Efficient Implementation of Dynamic Fuzzy Q-Learning

Chang Deng; Meng Joo Er

School of Electrical and Electronic Engineering
Nanyang Technological University
Email: dchang@pmail.ntu.edu.sg; emjer@ntu.edu.sg

Abstract

This paper presents a Dynamic Fuzzy Q-Learning (DFQL) method that is capable of tuning the Fuzzy Inference Systems (FIS) online. Online self-organizing learning is developed so that structure and parameters identification are accomplished automatically and simultaneously. Self-organizing fuzzy inference is introduced to calculate actions and Q-functions so as to enable us to deal with continuous-valued states and actions. We provide the conditions of the convergence of the algorithm. Furthermore, the learning methods based on bias component and eligibility traces for rapid reinforcement learning are discussed.

1. Introduction

Fuzzy logic is a mathematical approach to emulate the human way of thinking and learning. The conventional way of designing Fuzzy Inference Systems (FIS) has been a subjective approach. If the fuzzy system somehow possesses learning abilities, an enormous amount of human efforts would be saved from tuning the system. In this paper, we consider to design an efficient FIS based only on reinforcement signals. One of the most important Reinforcement Learning (RL) method is Q-learning, however, it is difficult to deal with continuous states and actions. Some authors have also extended the Q-Learning to deal with continuous situations spaces by means of function approximation [6]. However, these works still assume discrete actions. In [3], a Q-Learning method which works in continuous domains employing an incremental topology preserving map to partition the input space, is used. Unfortunately, it is hard to quantify or analyze the performance of neural-network-based techniques. In [2], the approach named Fuzzy Q-Learning (FQL), whereby fuzzy logic is used to introduce generalization in the state space and to generate continuous actions, has been developed. However, only the conclusion part of FIS is tuned online. In this paper, a new reinforcement learning method termed Dynamic Fuzzy Q-Learning (DFQL) is proposed. It is an automatic method capable of self-tuning FIS based only on reinforcement signals. Continuous states are handled and continuous actions are generated by fuzzy reasoning. Prior knowledge can be embedded into the fuzzy rules, which can reduce the training time significantly. The DFQL is an efficient learning method whereby not only the conclusion part of a FIS can be constructed online, but also the structure of a FIS can be generated automatically.

The organization of this paper is as follows: Section 2 introduces the architecture of DFQL. Then, the learning algorithm is presented in Section 3. Section 4 provides the convergence theorem. And Section 5 discusses some extensions for rapid learning based on bias component and eligibility traces.

2. Architecture of DFQL

Dynamic Fuzzy Q-Learning is an extension of the original Q-Learning method into a fuzzy environment. At each time step $t$, the learner observes the current state, $S_t$ and selects an action, $U_t$ from the set of possible actions corresponding to that state, $A(S_t)$, according to the state-action pair value function $Q(S,A)$. One time step later, in part as a consequence of its action, the learner receives a numerical reward, $r_{t+1}$, and finds itself in a new state, $S_{t+1}$. State-space coding is realized by the input variable fuzzy sets.

We describe a FIS based on the Takagi-Sugeno method. Let $r$ be the number of input variables. Each input variable $x_i (i = 1, 2, \ldots, r)$ has $n$ membership functions in the form of Gaussian functions given by

$$
\mu_{ij}(x_i) = \exp \left[ - \frac{(x_i - c_{ij})^2}{\sigma_{ij}^2} \right] \quad i = 1, 2, \ldots, r, \quad j = 1, 2, \ldots, n
$$

where $\mu_{ij}$ is the $j$th membership function of $x_i$, $c_{ij}$ and $\sigma_{ij}$ are the center and width of the $j$th Gaussian membership function of $x_i$, respectively. If the T-norm operator used to compute each rule’s firing strength is multiplication, the output of the $j$th rule $R_j (j = 1, 2, \ldots, n)$ in layer 3 is

$$
\Phi_j(x_1, x_2, \ldots, x_r) = \exp \left[ - \frac{\sum_{i=1}^{r} (x_i - c_{ij})^2}{\sigma_{ij}^2} \right] \quad j = 1, 2, \ldots, n
$$

Normalization takes place in layer 4

$$
\alpha_i = \frac{\Phi_j}{\sum_{j=1}^{n} \Phi_j} \quad i = 1, 2, \ldots, n
$$
If defuzzification is performed in layer 5 using the center-of-gravity method, the output variable as the weighted summation of incoming signals, is given by

$$y = \sum_{j=1}^{n} \alpha_j o_j$$  \hspace{1cm} (4)

where $y$ is the value of an output variable and $o_j$ is the consequent parameter of the $j$th rule which is defined as a singleton.

The firing strength of each rule shown in Eq. (2) can be regarded as a function of regularized Mahalanobis distance (M-distance), i.e.

$$\Phi_j = \exp(-md^2(j))$$

where

$$md(j) = \sqrt{(X-C_j)^T \sum_j^{-1} (X-C_j)}$$  \hspace{1cm} (5)

is the M-distance.

In the DFQL approach, each rule $R_i$ has M possible discrete actions $A'$ and it memorizes the parameter vector $q'$ associated with each of these actions. These Q-values are then used to select actions so as to maximize the discounted sum of reward obtained while achieving the task. We build the FIS with competing actions for each rule $R_i$ as follows:

- If $X$ is $S_i$, then $A'(1)$ with $q'(1)$
- or $A'(M)$ with $q'(M)$

The continuous action performed by the learner for a particular state is the weighted sum of the actions elected in the fired rules that describe this state, whose weights are normalized firing strengths vector, $\alpha$ of rules. Then, the TD method updates the Q-values of the elected actions according to their contributions.

### 3. DFQL Algorithm

#### 3.1 Generation of Continuous Actions

The generation of continuous actions depends upon a discrete number of actions of every fuzzy rule and the firing strengths vector of fuzzy rules. In order to explore the set of possible actions and acquire experience through the reinforcement signals, the actions are selected using an exploration-exploitation strategy. It should be clear in order for optimistic policy iteration to perform well, all potentially important parts of the state space and all potential actions should be explored. The convergence theorem requires that all fuzzy state-action pairs be tried infinitely often. For this reason, several methods have been suggested that occasionally depart from the greedy policy. See [2] for more information about action selection methods.

At time step $t$, the input state is $S_t$. Assume that $n$ fuzzy rules have been generated and the firing strength vector of rules is $\alpha_i$. Each rule $R_i$ has $M$ possible discrete actions $A'$. Local actions selected from $A'$ compete with each other based on their $q$-values, while the winning local action $a'$ of every rule cooperates to produce the global action based on the rule’s normalized firing strengths, $\alpha_i$. The global action is given by

$$U_i(S_t) = \sum_{a=1}^{n} \pi_j(q'_i, a'_i) = \sum_i a'_i \alpha_i$$  \hspace{1cm} (6)

where $a'_i$ is the selected action of rule $R_i$ at time step $t$.

#### 3.2 Update of Q-values

As in DFQL, we define a function $Q$, which gives the action quality with respect to states. Q-values are also obtained by the FIS outputs, which are inferred from the quality of the local discrete actions that constitute the global continuous action. Under the same assumption used for generation of continuous actions, the Q function is given by

$$Q(S_t, U_j) = \sum_{a=1}^{n} q'_i(a'_i)$$  \hspace{1cm} (7)

where $U_j$ is the global action, $a'_i$ is the selected action of rule, $R_i$ at time step $t$ and $q'_i$ is the $q$-value associated action, $a'_i$.

Based on TD learning, the Q-values corresponding to the rule optimal actions are used to estimate the TD error, which is defined as follows:

$$V_i(S_t) = \sum_{a=1}^{n} \max_{a_i} q'_i(a'_i)$$  \hspace{1cm} (8)

and the TD error is defined by

$$\tilde{\delta}_{t+1} = r_{t+1} + \gamma V_i(S_{t+1}) - Q_i(S_t, U_j)$$  \hspace{1cm} (9)

where $r_{t+1}$ is the reinforcement signal received at time $t+1$ and $\gamma$ is the discount factor used to determine the present value of future rewards. This TD error can be used to evaluate the action just selected. The learning rule is given by

$$q'_i(a'_i) = q'_i(a'_i) + \beta \tilde{\delta}_{t+1} \alpha'_i \hspace{1cm} i = 1,2,...,n$$  \hspace{1cm} (10)

where $\beta_i$ is the critic learning rate.

#### 3.3 $\epsilon$-Completeness Criteria for Rules Generation

Definition 1: $\epsilon$-Completeness of Fuzzy Rules [1]:

For any input in the operating range, there exists at least one fuzzy rule so that the match degree (or firing strength) is no less than $\epsilon$.

According to $\epsilon$-completeness, when an input vector $X \in R^t$ enters the system, we calculate the M-distance $md(j)$ between the observation $X$ and centers $C_j (j=1,2,...,n)$ of existing Ellipsoidal Basis Function (EBF) units. Next, find
\[ J = \arg \min_{j \in J} (md(j)) \]  \hspace{1cm} (11)

If
\[ md_{\text{min}} = md(J) > k_d \]  \hspace{1cm} (12)

where \( k_d \) is a pre-specified threshold, this implies that the existing system is not satisfied with \( \varepsilon \)-completeness and a new rule should be considered. Here, \( k_d \) is chosen as follows:
\[ k_d = \sqrt{\ln(1/\varepsilon)} \]  \hspace{1cm} (13)

### 3.4 TD Error Criteria for Rules Generation

It is not sufficient to only consider \( \varepsilon \)-completeness of fuzzy rules as the criterion of rule generation. New rules need to be generated in regions of the input fuzzy subspace where the approximation performance of the DFQL is unsatisfactory. Here, we introduce a separate performance index, \( \xi^i \), for each fuzzy subspace which enables the discovery of “problematic” regions in the input space. The performance index is updated as follows:
\[ \xi^i = \frac{[(K - \alpha^i) \xi^i + \alpha^i (\xi^i)^2]}{K} \quad K > 0 \]  \hspace{1cm} (14)

Using the squared TD error as the criterion, the rule firing strength \( \alpha^i \) determines how much the fuzzy rule \( R_j \) affects the TD error. Thus, if \( \xi^i \) is lower than a certain threshold, further segmentation should be considered for this fuzzy subspace at least.

### 3.5 Estimation of Rule Parameters

Once a new rule is generated, the next step is to assign centers and widths of the corresponding membership functions. Assume that \( n \) fuzzy rules have been generated. A new rule will be formed when the input pattern \( X \) enters the system according to the criteria of rules generation. Next, the incoming multidimensional input vector \( X \) is projected to the corresponding one-dimensional membership for each \( i^{th} \) input variable and the Euclidean distance \( ed_j \) between the data \( x_j \) and the boundary set \( \Phi_j \) is computed as follows:
\[ ed_j(j) = |x_j - \Phi_j(j)| \quad j = 1, 2, \ldots, n + 2 \]  \hspace{1cm} (15)

where \( \Phi_j \in \{x_{\text{min}}, c_1, c_2, \ldots, c_n, x_{\text{max}}\} \) and find
\[ f_i = \arg \min_{j=1,2,\ldots,n} (ed_j(j)) \]  \hspace{1cm} (16)

If
\[ ed_j(f_i) \leq k_{ed} \]  \hspace{1cm} (17)

where \( k_{ed} \) is a predefined constant that controls the similarity of neighboring membership functions, we assume that \( x_j \) can be completely represented by the existing fuzzy set \( E_{ij} \left( c_{ij}, \sigma_{ij} \right) \) without generating a new membership function. Otherwise, a new Gaussian membership function whose center is
\[ c_{ij+1} = x_j \]  \hspace{1cm} (18)

and the widths of all fuzzy sets in \( i^{th} \) input variable are adjusted as follows:
\[ \sigma_i = \frac{\max\{c_{i,j}, c_{i,j+1}\}}{\sqrt{\ln(1/\varepsilon)}} \]  \hspace{1cm} (19)

where \( c_{i,j} \) and \( c_{i,j+1} \) are the two centers of neighboring membership functions of the fuzzy set whose center is \( c_i \). By this approach, the fuzzy sets of input variables can satisfy \( \varepsilon \)-completeness of fuzzy rules. More details of the proof can be found in [1].

Combining the \( \varepsilon \)-completeness criterion and the TD error criterion together, we obtain the procedure of generating a new rule: When an input vector \( X \in R^r \) enters the system, we calculate the M-distance \( md(j) \) between the observation \( X \) and centers \( C_j (j = 1, 2, \ldots, n) \) of existing EBF units. Next, find
\[ J = \arg \min_{j \in J} (md(j)) \]  \hspace{1cm} (20)

If
\[ md_{\text{min}} = md(J) > k_d \]  \hspace{1cm} (21)

where \( k_d \) is an \( \varepsilon \)-completeness threshold, this implies that the existing system is not satisfied with \( \varepsilon \)-completeness and a new rule should be considered. Otherwise, if
\[ \varepsilon^i > k_e \]  \hspace{1cm} (22)

where \( k_e \) is a TD error criteria threshold, this fuzzy rule \( R_j \) does not satisfy the TD error criterion and a new rule should be considered.

### 4. Convergence Theorem

Reinforcement learning has been applied successfully to a variety of practical applications. However, a fundamental obstacle to a widespread application of reinforcement learning is that these algorithms seldom give the proof of the convergence to a solution. This is particularly true for variants of temporal difference learning with function approximation, for example, linear combination of feature vectors or neural networks. Fortunately, [4] provided the convergence of feature-based methods. On the other hand, the fuzzy system is equivalent to a series expansion of some basis functions which are named fuzzy basis functions. Referring to [4], we can give the convergence theorem.

#### 4.1 Conditions of Convergence

From the original Q-learning algorithm, we update the global Q value of the state-action pair according to
\[ Q_{\alpha+1}(S, U_{\alpha}) = Q(S, U_{\alpha}) + \beta \left[ r_{\alpha+1} + \gamma \max_{a \in A} Q(S_{\alpha+1}, U_{\alpha}) - Q(S, U_{\alpha}) \right] \]

This algorithm converges to the optimal Q-values, with probability 1, provided that all state-actions pairs are considered infinitely many times and the stepsize \( \beta \) diminished to zero at a suitable rate. In more detail, Q-
learning can be viewed as a combination of value iteration and simulation. Equivalently, Q-learning is the Robbins-Monro stochastic approximation method. The exploration-exploitation strategy satisfies the additional requirement, namely, that all potentially beneficial actions be explored. We introduce the following standard assumption concerning the stepsize sequence:

Assumption 1:
(a) The stepsize sequence satisfies \( \sum_{t=0}^{\infty} \beta(t) = \infty \)
(b) There exists some constant \( C \) such that \( \sum_{t=0}^{\infty} \beta^2(t) \leq C \)

We suggest define a value iteration operator \( T \)

\[
T(Q_t) = T(Q)
\]

It is easy to check that \( T \) is a contraction with respect to the maximum norm, for all \( Q, Q' \)

\[
\| T(Q) - T(Q') \|_\infty \leq \gamma \| Q - Q' \|_\infty
\]

For this reason, the sequence \( Q_t \) produced by the value iteration algorithm converges to \( Q' \) at the rate of a geometric progression.

In [4], value iteration with linear architecture has been discussed. More formally, the compact representations of the form is given by

\[
Q(W) = \sum_{i=1}^{K} W_i f_i(i) = W^T F(i), \quad \forall i \in S
\]  

where \( W \) is the parameter vector, \( F(i) = (f_1(i), \ldots, f_K(i)) \) is the feature vector associated with state \( i \).
Assumption 2: Let \( i_1, \ldots, i_k \in S \) be the pre-selected states used by the algorithm.
(a) The vectors \( F(i_1), \ldots, F(i_k) \) are linearly independent.
(b) There exists a value \( \gamma' \in [\gamma, 1] \) such that for any state \( i \in S \), there exist \( \theta_1(i), \ldots, \theta_k(i) \in \mathbb{R} \) with \( \sum_{i=1}^{k} \theta_i(i) \leq 1 \) and

\[
F(i) = \frac{\gamma'}{k} \sum_{i=1}^{k} \theta_i(i) F(i) \]

And the Convergence Theorem is [4]:

Let assumption 2 hold.
(a) There exists a vector \( W^* \in \mathbb{R}^k \) such that \( W(t) \) converges to \( W^* \)
(b) Let \( \hat{Q} \) be the optimal vector, and define \( e \) by letting

\[
e = \inf_{W^* \in \mathbb{R}^k} \left\| \hat{Q} - \hat{Q} (W^*) \right\|_e, \quad \text{the following hold:}
\]

\[
\left\| \hat{Q} - \hat{Q}(W^*) \right\|_e \leq \frac{\gamma + \gamma'}{\gamma (1 - \gamma') e}
\]

Recall \( \hat{Q}(q) = \sum_{i=1}^{K} q_i \alpha_i(i), \quad \forall i \in S \) in our learning algorithm. To bring this into the feature-based representation framework, we can view an individual fuzzy basis function \( \alpha_i \) as a feature.

Then, given a basis function architecture which linearly combines \( K \) basis functions, we can define

\[
f_i(i) = \alpha_i(i), \quad \forall i \in S
\]

and a feature mapping \( F(i) = (f_1(i), \ldots, f_K(i)) \). The architecture becomes a special case of the feature-based representation from eq. (20). We choose the states \( i_1, \ldots, i_k \) to be those corresponding to the centers of EBF units. For all \( k \in \mathbb{N} \), define \( \Psi = \min_{k \in \mathbb{N}} f_k(i) \). If \( \Psi \geq 0.5(1 + \frac{\gamma'}{\gamma'}) \), assumption 2 can be satisfied. The proof is omitted due to the paper length.

4.2 Alternative Membership Functions

There are various possible types of fuzzy membership functions. We choose Gaussian functions as membership functions since they represent a very easy way to describe the generation of new fuzzy rules. Similar ideas can be developed in learning algorithms except some complicated descriptions if the FIS considered uses triangular shapes of membership functions. The membership functions is defined by the following expression:

\[
\mu_i(x) = \left\{ \begin{aligned}
1.0 & \quad x = C_i \\
\max & \quad 0.0, 1.0 - \frac{x - C}{C_{i+1} - C_i}, \quad x > C_i \\
\max & \quad 0.0, 1.0 - \frac{C - x}{C_{i+1} - C_i}, \quad x < C_i
\end{aligned} \right.
\]

where \( C_i \) and \( C_{i+1} \) are the two centers of neighboring membership functions of the fuzzy set whose center is \( C_k \).

This kind of partition implies that there are no more than two fuzzy sets “activated” for an input value and all values “activate” at least one fuzzy set. With these conventions, the assumption 2 can be satisfied without more assumptions.

5. Extensions

5.1 Initial Values Based on Prior Knowledge

The if-then fuzzy rules corresponding to the domain knowledge about the task can be incorporated into the DFQL design. The premise of rules can be used to generate EBF units over the fuzzy input space and the consequents of rules can be used to generate the initial Q-values, which are called bias. The rule’s parameter vector \( q \) associated with discrete actions is initialized so that a greedy policy would select the action \( a \) suggested by this rule. The Q-value of the selected discrete action \( a \) is initialized to a fixed value \( k_q \), while all the others are given random values according to a uniform distribution in \([0, k_q/2]\).
5.2 Eligibility Traces

In order to speed up learning, eligibility traces are used to memorize previously visited rule-action pairs, weighted by their proximity to time step \( t \). The trace value indicates how rule-action pairs are eligible for learning. Thus, it not only permits tuning of parameters used at time step \( t \), but also those involved in past steps. However, the Sarsa(\( \lambda \)) cannot be used directly because we need TD errors before tuning the structure of FIS. Fortunately, a \( Q(\lambda) \)-learning algorithm has been proposed in [5]. We can easily extend DFQL to DFQL(\( \lambda \)) based on \( Q(\lambda) \)-learning algorithm. The learning algorithm is summarized as follows:

1. \( q(s, a) = 0 \) and \( Tr(s, a) = 0 \) for all fuzzy state \( s \) and local action \( a \).
2. Do Forever:
   (a) \( S_t \leftarrow \) the current state
   (b) Choose a global action \( U_t \) according to eq. (6).
   (c) Carry our action \( U_t \) in the world. Let the short-term reward be \( r_{t+1} \), and the new state be \( S_{t+1} \)
   (d) \( \tilde{e}_{t+1} = r_{t+1} + \gamma V_{t+1}(S_{t+1}) - V_t(S_t) \)
   (e) \( e_{t+1} = r_{t+1} + \gamma V_{t+1}(S_{t+1}) - V_t(S_t) \)
   (f) For each fuzzy state-action pair \( (s, a) \) does
   \[ Tr(s, a) = \gamma \lambda Tr(s, a) + q_{t+1}(s, a) + \beta Tr(s, a) \tilde{e}_{t+1} \]
   (g) For "activate" fuzzy rules
   \[ q_{t+1}(s, a_i) = q_{t+1}(s, a_i) + \beta \tilde{e}_{t+1} \alpha_i(s_t) \]
   \[ Tr(s_t, a_i) = Tr(s_t, a_i) + \alpha_i(s_t) \]
   (h) Tuning of parameter \( \xi \) with Eq (14) being used as a TD error criterion; Checking the \( \varepsilon \)-completeness and TD error criteria according to the current state. If a new fuzzy rule need to be generated, tune the structure of FIS and initialize the parameter vector \( q \) according to the algorithm described in Section 3.5.

6. Conclusions

In this paper, we have presented a new Q-learning method, termed DFQL, to generate fuzzy controllers. Based on the criteria of \( \varepsilon \)-completeness and the TD errors, new fuzzy rules can be generated automatically, which allow us to circumvent the problem of setting up the fuzzy rules by hand. DFQL generates continuous-valued actions using fuzzy reasoning. We have already implemented experiments for the control of mobile robots using this approach. Experimental results and comparative studies with the Fuzzy Q-Learning and Continuous-Action Q-learning in the wall following task of mobile robots demonstrate the superiority of the proposed DFQL method.

References