A New Hybrid Model for Associative Reinforcement Learning

H. Montazeri
Soft Computing Laboratory
Computer Engineering and IT Department
Amirkabir University of Technology
Tehran, Iran
montazeri@autac.ir

M. R. Meybodi
Soft Computing Laboratory
Computer Engineering and IT Department
Amirkabir University of Technology
Tehran, Iran
mmeybodi@autac.ir

Abstract—**In this paper, a new model, addressing the Associative Reinforcement Learning (ARL) problem, based on learning automata and self organizing map is proposed. The model consists of two layers. The First layer comprised of a SOM which is utilized to quantize the state (context) space and the second layer contains of a team of learning automata which is used to select an optimal action in each state of the environment. First layer is mapped to the second layer via an associative function. In other words, each learning automaton is in correspondence with only one neuron of the self organizing map. In order to show the performance of the proposed method, it has been applied successfully to classification applications on Iris, Ecoli, and Yeast data sets, as examples of ARL task. The results of experiments show that the proposed method is reached the accuracy near to or even higher than the highest reported accuracy. The results obtained for Ecoli and Yeast data sets indicate that the method is able to classify in relatively high dimensional context space and high number of classes.**

I. INTRODUCTION

Associative reinforcement learning (ARL) task defined originally by Barto and Anandan [1] is one that requires the learning element to establish a connection between input and output. ARL tasks involve the following interaction between the environment and the learning system. At time step n, the environment provides the learning system with some context vector $X(n)$ selected from a set of vectors $X \subseteq \Re^n$, where $\Re$ is set of real numbers. Based on this input, the learning system selects an action $\alpha(n)$ among its action set. The environment evaluates the action $\alpha(n)$ in the context of the input $X(n)$ and sends to the learning system a real-valued evaluation signal $\beta(n+1) \in [0,1]$ at time step *n+1*, with $\beta(n+1) = 0$ denoting the maximum evaluation.

Many works have been done on ARL, from which only a few are mentioned here. In original definition of ARL tasks, Barto and Anandan [1] only considered learning tasks for which the evaluation is a binary-valued success/failure signal (i.e. $\beta(n) \in \{0,1\}$). Their work in such tasks led to the development of the associative reward-penalty algorithm (A_{R-P}).

Some solutions of ARL tasks utilize a set of parameters. In this case, the learning task is to find the optimal values of the parameters. The complementary reinforcement backpropagation algorithm (CRBP) [2], an approach based on neural network, consists of a feed-forward network mapping an encoding of the context vector to an encoding of the action. The action is determined probabilistically from the activation of the output units. A supervised training procedure is used to adapt the network as follows. If the evaluation of the selected action is success ($\beta = 1$), then the network is trained to use the action whenever this context vector is provided. If an action fails to generate reward, CRBP will endeavor to generate an action that is different from current choice. Another work take advantage of set of parameters is presented In [3]. In this algorithm, each action (or alternative) has an attribute vector associated with it, and for each the action it selects, it gets either a success or failure. The method incorporates a learning method like Widrow-Hoff rule and a probabilistic selection strategy to solve the ARL problem. Another method of this category is an automaton-based approach which will be described later.

Some methods consider ARL tasks from an analytical point of view. Streth shows that associative prediction problem, a version of ARL, can be reduced to cost-sensitive classification and then to standard classification [4]. In [5], confidence bounds are used to deal with situations which exhibit an exploitation-exploration trade-off. In [6], efficient algorithms to solve a restricted class of RL problem are proposed. These algorithms can learn efficiently action policies that can be expressed as propositional formula in k-DNF (disjunctive normal form). Abe et al. consider the problem of reinforcement learning with immediate rewards in a worst-case theoretical framework. This work provides bounds on the per-trial regret that go to zero as the number of trials approaches infinity [7]. In [8, 9], immediate reward reinforcement learning problem is reduced to a reward-weighted nonlinear regression problem, which greatly

accelerates the speed of learning. The method is successfully applied on simulated anthropomorphic robot arms.

Two learning automata-based methods dealing with associative reinforcement learning problems are Generalized Learning Automata (GLA) and a team of Finite Action-Set Learning Automata (FALA) [10]. In the GLA approach, the structure of LA is modified to allow for context vector input [11, 12]. In the other method, a parameterized class of discrimination function is defined. The learning problem is to obtain the optimal values of the parameters. Considering actions of automata as parameter values, optimal actions can be learned without reference to any context. In [13], a team of Finite Action-Set Learning Automata (FALA) is used to learn the optimal parameters in ARL problems.

In this paper, a new model, dealing with the ARL problem, based on learning automata and self organizing map is proposed. The model is composed of two layers. First layer comprised of a SOM which is used to represent the state (context) space. The second layer consists of a learning automaton team. The LA team is responsible for selecting an optimal action in each state of the environment and receiving the environment feedback. Two layers are related with each other via an associative function. In other words, each learning automaton is in correspondence with only one neuron of self organizing map. In order to show the performance of the proposed method, it has been applied successfully to classification applications on Iris, Ecoli, and Yeast data sets, as examples of ARL task. The results of experiments show that the proposed method is reached the accuracy near to or even higher than the highest reported accuracy. The results obtained for Ecoli and Yeast data sets indicate that the method is able to classify in relatively high dimensional context space and high number of classes.

The rest of the paper is organized as follows. Section 2 briefly introduces self organizing map. Section 3 describes learning automata. In section 4 the proposed model is described. Section 5 discusses the results of applying the proposed model on several classification problems. The conclusion remark is presented in section 6.

II. Self Organizing Map

The SOM usually consists of an array of units arranged in a grid. Associated with each unit, *t*, is a weight vector, $\boldsymbol{w^t} = [w_1^t, w_2^t, \dots, w_D^t]$ where *D* is the dimensionality of the input data. The aim is to find a suitable set of weights for each unit so that the network models the distribution of the input data in the input space. In many cases, the intrinsic dimensionality of the distribution may be low, even though the dimensionality of the input data itself is high. In these cases, the SOM can be used as a dimensionality reduction technique [14].

The learning rule responsible for finding a suitable set of weights is simple: given an input vector, $x = [x_1, x_2, \dots, x_D]$ the distance between each unit, *t*, of the SOM and the input vector is calculated by (1).

$$\sum_{d=1}^{D} (x_d - w_d^t)^2 \qquad (1)$$

The unit with the smallest distance is the one which most closely represents the current input, and is thus considered the winner for that input. The weights of the winning unit are updated towards that input. In addition to the winning unit, the neighbors of the winning unit are also updated towards the input vector but by an amount that decays with the distance of those neighbors from the winning unit. The neighborhood function which controls this is often Normally distributed around the winning unit.

The weights of the map are initialized to random values and then the above process is iterated for each input vector in the data set, effectively resulting in a competition between different regions of the input space for units of the map. Dense regions of the input space will tend to attract more units than sparse ones, with the distribution of units in weight space ultimately reflecting the distribution of the input data in the input space. Neighborhood learning also encourages topology preservation with units close in the topology of the map ending up close in the weight space too [14].

III. Learning Automata

Learning automaton can be defined as an abstract model which randomly selects one action out of its finite set of actions and performs it on a random environment. Environment then evaluates the selected action and responses to the automaton with a reinforcement signal. Based on selected action, and received signal, the automaton updates its internal state and selects its next action. Figure 1 depicts the automaton's interaction with its environment.

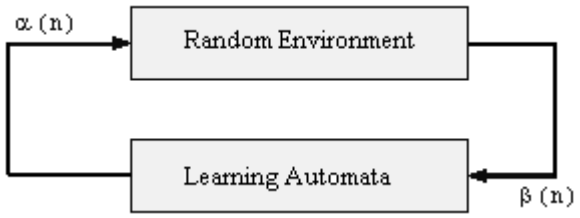

Figure 1: Learning automaton acting in an environment

Environment can be defined by the triple $E = \{\alpha, \beta, c\}$ where $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ represents a finite input set, $\beta = \{\beta_1, \beta_2, \dots, \beta_r\}$ represents the output set, and $c = \{c_1, c_2, \dots, c_r\}$ is a set of penalty probabilities, where each element c_i of c corresponds to one input of action α_i. An environment in which β can take only binary values 0 or 1 is referred to as P-model environment. A further generalization of the environment, known as Q-model, allows finite output set with more than two elements that take values in the interval [0, 1]. A further step in this direction is the S-model whose responses can take continuous values over the unit interval [0, 1].

Learning automata can be classified into fixed-structure LA, and variable-structure LA. Fixed-structure automata are characterized by state transition probabilities that are fixed.

Tsetline, Krinsky, and Krylov are examples of this kind of learning automata. A variable-structure automaton is defined by the quadruple $\{\alpha, \beta, p, T\}$ in which $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ represents the action set of the automata, $\beta = \{\beta_1, \beta_2, \dots, \beta_r\}$ represents the input set, $p = \{p_1, p_2, \dots, p_r\}$ represents the action probability set, and finally learning algorithm is defined as $p(n+1) = T[\alpha(n), \beta(n), p(n)]$. This automaton operates as follows. Based on the action probability set *p*, automaton randomly selects an action α_i, and performs it on the environment. After receiving the environment's reinforcement signal, automaton updates its action probability set based on following reinforcement scheme, equations (2) for favorable response, and equations (3) for unfavorable one. It is noteworthy that this reinforcement scheme is for multi-action learning automata acting in the P-model environment [15].

$$\begin{aligned} p_i(n+1) &= p_i(n) + a\big(1 - p_i(n)\big) \\ p_j(n+1) &= p_j(n) - a p_j(n) \qquad \forall\, j \neq i \end{aligned} \tag{2}$$

$$\begin{aligned} p_i(n+1) &= (1-b)p_i(n) \\ p_j(n+1) &= \frac{b}{r-1} + (1-b)p_j(n) \quad \forall\, j \neq i \end{aligned} \tag{3}$$

In these two equations, a and b are reward and penalty parameters respectively. For a=b, learning algorithm is called L_{R-P}, for $b << a$, it is called $L_{R\varepsilon P}$, and for $b = 0$, it is called L_{R-I}.

IV. The Proposed Model

In this section, a new hybrid model for associative reinforcement learning (ARL) obtained by combining self organizing map and learning automata is proposed. An associative reinforcement learning (ARL) agent is shown in figure 2. The ARL agent's interaction with its environment is similar to the Interaction of learning automata with its environment (figure 1) except that in addition to the environment response it also gets a context vector from the environment.

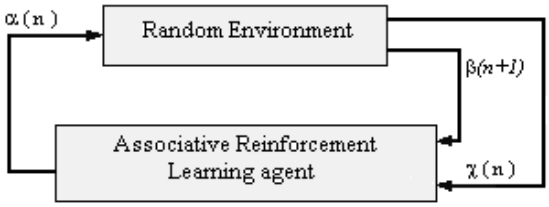

Figure 2: ARL agent acting in an environment

The proposed model can be defined by $\{S, \zeta, N, \phi, T\}$ in which *S* is the self organizing map used to represent state space; $\zeta = \{L_i | \, 1 \leq i \leq M\}$ *is* a team of *M* learning automata, where M is less than or equal to the number of SOM neurons; *N* is the topology of learning automaton team, which is assumed to be the same as the topology of SOM in this paper; ϕ is a function which specifies associations between each neuron of SOM and each learning automaton of LA team (ζ); and finally *T* is the learning algorithm which is explained in more details later.

Figure 3 depicts the model structure and its interaction with the environment. The model contains two layers. First layer incorporates a SOM which is used to quantize the state (context) space. The second layer comprised of a learning automata team which is used to select the optimal action. First layer is mapped to the second layer via ϕ function. In other words, each learning automaton is in correspondence with only one neuron of the SOM. The interaction of the model with the environment is as follows: environment provides the ARL agent with a context vector; the winner neuron which most closely represents this context vector is determined; learning automaton associated to the winner neuron is determined using ϕ function; the selected learning automaton chooses an action and performs it in the environment; the environment evaluates the action and provides the LA with a reinforcement signal; using this signal, LA and Its neighbors update their action selection strategies.

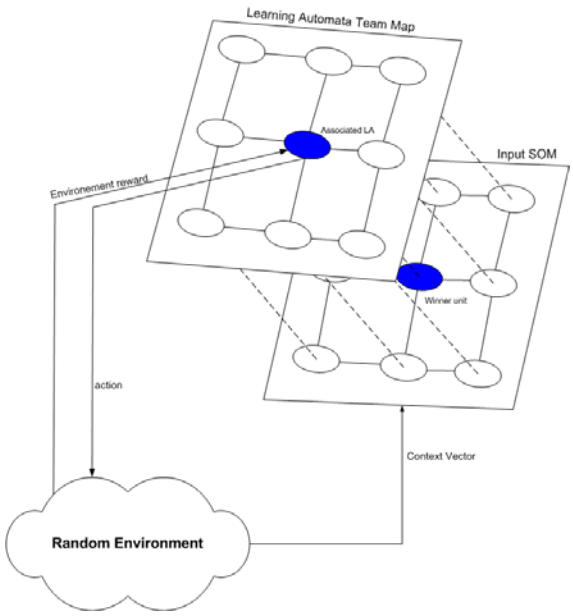

Figure 3: The Proposed model

The algorithm is summarized as follow:

1. ARL agent gets the context vector from the environment.
2. The SOM determines the winner neuron.
3. The learning automaton in correspondence with the winner neuron is selected using ϕ function.
4. The learning automaton selects an action and performs it on the environment.
5. The environment evaluates the action of LA and gives a reinforcement signal to the LA team.
6. SOM updates its neurons towards the context vector.
7. LA is updated based on the reinforcement signal.

8. (Only if variable structure LA is used) LA neighbors are updated based on the reinforcement signal provided to executing LA but with smaller reward and penalty parameter.

V. Experimental Results

Experimental results for several pattern recognition problems are reported in this section. Figure 4 demonstrates the first problem. This simple problem is selected to illustrate various feature of our model, later we use the model in more difficult problems. In this problem, the environment provides the context vectors uniformly from $[0,1] \times [0,1]$. The discrimination function to be learned is shown in figure 4 and mathematically defined by $[2x_1 - x_2 > 0] \wedge [-x_1 + 2x_2 > 0]$. The task of ARL agent is to classify the given context vector as *class A* or *class B*. In this experiment, the environment provides the ARL agent with 15000 samples. After learning each sample, the performance of the ARL agent is calculated. Performance is the number of correct classified context vector of the test set. The experiment is repeated over 10 independent trials. The figure 5 shows the average performance against time over 10 independent trials and the bars show the best and worst trial among 10 trials. The figure indicates that the method learns to classify the provided context vector properly.

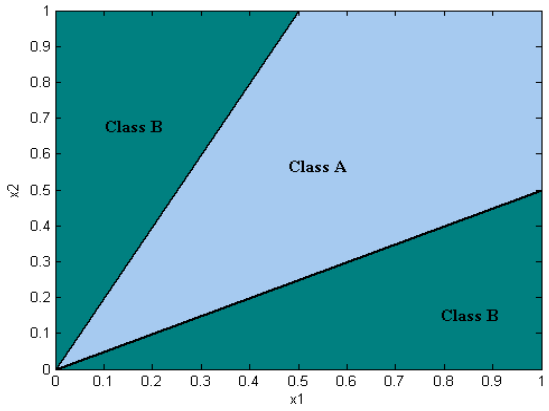

Figure 4: Discrimination function

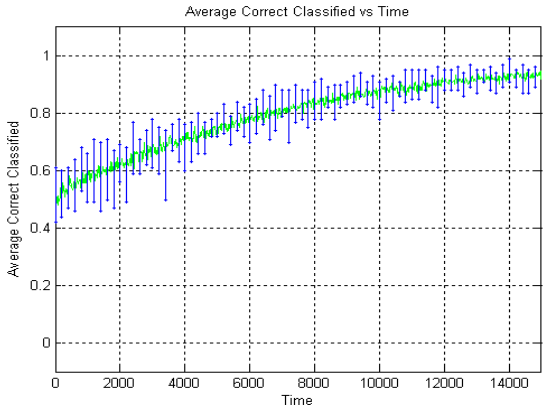

Figure 5 average performance of proposed model against time over 10 independent trials

Figure 6 demonstrates how the model changes its internal structure against the time. It shows change in the SOM neurons positions and the probabilities of LA actions during learning. The figure 6 comprise of four snapshots (step 1, 1000, 5000, 15000) of learning. For each step, left diagram shows the positions of SOM neurons and the topology of the SOM. Each neuron is determined with black or white circle. Black circle determines that selection probability of class B in corresponding LA is greater than class A and white circle is defined vice versa. Right diagram shows the Vornoi diagrams for the left diagram. The diagram determines that each neuron which portion of the state space is classifying as *class A* or *class B*.

Figure 6-a shows the internal structure of the model in the step 1. In this step, the neuron weights of the SOM are selected randomly. So the topology of the SOM neurons is not preserved in the weight space. Irregular distribution of black or white circles in the Vornoi diagram implies that the selection probability of *class A* or *B* for each LA is also random.

In the step 1000, in spite of preserving the topology in the weight space, the SOM could not properly estimate the uniform distribution of context space. Also several misclassifications can be seen which some of them is shaded in red in corresponding Vornoi diagram. The shaded region shows that associated LA has not converged yet. Misclassifications can be seen either in the middle of the *class A/B* region or boundary of the *class A* and *B*. The figure 6-c shows the step 5000 of learning. In this step, the topology is properly preserved but the SOM could not represent the uniform distribution of context space very well. The neurons need minor change in order to elegantly represent the space. For instance, distribution of neurons in regions which are bounded in red ellipse is not uniform, so that the SOM need more learning sample to converge to a suitable representation. In the Vornoi diagram of this figure, some misclassifications are shaded in red. In this case, misclassifications are restricted to some neurons in the boundary of *class A* and *B*.

Figure 6-d shows the step of 15000 learning. In this step, the topology is preserved; the neurons uniformly distributed in the state space; and as can be seen misclassifications are occurred neither in the middle of the *class A/B* region nor boundary of the *class A and B*.

Table 1 Evaluation of the model on three standard data sets

Name	**#instances**	**#attributes**	**#classes**	**Highest reported accuracy**	**Accuracy of proposed method**
Iris	150	4	3	100%	%99
Ecoli	336	7	8	81%	%85
Yeast	1484	8	10	55%	%52

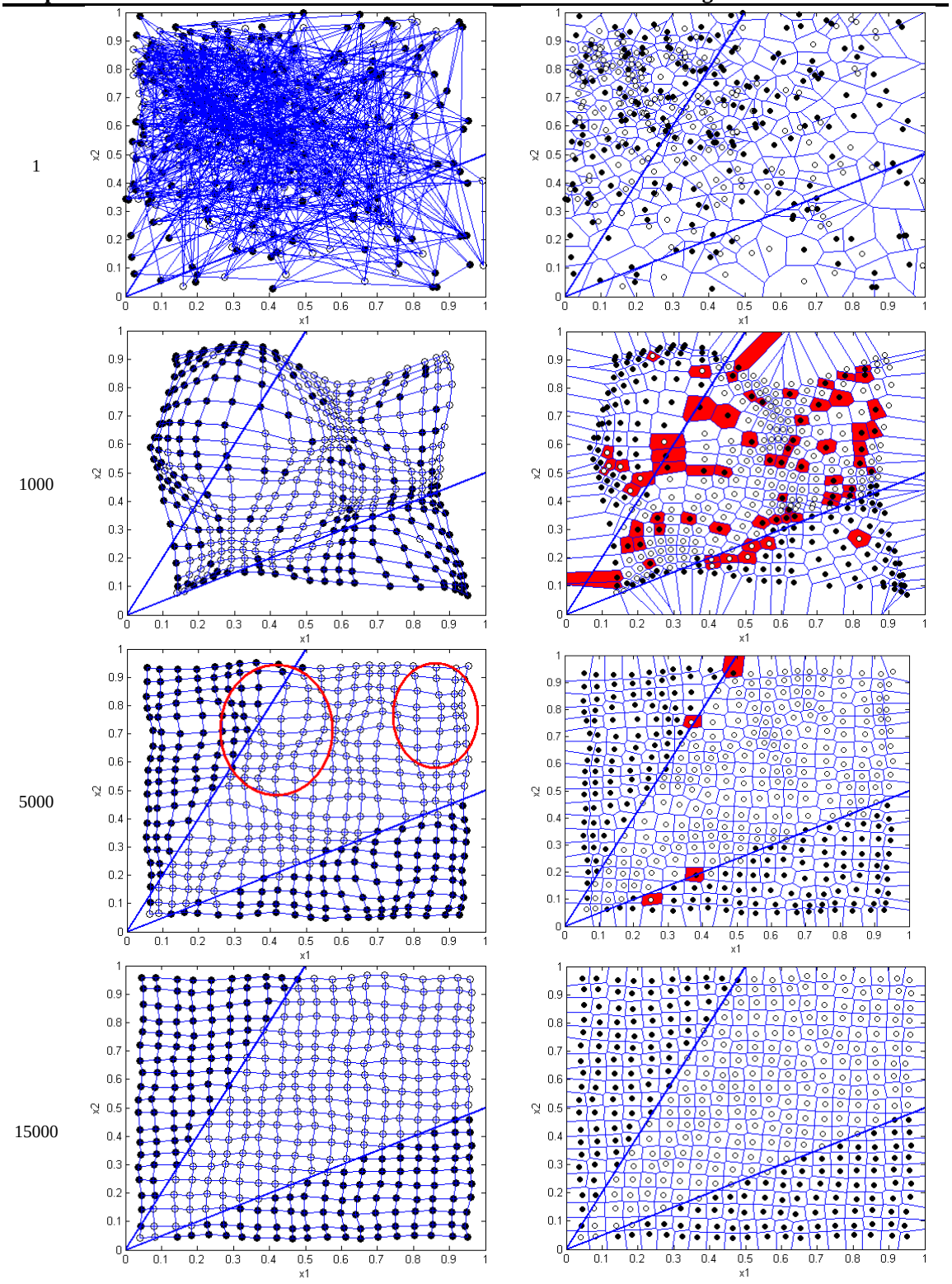

Figure 6: The internal structure of the model in the step 1, 5000, 10000, and 15000. Diagrams in the left column demonstrate SOM topology in each step. Right diagrams indicate Vornoi diagram for each step. Each neuron is determined with black or white circle. Black circle determines that selection probability of class B in corresponding LA is greater than class A and white circle is defined vice versa.

The previous example illustrates how the proposed model works in a simple problem. In the following, we present the evaluation of the proposed model on Iris, Ecoli, and Yeast data sets [16]. The characteristics of each data set, highest accuracy of previous methods, and accuracy of the proposed method is presented in table 1. The results for the proposed model are calculated using 10-fold cross validation procedure and each result is averaged over 10 runs. The results demonstrate that the proposed method is reached the accuracy near to (for Ecoli, better than) highest reported accuracy. Applying the model on Ecoli and Yeast dataset shows that the method is able to deal with classification problems with relatively high dimensional context space and high number of classes.

VI. CONCLUSION

In this paper, we described a new hybrid model that uses evaluative performance feedback to learn associative maps from context vector to a set of actions. The new hybrid model is based on the SOM and a team of Learning Automata. Various features of the model were illustrated by a simple classification problem. This simple problem indicated that the proposed model learns to preserve topology in the weight space, to represent properly the distribution of the context vector, and to classify correctly the given context vector. Experimental results on three standard data sets showed that the proposed method can perform near to or even better than the best previous method. The experiments also indicated that the method can be used in high dimensional context space and high number of classes.

REFERENCES

[1] A. G. Barto and P. Anandan, "Pattern recognizing stochastic learning automata," *IEEE Transactions on Systems, Man, and Cybernetics* vol. 15, pp. 360-375, 1985.

[2] M. L. Littman and D. H. Ackley, "Generalization and scaling in reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 2, pp. 550-557, 1990.

[3] N. Abe and P. M. Long, "Associative reinforcement learning using linear probabilistic concepts," Proceedings of *16th International Conf. on Machine Learning* Morgan Kaufmann, San Francisco, CA, pp. 3-11, 1999.

[4] A. L. Strehl, C. Mesterharm, M. L. Littman and H. Hirsh, "Experience-efficient learning in associative bandit problems," Proceedings of *23rd international conference on Machine learning* Pittsburgh, PA, , pp. 889-896, 2006.

[5] P. Auer, "Using confidence bounds for exploitation-exploration trade-offs," *Journal of Machine Learning Research*, vol. 3, pp. 397-422, 2003.

[6] L. P. Kaelbling, "Associative reinforcement learning: Functions in k-DNF," Machine *Learning*, vol. 15, pp. 279-298, 1994.

[7] N. Abe, A. W. Biermann and P. M. Long, "Reinforcement learning with immediate rewards and linear hHypotheses," *Algorithmica*, vol. 37, pp. 263-293, 2003.

[8] J. Peters and S. Schaal, "Using reward-weighted regression for reinforcement Learning of Task Space Control," in IEEE Symposium on Approximate *Dynamic Programming and Reinforcement Learning* Honolulu, HI, 2007, pp. 262-267.

[9] J. Peters and S. Schaal, "Reinforcement learning for operational space control," Proceedings of *IEEE International Conference on* Robotics *and Automation* Roma, Italy, pp. 2111-2116, 2007.

[10] M. A. L. Thathachar and P. S. Sastry, "Varieties of learning automata: an overview," *IEEE Transactions on Systems, Man and Cybernetics-Part B*, vol. 32, pp. 711-722, 2002.

[11] V. V. Phansalkar, "Learning automata algorithms for connectionist systems-Local and global convergence," *Department of* Electrical *Engineering*, Ph.D. dissertation, Bangalore, India: Indian Institute of Science, 1991.

[12] V. V. Phansalkar and M. A. L. Thathachar, "Local and global optimization algorithms for generalized learning automata," *Neural Computation*, vol. 7, pp. 950-973, 1995.

[13] M. A. L. Thathachar and V. V. Phansalkar, "Convergence of teams and hierarchies of learning automata in connectionist systems," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, pp. 1459-1469, 1995.

[14] T. Kohonen, *Self*-Organizing *Maps*: Springer, 2001.

[15] K. S. Narendra and M. A. L. Thathachar, *Learning automata: an* introduction: Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1989.

[16] A. Asuncion and D. J. Newman, "UCI machine learning repository,"http://www.ics.uci.edu/~mlearn/MLRepository.html, University of California, Irvine, School of Information and Computer Sciences, 2007.