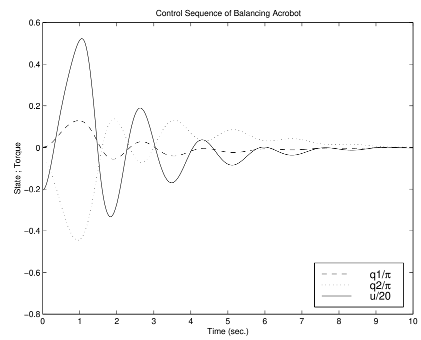

Figure 4: State-control sequence

5. Conclusion

In this article, we have proposed a new RL method, which has an architecture like the actor-critic model. The critic approximated the Q-function. The actor approximated a stochastic soft-max policy defined by the Q-function. The on-line EM algorithm is used to train the critic and the actor. The new method was applied to two automatic control problems and fairly good control was obtained after a few learning trials. We can thus confirm that our new RL method is based on a good function approximator and a fast learning algorithm, which are crucial for tasks that have continuous state and action spaces.

6. References

[1] Sato, M. & Ishii, S., in *Advances in Neural Information Processing Systems 11*, pp.1052-1058, 1999

[2] Sato, M. & Ishii, S., *Neural Computation*, **12**(2), 2000

[3] Barto, A. G., Sutton, R. S. & Anderson, C. W., *IEEE Transactions on Systems, Man, and Cybernetics*, **13**, pp.834-846, 1983

[4] Moody, J. & Darken, C. J., *Neural Computation*, **1**, pp.281-294, 1989

[5] Doya, K., in *Advances in Neural Information Processing Systems 8*, pp.1073-1079, 1996

[6] Yoshimoto, J., Ishii, S. & Sato, M., in *1999 IEEE International Conference on Systems, Man and Cybernetics*, **V**, pp.516-521, 1999

[7] Dempster, A. P., Laird, N. M. & Rubin, D. B., *Journal of Royal Statistical Society B*, **39**, pp.1-22, 1977

[8] Xu, L., Jordan, M. I. & Hinton, G. E., in *Advances in Neural Information Processing Systems 7*, pp.633-640, 1995