A Reinforcement-Learning Approach to Robot Navigation

Mu-Chun Su, De-Yuan Huang, Chien-Hsing Chou*, and Chen-Chiung Hsieh**
Department of Computer Science & Information Engineering, National Central University, Taiwan, R.O.C.
*Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C.
**Integration Technology Laboratory, Institute for Information Industry, Taiwan, R.O.C.
muchun@csie.ncu.edu.tw

Abstract - *This paper presents a reinforcement-learning approach to a navigation system which allows a goal-directed mobile robot to incrementally adapt to an unknown environment. Fuzzy rules which map current sensory inputs to appropriate actions are built through the reinforcement learning. Simulation results illustrate the performance of the proposed navigation system.*

Keyword: robot, navigation, reinforcement learning, neural network, neuro-fuzzy system

1 Introduction

Efficient navigation strategies are very crucial for autonomous mobile robots operating in unknown environments. If a priori complete information about the environment is known the global path planning, such as configuration space method [1], potential field method [2], and generalized Voronoi diagram [3] is preferred. However, global path planning is not suitable for an unknown and dynamically changing environment. Thus, reaction or behavior-based systems have been offered as alternative methods to designing autonomous mobile robots [4]-[14]. Recently, several different approaches to applying a reinforcement learning mechanism in the designing of autonomous mobile robots have been proposed [15]-[17]. With no need of a lot of training patterns, the reinforcement-learning-based robots can incrementally infer a reactive control policy based on a reinforcement signal.

An overview of reinforcement learning can be found in the article [18]. This article surveys the field of reinforcement learning from a computer-science perspective. The approaches to solving reinforcement-learning problems can be roughly divided into two strategies. The first is to search in the space of behaviors in order to find one that performs well in the environment. This approach has been taken by work in genetic algorithms and genetic programming as well as some more novel search techniques [19]. The second is to use statistical techniques and dynamic programming methods to estimate the utility of taking actions in states of the world, for example, adaptive heuristic critic [20], temporal difference ($TD(l)$) [21] and Q-learning [22]-[23], etc. Each approach has its own advantages and disadvantages. For example, some reinforcement learning algorithms may suffer from the amount of time required to learn an effective strategy.

In our previous work [24], we developed an adaptive classifier-system-based neuro-fuzzy inference system called ACSNFIS, which uses a special reinforcement learning algorithm to incrementally construct its architecture and tune the system's parameters. The special reinforcement learning algorithm was motivated by the so-called *classifier system* which was a machine learning system that learns syntactically simple string rules (called classifiers) [25]-[26]. In this paper, we use the ACSNFIS as the main network architecture to implement the reinforcement-learning-based navigation system.

The organization of the paper is as follows. In Section II, we first briefly introduce the ACSNFIS. Then the problem description and simulation results of an autonomous robot are described in Section III. Finally, Section IV concludes the paper.

2 Brief Review of ACSNFIS

In our previous work [24], a new approach to fuzzy classifier systems was proposed and a class of adaptive classifier-system-based neuro-fuzzy inference system referred to as ACSNFIS was developed to implement such new fuzzy classifier systems. The proposed ACSNFIS can not only tune its parameters but also incrementally construct its architecture in a reinforcement-learning environment.

Through injecting new rules into the system, an ACSNFIS can explore the possible solution space. The major difference between the proposed ACSNFIS and other neuro-fuzzy systems is that each rule in an ACSNFIS is associated with a strength parameter updated by a credit assignment method such as the fuzzy bucket brigade algorithm.

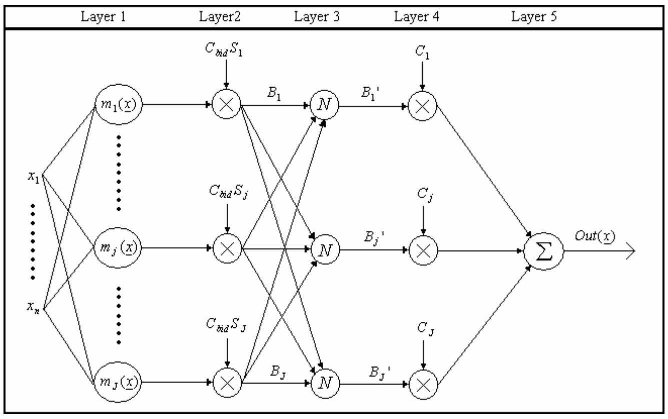


Figure. 1 The Architecture of an ACSNFIS

2.1 The Architecture of an ACSNFIS

An ACSNFIS is a five-layer feedforward network, as shown in Figure. 1. The outputs in each layer are computed as follows:

$$O_{1,j} = m_j(\underline{x}) = \exp\left[\frac{-1}{2}\sum_{i=1}^{n}\left(\frac{(x_i - w_{ji})}{s_{ji}}\right)^2\right] \quad (1)$$

$$O_{2,j} = O_{1,j} \cdot S_j \cdot C_{bid} = m_j(\underline{x}) \cdot S_j \cdot C_{bid} = B_j \quad (2)$$

$$O_{3,j} = B_j{}' = \frac{B_j}{\sum_{i=1}^{J} B_i} \quad (3)$$

$$O_{4,j} = C_j B_j{}' \quad (4)$$

$$O_{5,j} = Out(\underline{x}) = \sum_{j=1}^{J} C_j B_j{}' = \sum_{j=1}^{J} C_j \frac{B_j}{\sum_{i=1}^{J} B_i} = \sum_{j=1}^{J} C_j \frac{C_{bid} S_j m_j(\underline{x})}{\sum_{i=1}^{J} C_{bid} S_i m_i(\underline{x})} \quad (5)$$

where B_j is the bid of the *j*th hidden node, C_{bid} is a constant about the bid coefficient ($0 < C_{bid} \le 1$), C_j is an adaptive parameter which represents the consequent part of the *j*th rule, $O_{i,j}$ represents the output of the *i*th node in the *j*th layer, and S_j is the strength of the *j*th fuzzy rule.

The fuzzy rules in the system then can be represented as

R_j : ***IF*** ($\underline{x}$ *is* HE_j)

THEN *the system output is* C_j (6)

2.2 Learning Mechanisms

In order to learn behavior through trail-and-error interaction with a dynamic environment, the proposed system must explicitly explore its environment and exploit the new rules. The architecture of the neuro-fuzzy system is incrementally constructed during the training procedure. First, the system starts with no hidden node (*i.e.* fuzzy rule). Then new hidden nodes are sequentially injected into the system during the learning procedure. During the learning procedure, we also need to eliminate useless rules so as to keep the number of rules as least as possible. Those fuzzy rules whose strengths are smaller than a pre-specified threshold (*i.e.* $S_j < q_s$) will be eliminated.

In our system, there are several parameters, such as the strength, S_j, and the weights in the hidden node, $\underline{w}_j$ and s_{ji}, which need to be updated during the learning procedure. These parameters are updated according to different learning mechanisms.

2.2.1 Reward Mechanism

The reward mechanism is used to update the strength, S_j. The idea of the reward mechanism is very simple. In our system all fuzzy rules cooperate to the computation of the system's output .If the performance improves, then the strength of each fuzzy rule will be increased according the degree of its contribution. Otherwise, its strength will be decreased. The internal reinforcement signal $r(t+1)$ is used to indicate the performance of the neuro-fuzzy network at time *t*. If the internal reinforcement signal $r(t+1)$ is a positive value, it represents the performance of the current state at time $t+1$ is better than the latest state at time *t*. Otherwise, the current performance is worse than the latest state. In addition to a receipt from the environmental reward, each rule can also receive a receipt from its previous message-sending activity (*i.e.* the contribution to the computation of the action at the previous step). Taking together, we may write an equation governing the deletion or accretion of the strength of *j*th rule as follows:

$$\begin{aligned} S_j(t+1) &= S_j(t) - m_j(\underline{x}(t)) \cdot S_j(t) \cdot C_{bid} + R_j(t) \\ &= S_j(t) - B_j(t) + R_j(t) \quad (7) \\ &= S_j(t) - B_j(t) + (R_j^p(t) + R_j^e(t)) \end{aligned}$$

where

$$R_j^p(t) = B_j{}'(t-1)\sum_{i=1}^{J} B_i(t) \quad (8)$$

$$R_j^e(t) = B_j{}'(t)R(t) = B_j{}'(t) \cdot S_r \cdot r(t+1) \quad (9)$$

where the parameter S_r is a positive value that is determined by the user. The term, $R(t) = S_r \cdot r(t+1)$, represents the environmental reward. Obviously, if the reinforcement signal (*i.e.* $r(t+1)$) is positive, the *j*th rule will receive the reward $R_j^e(t) = B_j{}'(t)R(t)$ from the environment. Otherwise, it has to pay $B_j{}'(t)R(t)$ back as a punishment when $r(t+1)$ is negative. The term $\sum_{i=1}^{J} B_i(t)$ is the total amount of bids at time *t* and $B_j{}'(t-1)$ is the normalized firing strength of the *j*th rule at time *t*-1. Naturally, the *j*th rule can get a receipt, $R_j^p(t) = B_j{}'(t-1)\sum_{i=1}^{J} B_i(t)$ from its previous message-sending activity. Note that we limit the updated strength $S_j(t+1)$ in the range between 0 to $S_{\max}$ which is a pre-specified constant scalar.

2.2.2 Updating Weights

Here we present how we update the weights $\underline{w}_j$ and

$\underline{s}_j$. First, we let P_j represent the *j*th hidden node's weights in the network, which includes the weighting vector $\underline{w}_j$ and the regulating parameter $\underline{s}_j$. Note that the connection weight C_j is fixed during the training procedure.

The intent of computing $Out(\underline{x}(t))$ is to maximize $v(t)$ so that the system ends up in a good state and avoids failure. Thus, the learning task is done by gradient descent, which needs to be maximized the objective function $v(t)$ as a function of P_j. The learning rule is defined as

$$\begin{aligned} P_j(t+1) &= P_j(t) + h(t)\frac{\partial v(t)}{\partial P_j(t)} \\ &= P_j(t) + h(t)\frac{\partial v(t)}{\partial Out(\underline{x}(t))}\frac{\partial Out(\underline{x}(t))}{\partial P_j(t)} \end{aligned} \quad (10)$$

To compute this equation, we need the two derivatives on the right-hand side, which will depend on the state in a general way. Since it is complex to compute the derivative $\partial v(t) / \partial Out(\underline{x}(t))$, Berenji and Protap [27] proposed as a method to make the approximation that $\partial v(t) / \partial Out(\underline{x}(t))$ can be computed by the instantaneous difference ratio

$$\begin{aligned} \frac{\partial v(t)}{\partial Out(\underline{x}(t))} &\approx \frac{dv(t)}{dOut(\underline{x}(t))} \approx \frac{v(t)-v(t-1)}{Out(\underline{x}(t))-Out(\underline{x}(t-1))} \\ &\approx \operatorname{sgn}\left(\frac{v(t)-v(t-1)}{Out(\underline{x}(t))-Out(\underline{x}(t-1))}\right) \end{aligned} \quad (11)$$

We then reword the learning rule as

$$P_j(t+1) = P_j(t) + h(t)\operatorname{sgn}\left(\frac{v(t)-v(t-1)}{Out(\underline{x}(t))-Out(\underline{x}(t-1))}\right)\frac{\partial Out(\underline{x}(t))}{\partial P_j(t)} \quad (12)$$

and

$$h(t) = h_1(1-v(t)^2)+h_2 \quad (13)$$

where $h(t)$ is a function of learning rate, h_1 and h_2 are two learning parameters to control the learning rate.

In fact, we also developed a second method of updating these weights. The second method is similar to the learning vector quantization method [28]. Due to the limitation on the space, we cannot present the second method here.

3. Experimental Results

This navigation system is simulated for a wheeled mobile robot. As shown in Figure. 2, it is equipped with five ultrasonic sensors, six tactile bumpers, and a light sensor. It was assumed that the sizes of the robot and the environment are $0.2m \times 0.2m$ and $6m \times 3m$, respectively. The ultrasonic sensors were assumed to be able to ideally detect the obstacles within their beams and within the range of 2-15m.

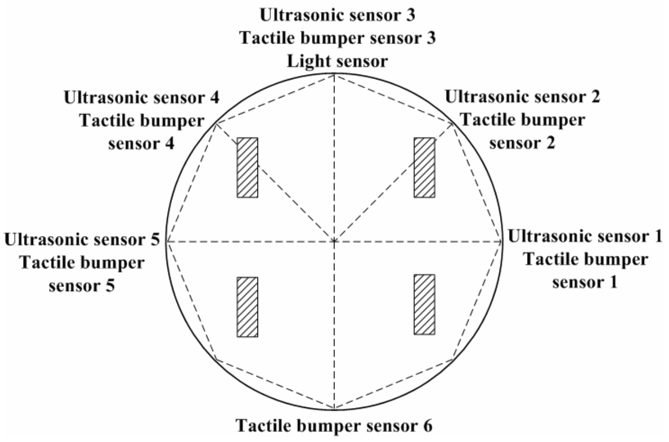


Figure. 2 The experimental wheeled mobile robot.

The task of the robot is to survive (*i.e.* without running out of energy) in an arbitrarily constructed environment. The robot was assumed to consume energy at every time step. In addition, it consumes a lot of energy when it collides with obstacles or walls. To gain energy, the robot must search for a light source to follow and then stop in front of it.

To make the robot to achieve the goal, the evaluation function is chosen as follows:

$$v(t) = \exp\left(w\left(\Delta I(t) - q_I\right)\right) \quad (14)$$

where

$$\Delta I(t) = w_l l(t) - w_s \sum_i s_i(t) - w_h \sum_i h_i(t) - K \quad (15)$$

$$l(t) = \begin{cases} e^{-D_l}\cos(l) & \text{if } \cos(l) > 0 \\ -3 & \text{if } \cos(l) \le 0 \end{cases} \quad (16)$$

$$s_i(t) = \begin{cases} e^{-20} & \text{if } D_{s_i} \ge 15 \\ e^{-(D_{s_i}-2)} & \text{if } D_{s_i} < 15 \end{cases} \quad for\ 1 \le i \le 5 \quad (17)$$

$$h_i = \begin{cases} 1 & \text{if tactile bumper i is touched} \\ 0 & \text{otherwise} \end{cases} \quad for\ 1 \le i \le 6 \quad (18)$$

The functions, $l(t)$, $s(t)$ and $h(t)$, represent the amounts of energy gained when the robot is approaching the light source, consumed when the robot is approaching the obstacle or the walls, and consumed when the robot collides with the obstacle or the walls, respectively. The parameters, *K* and q_I, represent the amount of energy consumed at each time interval and the maximum variation of energy during the time interval, respectively. The parameters, D_l and D_{s_i}, represent the distance between the robot and the light source and the distance between the robot and the obstacle, respectively. The parameters, w, w_l, w_s and w_h are weighting factors.

To complete the training task, we divide the training task into three sub-tasks. The difficulties and complexities of the sub-tasks progressively increase from low-level sub-task to high-level sub-task. These sub-tasks are as follows:

Sub-task 1: One light source and no obstacle

This sub-task is to ask the robot to develop a searching skill such that the robot won't sit still waiting for death and instead it can efficiently find the light source without bumping walls and then stop in front of the light source.

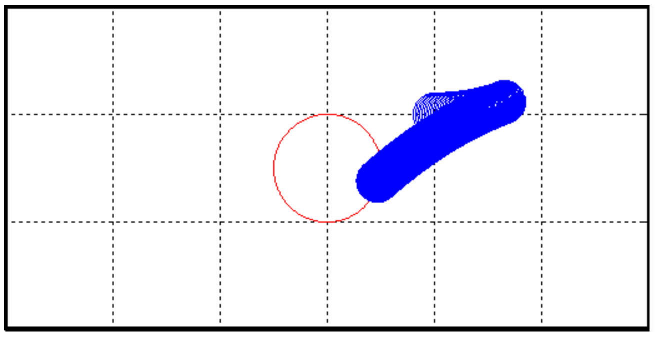
Sub-task 2: One light source and one obstacle

After the robot has developed the basic searching skill, the robot is then deployed in an environment composed of one obstacle, which is placed in the middle of the light source and the robot. The goal of this sub-task is to train the robot to automatically develop the skill of obstacle avoidance.

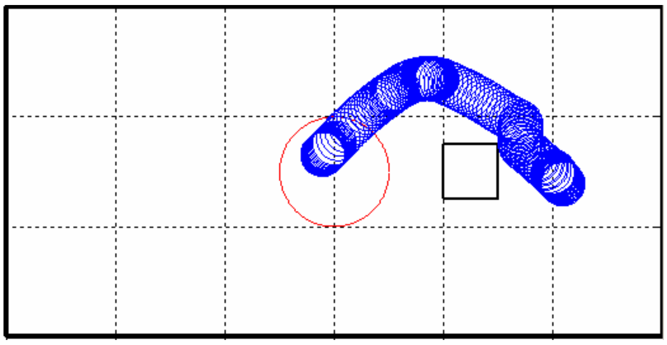
Sub-task 3: One light source and an arbitrarily constructed environment

After the robot has learned the skill of obstacle avoidance, the goal of this sub-task is to ask the robot to develop an efficient autonomous navigation scheme in an arbitrary environment without bumping to any obstacle. Furthermore, it will try its best to search the light source before it runs out of the energy.

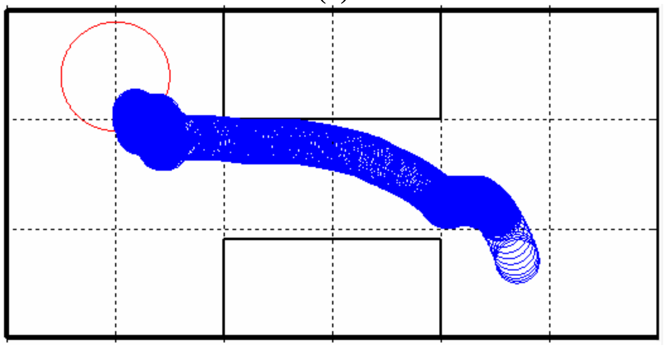
Figure. 3(a) shows the trajectory of the robot in the process of learning the first sub-task. As the learning proceeds in some cases the robot may be trapped. In such cases, a new start position and a heading angle must be given in order to continue the learning process. After several trials, the robot finally learned to search a light source without bumping to the walls. The simulation result shows that the robot can develop the searching skill that enforces it to change its position or rotate itself in order to increase the chance of finding a light source. In the learning procedure, three neurons were generated.



(a)



(b)



(c)

Figure. 3. Trajectories in each sub-task.
(a) Sub-task 1.(b) Sub-task 2. (c) Sub-task 3.

With the developed searching skill, the robot was then deployed in the environment, as shown in Figure. 3(b). As the learning proceeds, whenever the robot bumped to the obstacle or the wall, the robot was positioned in a new start position. After a sequence of trials, the robot finally can keep away from the obstacle and then move around the environment to search the light source before running out of the energy. After the robot has learned the second sub-task, 70 more neurons were generated.

Finally, the robot was deployed in the environment shown in Figure. 3(c). After another sequence of trials, the robot finally can walk through the corridor and then move to the left space to search the light source before running out of the energy.

4. Conclusions

In this paper, we have shown how to apply a reinforcement learning algorithm to design an autonomous robot to navigate efficiently in an unknown environment. Reinforcement learning is used to generate the fuzzy rule for navigation in such a way that the correct actives can be acquired in unknown environment situations.

Acknowledgement

This work is partially supported by the National Science Council, Taiwan, R.O.C., under the NSC 91-2213-E-008-020 and by the MOE Program for Promoting Academic Excellent of Universities under the grant number EX-91-E-FA06-4-4.

References

[1] T. Lozano-Perez and M. A. Wesley, "An algorithm for planning collision-free paths among the polyhedral obstacles," Communications of the ACM, vol. 22, no. 10, pp. 560-570, Oct.1979

[2] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," Int. J. of Robotics Research, vol. 5, no.1, pp. 90-98, Spring 1986.

[3] O. Takahashi and R. J. Schilling, "Motion planning in a plane using generalized Voronoi diagrams," IEEE Trans. on Robotics Automat., vol. 5, no. 2, pp. 143-150,1989.

[4] R. A. Brooks, "A robust layered control system for a mobile robot," IEEE Trans. on Robotics Automat., vol. RA-2, no. 1, pp. 1-23, Mar. 1986.

[5] J. Borenstein and Y. koren, "Real-time obstacle avoidance for fast mobile robot," IEEE Trans. on Syst. Man Cyber., vol. 19, no. 5, pp. 1179-1187, Sept./Oct. 1989.

[6] J. Borenstein and Y. Koren, "Potential field methods and their inherent limitations for mobile robot navigation," in Proc. IEEE Int. Conf. Robotics and Automation (Sacramento, CA, Apr. 9-11, 1991), pp. 818-823, 1991.

[7] J. H. Connell, "Minimalist Mobile Robotics: A colony-Style Architecture for an artificial creature," San Diego, CA: Academic Press, 1990.

[8] M. J. Schoppers, "Universal plans for reactive robots in unpredictable environments, " in the 10th International Joint Conference on Artificial Intelligence, pp. 1039-1046, 1987.

[9] J. H. Connell, "A hybrid architecture applied to robot navigation," in IEEE International Conference on Robotics and Automation, pp. 2719-2724, 1992.

[10] D. P. Miller and M. G. Slack, "Global symbolic maps from local navigation," in the 9th National Conference on Artificial Intelligence, pp. 750-755, 1991.

[11] T. M. Mitchell and S. B. Thrun, "Explanation-based neural networks learning for robot control," In C. L. Giles, S. J. Hanson and J. D. Cowan (eds.), Advances in Neural information Processing Systems 5, pp. 287-294. San Mateo, CA: Morgan Kaufmann.

[12] J. del R. Millan and C. Torras, "A reinforcement connectionist approach to robot path finding in non-maze-like environments," Machine Learning, 8, pp. 1992.

[13] G. A. Bekey and R. Tomovic ,"Reflex control of robot actions," in IEEE Int. Conf. on Robotics and Automation, pp. 240-247, 1986.

[14] R. C. Arkin, "Integrating behavioral, perctptual, and world knowledge in reactive navigation," Robotics and Autonomics Systems, vol.6, pp. 105-122, 1990.

[15] J. del R. Millan, "Learning Efficient Reactive Behavioral Sequences from Basic Reflexes in a Goal-Directed Autonomous Robot," From Animals to Animats: Third International Conference on Simulation of Adaptive Behavior, Brighton, UK, August 8-12, 1994.

[16] A. H. Fagg, D. Lotspeich and G. A. Bekey, "A reinforcement-learning approach to reactive control policy design for autonomous robots," IEEE International Conference on Robotics and Automation, vol. 1, pp. 39-44, 1994.

[17] H. R. Beom and H. S. Cho, "A Sensor-based navigation for a Mobile Robot Using Fuzzy Logic and Reinforcement Learning," IEEE Transactions on Systems, Man, and Cybernetics, vol. 25, no. 3, March 1995.

[18] L. P. Kaelbling, M. L. Littman and A. W. Moore, "Reinforcement learning: a survey," Journal of Artificial Intelligence Research, vol. 4, pp. 237-285, 1996.

[19] J. Schmidhuber ,"A general method for multi-agent learning and incremental self-improvement in unrestricted environments," In Yao, X. (Ed.), Evolutionary Computation: Theory and Applications. Scientfic Publ. Co., Singapore, 1996.

[20] A. G. Barto, R. S. Sutton and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," IEEE Trans. on System, Man, and Cybernetics, vol. 13, no. 5, pp. 834-846, 1983.

[21] R. S. Sutton, "Learning to predict by the methods of temporal differences," Machine Learning, vol. 3, pp. 9-44, 1988.

[22] C. J. C. H. Watkins, *Learning form Delayed Rewards*. Ph.D. thesis, King' s College, Cambridge, UK, 1989.

[23] C. J. C. H. Watkins and P. Dayan, "Q-Learning," Machine Learning, vol. 8, no. 3, pp. 279-292. 1992.

[24] M. C. Su, C. H. Chou, and E. Lai, "A Self-Generating Neuro-Fuzzy System Through Reinforcements," 2nd WSEAS Int. Conf. on Scientific Computation and Soft Computing, pp. 332-337, Crete, Greece, 2002.

[25] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.

[26] D. Goldberg, *Genetic Algorithm in Search, Optimization, and machine Learning*, Addison-Welsley, MA, 1989.

[27] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controls through reinforcements," IEEE Trans. on Neural Networks, vol. 3, no. 5, pp. 724-740, Sep. 1992.

[28] T. Kohonen, "Improved versions of learning vector quantization," in International Joint Conference on Neural Networks, pp. 545-550, San Diego, 1990.