Received: November 17, 2019. Revised: April 26, 2020. Accepted: May 21, 2020. Published: June 2, 2020.

A REINFORCEMENT LEARNING ALGORITHM WITH EVOLVING FUZZY NEURAL NETWORKS

Hitesh Shah
Professor, Department of Electronics & Communication
G H Patel College of Engineering & Technology
Vallabh Vidyanagar, Gujarat (INDIA)
iitd.hitesh@gmail.com

M.Gopal
Director, School of Engineering
Shiv Nadar University
Noida, Uttar Pradesh (INDIA)
mgopal@snu.edu.in

***Abstract*—The synergy of the two paradigms, neural network and fuzzy inference system, has given rise to rapidly emerging filed, neuro-fuzzy systems. Evolving neuro-fuzzy systems are intended to use online learning to extract knowledge from data and perform a high-level adaptation of the network structure. We explore the potential of evolving neuro-fuzzy systems in reinforcement learning (RL) applications. In this paper, a novel on-line sequential learning evolving neuro-fuzzy model design for RL is proposed. We develop a dynamic evolving fuzzy neural network (DENFIS) function approximation approach to RL systems. Potential of this approach is demonstrated through a case study—two-link robot manipulator. Simulation results have demonstrated that the proposed approach performs well in reinforcement learning problems.**

Keywords*— *Reinforcement learning, Neuro-fuzzy system

I. INTRODUCTION

Reinforcement learning (RL) paradigm is a computationally simple and direct approach to the adaptive optimal control of nonlinear systems [1]. In RL, the learning agent (controller) interacts with an initially unknown environment (system) by measuring states and applying actions according to its policy to maximize its cumulative rewards. Thus, RL provides a general methodology to solve complex uncertain sequential decision problems, which are very challenging in many real-world applications.

Often the environment of RL is typically formulated as a Markov Decision Process (MDP), consisting of a set of all states $\mathcal{S}$, a set of all possible actions $\mathcal{A}$, a state transition probability distribution $P:\mathcal{S}\times\mathcal{A}\times\mathcal{S}\to[0,1]$, and a reward function $R:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$. When all components of the MDP are known, an optimal policy can be determined, e.g., using dynamic programming.

There has been a great deal of progress in the machine learning community on value-function based reinforcement learning methods [2]. In value-function based reinforcement learning, rather than learning a direct mapping from states to actions, the agent learns an intermediate data structure known as a value function that maps states (or state-action pairs) to the expected long term reward. Value-function based learning methods are appealing because the value function has well-defined semantics that enable a straightforward representation of the optimal policy, and theoretical results guaranteeing the convergence of certain methods [3].

Q-learning is a common model-free value function strategy for RL [4]. *Q*-learning system maps every state-action pair to a real number, the *Q*-value, which tells how optimal that action is in that state. For small domains, this mapping can be represented explicitly by table of *Q*-values. For large domains, this approach is simply infeasible. If, one deals with large discrete or continuous state and action spaces, it is inevitable to resort to function approximation, for two reasons: first to overcome the storage problem (*curse of dimensionality*), second to achieve data efficiency (i.e., requiring only a few observations to derive a near-optimal policy) by generalizing to unobserved states-action pairs. There is a large literature on RL algorithms using various value-function estimation techniques.

Functionally, a fuzzy system or a neural network can be described as a function approximator. Theoretical investigations have revealed that neural networks and fuzzy inference systems are universal approximators [5, 6]. Neural networks are used to generalize the value function pertaining to specific situations. However, these works still assume discrete actions and cannot handle continuous-valued actions. In realistic applications, it is imperative to deal with continuous states and actions. Fuzzy Inference System (FIS) can be used to facilitate generalization in the state space and to generate continuous actions, in particular in conjunction with *Q*-learning widely known as fuzzy *Q*-learning (FQL). Glorennec [7] and the extension proposed by Jouffe [8] provided a fundamental contribution in the definition of FQL, this is the basis for many of the existing implementations. In FQL, the consequent parts of a FIS are selected by *Q*-learning. However, structure and premise parameters are still determined by a *priori* knowledge. To circumvent this problem, Er and Deng [9] proposed a dynamic fuzzy *Q*-learning (DFQL) approach to construct self-tuning FIS based on reinforcement signals and deal with continuous state and action spaces.

Recently, the synergy of the two paradigms, neural network and fuzzy inference system, has given rise to rapidly emerging field, neuro-fuzzy systems. The neuro-fuzzy term means a type of system characterized for a similar structure of a fuzzy controller, where the fuzzy sets and rules are adjusted using neural network tuning techniques in an iterative way with the input-output data vectors. A Neuro-fuzzy system is widely termed as fuzzy neural network (FuNN) [10, 11] in the literature. Fuzzy neural network systems are intended to capture the advantages of both fuzzy logic (approximate reasoning) and neural networks (learning) i.e. acquire fuzzy rules based on the learning ability of neural networks [12].

Many researchers have developed such a neuro-fuzzy system for solving real-world problem effectively. The evolving fuzzy neural network (EFuNN) was proposed by Kasabov in [13], one of the hybrid neuro-fuzzy architecture. Dynamic evolving neural fuzzy inference system (dmEFuNN/DENFIS) [14] is a modified version of the EFuNN with the idea that, depending on the position of the input vector in the input space, a FIS for calculating the output is formed dynamically bases on *m* fuzzy rules that had been created during the past learning process. The application of these networks has been in the areas of classification and regression using supervised learning methods. DENFIS when used especially for online learning adaptive systems [14][15].

Use of neuro-fuzzy systems for value function approximation for RL setup has not yet been explored. In this paper, we explore the potential of an alternative dynamic evolving fuzzy-neural network (dmEFuNN) for reinforcement learning algorithms. We compare the learning performances of dmEFuNN and Dynamic FNN (here, dynamic fuzzy *Q*-learning) in reinforcement learning framework, using simulation experiment on two-link robot manipulator tracking control problem. Further, we examine the robustness performance of the proposed approach for handling the uncertainty in terms of parameter variations and external disturbances.

The paper is organized as follows. Section II presents the theoretical background of fuzzy inference system with reinforcement learning approach and recent trends of neuro-fuzzy systems. Section III proposes architecture and learning framework of dmEFuNN function approximator for RL systems. Section IV exhibits the empirical performance based on the experimental results of the system-two-link robot manipulator simulations. Section V, conclusions are drawn in the last section.

II. Theoretical Background

A neuro-fuzzy system is widely termed as fuzzy neural network (FuNN) [10, 11] in the literature. Fuzzy neural network systems are intended to capture the advantages of both learning and computational power of neural network and the high-level human-like thinking and reasoning of fuzzy system. Evolving fuzzy neural network and dynamic evolving fuzzy neural network are the hybrid neuro-fuzzy architecture.

A. Evolving Fuzzy Neural Network (FEuNN)

EFuNN implements five layers Mamdani type FIS. The first layer passes crisp input variable to the second layer that calculates the degrees of compatibility in relation to the predefined membership functions. The third layer is the rule layer and each node in this layer represents either an existing rule, or a rule anticipated after training. The rule nodes represent prototypes of input-output data as an association of hyperspheres from the fuzzy input and the fuzzy output spaces. Each rule node is defined by two vectors of connection weights, which are adjusted through a hybrid learning technique. The fourth layer represents a fuzzy quantization of each output variable and calculates the degree to which output membership functions are matched the input data. The fifth layer carries out defuzzification and calculates the crisp value for the output variable. In EFuNN, all the rule nodes are created during the learning phase. We used EFuNN as an function approximator in RL framework, where input to the EFuNN is the state or state-action pair resulted in to the output *Q*-value.

B. Dynamic Evolving Fuzzy Neural Network (DENFIS)

The dynamic evolving neural-fuzzy inference system, DENFIS (also known as dmEFuNN), uses the first-order Takagi-Sugeno type of inference engine [14]. DENFIS is similar to EFuNN in some principles. It inherits and develops EFuNN's dynamic features that make DENFIS suitable for on-line adaptive systems. The DENFIS model uses a local generalization. Principally structure of EFuNN and DENFIS is somewhat similar. Dynamic feature of EFuNN developed with the idea that, depending on the position of the input vector in the input space, a FIS for calculating the output value is formed dynamically bases on *m* fuzzy rules that has been created during the past learning process. Evolving clustering method (ECM) [15] is used for fuzzy rules creation and updation within the input space partitioning.

Although DENFIS meets the requirements of online learning to form adaptive intelligent systems to a great extent, however there is still scope of advancement. Our objective is to use DENFIS as a function approximator in reinforcement learning framework.

III. Approximation of Value Function Using DENFIS

A novel value function approximator for online sequential learning on continuous state-action domain based on DENFIS is proposed in this paper. Fig. 1shows architectural view of the DENFIS function approximation approach to RL system.

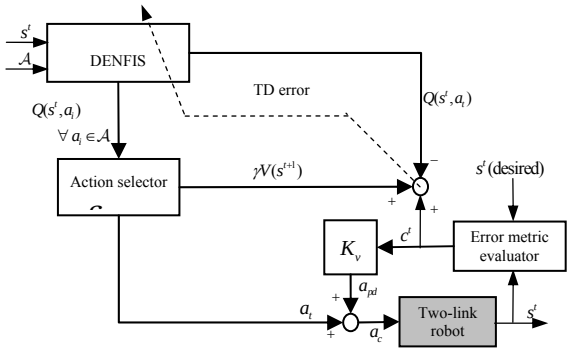

Fig. 1 DENFIS controller architecture

The state-action pair (s^t, a_t); where $s^t = \{s_1^t, s_2^t, \ldots, s_n^t\} \in \mathcal{S}$ is the current system state and a_t is the each possible discrete control action in action set $\mathcal{A} = \{a_i\}; i = 1, \ldots, m$, is the input of DENFIS model and the estimated *Q*-value corresponding to (s^t, a_t) is the output of the network.

$$Q(s^t, a_t) = y^t = f_1(x^t) = f_1(x_1, x_2, \ldots, x_q) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_q x_q \quad (1)$$

where x^t is the input vector ($x^t = [x_1, x_2, \ldots, x_q]^T = (s^t, a_t)$) of the DENFIS model and output y^t corresponds to estimated Q-value associated with each state-action in rule $R_i; i = 1, 2, ..., m$.

Training samples are obtained online as the interaction between the learning agent (controller) and its environment (plant). The online learning process of DENFIS involves the creation of new fuzzy rules, and existing fuzzy rules can be updated incrementally. In addition, evolving clustering method (ECM) is used to partition the input sample space to determine the fuzzy sets in the antecedent part, *i.e.*, ECM is used to determine cluster centers and membership functions of the antecedent part, and wRLS with forgetting factor determine the parameters of the consequent part of a fuzzy rule.

The agent's action is selected based on the outputs of DENFIS. In specific, control actions are selected using an exploration/exploitation policy [4] in order to explore the set of possible actions and acquire experience through the online RL signals. We use a pseudo-stochastic exploration ε-greedy as in [4]. In ε-greedy exploration, we gradually reduce the exploration (determined by the ε parameter) according to some schedule; we have reduced ε to its 90 percent value after every 10 iterations. The lower limit of parameter ε has been kept fixed at 0.002 (to maintain exploration).

It is an online learning algorithm that learns an approximate state-action value function $Q(s^t, a_t)$ that converges to the optimal function Q^* (commonly called Q-value). Online version is given by

$$Q(s^t, a_t) \leftarrow Q(s^t, a_t) + \eta[c^t + \gamma V(s^{t+1}) - Q(s^t, a_t)] \qquad (2)$$

where $s^t \xrightarrow{c^t} s^{t+1}$ is the state transition under the control action $a_t \in \mathcal{A}(s^t)$ (in fact $a_c = a_t(s^t) + a_{pd}(s^t)$; where $a_{pd}(s^t)$ is the action generated by inner PD loop), c^t is the cost incurred by the controller, $\eta \in (0,1]$ is the learning rate parameter that can be used to optimize the speed of learning, and $\gamma \in (0,1]$ is the discount factor that controls the trade-off between immediate and future costs.

A. Learning Process in DENFIS online model

The first-order Tagaki-Sugeno fuzzy rules [58] are employed in DENFIS online model. The linear functions in the consequence parts are created and updated by linear least-square estimator (LSE) [15] on the learning data. The linear function for a learning data set of p data pairs, $\{([x_{i1}, x_{i2}, \ldots, x_{iq}], y_i), i = 1, 2, \ldots, p\}$, can be expressed as

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_q x_q \qquad (3)$$

The least-square estimator (LSE) of $\beta = \begin{bmatrix} \beta_0 & \beta_1 & \beta_2 & \cdots & \beta_q \end{bmatrix}^T$ is calculated as the coefficients of $b = \begin{bmatrix} b_0 & b_1 & b_2 & \cdots & b_q \end{bmatrix}^T$, by applying the following weighted least-square estimator formula:

$$b = (\mathrm{A}^T \mathrm{W} \mathrm{A})^{-1} \mathrm{A}^T \mathrm{W} y \qquad (4)$$

where

$$\mathrm{A} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1q} \\ 1 & x_{21} & x_{22} & \cdots & x_{2q} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{p1} & x_{p2} & \cdots & x_{pq} \end{bmatrix}; \; y = \begin{bmatrix} y_1 & y_2 & \cdots & y_p \end{bmatrix}^T \text{ and } \mathrm{W} = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & w_p \end{bmatrix}$$

Here W is the weight matrix and its elements, w_{ij} , are defined by $1 - d_j$ (d_j is the distance between the j^{th} sample and the corresponding cluster center), $j = 1, 2, \ldots, p$. We can rewrite equation (4) with the use of recursive LSE formula [104] as follows:

$$\begin{aligned} \mathrm{P} &= (\mathrm{A}^T \mathrm{W} \mathrm{A})^{-1} \\ b &= \mathrm{P} \mathrm{A}^T \mathrm{W} y \end{aligned} \qquad (5)$$

In the DENFIS online model, Kasabov and Song [99] used a weighted recursive LSE with a forgetting factor defined as follows. Let the k^{th} row vector of matrix A is denoted as a_k^T and the k^{th} element of y is denoted as y_k . Then b can be calculated iteratively as follows:

$$\begin{aligned} b_{k+1} &= b_k + w_{k+1} \mathrm{P}_{k+1} \mathrm{a}_{k+1} (y_{k+1} - \mathrm{a}_{k+1}^T b_k) \\ \mathrm{P}_{k+1} &= \frac{1}{\lambda} \left(\mathrm{P}_k - \frac{w_{k+1} \mathrm{P}_k \mathrm{a}_{k+1} \mathrm{a}_{k+1}^T \mathrm{P}_k}{\lambda + \mathrm{a}_{k+1}^T \mathrm{P}_k \mathrm{a}_{k+1}} \right) \end{aligned} \qquad (6)$$

where $k = n, n+1, \ldots p-1$; w_{k+1} is the weight of $k+1$-th sample defined by $1 - d_{k+1}$ (d_{k+1} is the distance between the $k+1$-th sample and the corresponding cluster centre); and $\lambda \in (0.8, 1)$ is forgetting factor. The initial values of P_n and b_n can be calculated directly from (5) with the use of first n data pairs from the learning data set.

In online DENFIS model, the rules are created and updated at the same time with the input space partitioning using online ECM, and equations (4) and (6).

IV. Simulation Expriments

To demonstrate the usefulness of dynamic evolving fuzzy neural network function approximator in reinforcement learning framework, we conducted experiments using the well-known two-link robot manipulator tracking control problem.

In implementation, the DENFIS has as input the state-action pair and as output, the Q-value corresponding to the state-action pair. In particular, the DENFIS network begins with zero cluster. We first obtained a group of fuzzy rules using an DENFIS off-line learning model, with the use of training samples available from well defined reinforcement fuzzy systems (here we take training samples from dynamic fuzzy Q-learning controller). Then with agent-environment interaction, the training samples available and the DENFIS model build-up an online mode based on dynamic inference, *i.e.*, clustering and reformulation of the rules are performed whenever a new training example is presented to the network. The DENFIS off-line learning model when used as an

initialization, improves the generalization (e.g., improves the learning efficiency).

For simplicity, the controller uses two DENFIS models as function approximators; one each for the two-links. DENFIS is one module of ECOS toolbox working in the MATLAB numeric computing environment. The distance threshold *Dthr* is set to 0.08 and default value of the number of rules in dynamic fuzzy inference system is set to 3 for constructing DENFIS.

A. Simulation Results and Discussion

Simulations were carried out to study the learning performance, and robustness against uncertainties, for DENFIS learning approach on two-link robot manipulator control problem. To analyze the DENFIS algorithm for computational cost, accuracy, and robustness, we compare the proposed approach with dynamic fuzzy reinforcement learning approach. MATLAB 7.10 (R2010a) has been used as simulation tool.

Learning performance study

The physical system has been simulated for a single run of 10 sec using fourth-order Runge-Kutta method, with fixed time step of 10 msec. Fig. 2 and Fig. 3 show the output tracking error (both the links), for both the controllers ─ DFQC and DENFISQC. Table 1 tabulates the mean square error, absolute maximum error ($\max|e(t)|$), and absolute maximum control effort ($\max|\tau|$) under nominal operating conditions.

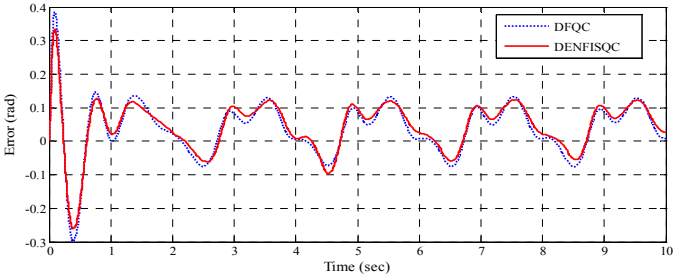

Fig. 2 Standard two-link controller comparison: output tracking errors (link 1)

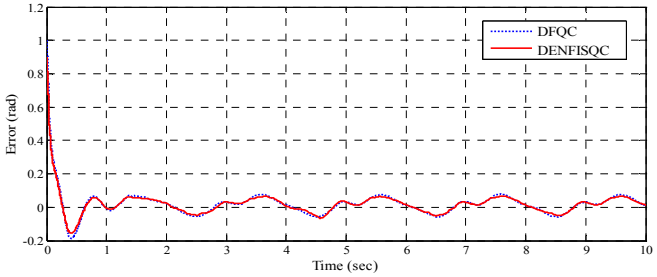

Fig. 3 Standard two-link controller comparison: output tracking errors (link 2)

Table 1 Comparison of controllers: learning performance study

Controller	MSE (rad)		$\max\|e(t)\|$ (rad)		$\max\|\tau\|$ (Nm)		Training Time (sec)
	Link 1	Link 2	Link 1	Link 2	Link 1	Link 2	------
DFQC	0.0083	0.0066	0.1336	0.0788	89.1723	35.7915	9.8383
DENFISQC	0.0077	0.0054	0.1242	0.0676	84.1694	33.0757	72.2333

From the results (Figs. 2−3 and Table 1), we observe that training time for DENFISQC is higher than DFQC. DENFISQC outperforms DFQC, in terms of lower tracking errors and the low value of absolute error and control effort for both the links

Robustness study

In the following, we compare the performance of DFQ and DENFISQC under uncertainties. For this study, we trained the controller for 20 episodes, and then evaluated the performance for two cases:

Effect of payload variations **:** The end-effector mass is varied with time, which corresponds to the robotic arm picking up and releasing payloads having different masses. Fig. 4 and Fig. 5 show the output tracking errors for link 1 and link 2, respectively, and Table 2 tabulates the mean square error, absolute maximum error and absolute maximum control effort at payload variations with time.

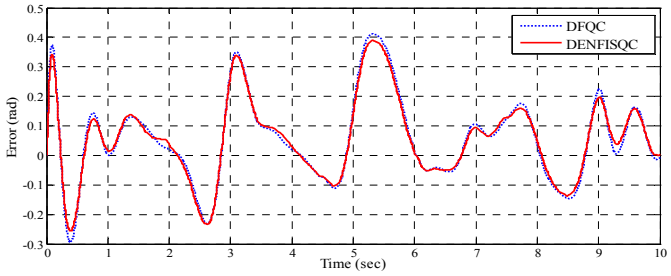

Fig. 4 Effect of payload variation comparison: output tracking errors (link 1)

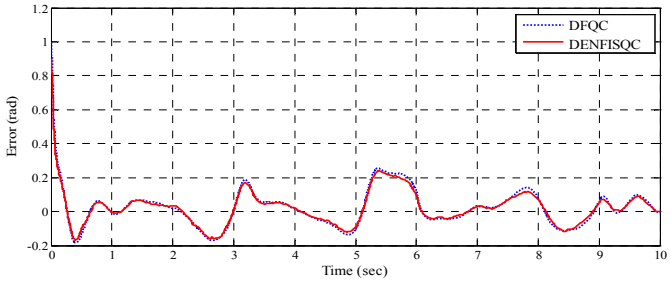

Fig. 5 Effect of payload variation comparison: output tracking errors (link 2)

Table 2 Comparison of controllers: effect of payload variations

Controller	MSE (rad)		max |e(t)| (rad)		max | τ | (Nm)	
	Link 1	Link 2	Link 1	Link 2	Link 1	Link 2
DFQC	0.0237	0.0126	0.4112	0.9052	267.9725	399.1648
DENFISQC	0.0215	0.0103	0.3902	0.8157	263.9985	379.0526

Effects of external disturbances**:** A torque disturbance τ_{dis} with a sinusoidal variation of frequency 2π rad/sec, was added with time to the model. The magnitude of torque disturbance is expressed as a percentage of control effort.Fig. 6 and Fig. 7 show the output tracking errors for link 1 and link 2, respectively, and Table 3 tabulates the mean square error, absolute maximum error ($\max|e(t)|$), and absolute maximum control effort ($\max|\tau|$) for torque disturbances added with time to the model variation.

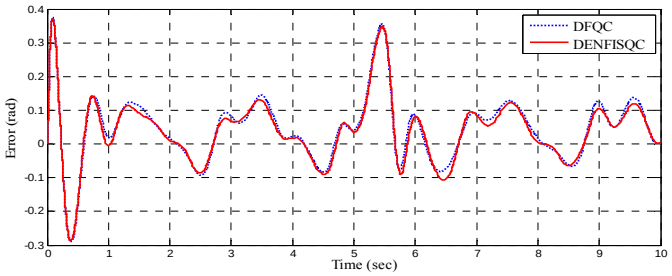

Fig. 6 Effect of external disturbances comparison: output tracking errors (link 1)

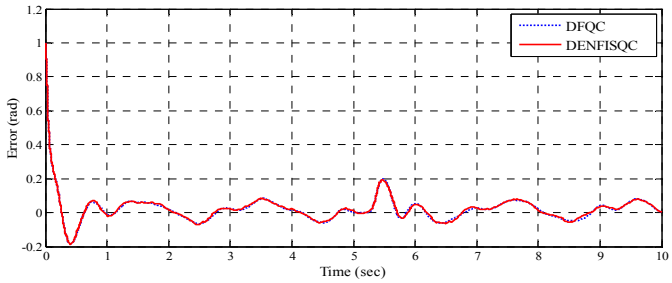

Fig. 7 Effect of external disturbances comparison: output tracking errors (link 2)

Table 3 Comparison of controllers: effect of external disturbances

Controller	MSE (rad)		$\max \lvert e(t) \rvert$ (rad)		$\max \lvert \tau \rvert$ (Nm)	
	Link 1	Link 2	Link 1	Link 2	Link 1	Link 2
DFQC	0.0115	0.0063	0.3748	0.9054	259.2883	399.0790
DENFISQC	0.0106	0.0063	0.3711	0.9068	260.9928	399.5970

Simulation results (Figs 4−7, Table 2 and Table 3) show comparable robustness property for DENFISQ-learning based controller and Dynamic fuzzy *Q*-learning based controller.

V. Concluions

We have explored the potential of dynamic evolving fuzzy-neural network (DENFIS) for reinforcement learning algorithms. DENFIS is a sequential learning architecture and has ability to grow and prune to ensure a parsimonious structure that is well suited for real-time control applications.

From the simulation results, it is obvious that training time in DENFIS based RL system is larger compared to the dynamic fuzzy *Q*-learning based RL system. This feature is achieved without any loss of performance.

References

[1] R. S. Sutton, A. G. Barto, and R. J. Williams, "Reinforcement learning is direct adaptive optimal control," *IEEE Control Syst. Mag.*, vol. 12, no. 2, pp. 19−22, 1992.

[2] J. A. Boyan, and A. W. Moore, "Generalization in reinforcement learning: Safely approximating the value function," *Advances in Neural Information Proc. Sys.*, pp. 369–376., 1995.

[3] B. Ratitch, *On characteristics of Markov decision processes and reinforcement learning in large domains*, PhD thesis, Montréal: McGill University, School of Computer Science, 2004.

[4] R. S. Sutton, and A. G. Barto, *Reinforcement Learning: An Introduction* (*adaptive computation and machine learning*), Cambridge: MIT Press, 1998.

[5] K. Hornic, M. Stinchcombe, and H. White, "Multilayer feed forward networks are universal approximators," *Neural Networks*, vol. 2, pp.359–366, 1989.

[6] L. Wang, "Fuzzy systems are universal approximators," *in Proc. Int. Conf. Fuzzy System*, 1992.

[7] P. Y. Glorennec, L. Jouffe, "Fuzzy *Q*-learning," *Proc. IEEE Int. Conf. Fuzzy Systems*; vol. 2, pp. 659–662, 1997.

[8] L. Jouffe, "Fuzzy inference system learning by reinforcement methods," *IEEE Trans. System, Man, and Cybernetics*, Part C, vol. 28, no. 3, pp. 338–355, 1998.

[9] M. J. Er, and C. Deng, "Online tuning of fuzzy inference systems using dynamic fuzzy *Q*-learning," *IEEE Trans. on Systems, Man, and Cybernetics*, Part B, vol. 34, no. 3, pp. 1478–1489, 2004.

[10] N. Kasabov, Foundation of Neural networks, Fuzzy systems and Knowledge engineering, The MIT Press, CA, MA, 1996.

[11] J. Vieira, F.M Dias, and A. Mota, "Neuro-fuzzy systems: A survey," WSEAS *Trans on Systems*, vol. 3, no. 2, April 2004.

[12] D. A. Linkes, and H. O. Nyongesa, "Learning systems in intelligent control: On appraisal of fuzzy, neural and genetic algorithm control applications," In *Proc. Inst. Elect. Eng. Control Theory Applications*, vol. 143, pp. 367–386, 1996.

[13] N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning," *IEEE Trans. Syst., Man, Cybern.*, Part B, vol. 13, no. 6, pp. 902–918, Dec. 2001.

[14] N. Kasabov, and Q. Song, "DENFIS: Dynamic evolving neuro-fuzzy inference system and its application for time-series prediction," *IEEE Trans. Fuzzy Sys.*, vol. 10, no. 2, pp. 144–154, April 2002.

[15] M J Watts, "A decade of Kasabov's evolving connectionist systems: A review," *IEEE Trans. Systems, Man, and Cybernetics*-Part C: *Applications and Reviews*, vol. 39, no. 3, pp. 253–269, May 2009.