

Fuzzy Q-Learning

Pierre Yves Glorennec Lionel Jouffe
Département d'Informatique
INSA de Rennes
(glorenne,jouffe)@irisa.fr

Abstract

This paper proposes an adaptation of Watkins' Q-Learning for Fuzzy Inference systems where both the actions and the Q-functions are inferred from fuzzy rules. This approach is compared with Genetic Algorithm on the Cart-Centering Problem, showing its effectiveness.

Keywords: Q-Learning, reinforcement, Fuzzy Inference Systems, process control, Genetic Algorithm.

1 Introduction

Reinforcement learning is learning with a critic instead of a teacher. The only feedback provided by the critic is a scalar signal, r , called reinforcement, which can be thought of as a reward or a punishment. Reinforcement learning performs an on-line search to find an optimal decision policy in multi-stage decision problems.

Q-Learning [21] is a reinforcement learning method where the learner builds incrementally a Q-function which attempts to estimate the discounted future rewards for taking actions from given states. The output of the Q-function for state x and action a is denoted by $Q(x, a)$.

When action a has been chosen and applied, the system moves to a new state, y , and a reinforcement signal, r , is received. $Q(x, a)$ is updated by:

$$Q(x, a) = (1 - \alpha)Q(x, a) + \alpha\{r + \gamma V(y)\} \quad (1)$$

where $V(y)$ is the value of state y , defined by:

$$V(y) = \max_{b \in A(y)} Q(y, b) \quad (2)$$

$A(y)$ is the set of possible actions in state y , α is the learning rate and γ a discount factor.

The Q-values are usually stored in a look-up table, but this method is either unpracticable in case of large state-action spaces, or impossible with continuous state space. Different authors have proposed to use the generalization ability of feedforward Neural Networks [15, 17] or Self-Organizing Maps [19] to store the Q-values. Unfortunately this approach combines two slow processes: Neural Networks and Reinforcement Learning are known for their slow learning rate. Another approach consists in extending Q-Learning into fuzzy environments [4, 12]. The motivations are twofold:

- Fuzzy Inference Systems are universal approximators [9, 20] and are good candidates to store the Q-values.
- Prior knowledge can be embedded into the fuzzy rules, which can reduce training significantly.

This paper presents one extension of Watkin's Q-Learning into a fuzzy environment. A more complete description of Fuzzy Q-Learning (FQL) and Fuzzy Actor-Critic Learning (FACL) can be found in [14]. This paper is organized as follows: in Section 2, we present the FQL algorithm. We compare FQL and a Genetic Algorithm approach in Section 3 with the Cart-Centering Problem, and we conclude.

2 Fuzzy Q-Learning

2.1 Presentation

We have proposed earlier [12] to represent both the actions and the Q-function by FIS. The rules have the form:

if S_i **then** action = a_i , $a_i \in A_i$

if S_i **and** action = a_i **then** q-value = $q(S_i, a_i)$

where the state S_i is defined by “ x_1 is $S_{i,1}$ **and** x_2 is $S_{i,2}$... **and** x_n is $S_{i,n}$ ”, $(S_{i,j})_{j=1}^n$ are fuzzy labels and A_i is the set of possible actions that can be chosen in state S_i .

We use simplified Takagi-Sugeno FIS with N rules; therefore, the inferred action $a(x)$ for input vector $x = (x_1, \dots, x_n)$ and rule consequents $(a_i)_{i=1}^N$ is:

$$a(x) = \frac{\sum_{i=1}^N \alpha_i(x) \times a_i}{\sum_{i=1}^N \alpha_i(x)} \quad (3)$$

and the corresponding Q-value:

$$Q(x, a) = \frac{\sum_{i=1}^N \alpha_i(x) \times q(S_i, a_i)}{\sum_{i=1}^N \alpha_i(x)} \quad (4)$$

The function $x \rightarrow \alpha_i(x)$ represents the truth value of rule i given the input vector x .

To simplify, we suppose that the learning system can choose one action among J for each rule; we call $a[i, j]$ the j^{th} possible action in rule i and $q[i, j]$ its corresponding q-value. We build a FIS with competing actions for each rule:

if x is S_i **then** $a[i, 1]$ with $q[i, 1]$
or $a[i, 2]$ with $q[i, 2]$

or $a[i, J]$ with $q[i, J]$

The learning agent has to find the best conclusion for each rule, i.e. the action with the best q-value.

The q-values are zeroed initially and are not significant in the first stages of the learning process. In order to explore the set of possible actions and acquire experience through the reinforcement

signals, the actions are selected using an Exploration/Exploitation Policy (EEP). Instead of the usual Boltzmann exploration, we use either a Pseudo-Stochastic [8] or a combined directed/non-directed exploration [14], in order to maximize the escounted knowledge gain.

Let i^\dagger be the selected action in rule i using an EEP and i^* such as $q[i, i^*] = \max_{j \leq J} q[i, j]$. The actual Q-value of the inferred action, a , is:

$$Q(x, a) = \frac{\sum_{i=1}^N \alpha_i(x) \times q[i, i^\dagger]}{\sum_{i=1}^N \alpha_i(x)} \quad (5)$$

and the value of state x :

$$V(x) = \frac{\sum_{i=1}^N \alpha_i(x) \times q[i, i^*]}{\sum_{i=1}^N \alpha_i(x)} \quad (6)$$

2.2 Updating the q-values

Let x be a state, a the action applied to the system, y the new state and r the reinforcement signal. $Q(x, a)$ is updated using Equations (1) and (6). The difference between the “old” and the “new” $Q(x, a)$ can be thought of as an error signal, $\Delta Q = r + \gamma V(y) - Q(x, a)$, than can be used to update the action q-values. By an ordinary gradient descent we obtain:

$$\Delta q[i, i^\dagger] = \epsilon \times \Delta Q \frac{\alpha_i(x)}{\sum_{i=1}^N \alpha_i(x)} \quad (7)$$

where ϵ is a learning rate.

To speed up learning, we can combine Q-Learning and Temporal Difference (TD(λ)) Method [18]. We define the eligibility $e[i, j]$ of an action by:

$$e[i, j] = \begin{cases} \lambda \gamma e[i, j] + \frac{\alpha_i(x)}{\sum_{i=1}^N \alpha_i(x)} & \text{if } j = i^\dagger \\ \lambda \gamma e[i, j] & \text{elsewhere} \end{cases} \quad (8)$$

Therefore, the updating equation (7) becomes:

$$\Delta q[i, i] = \epsilon \times \Delta Q \times e[i, j] \quad (9)$$

The global algorithm is described in Table 1.

- 1 Observe the state x
- 2 For each rule: choose the actual consequence using some EEP
- 3 Compute the global consequence $a(x)$ and its corresponding Q-value $Q(x, a)$
- 4 Apply the action $a(x)$. Let y be the new state
- 5 Receive the reinforcement r
- 6 Update q-values using Eq. (9)

Table 1: Fuzzy Q-Learning algorithm

2.3 Other FQL variants

Other fuzzy adaptations of Q-Learning have been proposed. In [11], a set of FIS are competing to control the process and $Q(x, A_k)$ represents the Q-value of action A_k proposed by the FIS number k when the system is in state x . The rules have the form:

- if** x is S_i **then** $a[i, 1]$ with $q[i, 1]$
and $a[i, 2]$ with $q[i, 2]$

and $a[i, J]$ with $q[i, J]$

and the Q-value of the inferred output of FIS k is:

$$Q(x, A_k) = \frac{\sum_{i=1}^N \alpha_i(x) \times q[i, k]}{\sum_{i=1}^N \alpha_i(x)} \quad (10)$$

The system is depicted in Figure 1. It can be used for knowledge fusion between several experts/FIS.

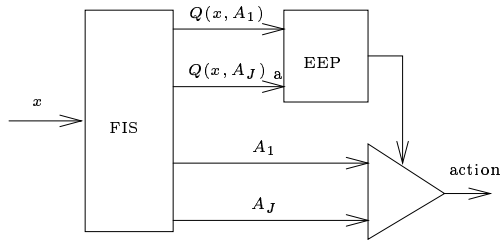


Figure 1: Architecture for knowledge fusion

In [14], several improvements and extensions are proposed: FQL with discrete action, called QFUZZ by analogy with Lin's QCON [15], self-adaptation of FQL parameters, comparison between FQL and Fuzzy Actor-Critic Learning. QFUZZ architecture is depicted in Figure 2.

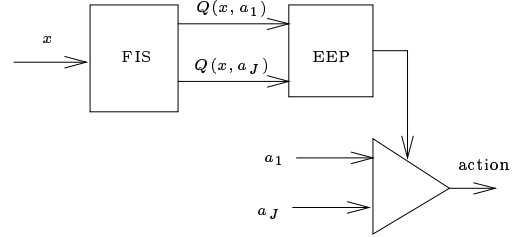


Figure 2: QFUZZ architecture

3 Example: the cart-centering problem

In order to show the effectiveness of FQL, we have applied it on the famous Cart-Centering Problem. A cart is moving on a one-dimensional track. The state is defined by position, x and velocity, v , of the cart. The goal is to bring it to $x = 0$ and $v = 0$ from any arbitrary initial condition, in minimum time. The equations of motion are:

$$\begin{cases} x_{t+\tau} = x_t + \tau v_t \\ v_{t+\tau} = v_t + \tau \frac{F_t}{m} \end{cases} \quad (11)$$

where $\tau = 0.02sec$ is the time step, $F \in [-150, 150]$ is a force and $m = 20kg$ the mass.

In [13], a Genetic Algorithm is used to find the best fuzzy controller performing the given task. The so-called "555" Fuzzy Controller is defined by five isoscele triangular membership functions for x and v , centered at -2, -1, 0, 1 and 2, with variable width. The conclusions of the 25 fuzzy rules take their values in $\{-150, -75, 0, 75, 150\}$. The GA is characterized by probability of crossover = 0.7, probability of mutation = 0.03 and a population size of 100 strings. Learning was divided into two stages:

	NM	NS	ZR	PS	PM
NM	150	150	150	150	75
NS	75	150	150	150	-75
ZR	75	150	0.0	-150	-75
PS	75	-150	-150	-150	-75
PM	-75	-150	-150	-150	-150

Table 2: The optimal rule base found by GA

- 30 generations to find satisfactory controllers
- 70 additional generations to find the best controller.

The optimal rule base found by GA is depicted in Table 3 and the trajectories in the plane x, v in Figure 3.

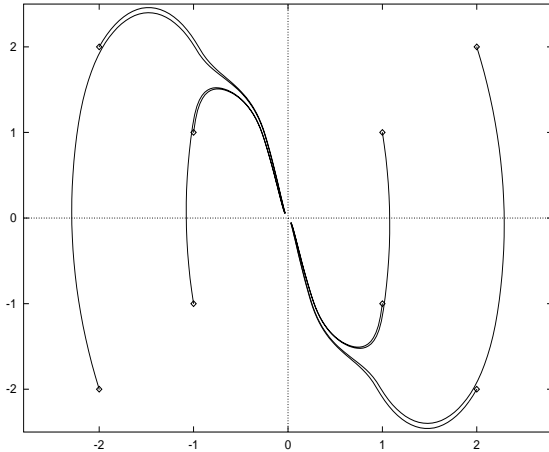


Figure 3: Trajectories in the plane x, v with the best GA controller.

We have used FQL on the same problem with two series of ten experiments. The number of trials was limited to 200. In both experiments, the parameters of FQL were: $\lambda = \gamma = 0.9$, $\epsilon = 0.001$ and the reinforcement signal:

	failure	first	average
without knowledge	3	8	30
with knowledge	1	4	20

Table 3: Results of the FQL search

$$r = \begin{cases} +1 & \text{if } |x| < 0.1 \text{ and } |v| < 0.1 \\ -1 & \text{if } |x| > 4 \text{ or } |v| > 3 \text{ or iter} > 175 \\ 0 & \text{elsewhere} \end{cases} \quad (12)$$

- Without any knowledge. In three cases, FQL was not able to find a satisfactory controller in less than 200 trials.
- With prior knowledge. We have introduced the rules:
if $x = v = 0$ then $F = 0$
if $x = v = 2$ then $F = -150$
if $x = v = -2$ then $F = 150$

The results are shown in Table 3 and the trajectories in Figure 4.

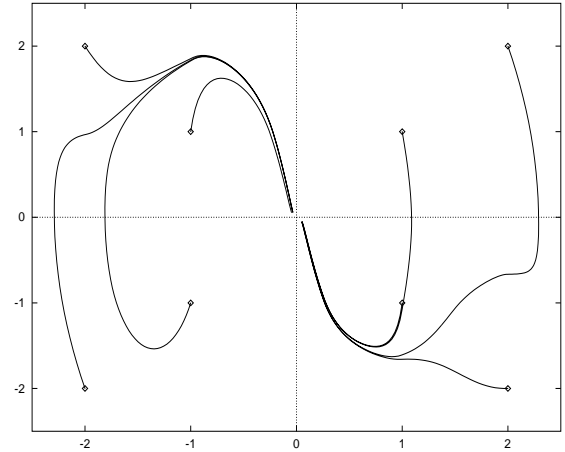


Figure 4: Trajectories in the plane x, v with the best FQL controller.

3.1 Comparison between GA and FQL

GA and FQL are based on stochastic search: mutation and cross-over for GA, and exploration for FQL. Both methods exhibit interesting performances when only a poor feedback signal is available. The methods differ in two main points:

- With GA, the fitness function is evaluated once only, at the end of a trial, whereas with Q-Learning, the Q-values are evaluated on-line.
- With GA, we know the performance for the whole set of conclusions, whereas FQL allows to examine the quality of individual actions and gives more acute insight on the acquired knowledge.

4 conclusion

We propose efficient adaptations of Watkins' Q-Learning allowing on-line optimization of the consequent part of fuzzy rules. Thanks to the knowledge representation ability of fuzzy rules, prior knowledge can be incorporated into the initial set of rules, speeding up the learning stage. Moreover the action and state spaces can be continuous, extending the application domain of Q-Learning.

References

- [1] Aubin J.P., "Learning rules of cognitive processes", C.R. Acad. Sc. Paris, T. 308, Série I, 1989.
- [2] Barto A., Sutton R., Anderson C. "Neuron-like elements can solve difficult learning control problems", *IEEE Trans. on SMC*, Vol. 13, Sept. 83.
- [3] Barto A., Sutton R., Watkins C. "Sequential decision problems and neural networks", in *Advances in Neural Information Processing Systems 2*, D. Touretzky, Ed. Morgan Kaufmann, San Mateo, CA, 1990.
- [4] Berenji H., Khedkar P., "Learning and tuning fuzzy logic controllers through reinforcement", *IEEE Trans. on Neural Networks*, 3(5), Sept. 92.
- [5] Berenji H., "Fuzzy Q-Learning for generalization of reinforcement learning", *Proc. of FUZZIEEE'96*, New Orleans, Sept. 1996.
- [6] Bersini H., "Reinforcement learning and recruitment mechanism for adaptive distributed control", TR. IR/IRIDIA/92-4, Université Libre de Bruxelles, 1992.
- [7] Bonarini A., "Reinforcement learning of hierarchical fuzzy behaviors for autonomous robots", *WSC1*, special session on Fuzzy Logic in Autonomous Robotics, 1996.
- [8] Caironi P., Dorigo M., "Training Q-agents", *Tech. Rep. IRIDIA/94-14*, Université Libre de Bruxelles, Belgium, 1994.
- [9] J.L. Castro, "Fuzzy logic controllers are universal approximators", *IEEE Trans. on SMC*, Vol. 25 n° 4, April 95.
- [10] Clouse J.A., Utgloff P.E., "A teaching method for reinforcement learning", *Proc. of 9th Workshop on Machine Learning ML'92*, 1992.
- [11] Glorennec P.Y., "Fuzzy Q-Learning and Dynamical Fuzzy Q-Learning", *Proc. of the 3th IEEE Int. Conf. on Fuzzy Systems*, Orlando, June 1994
- [12] Glorennec P.Y., Jouffe L., "Reinforcement learning for autonomous robots", *Proc. of EUFIT'96*, Aachen, Germany, Sept. 1996.
- [13] Homaifar A., McCormick E., Simultaneous Design of Membership Functions and Rule Sets for Fuzzy Controllers Using Genetic Algorithms", *IEEE Trans. on Fuzzy Systems*, Vol. 3, Num. 2, May 1995.
- [14] Jouffe L., "Fuzzy Inference System learning by reinforcement methods", *Tech. Rep. INSA 96081*, submitted, 1996.

- [15] Lin L-J., "Self-improvement based on reinforcement learning, planning and teaching", *Proc. of 8th Workshop on Machine Learning ML'91*, 1991.
- [16] McCallum R.A., "Using transitional proximity for faster reinforcement learning", *Proc. of 9th Workshop on Machine Learning ML'92*, 1992.
- [17] Rummery G., Niranjan M. "On-line Q-Learning using connectionist systems", *Tech. Rep. CUED/F-INFENG/TR 166*, Cambridge University, 1994.
- [18] Sutton R.S., "Learning to predict by the method of temporal differences", *Machine Learning*, 3, pp. 9-44, 1989.
- [19] Touzet P., "Neural reinforcement learning for behavior synthesis", *Proc. of CESA'96, IMACS Multiconference*, Lille, July 1996.
- [20] Wang L.X., "Fuzzy systems are universal approximators", *Proc. First IEEE Conf. on Fuzzy Systems*, pp.1163-1169, San Diego, 1992.
- [21] Watkins C., "Learning from delayed rewards", PhD Thesis, University of Cambridge, England, 1989.
- [22] Watkins C., Dayan P., "Q-Learning, Technical Note", *Machine Learning*, Vol. 8, p.279-292, 1992.
- [23] Whitehead S.D. "A complexity analysis of cooperative mechanisms in reinforcement learning" in *Proc. of the 9th AAAI Conf.*, 1991.