Available online at www.sciencedirect.com
ScienceDirect
Fuzzy Sets and Systems 160 (2009) 1420–1443
FUZZY sets and systems
www.elsevier.com/locate/fss

Reinforcement distribution in fuzzy Q-learning

Andrea Bonarini, Alessandro Lazaric, Francesco Montrone, Marcello Restelli*

Department of Electronics and Information, Politecnico di Milano, Milan, Italy

Available online 6 December 2008

Abstract

Q-learning is one of the most popular reinforcement learning methods that allows an agent to learn the relationship between interval-valued state and action spaces, through a direct interaction with the environment. Fuzzy Q-learning is an extension to this algorithm to enable it to evolve fuzzy inference systems (FIS) which range on continuous state and action spaces. In a FIS, the interaction among fuzzy rules plays a primary role to achieve good performance and robustness. Learning a system where this interaction is present gives to the learning mechanism problems due to eventually incoherent reinforcements coming to the same rule due to its interaction with other rules. In this paper, we will introduce different strategies to distribute reinforcement to reduce this undesired effect and to stabilize the obtained reinforcement. In particular, we will present two strategies: the former focuses on rewarding the actions chosen by each rule during the cooperation phase, the latter on rewarding the rules presenting actions closer to those actually executed rather than the rules that contributed to generate such actions.

Keywords: Fuzzy systems; Fuzzy Q-learning; Reinforcement learning; Reinforcement distribution

1. Introduction

Fuzzy Q-learning is an approach to learn a set of fuzzy rules by reinforcement. It is an extension of the popular Q-learning [1] algorithm, widely used to learn tabular relationships among states, described by a finite number of values for each variable, and discrete actions. Learning fuzzy rules makes it possible to face problems where inputs are described by real-valued variables, matched by fuzzy sets, and also actions are real-valued. Fuzzy sets play the role of the ordinal values used in Q-learning, thus making possible an analogous learning approach, but overcoming the limitations due to the interval-based approximation needed by Q-learning to face the same type of problems. The partial overlapping among close fuzzy sets covering the range of each variable, although suitable for improving robustness, smoothness, and many other desired characteristics of fuzzy inference systems (FIS), induces problems for the evaluation of the contribution of the single rules, since each of them is activated in turn with different rules, and may obtain from this *collaboration* [2] different reinforcements. This may result in an incoherent reinforcement assignment which makes convergence more difficult. Thus, reinforcement distribution becomes a relevant issue in the definition of fuzzy Q-learning.

In this paper, we propose a new reinforcement distribution method for fuzzy Q-learning that gives better performances than the traditional one when the domain is described by a large number of fuzzy sets, and that combines favorably with the traditional one when equivalently optimal actions are possible.

* Corresponding author.
E-mail addresses: bonarini@elet.polimi.it (A. Bonarini), lazaric@elet.polimi.it (A. Lazaric), montrone@elet.polimi.it (F. Montrone), restelli@elet.polimi.it (M. Restelli).

0165-0114/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.fss.2008.11.026

The paper is structured as follows: the next section describes the main results about the use of function approximation techniques in reinforcement learning (RL) algorithms, with particular attention to methods based on the fuzzy theory. Section 3 introduces the basic notation of the RL framework and the FIS with particular attention to the problem of function approximation. Section 4 describes in detail the main phases of the fuzzy Q-learning algorithm. Two different strategies for reinforcement distribution are formally defined in Section 5. Their main differences are discussed in Section 6 and highlighted by experimental results presented in Section 7. In the last section we draw conclusions and propose new directions for future research activities.

2. State of the art

RL is a learning paradigm that has become the standard framework in machine learning for designing agents able to learn how to behave optimally in uncertain environments. An agent performing RL gets no direct information about the optimal control strategy, but simply receives instantaneous reinforcement signals. In its simplest form, it is based on the idea that, if an action is followed by a reward (i.e., an "improvement" in the state of affairs), then the tendency to produce that action is reinforced; otherwise, the tendency to produce that action is weakened [3,4]. The objective is to find a policy, a function that maps the states of the system to actions, that maximizes the utility in each state of the environment. It is often considered that the utility is the expected sum of the reinforcements collected from a state on. This requires the computation of the action-value function $Q(s, a)$, that is the expected sum of reinforcements starting from a state s and taking an action a.

2.1. Function approximation in RL

Most of the RL algorithms [5] represent the action-value function as a look-up table with one entry for each state–action pair. While this approach has strong theoretical foundations [6–9] and is effective in many applications, it is a severe limitation when applied to problems characterized by large state and action spaces or with continuous domains, due to the phenomenon called *curse of dimensionality* [5]. Several approaches try to overcome this problem by applying function approximation so as to approximate the action-value function with few parameters. Therefore, the agent can experience only a limited subset of the state space and then, through *generalization*, can produce a good approximation on a larger portion of the state space.

The problem of building a function approximator from a set of input–output mappings has been extensively studied by supervised learning [10]. However, function-approximation and generalization in RL are harder to implement than in supervised learning, because the training data are not given in advance by a trainer, but are in part determined by the output of the learned function. Since learning must occur on-line, while interacting with the environment, not all function-approximation methods used in supervised learning are well-suited for use in RL.

State aggregation is one of the most-studied function approximators in which subsets of the original state space are aggregated, thus reducing the size of the state space. Several algorithms use multiple overlapping partitions of the state space so that the value function is approximated by a linear combination of the values in each partition. Many learning algorithms [11,12] are based on a *multigrid* approach that uses partitions with different resolutions to speed-up the learning process. Another solution is the introduction of *soft-state aggregations* [13], in which states are soft clustered. Most of the state-aggregation approaches suffers from two problems. First of all, it may turn necessary to partition the state space into tiny regions in order to solve the problem, thus obviously reducing the advantages of the aggregation. Moreover, the criterion used to partition the state space is crucial in order to obtain an effective approximation of the value function over all the state space. Nonetheless, when there are no clues about how to divide the state space (e.g., knowledge about the shape of the function), uniform partitioning is commonly used, and this may result in a huge set of input features. Multiple overlapping partitions, called *tilings*, are also used in *CMAC* (or *tile coding*) [14]. The shape of the tilings determines the nature of the generalization. The generalization is mainly affected by the size and the shape of the tilings, while the resolution, that is the finest discrimination available, depends on the total number of tilings. radial basis functions (RBFs) are the natural generalization of tile coding to continuous state space [15,16]. Each tiling is no longer associated to a binary activation value, but it can take any value in the interval [0, 1]. The primary advantage of RBFs over binary activations is that they produce approximate functions that are smooth and differentiable. The main drawbacks of RBF networks are in their great computational complexity and in the fact that they often require long manual tuning in order to make the learning robust and efficient.

In many real applications, the agent has to choose actions very precisely in the action domain. Nonetheless, while the previous approaches are suitable for continuous state space, the problem of continuous action spaces received less attention. Most of the proposed solutions adapt the techniques used for state spaces to action spaces [17–19].

2.2. Fuzzy systems in RL

FIS have been widely adopted as function approximators for RL problems, in particular in conjunction with Q-learning [20].

Bellman and Zadeh [21] introduced the first method for decision making in fuzzy environments, which is defined by fuzzy goals and fuzzy constraints. The basic idea behind this method is that a decision is given by the confluence of goals and constraints. Since then, a large variety of algorithms for decision making under fuzzy goals and constraints have been developed [22,23].

Glorennec [24] and the extension proposed by Jouffe [25] provided a fundamental contribution in the definition of fuzzy Q-learning, that is the basis for many of the existing implementations. In this approach, agents represent different (predefined) rule bases that are evaluated by Q-values. In a given state the approach selects the agent whose active rules have the largest Q-values. This agent may then execute the action proposed by its rule base. Afterwards, the Q-values of this agent are updated. Berenji and Khedkar [26] introduced the Generalized Approximate Reasoning based Intelligent Control (GARIC) architecture, an extension of this approach. The GARIC system can be considered as an actor–critic approach, where both the actor and the critic are represented by neural networks. The structure of the actor network simulates the behavior of a Mamdani fuzzy controller. By adapting the weights of the actor network the parameters of the corresponding Mamdani controller are tuned. In this framework each agent represents a GARIC system and both the Q-values and the rule base of the active agent are adapted after the execution of an action. Beom and Cho [27] presented a fuzzy actor–critic approach that can be considered as a fuzzy extension of the actor–critic method. Interesting ideas come from Gu and Hu [28], in which they expose a Jouffe-like fuzzy Q-learning, called *accuracy-based fuzzy Q-learning*, in which the consequent part of the rules takes into consideration the variation of the parameter, which is then used as fitness value of the rule for a genetic algorithm that modifies the actions available in each rule.

Horiuchi et al. [29] presented a fuzzy Q-learning algorithm for continuous state and action spaces. In their approach the (continuous) Q-function is represented by a Takagi–Sugeno FIS and it can be used also in conjunction with a genetic algorithm for the generation of suitable discrete action space [30]. Dai et al. [31] proposed a system composed by a neural network that computes the action-value function, a FIS with one output per rule that interpolates the optimal action function, and a stochastic action modifier that changes dynamically the output of the rules in order to find the most appropriate ones.

In [32–34], Meng Joo Er and Chang Deng present a version of fuzzy Q-learning that adapts the representation of the fuzzification called Dynamic Fuzzy Q-learning (DFQL). This method is based on the one proposed by Jouffe [25] and focuses on the domain representation: beside the ability of tuning the system parameters on-line, the algorithm is capable to self-organize itself on-line so that structure and parameters identification are accomplished automatically and simultaneously based only on Q-learning. The fuzzification of the domain is accomplished by a RBF network that is initialized with few RBFs and progressively grows when there are regions in the state space that are not covered "enough", or when the moving average of the quadratic temporal error is greater than a certain threshold. Other methods for identifying the position and the widths of new RBFs are proposed in [35] where the distribution of the approximation error throughout the space is considered.

3. Formalization of fuzzy RL

In this section we formalize the problem of function approximation in RL and we introduce FIS.

3.1. Reinforcement learning

In this section we introduce the notation of RL and we focus on the problem of approximating the action value function when a tabular approach cannot be used (e.g., in problems with continuous state and action spaces). Finally, we introduce FIS as a mean for function approximation.

3.1.1. Markov decision processes

In a RL problem, the environment and its interaction with an agent are formally defined as a finite Markov decision process (MDP), characterized by:

- a state space $\mathcal{S}$,
- an action space $\mathcal{A}$,
- a state transition function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$,
- a reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,

where the transition function $P(s, a, s')$ gives the probability to get from s to s' by taking action a and the reward function $R(s, a)$ gives the agent a reinforcement signal for the state–action pair (s, a). The goal of the agent is to learn the policy (i.e., the best sequence of actions) that is optimal according to a specific evaluation criterion. In case of infinite horizon MDPs, the most used criterion is the expected sum of discounted rewards accumulated in time:

$$E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots] = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+1+k}\right]$$

where $\gamma \in [0, 1)$ is the *discount factor* that weights the relevance of future rewards with respect to recent ones. A policy is defined as a function $\pi : S \times A \to [0, 1]$, that maps each state–action pair (s, a) to the probability to take action a when in state s. The value function $V^{\pi}(s)$ is defined as the sum of discounted rewards when following policy π and represents the utility of each state. For any MDP, there exists at least one deterministic optimal policy (π^*) that maximizes the expected sum of discounted rewards. The optimal policy is defined as the greedy policy with respect to the optimal value function $V^*(s)$, defined by the recursive Bellman equation [36]:

$$V^*(s) = \max_a \left[R(s, a) + \gamma \sum_{s'} P(s, a, s')V^*(s')\right] \tag{1}$$

Besides the value function $V(s)$, it is possible to define also the optimal action-value function $Q^*(s, a)$ that maps each state–action pair to the highest expected sum of discounted rewards that can be obtained by taking action a in

Algorithm 1. The Q-learning algorithm

```
Initialize Q(s, a) arbitrarily
for all episode do
  Initialize s^t
  while s^t is not terminal do
    a^t ← π(s^t)
    Take action a^t; observe r^{t+1} and s^{t+1}
    Q(s^t, a^t) ← Q(s^t, a^t) + α[r^{t+1} + γ max_{a'} Q(s^{t+1}, a') − Q(s^t, a^t)]
    t ← t + 1
  end while
end for
```

state s and following the greedy policy thereafter. The Bellman equation for the action-value function is

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} P(s, a, s') \max_{a'} Q^*(s', a') \tag{2}$$

The optimal policy is defined as the greedy action in each state: $\pi^*(s) = \arg\max_a Q^*(s, a)$. Both versions of the Bellman equations can be solved using dynamic programming algorithms such as policy iteration and value iteration [37], but both the transition model and the reward function are required to compute the optimal value functions.

3.1.2. Q-learning

Unlike dynamic programming algorithms, RL algorithms such as Q-learning and SARSA [5] can iteratively estimate the optimal action-value function without any information about transition probabilities and rewards. In particular, when

the state and action spaces are discrete, Q-learning [38] (Algorithm 1) enables the agent to compute the optimal action-value function by a direct interaction with the environment. When the agent takes action a in state s, receives a reward $r = R(s, a)$ and gets to state s', the estimation of the action-value function is updated as

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a')) \tag{3}$$

where $\alpha \in [0, 1]$ is a learning rate. Q-learning is proved to converge to the optimal action-value function under very loose hypotheses on the learning rate, derived from stochastic approximation convergence theorem [39], and under the hypothesis that each state–action pair is visited infinitely often.

The performance of Q-learning, and of most of the RL algorithms, is strongly influenced by the way the agent faces the exploration–exploitation dilemma. In fact, at each time instant the agent has to exploit what it has already learned in order to obtain high rewards, but at the same time it also has to explore the environment in order to discover unexplored and potentially more rewarding regions of the state space. Therefore, the agent must continue to explore different actions, but progressively favor those that are estimated to be the best. For each state s, the estimation of the quality of an action a is stored in the corresponding action value $Q(s, a)$.

One of the most simple and frequently used exploration strategy is the ε-greedy [5]. According to this strategy, with probability $1 - \varepsilon$, the action with highest action value is chosen, and with probability ε the agent takes one action at random. Typically the value of ε decreases in time, so that the more the agent learns, the greedier its behaviour is.

3.1.3. Function approximation in RL

In problems characterized by continuous state and action spaces, Q-learning must be used in conjunction with function approximators in order to approximate the action-value function. Starting from the solutions adopted in supervised learning, many function-approximation techniques have been widely adopted in RL problems [1] to face domains with high dimensionality or continuous spaces. However, the application of function approximation in RL is more complex than in supervised learning, because the training data are not given in advance by a trainer, but determined by the value function currently estimated.

The goal of function approximation is to approximate the function V^* with the estimate $\hat{V}^t(s, \boldsymbol{\theta}^t)$, which is of the continuous state s and of a set of *parameters* $\boldsymbol{\theta}^t$. The approximation error is usually defined as the Mean Squared Error (MSE), and the best values for the parameters are those which minimize the MSE over a distribution P:

$$MSE(\boldsymbol{\theta}^t) = \sum_{s \in S} P(s)[V^*(s) - \hat{V}^t(s, \boldsymbol{\theta}^t)]^2 \tag{4}$$

where P weights the error with the frequency of visit to each state s. The most straightforward technique for the minimization of the error is the gradient descent: each parameter of $\boldsymbol{\theta}^t$ is updated in the direction of the maximum effect achievable, i.e., along the gradient direction. The gradient $\nabla_{\boldsymbol{\theta}^t} f(\boldsymbol{\theta}^t)$ of a function f is the vector of the partial derivatives $\nabla_{\boldsymbol{\theta}^t} f(\boldsymbol{\theta}^t) = \left\{\partial f(\boldsymbol{\theta}^t)/\partial\theta^t_1, \ldots, \partial f(\boldsymbol{\theta}^t)/\partial\theta^t_N\right\}$.

Since RL works on-line, the parameters are updated as soon as a sample of the value function is available. The update of $\boldsymbol{\theta}^t$ with gradient descent method applied to the error function is

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \tfrac{1}{2}\alpha\nabla_{\boldsymbol{\theta}^t}[V^*(s) - \hat{V}^t(s, \boldsymbol{\theta}^t)]^2 = \boldsymbol{\theta}^t - \alpha[V^*(s) - \hat{V}^t(s, \boldsymbol{\theta}^t)]\nabla_{\boldsymbol{\theta}^t}\hat{V}^t(s, \boldsymbol{\theta}^t) \tag{5}$$

where α is a learning rate.

Unfortunately, in RL no sample of the optimal action-value function is actually available. Thus, at each time instant, the update rule derived from the gradient descent becomes

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \alpha[R(s, a) + \gamma V(s') - \hat{V}^t(s, \boldsymbol{\theta}^t)]\nabla_{\boldsymbol{\theta}^t}\hat{V}^t(s, \boldsymbol{\theta}^t) \tag{6}$$

While this update rule has been successfully used with many functions in some relevant applications [42,14], several studies [43–46] showed that it may lead the approximator to unpredictable results and, in some cases, to divergence. Generally, function approximation in problems with continuous state space requires long hand-tuning in order to achieve good solutions.

[1] For a detailed review about function approximation in RL we refer the reader to Chapter 2 from [40] and Chapter 5 from [41].

3.2. Fuzzy inference systems

A FIS is a set of rules, each of which infers fuzzy propositions (consequent) given the evidence of other fuzzy propositions (antecedent). In this paper we refer to the *Tagaki–Sugeno* FIS:

$$\begin{aligned}
&\mathcal{R}_1: \quad \textbf{IF } x \textit{ is } \mathcal{X}_1 \textbf{ THEN } y = p_1^{(n)}(x)\\
&\mathcal{R}_2: \quad \textbf{IF } x \textit{ is } \mathcal{X}_2 \textbf{ THEN } y = p_2^{(n)}(x)\\
&\vdots\\
&\mathcal{R}_N: \quad \textbf{IF} x \textit{ is } \mathcal{X}_N \textbf{ THEN } y = p_N^{(n)}(x)
\end{aligned} \tag{7}$$

where "x *is* $\mathcal{X}_i$" is a *proposition* with *fuzzy variable* x taking values in X and $\mathcal{X}_i$ modelled by a fuzzy set defined on the universe of discourse X by its membership function $\mu_{\mathcal{X}_i}(x)$.[2] The output function $p^{(n)}(x)$ is a polynomial function of x of order n defined by a vector of $M+1$ parameters $P^n = \{p_0, \ldots, p_M\}$: if $n = 0$, the output is a constant p_0, if $n = 1$, the output is a linear combination $p_0 + p_1x_1 + \cdots + p_Mx_M$, and so on. In the following, we will refer to the case with $n = 0$ that can be seen also as a particular (and quite common) Mamdani FIS, with singleton membership functions for the output fuzzy sets.

FISs can be used to define a function $\widehat{f}: \mathrm{X} \to \mathbb{R}$ mapping a N-dimensional input domain $\mathrm{X} = \mathrm{X}_1 \times \cdots \times \mathrm{X}_N$ to real numbers. In the following, we assume that a fuzzy partition is defined over X, so that each dimension X_k of the state space has n_{X_k} fuzzy sets defined on it. Thus, we call $\mu_{\mathcal{X}_{i_k}}$ the membership function of the i_kth fuzzy set $\mathcal{X}_{i_k}$ on the kth variable X_k, where $i_k = 1, \ldots, n_{\mathrm{X}_k}$. The fuzzy sets belonging to the partition are the premises of a FIS. The FIS is a black box in which crisp values $\boldsymbol{x}$ enter and from which crisp values $\hat{f}(\boldsymbol{x})$ come out.

In general, the rules that compose the FIS are in the form

$$\begin{aligned}
\mathcal{R}_{j_1,\ldots,j_N}: \quad &\textbf{IF } x_1 \textit{ is } \mathcal{X}_{j_1} \textbf{ AND } \ldots \textbf{ AND } x_N \textit{ is } \mathcal{X}_{j_N}\\
&\textbf{THEN } y = p^{(n)}_{j_1,\ldots,j_N}(\boldsymbol{x})
\end{aligned} \tag{8}$$

where $p^{(n)}_{j_1,\ldots,j_N}(\boldsymbol{x})$ is the polynomial function defined by the order n and the parameters $P^n_{j_1,\ldots,j_N}$ associated to the rule $\mathcal{R}_{j_1,\ldots,j_N}$.

The premises of the fuzzy rules give the "fuzzy coordinates" of a region in the state space. The rule $\mathcal{R}_{j_1,\ldots,j_N}$ covers the region of the intersection (with respect to a T-norm) of the j_1th membership function $\mu_{\chi_{j_1}}(x_1)$ defined on the first variable X_1, with the j_Nth membership function $\mu_{\mathcal{X}_{j_N}}(x_N)$ defined on the last variable X_N. When the crisp input $\boldsymbol{x} = \{x_1, \ldots, x_N\}$ is presented to the system, its matching degree with respect to all the fuzzy sets is computed and used to activate the corresponding rules. The premises of the active rules have the degree of truth given by the T-norm applied in $\boldsymbol{x}$. In particular, in this paper we adopt the product as T-norm and we obtain

$$\phi_{j_1,\ldots,j_N}(\boldsymbol{x}) = \prod_{i=1}^{N} \mu_{\mathcal{X}_{j_i}}(x_i) \tag{9}$$

Then the output of each rule is computed and defuzzified. The defuzzification is a mapping from the fuzzy output of the FIS to a real value, which is the final crisp output $\widehat{f}(\boldsymbol{x})$. In a Takagi–Sugeno FIS the steps of overall output generation and defuzzification are usually computed all at once: when the outputs of each rule are available, they are summed up and normalized by the sum of all $\phi_{j_1,\ldots,j_N}(\boldsymbol{x})$. Since, in each rule, the single output is obtained by scaling the consequent evaluated in $\boldsymbol{x}$, $p^{(n)}_{j_1,\ldots,j_N}(\boldsymbol{x})$, by the factor $\phi_{j_1,\ldots,j_N}(\boldsymbol{x})$, the overall computation of the final crisp output can be viewed as the mean of the consequents over the $\phi_{j_1,\ldots,j_N}(\boldsymbol{x})$:

$$\hat{f}(\boldsymbol{x}) = \frac{\sum_{j_1,\ldots,j_N} p^{(n)}_{j_1,\ldots,j_N}(\boldsymbol{x})\phi_{j_1,\ldots,j_N}(\boldsymbol{x})}{\sum_{j_1,\ldots,j_N} \phi_{j_1,\ldots,j_N}(\boldsymbol{x})} \tag{10}$$

[2] In general, each rule is described also by a membership function $\mu_{\mathcal{R}}(x, y)$ that gives the degree of *reliability* of the rule. In this paper, we consider the reliability to be a constant equal for all the rules.

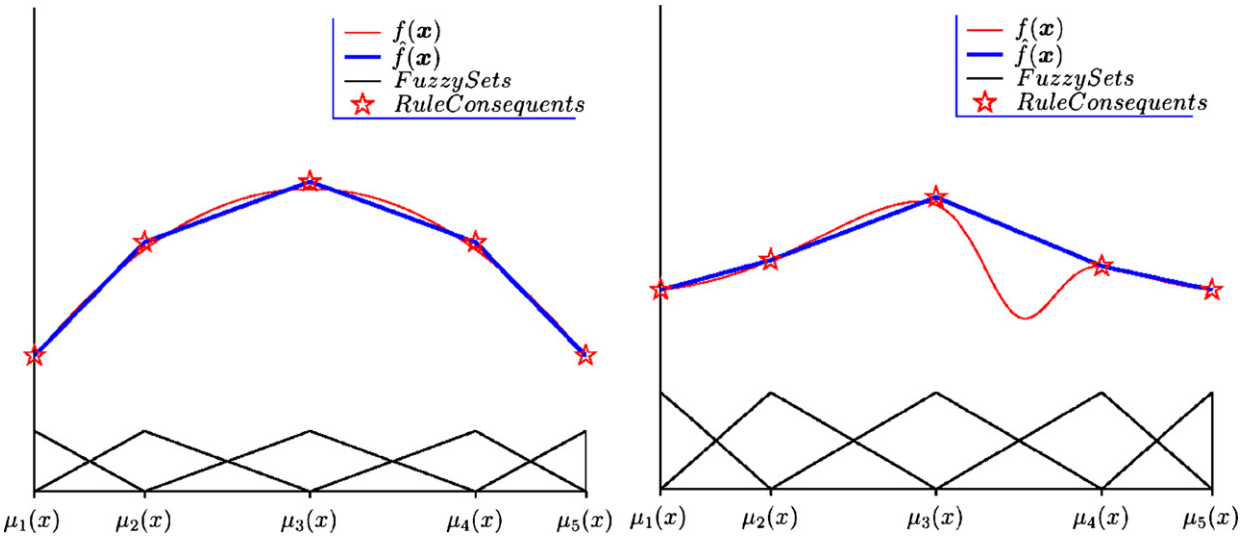

Fig. 1. Each fuzzy rule produces an output that is its matching degree multiplied by the value of its consequent. The final result is an interpolation between consequents. The kind of interpolation is bounded to the shape of membership functions and, depending on the function to be approximated, it could be more (left) or less reliable (right).

When $n = 0$, the expression of $\hat{f}(\boldsymbol{x})$ becomes

$$\hat{f}(\boldsymbol{x}) = \frac{\sum_{j_1,\ldots,j_N} p_{j_1,\ldots,j_N}[\prod_{i=1}^{N} \mu_{\mathcal{X}_{j_i}}(x_i)]}{\sum_{j_1,\ldots,j_N}[\prod_{i=1}^{N} \mu_{\mathcal{X}_{j_i}}(x_i)]} \tag{11}$$

Since the Takagi–Sugeno FISs can be used to represent a function, they can be profitably used to approximate a given function $f(\boldsymbol{x})$ by changing the values of the consequents. Tuning *appropriately* $P^n_{j_1,\ldots,j_N}$, the parameters of the different $p^{(n)}_{j_1,\ldots,j_N}(\boldsymbol{x})$, it is possible to manipulate, *within the limits imposed by the structure of the fuzzification of the domain*, the shape of the function $\hat{f}(\boldsymbol{x})$ so that it is a *good* approximation of $f(\boldsymbol{x})$. In this way, it could be possible to represent a function with a small number of parameters (Fig. 1-left). Conversely, since the computation is not point-wise but generalized over a region, the approximation can be rough (Fig. 1-right). According to the definition of MSE in Eq. (4), we can consider as the best approximation the one that minimizes

$$MSE(\boldsymbol{\theta}^t) = \sum_{s \in S} P(s)[f(\boldsymbol{x}) - \widehat{f}_{\boldsymbol{\theta}^t}(\boldsymbol{x})]^2 \tag{12}$$

where parameters $\boldsymbol{\theta}^t$ in this case are the values of the consequents.

The main issue in using FIS for function approximation in RL is about the process of minimization in order to tune the consequences of the rules. Many methods have been studied in the past, especially in the framework of supervised learning. In the following sections we will focus on the solutions that can be used to approximate the action-value function in RL problems through Takagi–Sugeno FIS.

4. Fuzzy Q-learning

In the rest of the paper we will refer to zero-order Takagi–Sugeno FISs described in the previous section. In the RL framework, the FIS is used to map the state, represented as an N-dimensional real-valued domain $\mathrm{X} = \mathrm{X}_1 \times \cdots \times \mathrm{X}_N$, to real-valued *actions* together with their estimated *action values*. In the following, we do not assume any particular choice for the fuzzy partition defined over X, since the algorithm works independently from it. However, its performance is affected by the *quality* of the particular structure, thus we will only assume that the partitioning is adequate to the problem.

On each dimension X_k of the state space n_{X_k} fuzzy sets are defined. Let $\mu_{\mathcal{X}_{j_k}}$ be the membership function of the j_kth fuzzy set $\mathcal{X}_{j_k}$ on the kth variable X_k, where $j_k = 1, \ldots, n_{X_k}$. Each rule is associated to a set of $n_{j_1,\ldots,j_N}$ possible discrete actions $A_{j_1,\ldots,j_N} = \{a_{j_1,\ldots,j_N,1}, \ldots, a_{j_1,\ldots,j_N,n_{j_1,\ldots,j_N}}\}$ and to the related action values $Q_{j_1,\ldots,j_N} = \{q_{j_1,\ldots,j_N,1}, \ldots, q_{j_1,\ldots,j_N,n_{j_1,\ldots,j_N}}\}$, so that $q_{j_1,\ldots,j_N,i}$ is associated to the ith discrete action available in the rule $\mathcal{R}_{j_1,\ldots,j_N}$. To simplify the notation, let us associate the set of indexes $\{j_1, \ldots, j_N\}$ of each rule to a numeric index $r = 1, \ldots, n_R$, where n_R is the number of rules. Thus, the rule $\mathcal{R}_{j_1,\ldots,j_N}$, the discrete actions $a_{j_1,\ldots,j_N,i}$, and the corresponding action value $q_{j_1,\ldots,j_N,i}$ simply become, respectively, $\mathcal{R}_r$, $a_{r,i}$ and $q_{r,i}$. The generic rule r may be written as follows:

$\mathcal{R}_r$: **IF** x_1 *is* $\mathcal{X}_{j_1}$ **AND** ... **AND** x_N *is* $\mathcal{X}_{j_N}$
THEN $y = a_{r,1}$ **with** $q_{r,1}$
OR
⋮
OR
$y = a_{r,n_r}$ **with** q_{r,n_r}.

where the T-norm used to implement the **AND** operator and to compute the truth degree of the rule premise is the product. At each time step, only one action for each active rule participates to the inference, while all the others remain inactive.

4.1. Activation degree

When a crisp input $\boldsymbol{x}^t$ enters the system at time t, all the rules $\mathcal{R}_r$ that cover a region to which $\boldsymbol{x}^t$ partially belongs are said to be partially active by a certain activation degree ϕ_r. As stated above, the rule $\mathcal{R}_r$ is defined by the intersection of fuzzy sets along each dimension $\mathcal{X}_{j_1}, \ldots, \mathcal{X}_{j_N}$ with truth degrees $\mu_{\mathcal{X}_{j_1}}(x_1), \ldots, \mu_{\mathcal{X}_{j_N}}(x_N)$, and the T-norm is implemented by the product. The activation degree of rule $\mathcal{R}_r$ is

$$\phi_r(\boldsymbol{x}^t) = \prod_{j_i \in r} \mu_{\mathcal{X}_{j_i}}(x_i) \tag{13}$$

4.2. Action selection

Since the state domain is covered by a number of fuzzy sets, a generic point $\boldsymbol{x}^t$ in the continuous state domain, belongs with some degree to some of the fuzzy sets. We can equivalently say that an agent being in $\boldsymbol{x}^t$ has fuzzy (partial) presence in some of the fuzzy states and activates the corresponding rules. For each fuzzy state the agent is partially in, it chooses a discrete action, that is, each active rule should infer a discrete action, and the overall executed action in correspondence of $\boldsymbol{x}^t$ is a composition of such discrete actions. The **OR** in the representation of the rule is to be intended as exclusive and it stands for a local competition operator among possible actions of the same rule, thus *only one* action per state is employed as effective consequent of the rule during the inference. As discussed in Section 3.1.2, in each fuzzy state the agent choice must face the exploration–exploitation dilemma. For each rule the expected reward of an action $a_{r,i}$ is its corresponding action value $q_{r,i}$. According to the ε-greedy exploration strategy, in each rule the agent selects the best action (i.e., the one with highest action value) with probability $1 - \varepsilon$ and a random action with probability ε. At the end of the action selection phase, for each active rule the agent has a winning action $\breve{a}_r^t$.

4.3. Continuous action computation

Once the degree of activation of the premises is computed and the action in each active fuzzy rule is chosen, the rules must cooperate with each other in order to choose the unique continuous action $A^t(\boldsymbol{x}^t)$ that will be actually executed in $\boldsymbol{x}^t$ at time t. The overall inference of the FIS is computed as the usual output of a Takagi–Sugeno

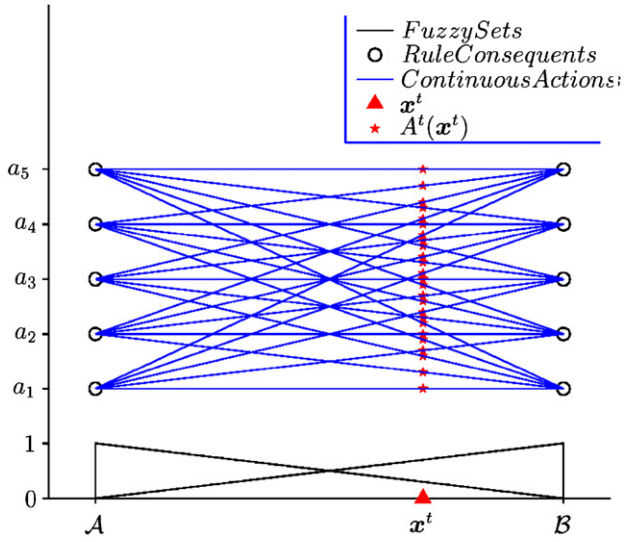

Fig. 2. All the possible combinations of the discrete actions of the rules for every $\boldsymbol{x}^t$ determine all the possible continuous actions. They are said continuous because they vary smoothly with $\boldsymbol{x}^t$, but for a given value of $\boldsymbol{x}^t$, there is a finite set of possible values of $A^t(\boldsymbol{x}^t)$.

FIS (see Section 3.2), by doing a sum of the winning actions of each rule $\check{a}_r^t$ weighted by corresponding activation degrees $\phi_r(\boldsymbol{x}^t)$:

$$A^t(\boldsymbol{x}^t) = \frac{\sum_{r=1}^{n_R} \phi_r(\boldsymbol{x}^t)\, \check{a}_r^t}{\sum_{r=1}^{n_R} \phi_r(\boldsymbol{x}^t)} \tag{14}$$

where only the terms corresponding to the active rules are significant. In order to simplify the notation, we define the normalized activation degree as

$$\psi_r(\boldsymbol{x}^t) = \frac{\phi_r(\boldsymbol{x}^t)}{\sum_{r=1}^{n_R} \phi_r(\boldsymbol{x}^t)} \tag{15}$$

Thus, Eq. (14) can be rewritten as

$$A^t(\boldsymbol{x}^t) = \sum_{r=1}^{n_R} \psi_r(\boldsymbol{x}^t)\check{a}_r^t \tag{16}$$

Actually $A^t(\boldsymbol{x}^t)$ does not depend only on $\boldsymbol{x}^t$, as explicitly stated, but it depends also on the set of the winning actions $\check{a}_r^t$ for each active rule, $\check{\boldsymbol{a}}^t$, so by $A^t(\boldsymbol{x}^t)$ we actually mean $A(\boldsymbol{x}^t, \check{\boldsymbol{a}}^t)$.

It is worth noting that $A^t(\boldsymbol{x}^t)$ is a *continuous action* in the sense that, given the winning discrete actions, a continuous variation of $\boldsymbol{x}^t$ implies a continuous and smooth variation of $A^t(\boldsymbol{x}^t)$. On the other hand, this does not mean that in a generic point $\boldsymbol{x}^t$ all the values of a continuous action are available; indeed, all the possible combinations of consequents of rules result only in a discrete set of possible actions, even if much richer than the sets of each rule (see Fig. 2). This leads to the consideration that an adequate distribution of the discrete actions in each rule is fundamental in order to have an adequate distribution of actions in all the points in the space.

4.4. Action value computation

While the computation of the continuous action is straightforward, the computation of its corresponding action value is more complicated, since it is not obvious the best way to interpret it. The more simple approach in the computation of the action values is to use a linear combination of the action values associated to the chosen actions. This is the approach followed by Jouffe in [25]. We call $\mathcal{Q}$ the continuous action value obtained with this technique.

On the other hand, we can observe that the value of the continuous action $A^t(\boldsymbol{x}^t)$ can be obtained by combinations of different discrete winning actions. According to the definition of $\mathcal{Q}$, the action $A^t(\boldsymbol{x}^t)$ inherits its action value by the particular set of discrete winning actions $\check{a}_r^t$ at time t. If we choose all the discrete winning action sets that are able to interpolate one of the intersection points cited above, then the resulting action $A^t(\boldsymbol{x}^t)$ is the same, but inferred action value $\mathcal{Q}(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t))$ is different depending on the choice of the winning action set. Thus we can associate as many $\mathcal{Q}$-values to $A^t(\boldsymbol{x}^t)$ as the number of combinations of discrete actions that can interpolate $A^t(\boldsymbol{x}^t)$. By drawing a parallel with traditional tabular Q-learning, this condition resembles the situation in which the Q-table has the same actions repeated several times.

To avoid this replication of information that may lead to longer learning times, we propose a different strategy that associates a unique action value to $A^t(\boldsymbol{x}^t)$. A way to achieve this result is binding every possible $A^t(\boldsymbol{x}^t)$ to a linear combination of a uniquely identifiable set of discrete actions and consequently computing the associated action value. A criterion for identifying uniquely a combination of discrete actions is choosing among the active rules the one proposing nearest action to $A^t(\boldsymbol{x}^t)$. In other words, as explained in Section 5.2, we identify the features as the counterpart of the *executed* action. We call $\mathfrak{Q}$ the continuous action value obtained with this technique.

5. Different action-value estimation criteria

In the previous section, we provided a preliminary interpretation for the inference of the action values followed by Jouffe, and then we introduced a different way to interpret the action value function $\mathcal{Q}(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t))$. In the following, we describe both methods in deeper detail, and we provide a comparison between them under a number of different points of view.

5.1. Definition of $\mathcal{Q}$-values

The $\mathcal{Q}$-function is defined on the basis of the values associated to the actions $\check{\boldsymbol{a}}_r^t$ chosen at time t in the active rules. At each time instant t, given the current state of the environment $\boldsymbol{x}^t$, the FIS evaluates which rules are active, that is, which fuzzy sets cover the point $\boldsymbol{x}^t$, and then, for each active rule $\mathcal{R}_r$, a winning action is determined on the basis of the ε-greedy strategy. If $\check{a}_r^t$ is the winning action determined by rule $\mathcal{R}_r$, we use $\check{q}_r^t$ to refer to its associated $\mathcal{Q}$-value. The continuous action $A^t(\boldsymbol{x}^t)$ is obtained by a cooperation of all the active rules; it is computed as the sum of their winning actions weighted by the activation values of the respective rules, as shown in Eq. (14). The $\mathcal{Q}$-value associated to $A^t(\boldsymbol{x}^t)$ in the state $\boldsymbol{x}^t$ at time t, namely $\mathcal{Q}^t(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t))$, is computed in the same way

$$\mathcal{Q}^t(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t)) = \sum_{r=1}^{n_R} \psi_r(\boldsymbol{x}^t)\check{q}_r^t \tag{17}$$

The continuous action $A^t(\boldsymbol{x}^t)$ might be very different from the discrete actions $\check{a}_r^t$ that it has been generated from. Despite of this situation, $A^t(\boldsymbol{x}^t)$ inherits its $\mathcal{Q}$-value from the action values $\check{q}_r^t$ associated to the winning actions $\check{a}_r^t$. Furthermore, a generic point $(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t))$ belongs to the state–action space, but not all the points in the state–action space can be expressed by $(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t))$. Indeed, $A^t(\boldsymbol{x}^t)$ is a restriction of the action space to a discrete set of actions in correspondence of $\boldsymbol{x}^t$. Therefore, the value of $\mathcal{Q}$ is defined only on a subspace of the state–action space. Nevertheless, $\mathcal{Q}(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t))$ is an explicit function of $\boldsymbol{x}^t$ and $A^t(\boldsymbol{x}^t)$ (and, implicitly, of the structure of the FIS), but above all it is an implicit function of the values associated to the winning discrete actions $\check{a}_r^t$, so $\mathcal{Q}$ actually should be written as $\mathcal{Q}(\boldsymbol{x}^t, \check{\boldsymbol{q}}^t)$, where $\check{\boldsymbol{q}}^t$ is the set of action values associated to the winning action at time t. Thus, the point $(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t))$ in the state-action space is not associated to any $\mathcal{Q}$-value *per se*, as in the classic definition of the action-value function. A graphical description of the way $\mathcal{Q}$ values are computed is shown in Fig. 3.

The way $\mathcal{Q}$ values are computed is similar to connectionist approaches, such as neural networks, in which the information about a function is distributed throughout a set of simple, individually meaningless, parameters. The action value associated to each action of each rule $(q_{r,i}^t)$ does not have a meaning *per se*, it provides a piece of information that is complete only relatively to the *combination* with the other winning actions. Thus, its corresponding action value gives a measure of the goodness of such a combination. As a result, the meaning of the individual $q_{r,i}^t$ is not the same

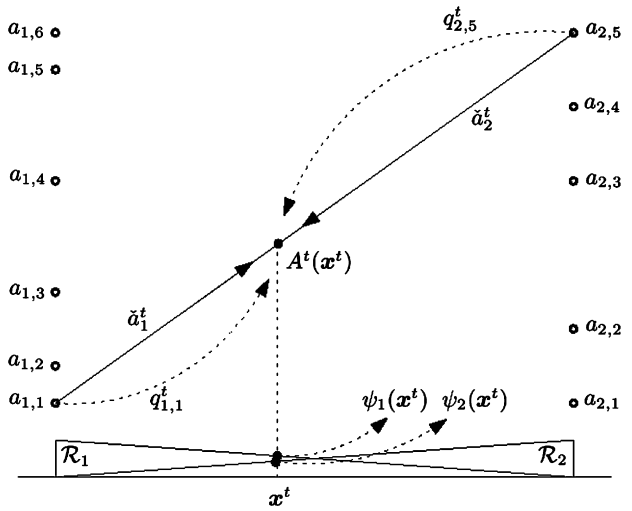

Fig. 3. The computation of $\mathcal{Q}$-values. Once the winning actions (and the corresponding $\mathcal{Q}$-values) are chosen, the continuous action $A^t(\boldsymbol{x}^t)$ and its $\mathcal{Q}$-value are computed by means of the same linear combination: $A^t(\boldsymbol{x}^t) = \psi_1(\boldsymbol{x}^t)a_{1,1} + \psi_2(\boldsymbol{x}^t)a_{2,5}$ and $\mathcal{Q}(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t)) = \psi_1(\boldsymbol{x}^t)q_{1,1} + \psi_2(\boldsymbol{x}^t)q_{2,5}$.

of the classic action value. In fact, as discussed in Section 5.3, they give a measure of the goodness of choosing the corresponding actions as winners in the local competition.

5.2. Definition of the $\mathfrak{Q}$ function

The $\mathfrak{Q}$ function in the point $(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t))$ is computed on the basis of the values associated to the actions that belong to the sets of consequents of the active rules nearest to $A^t(\boldsymbol{x}^t)$. When a continuous action $A^t(\boldsymbol{x}^t)$ is computed, we want to univocally associate to it a $\mathfrak{Q}$ value. Since the information about the action values resides in the $\mathfrak{Q}$ values of the discrete actions, we must find a way to associate $A^t(\boldsymbol{x}^t)$ univocally to a set of $a_{r,i}$ and then use their values to derive $\mathfrak{Q}(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t))$. To do this, we need to generalize the FIS in order to deal with continuous input of the state–action space. We *strongly* fuzzify the action dimension in each rule $\mathcal{R}_r$ by one fuzzy set $\mathcal{A}_{r,i}$ centered in each discrete action $a_{r,i}$. The fuzzification of the action dimension is specific for each rule, but it is still strong, so a general action a^t is covered by *only and always* two fuzzy sets. Thus, for *each rule* $\mathcal{R}_r$, we can find a unique pair of actions that can express $A^t(\boldsymbol{x}^t)$ (or a generic action a^t), and from which it can inherit its $\mathfrak{Q}$-value by means of the following simple FIS:

$$\widetilde{\mathcal{R}}_{r,1}: \textbf{IF } a \textit{ is } \mathcal{A}_{r,1} \textbf{ THEN } \mathfrak{Q} = q_{r,1}$$
$$\vdots$$
$$\widetilde{\mathcal{R}}_{r,n_r}: \textbf{IF } a \textit{ is } \mathcal{A}_{r,n_r} \textbf{ THEN } \mathfrak{Q} = q_{r,n_r}$$

where n_r is the cardinality of the discrete action set associated to the rule $\mathcal{R}_r$. Note that since the action dimension is strongly fuzzified there is no need to normalize the expression above. The maintenance of this is very simple, since the vector of centers of each fuzzy set is already available in the set of the discrete actions, and all we should do is interpolating $\widetilde{\mathfrak{Q}}_r(a^t)$ (the $\mathfrak{Q}$-value of a^t in rule $\mathcal{R}_r$), with the two appropriate $q_{r,i}$ weighted on corresponding activation degrees $\mu_{\mathcal{A}_{r,i}}(a^t)$:

$$\widetilde{\mathfrak{Q}}_r(a^t) = \sum_{i=1}^{n_r} \mu_{\mathcal{A}_{r,i}}(a^t)\, q^t_{r,i} \tag{18}$$

The final $\mathfrak{Q}$-value of the point $(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t))$, or a generic pair $(\boldsymbol{x}^t, a^t)$, is computed as in Eq. (17) using $\widetilde{\mathfrak{Q}}_r(a^t)$ instead of $q^t_{r,i}$. Let a^t be the winning action of each rule, and $\widetilde{\mathfrak{Q}}_r(a^t)$ its value for rule $\mathcal{R}_r$, the $\mathfrak{Q}$-value is defined as

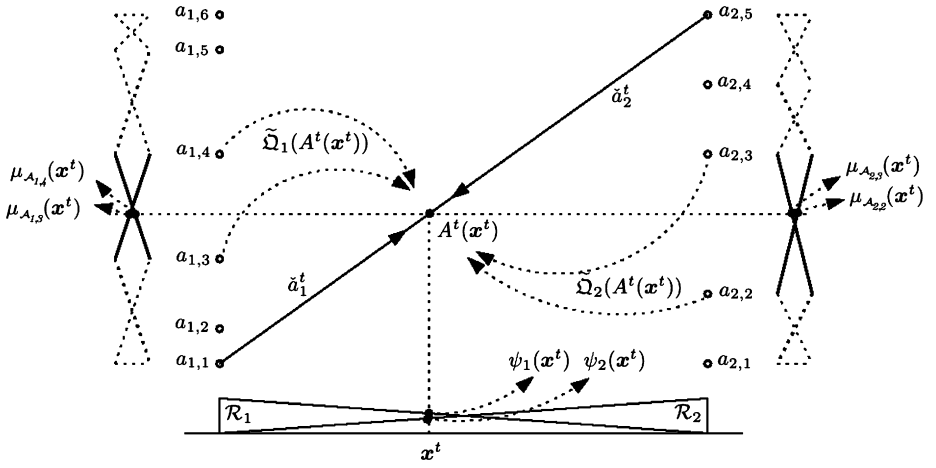

Fig. 4. The computation of $\mathfrak{Q}$-values. Once determined $A^t(\boldsymbol{x}^t)$, each rule $\mathcal{R}_r$ creates a $\widetilde{\mathfrak{Q}}_r(A^t(\boldsymbol{x}^t))$ with the secondary FIS: $\widetilde{\mathfrak{Q}}_1(A^t(\boldsymbol{x}^t)) = \mu_{\mathcal{A}_{1,3}}(\boldsymbol{x}^t)q_{1,3} + \mu_{\mathcal{A}_{1,4}}(\boldsymbol{x}^t)q_{1,4}$ and $\widetilde{\mathfrak{Q}}_2(A^t(\boldsymbol{x}^t)) = \mu_{\mathcal{A}_{2,2}}(\boldsymbol{x}^t)q_{2,2} + \mu_{\mathcal{A}_{2,3}}(\boldsymbol{x}^t)q_{2,3}$. Then these values are used to compute the $\mathfrak{Q}$-value of $A^t(\boldsymbol{x}^t)$ through their linear combination with parameters $\psi_r(\boldsymbol{x}^t)$: $\mathfrak{Q}(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t)) = \psi_1(\boldsymbol{x}^t)\widetilde{\mathfrak{Q}}_1(A^t(\boldsymbol{x}^t)) + \psi_2(\boldsymbol{x}^t)\widetilde{\mathfrak{Q}}_2(A^t(\boldsymbol{x}^t)) = \psi_1(\boldsymbol{x}^t)\mu_{\mathcal{A}_{1,3}}(\boldsymbol{x}^t)q_{1,3} + \psi_1(\boldsymbol{x}^t)\mu_{\mathcal{A}_{1,4}}(\boldsymbol{x}^t)q_{1,4} + \psi_2(\boldsymbol{x}^t)\mu_{\mathcal{A}_{2,2}}(\boldsymbol{x}^t)q_{2,2} + \psi_2(\boldsymbol{x}^t)\mu_{\mathcal{A}_{2,3}}(\boldsymbol{x}^t)q_{2,3}$.

$$\mathfrak{Q}^t(\boldsymbol{x}^t, a^t) = \sum_{r=1}^{n_R} \psi_r(\boldsymbol{x}^t)\widetilde{\mathfrak{Q}}_r(a^t) \tag{19}$$

A graphical description of the computation of $\mathfrak{Q}$ values is shown in Fig. 4.

A clear advantage of the $\mathfrak{Q}$-function with respect to the $\mathcal{Q}$-function is the ability of finding a univocal $\mathfrak{Q}$-value for any possible input $(\boldsymbol{x}^t, a^t)$. Furthermore, the definition of the $\mathfrak{Q}$-function respects the *principle of locality* between the effectively *executed* action a^t and the origin of its $\mathfrak{Q}$-value, since it is created by the $\mathfrak{Q}$-values of the actions in the discrete sets available in each active rule that are the nearest to a^t.

5.3. $\mathcal{Q}$ updates

As described in Section 2.1, the update of the parameter in algorithms with function approximators is often implemented as a gradient descent method with respect to the mean square error (MSE). The parameters are updated in order to produce the steepest reduction of the MSE evaluated in the last real-valued state $\boldsymbol{x}^t$ and this leads to the Eq. (5) for the value function. A similar equation can be obtained for the action-value function, by the following operations:

- substitute $\mathcal{Q}$ to V,
- since the features are represented by the fuzzy antecedents of the rules instantiated with their winning action $\breve{a}_r^t$, the parameters θ^t are the $\mathcal{Q}$-values $q_{r,i}^t$ associated to $a_{r,i}^t$, given that $a_{r,i}^t$ is the winning action $\breve{a}_r^t$:

  $$\boldsymbol{\theta}^t \longleftrightarrow q_{r,i}^t$$

  the sample of $\mathcal{Q}$ at time t is replaced by its estimate given the values at time $t+1$:

  $$q^t(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t)) \longleftrightarrow r_{\boldsymbol{x}^t, A^t(\boldsymbol{x}^t)}^t + \gamma \max_a \mathcal{Q}(\boldsymbol{x}^{t+1}, a)$$

so, the resulting equation is (Fig. 5):

$$q_{r,i}^{t+1} = q_{r,i}^t - \alpha\varepsilon_{\mathcal{Q}}^t \nabla_{q_{r,i}^t} \mathcal{Q}^t(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t)) \tag{20}$$

where $\varepsilon_{\mathcal{Q}}^t$ is defined as

$$\varepsilon_{\mathcal{Q}}^t = r_{\boldsymbol{x}^t, A^t(\boldsymbol{x}^t)}^t + \gamma \max_a \mathcal{Q}(\boldsymbol{x}^{t+1}, a) - \mathcal{Q}^t(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t)) \tag{21}$$

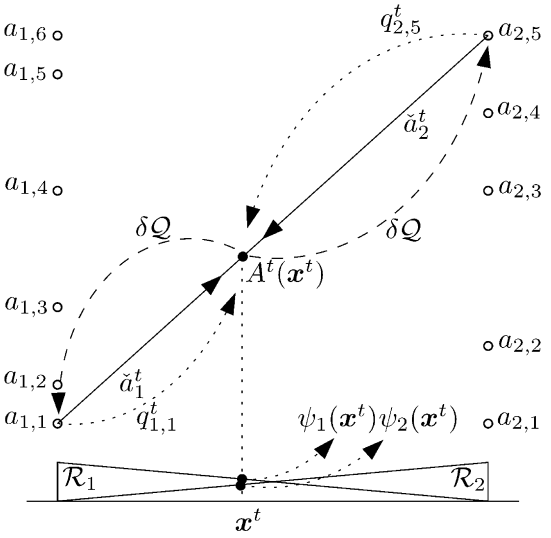

Fig. 5. The update of the features corresponding to $\mathcal{Q}$-values. The reinforcement is redistributed to the parameters corresponding to the chosen actions, weighted by the activation degree of the corresponding rule $\psi_r(\boldsymbol{x}^t)$.

In order to solve the gradient of $\mathcal{Q}$ with respect to $q^t_{r,i}$, it is enough to look at Eq. (17) and we immediately see that

$$\nabla_{q^t_{r,i}} \mathcal{Q}^t(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t)) = \nabla_{q^t_{r,i}} \left[\sum_{r=1}^{n_R} \psi_r(\boldsymbol{x}^t) q^t_{r,i} \right] = \begin{cases} \psi_r(\boldsymbol{x}^t) & \text{if } a^t_{r,i} = \check{a}^t_r \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

so the update rule becomes the following:

$$q^{t+1}_{r,i} = q^t_{r,i} - \alpha \varepsilon^t_{\mathcal{Q}} \psi_r(\boldsymbol{x}^t) \tag{23}$$

where we remind that $q^t_{r,i}$ are the $\mathcal{Q}$-values associated to the chosen actions.

It is worth noting that $\mathcal{Q}(\boldsymbol{x}^{t+1}, a)$ is the $\mathcal{Q}$-value achievable in the next state $\boldsymbol{x}^{t+1}$ associated to an action in the set of all the possible combinations of the discrete actions available in any active rule (see Fig. 2). The greatest $\mathcal{Q}$-value among them is the combination of all the greatest $q^t_{r,i}$ in each active rule $\mathcal{R}_r$:

$$\max_a \mathcal{Q}(\boldsymbol{x}^{t+1}, a) = \sum_{r=1}^{n_R} \psi_r(\boldsymbol{x}^{t+1}) \max_i \{q^t_{r,i}\} \tag{24}$$

and it is used as estimation of the highest expected sum of rewards from state $\boldsymbol{x}^{t+1}$ following the greedy policy.

5.4. $\mathfrak{Q}$ updates

This section describes the update mechanism for the $\mathfrak{Q}$-values, that is defined according to the actions $\check{a}^t_r$ chosen at time t in the active rules. The general considerations of the previous section are still valid, so we refer to Section 2.1 and to Eq. (5), and we obtain the following operations:

- substitute $\mathfrak{Q}$ to V,
- since the features are represented by the fuzzy antecedent of the rules instantiated in the discrete actions that are the nearest to the action actually executed, the parameters θ^t are the $\mathfrak{Q}$-values $q^t_{r,i}$ associated to $a^t_{r,i}$, given that $a^t_{r,i}$ contributes to generate $\widetilde{\mathfrak{Q}}_r(a^t)$ (see Section 5.2):

  $\boldsymbol{\theta}^t \longleftrightarrow q^t_{r,i}$

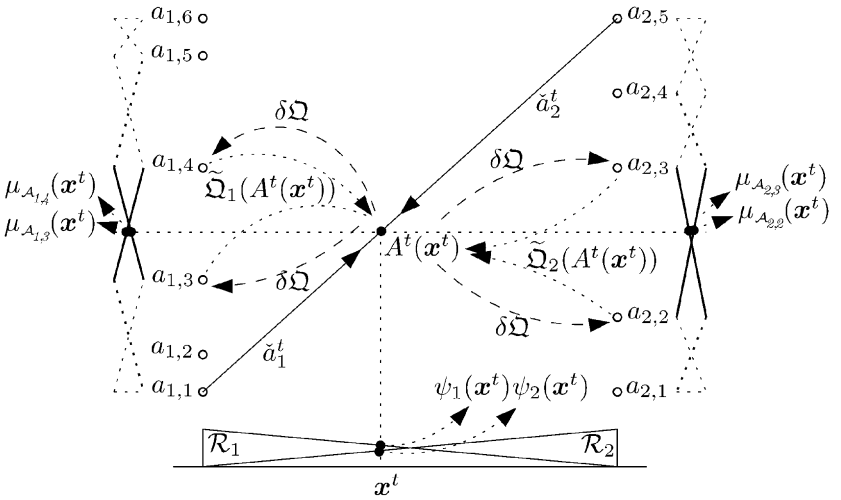

Fig. 6. The update of the features corresponding to $\mathfrak{Q}$-values. The reinforcement is redistributed to the parameters corresponding to those actions whose membership functions are activated by $A^t(\boldsymbol{x}^t)$. The reinforcement assigned to the parameter $q_{r,i}$ is weighted by the activation degrees of rule $\mathcal{R}_r$ and by the activation degrees of $\mathcal{A}_{r,i}$: $\psi_r(\boldsymbol{x}^t)\mu_{\mathcal{A}_{r,i}}(\boldsymbol{x}^t)$.

- the sample of $\mathfrak{Q}$ at time t is replaced by its estimate given the values at time $t+1$:

$$q^t(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t)) \longleftrightarrow r^t_{\boldsymbol{x}^t, A^t(\boldsymbol{x}^t)} + \gamma \max_a \mathfrak{Q}(\boldsymbol{x}^{t+1}, a)$$

although it makes the update rule not a real gradient it is useful to look at it as a gradient anyway [5],

so, the resulting equation is (Fig. 6):

$$q^{t+1}_{r,i} = q^t_{r,i} - \alpha \varepsilon^t_{\mathfrak{Q}} \nabla_{q^t_{r,i}} \mathfrak{Q}^t(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t)) \tag{25}$$

where $\varepsilon^t_{\mathfrak{Q}}$ is defined like in Eq. (21) as

$$\varepsilon^t_{\mathfrak{Q}} = r^t_{\boldsymbol{x}^t, A^t(\boldsymbol{x}^t)} + \gamma \max_a \mathfrak{Q}(\boldsymbol{x}^{t+1}, a) - \mathfrak{Q}^t(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t)) \tag{26}$$

In order to compute the gradient of $\mathfrak{Q}^t(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t))$, we consider Eq. (19):

$$\nabla_{q^t_{r,i}} \mathfrak{Q}^t(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t)) = \nabla_{q^t_{r,i}} \sum_{r=1}^{n_R} \sum_{i=1}^{n_r} \psi_r(\boldsymbol{x}^t)\mu_{\mathcal{A}_{r,i}}(A^t(\boldsymbol{x}^t))q^t_{r,i} = \psi_r(\boldsymbol{x}^t)\mu_{\mathcal{A}_{r,i}}(A^t(\boldsymbol{x}^t)) \tag{27}$$

so the update rule becomes the following:

$$q^{t+1}_{r,i} = q^t_{r,i} - \alpha \varepsilon^t_{\mathfrak{Q}} \psi_r(\boldsymbol{x}^t)\mu_{\mathcal{A}_{r,i}}(A^t(\boldsymbol{x}^t)) \tag{28}$$

where $q^t_{r,i}$ are the $\mathfrak{Q}$-values associated to the actions closest to the executed one.

It is worth noting that $\mathfrak{Q}(\boldsymbol{x}^{t+1}, a)$ is the $\mathfrak{Q}$-value achievable in the next state $\boldsymbol{x}^{t+1}$ and lies along the restriction of the $\mathfrak{Q}$ function over the straight line $(\boldsymbol{x}^{t+1}, a)$, where a is variable, so that the greatest $\mathfrak{Q}$ value is

$$\max_a \mathfrak{Q}(\boldsymbol{x}^{t+1}, a) = \max_a \left\{ \sum_{r=1}^{n_R} \sum_{i=1}^{n_r} \psi_r(\boldsymbol{x}^{t+1})\mu_{\mathcal{A}_{r,i}}(a)q^t_{r,i} \right\} \tag{29}$$

If the fuzzification along the action dimension is achieved though triangular fuzzy sets, it can be shown that the greatest $\mathfrak{Q}$-value lies in correspondence of one of the discrete action sets A_r of the active rules $\mathcal{R}_r$:

$$\max_a \mathfrak{Q}(\boldsymbol{x}^{t+1}, a) = \max_{a \in \bigcup_{i=1}^{n_R} A_i} \left\{ \sum_{r=1}^{n_R} \sum_{i=1}^{n_r} \psi_r(\boldsymbol{x}^{t+1})\mu_{\mathcal{A}_{r,i}}(a)q^t_{r,i} \right\} = \sum_{r=1}^{n_R} \psi_r(\boldsymbol{x}^{t+1}) \max_i \{q^t_{r,i}\}, \tag{30}$$

that is the same result obtained for the $\mathcal{Q}$-values (see Eq. (24)).

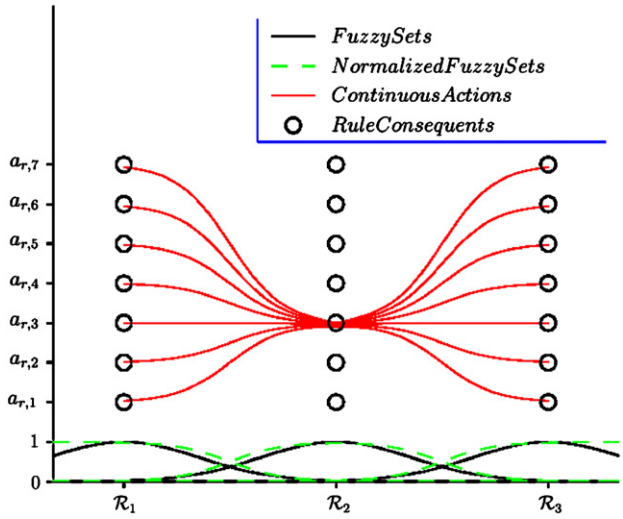

Fig. 7. The continuous actions in red and all the $q_{1,i}$ and $q_{3,i}$ influence the updates of the discrete action $a_{2,3}$.

6. Comparison between $\mathcal{Q}$ and $\mathfrak{Q}$ values

In the previous section we have described two different approaches to the computation of the action values in the framework of fuzzy Q-learning. The major distinction between these two approaches is due to the particular semantic of the action-values used for the linear approximation of the action-value function: the former focuses the computation and the update on the action to choose, the latter focuses on the actions that are actually performed by the FIS.

The main advantages and disadvantages of both can be summarized as follows:

- $\mathcal{Q}$ puts in evidence only the value of the discrete actions available by the FIS without dispersing the reward information, so that the policy can be more easily identified. However, its definition is restricted to optimal policy function and it does not reveal the action value of the other actions. Furthermore, the value associated to the actions of each rule does not represent how good is to execute that action when the rule is active, but it is strictly dependent from the goodness of combining that action with those selected by all the other rules.
- $\mathfrak{Q}$ gives a more natural view of the classic action-value function. Its values are stable, coherent, and defined over the whole domain. It gives an idea of distribution throughout the space of the values of the actions so we can make comparisons between optimal and suboptimal actions, and it allows to understand if the fuzzification is inadequate.

6.1. Coherence of updates

The $q^t_{r,i}$ are updated in a very different way in the two FISs. Let us focus on the ith discrete action of the rth rule $\mathcal{R}_r$ at time t, $a^t_{r,i}$. Using a FIS based on $\mathcal{Q}$-values, the action $a^t_{r,i}$ is updated every time it is chosen as the winning action of its rule $\breve{a}^t_r$, and the value of its update *strongly depends* on the other winning actions it is combined with, as shown in Fig. 5. The $\mathcal{Q}$-value $q^t_{r,i}$ associated to each action expresses the goodness of choosing $a^t_{r,i}$ as the winning action in rule $\mathcal{R}_r$, given the winning actions of all the other active rules. Thus, the value depends on the quality of the action generated by $a^t_{r,i}$ in the *overall cooperation* with *all* the discrete actions available in the neighboring rules; this is an extremely variable measure. Fig. 7 shows an example that puts in evidence all the continuous actions and all the discrete actions that influence the update of the action $a_{2,3}$: the continuous actions lie throughout the whole action space, so their quality may differ a lot from each other, consequently, action $a_{2,3}$ is susceptible to be affected in its updates by this big mass of heterogeneous information characterized by a high variance. As a result, the variations on the values $q^t_{r,i}$ give origin to oscillation around a certain value that is a mean of the heterogeneous amount of updating information weighted by the activation degrees of the proposing rules.

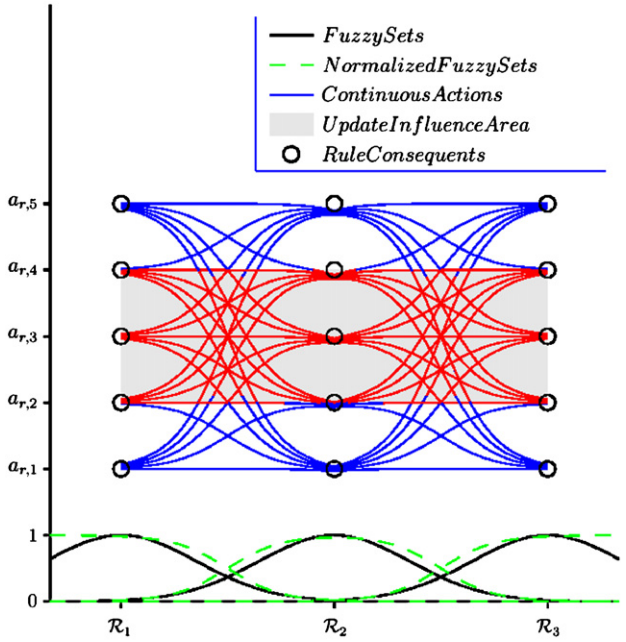

Fig. 8. All the continuous actions in the gray area are responsible for the updates of $a^t_{2,v}$ since they make its fuzzy set active. All these actions are similar and provoke the data set used for the updates to be more homogeneous.

On the other hand, in case of a FIS that computes the $\mathcal{Q}$-values, the action $a^t_{r,i}$ is updated every time that an executed action $A^t(\boldsymbol{x}^t)$ lies in the region covered by the intersection of the fuzzy set in the antecedent of $\mathcal{R}_r$ and the fuzzy set $\mu_{\mathcal{A}_{r,i}}$ associated to $a_{r,i}$, by construction (see Section 5.2 and Fig. 6). In Fig. 8 the attention is focused on the update of the discrete action $a_{2,3}$: we can see the region mentioned above, all the continuous actions that may be generated, and those lying in such region. The latter actions are characterized by being similar to each other.

Note that this similarity does not depend on the winning actions at all, since many combinations of them produce similar or equal continuous actions $A^t(\boldsymbol{x}^t)$. This is expressed by the *principle of locality* that introduced in Section 5.2. The locality refers to the proximity of the *executed* action $A^t(\boldsymbol{x}^t)$ to each other, and, above all, to the proximity of $A^t(\boldsymbol{x}^t)$ to the discrete actions in each active rule from which it inherits its $\mathcal{Q}$-value. Thanks to the principle of locality, the data set originated by the $q^t_{r,i}$ at the vertexes of the region and from the target received due to the executions of $A^t(\boldsymbol{x}^t)$, is, in general, more homogeneous, coherent and with smaller variance than in the case of $\mathcal{Q}$-values. It is worth noting that the update strength is not homogeneous throughout the region. In fact, the involvement of $q^t_{r,i}$ decreases the farther $A^t(\boldsymbol{x}^t)$ is from $a^t_{r,i}$, thus strengthening the principle of locality. This allows the updates to have less oscillations and, consequently, the reached values are more robust during the learning process under *any* policy (even random), and are more close to the optimal $\mathcal{Q}$-value throughout the whole region.

6.2. Influence of the structure of the FIS

Let us investigate the influence of the structure of the FIS on the performance of the two reward distribution approaches. In the former, the $\mathcal{Q}$-values are not bound spatially to the executed actions $A^t(\boldsymbol{x}^t)$, but they are inherited by the discrete actions that contributed to the interpolation of $A^t(\boldsymbol{x}^t)$. Thus, the optimal action function $A^*(\boldsymbol{x}^t)$ is approximated by the set of discrete actions $a^t_{r,i}$ associated to the greatest $q^t_{r,i}$ in each rule $\mathcal{R}_r$, and inherits from them its optimal value. The main limitation is due to the fact that all the action-value functions that the FIS can interpolate are a finite discrete set, i.e., all the possible choices of the set of all the winning actions $\breve{\boldsymbol{a}}^t$. Furthermore, given $\breve{\boldsymbol{a}}^t$, the shapes of these functions over the regions where a rule fades into the other depend on the shape of the fuzzy set employed in the FIS (see, for instance, the difference between the continuous actions in Fig. 2 and Fig. 7). The choice of the shapes of the fuzzy sets and the sets of discrete actions is therefore fundamental. On the other hand, there are not further limitations on the action-value function $\mathcal{Q}$. It is just defined over the restriction $(\boldsymbol{x}^t, A^t(\boldsymbol{x}^t))$ of the state–action

space, so its value is derived from the values associated to $\breve{\boldsymbol{a}}^t$ interpolated *throughout the action function*. It follows that the dependence of $\mathcal{Q}$ from the action dimension is not fixed a priori, rather it is determined only once the set $\breve{\boldsymbol{a}}^t$ is defined.

Conversely, the $\mathfrak{Q}$ values are computed according to the principle of locality, and the executed action is considered as an input value of the action dimension. The advantages of this approach are exposed in Section 5.2, while the disadvantages are very similar to those exposed above. Indeed, we discussed about the problems of fitting adequately the optimal action function $A^*(\boldsymbol{x}^t)$ with an interpolated one, due to the particular choices of the shapes of the fuzzy sets and the discrete actions sets employed in every rule: similar problems are still present in this case. Let us assume that a set of winning actions is able to interpolate perfectly the optimal action function $A^*(\boldsymbol{x}^t)$; it is reasonable to think that its action value is large along this continuous action region and decreases the more we get farther from it. When the FIS evolves, it distributes the $\mathfrak{Q}$ value of each point of $A^*(\boldsymbol{x}^t)$ to the action immediately greater and the one immediately less than $A^*(\boldsymbol{x}^t)$ in the discrete set of each rule (see Fig. 6). Thus, the updates are spread throughout the projection of $A^*(\boldsymbol{x}^t)$[3] to the action dimension of each rule $\mathcal{R}_r$. Such a dispersion makes the difference of the quality of actions in each state less sharp, and, as a consequence, makes this approach less reliable for the identification of the optimal policy. Fig. 9 shows an example of this condition.

In case of $\mathcal{Q}$ approach, the action values $q^t_{r,i}$ of discrete actions are simply not affected by what happens outside to the action-value function defined by the current winning action set. In particular, under the greedy policy, only optimal actions get optimal rewards, in a sharp distinction with respect to suboptimal actions. The limits of $\mathcal{Q}$ distribution are that the value of the estimated optimal action-value function depends on how accurately it fits the real optimal action function, and this fitness depends on the distribution of discrete actions of each rule and on the shapes of fuzzy sets.

In the case of a FIS that computes the $\mathfrak{Q}$-values, the action values $q^t_{r,i}$ of the discrete actions are influenced by what happens throughout a well defined region around $a_{r,i}$ and generalize over all the targets received in such region. As a result, if such region gets heterogeneous targets, an optimal action may have a $\mathfrak{Q}$-value $q^t_{r,i}$ less than the $\mathfrak{Q}$-value of a suboptimal action which lies on a more homogeneous region. Since the $\mathcal{Q}$ distribution does not give such a broad generalization over an a priori fixed region, it does not suffer of this problem at all.

7. Experiments

The main objective of the experiments presented in this section is to analyze pros and cons of the different approaches for reinforcement distribution, in order to determine the situations in which the use of the first system, or the latter, is preferable.

The shape of the fuzzy sets used for the input is triangular and the fuzzification is strong. The type of fuzzification is defined as $\{x_{min} : step : x_{max}\}$, meaning that the interval $[x_{min}, x_{max}]$ is divided in sub-intervals wide *step*. All the experiments use a discount factor $\gamma = 0.8$.

As testbed we considered an abstraction of a golf game, in which the agent should give an optimal velocity v^0 to the ball in order to put it in the hole, given the initial distance from the hole x^0. Any other information such as the direction of the shot, the wind and so on, are assumed to be irrelevant for this testbed. For each distance x^0, the ball falls in the hole if the initial velocity of the ball v^0 is in the range of values $[v^0_{min}(x^0), v^0_{max}(x^0)]$, where $v^0_{max}(x^0) = \sqrt{x^0 2gk}$ and $v^0_{min}(x^0) = \sqrt{x^0 2gk + {v_{max}}^2}$, where g is the universal constant of gravity, k is the coefficient of friction between the ball and the ground, and v_{max} is the maximum velocity allowed at the border of the hole in order to make the ball to enter the hole and not to overcome it. When the ball enters the hole the episode ends. If $v^0 > v^0_{max}(x^0)$, the ball is assumed to be lost and the episode ends. Finally, if $v^0 < v^0_{min}(x^0)$ the episode goes on and the agent can try another hit. At each time instant t, the ball is at position x^t and after the shot the agent senses the relative reward r^t and next state x^{t+1} as

- if $v^0_{min}(x^0) < v^0 < v^0_{max}(x^0)$, then

$$\begin{cases} r^t = 10 \\ x^{t+1} \text{ random position in X} \end{cases}$$

[3] With $\boldsymbol{x}^t$ limited to the region covered by the rule $\mathcal{R}_r$ taken in consideration.

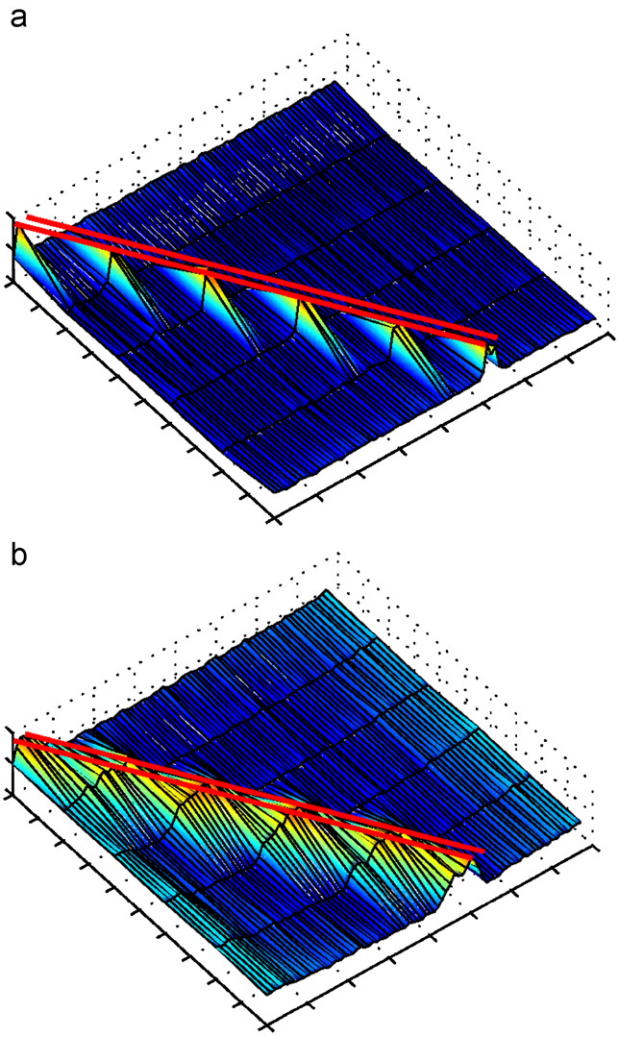

Fig. 9. The red lines in the pictures delimit a region of optimal actions. During the evolution of the FISs, the distribution of $\mathcal{Q}$ focuses on the discrete action to choose, and the value of the optimal action function is not represented explicitly, it is the interpolation of the peaks. The distribution of $\mathfrak{Q}$ disperses the values along the action dimension: it is less clear which is the optimal action in each rule, on the other hand we have an approximation of all the action value function: (a) $\mathcal{Q}$ values and (b) $\mathfrak{Q}$ values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- if $v^0_{min}(x^0) > v^0$, then[4]

$$\begin{cases} r^t = -2 \\ x^{t+1} = -\dfrac{(v^0)^2}{2gk} + x^t \end{cases}$$

- if $v^0 > v^0_{max}(x^0)$, then

$$\begin{cases} r^t = -5 \\ x^{t+1} \text{ random position in X} \end{cases}$$

[4] The expression of x^{t+1} is the simplification of the deceleration formula due to the friction $x^{t+1} = v^0 T_f + (gk/2){T_f}^2 + x^0$, where $T_f = v^0/gk$ is the instant in which the ball stops.

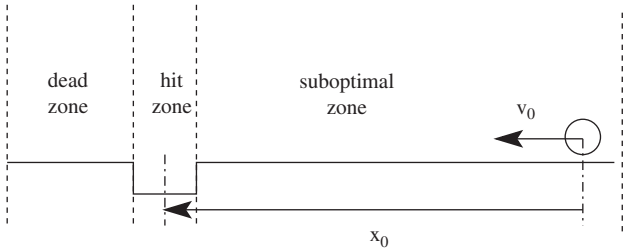

Fig. 10. The golf game.

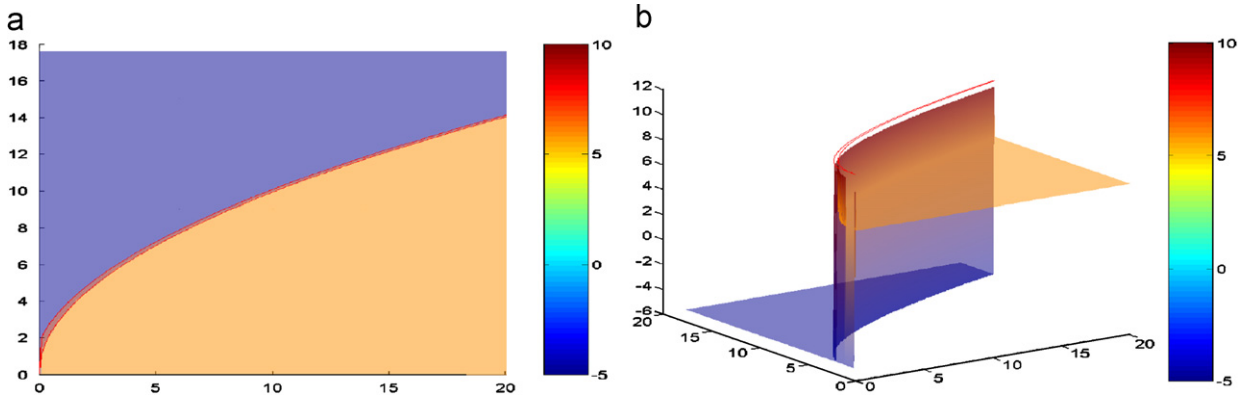

Fig. 11. The figure on the left shows a top view of the problem zones. From top left to bottom right we can see the dead, the optimal and the suboptimal zones. On the right, the figure shows the optimal action value function, where on x-axis we have the state (the distance to the hole), on y-axis the actions and z-axis is the corresponding action value.

The state–action space X × A can be divided into an *optimal* zone, a *suboptimal* zone, and a *dead* zone (Fig. 10). This problem is as simple as interesting, mainly due to the fact that the optimal function is on the edge of the dead zone, so the agent must try to get as near as possible to the optimal action without falling into the dead zone, and this requires it to be very accurate. Fig. 11 shows the theoretically optimal action-value function.

In the following we compare $\mathcal{Q}$ values with $\mathfrak{Q}$ values, analyzing the characteristics of the learned action-value functions in different situations. For each situation we show the graphs of the two action-value functions: the x-axis (the horizontal axis on the right) represents the state space, that is, the distance from the hole, while the y-axis (the horizontal axis on the left) represents the action space. The lines departing from the abscissae correspond to the vertexes of the fuzzy sets, while the lines departing from the ordinates correspond to the discrete actions available in each rule, $a_{r,i}$. At the intersection of these lines, on the z-axis the $\mathcal{Q}$ and $\mathfrak{Q}$ values are, respectively, represented. The red lines plotted above the surfaces represent, for each state, the range of optimal values (i.e., the minimum and maximum velocities to make the ball fall into the hole in one shot), while the blue line represents the learned policy.

7.1. Granularity of the fuzzification

A crucial aspect of fuzzy Q-learning is represented by the initial choice of the fuzzification of the state space and the choice of the discrete actions available in each fuzzy rule, since all the policies that the system is able to propose as estimates of the optimal action function strictly depend on these factors. The finer the representation, the more expensive the required exploration, and the better the approximation of the action function. Vice versa, the rougher the representation, the faster the convergence and the smaller the required computing effort.

In the first experiment we adopt a coarse representation (in particular for the fuzzification of the state variable): $\{0 : 5 : 20\} \times \{0 : 0.5 : 17.5\}$. The two action-value functions learned after 15,000 episodes are shown in Fig. 12. Looking at the policy learned using the $\mathfrak{Q}$-values, it can be noticed that, when the $\mathfrak{Q}$-values have an influence zone where both high and low action values may be achieved, their updates become unstable, so the estimated action function

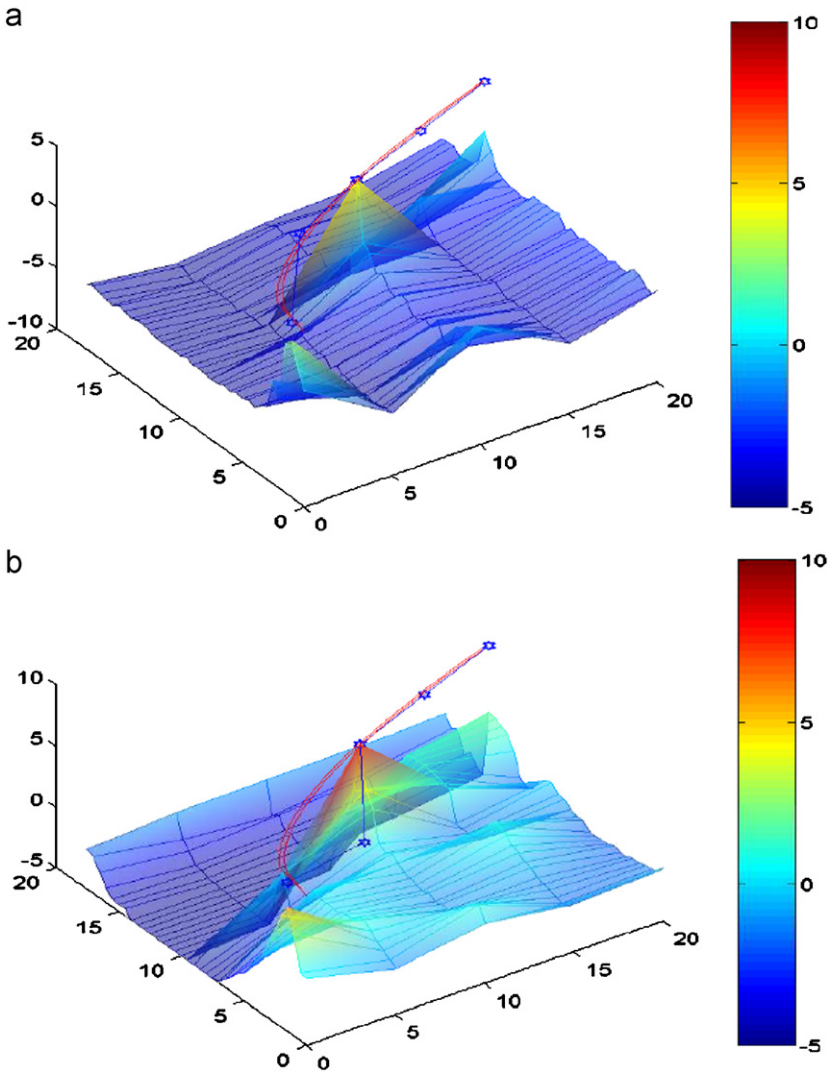

Fig. 12. Action-value functions obtained with a coarse representation: (a) $\mathcal{Q}$ value function and (b) $\mathfrak{Q}$ value function.

becomes *cautious* in the sense that a suboptimal action whose zone of influence gathers more coherent and constantly good targets may be estimated better than an optimal action whose zone of influence gathers both optimal and very bad targets. This is intrinsic in the nature of $\mathfrak{Q}$-values: they give generalization throughout their zone of influence, so the generalization over a zone will be influenced *in average* by all the values gathered in such zone, and a zone with optimal and very bad values may in average be worse than a zone with sub-optimal values.

The $\mathcal{Q}$-values do not suffer from this problem, even if a too coarse fuzzification of the state and action spaces may induce a piecewise linear policy that in some regions may bring away from the optimal policy function (see the difference between the red lines and the blue line in Fig. 12).

Nevertheless over-description of the problem is not always a good solution. First of all we may lose the advantage of generalization introduced by the concept of fuzziness. Furthermore, the advantages are not sure to come, especially in the case of $\mathcal{Q}$ values. As we discussed in Section 6.1, the values to update the $\mathcal{Q}$-values are structurally heterogeneous, and this heterogeneity grows with the number of active rules and with the cardinality of the corresponding discrete action sets, and in the meanwhile the frequency of the updates decreases. So the FIS working with $\mathcal{Q}$-values needs much more training in order to converge. Conversely, the $\mathfrak{Q}$-values exploit each update that they receive toward the final result: even if a $\mathfrak{Q}$-value has a little update rate, the targets it receives are more coherent since its zone of influence

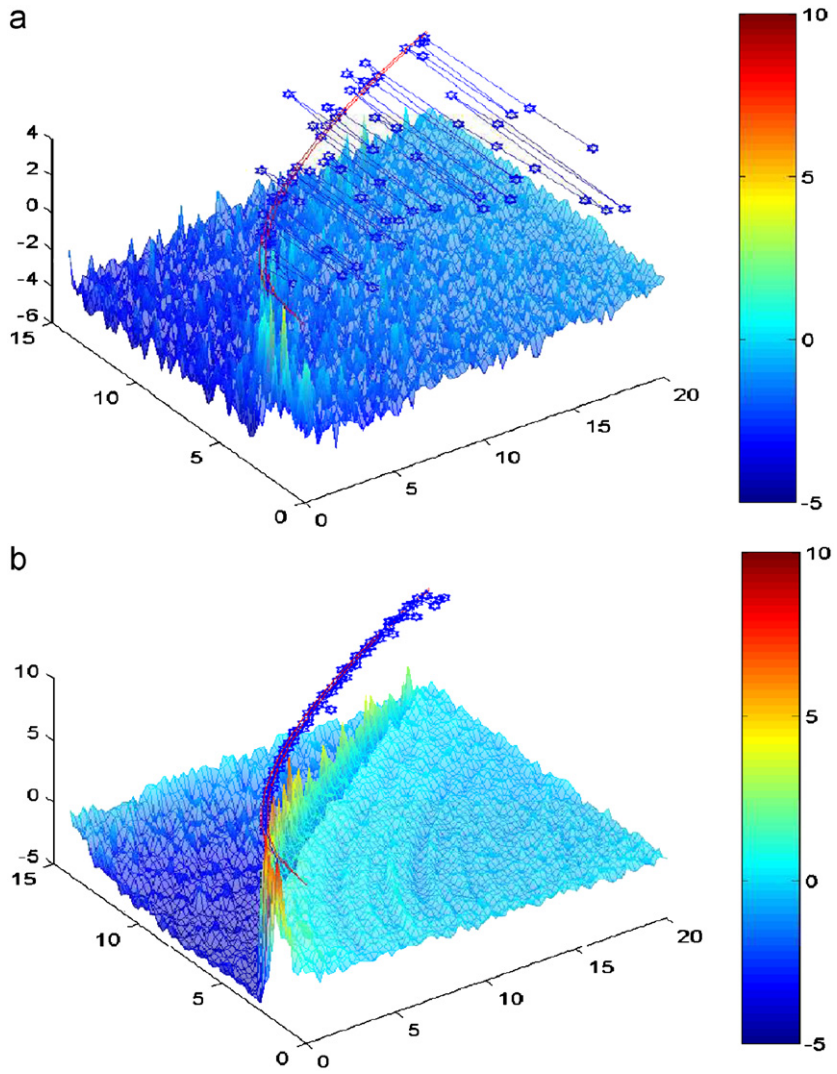

Fig. 13. Action-value functions obtained with a fine representation: (a) $\mathcal{Q}$ value function and (b) $\mathfrak{Q}$ value function.

is smaller and "more specialized", thus, each $\mathfrak{Q}$-value needs less training with respect to a coarser representation. In the next experiment shown in Fig. 13, we fuzzified the domain as $\{0 : 0.25 : 20\} \times \{0 : 0.25 : 17.5\}$. We can see that the $\mathfrak{Q}$-values work very well, interpolate the optimal action function with high accuracy, and are very stable, since their updates are more coherent even if less frequent in such fine representation. Conversely, the $\mathcal{Q}$-values do not work as well: they are not able to find a good approximation and they vary over time for much longer. Actually, the problem is that in a big space we need a lot of experience, and, at the same time, we need to exploit such experience as much as possible.

7.2. Equivalent actions

To study how the $\mathcal{Q}$-values and the $\mathfrak{Q}$-values behave when more than one optimal action is available in some states, we change the problem as follows: the ball can either fall in the hole or fall in a basket placed further in the dead zone. The agent can hit the ball within two ranges of optimal actions for every state, but it will have a second chance for hitting only if the ball remains in the suboptimal zone. This is an interesting problem, that cannot be solved so easily: indeed, since the optimal action is interpolated by many rules, the winning actions can belong to the two different optimal

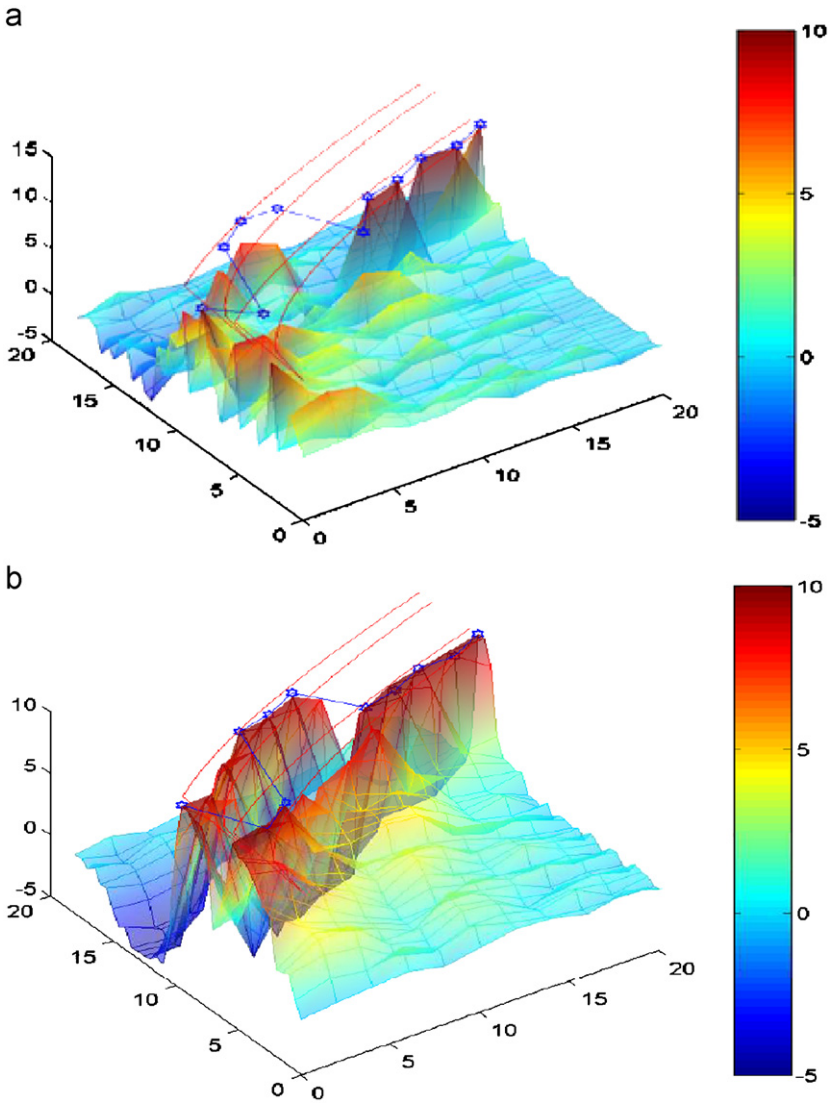

Fig. 14. Action value functions for the golf game with multiple optimal actions: (a) $\mathcal{Q}$ value function and (b) $\mathfrak{Q}$ value function.

ranges (being therefore optimal actions themselves), but they are still much bound to produce a very bad action. A possible solution consists of implementing a mechanism of *conformation* before the *cooperation*, i.e., let the active rule choose all the actions from either the former or the latter range of optimal values. This can be achieved in several ways, for example by letting the rule with the greatest activation degree choose first, and forcing the others to conform to this choice, or, for example, by proposing all the equivalent actions for each rule and then choosing the most voted first, or by combining these two criteria together. Future investigations will focus on this topic. The $\mathcal{Q}$-values and the $\mathfrak{Q}$-values learned in this situation are shown in Fig. 14.

Note that in the example the existence of equivalent actions is much more evident in the case of $\mathfrak{Q}$-values, since the quality of the single, executed, optimal actions is not affected by the existence of other equivalently optimal actions, so there are two evident optimal zones in the estimated optimal action function. The agent can easily recognize such a configuration and adopt strategies for choosing actions under the hypotheses of equivalent optimal actions. $\mathcal{Q}$-values instead have more chances to be confused by this situation, since the optimal discrete actions are bound to produce a bad executed action (if they belong to different optimal zones) and may receive, consequently, bad targets.

On the other hand, when the optimal zones are too close to each other, the zones of influence of $\mathfrak{Q}$-values may cover both of them, averaging the different quality of optimal, then dead, and then again optimal actions. As a result the agent may have a representation that merges these zones into only one (problem of the influence of the FIS structure, see Section 6.2). Conversely, the $\mathcal{Q}$-values are able to keep distinct the individual values of the discrete actions, so that there are more chances of identifying distinct actions belonging to the first optimal, the dead and the second optimal zone.

8. Conclusions and future work

The definition of fuzzy Q-learning involves many different concepts and many different ways of implementing the Q-learning taking advantage of the idea of fuzziness. We have considered the most promising algorithm found in literature [25], an evolution of the system proposed by Glorennec [24], and at the basis of many versions of Fuzzy Q-learning. We conducted an analysis on the nature of the particular reinforcement distribution of this algorithm, in comparison with a novel strategy of reinforcement distribution that we propose in this paper.

The experimental analysis and comparison of these two techniques has put in evidence the following results: when the fuzzification is coarse the $\mathfrak{Q}$-values may have difficulties to well approximate the optimal policy, especially when a fuzzy set covers a region characterized by heterogeneous rewards. On the other hand, when the domain is over-described $\mathcal{Q}$-values may have difficulties to converge, while $\mathfrak{Q}$-values correspond to a more efficient learning process. Furthermore, we have experimented how the combined use of both $\mathcal{Q}$-values and $\mathfrak{Q}$-values may turn very useful in those problems that present equivalently optimal actions.

The understanding of the behavior, limits, pros and cons of both the distribution strategies may help in the naturally following step, that is the addition of fuzzification adaptive strategies to the system. As we have underlined in Section 5.1, the value associated to the ith action of the rth rule, when the $\mathcal{Q}$ values are considered, is strictly dependent on the values associated to the winning actions selected by all the other rules. This implies that adding, modifying, and deleting a rule may affect the action values stored in any other rule, potentially wasting what has been previously learned. On the other side, using the $\mathfrak{Q}$ values each rule contains action values that have a meaning per se, thus being more suitable for being used in an adaptive system. A combined use of $\mathcal{Q}$ values and $\mathfrak{Q}$ values that exploit the benefits of both the approaches to build a fuzzy RL algorithm with adaptive rules will be considered by future research activity.

References

[1] C. Watkins, P. Dayan, Q-learning, Machine Learning 8 (1992) 279–292.
[2] A. Bonarini, Delayed reinforcement, fuzzy Q-learning and fuzzy logic controllers, in: F. Herrera, J.L. Verdegay (Eds.), Genetic Algorithms and Soft Computing (Studies in Fuzziness), Vol. 8, Physica-Verlag, Berlin, 1996, pp. 447–466.
[3] A.G. Barto, S.J. Bradtke, S.P. Singh, Learning to act using real-time dynamic programming, Artificial Intelligence 72 (1–2) (1995) 81–138.
[4] R. Sutton, A. Barto, R. Williams, Reinforcement learning is direct adaptive optimal control, in: Proc. American Control Conf., Boston, 1991, pp. 2143–2146.
[5] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, 1998.
[6] R.S. Sutton, Learning to predict by the methods of temporal differences, Machine Learning 3 (1988) 9–44.
[7] P. Dayan, T.J. Sejnowski, TD(λ) converges with probability 1, Machine Learning 14 (1994) 295–301.
[8] P. Dayan, The convergence of td(λ) for general λ, Machine Learning 8 (3/4) (1992) 341–362.
[9] T. Jakkola, M. Jordan, S. Singh, On the convergence of stochastic iterative dynamic programming, Neural Computation 6 (1993) 1185–1201.
[10] T. Mitchell, Machine Learning, McGraw-Hill, New York, 1997.
[11] P. Dayan, G.E. Hinton, Feudal reinforcement learning, in: S.J. Hanson, J.D. Cowan, C.L. Giles (Eds.), Advances in Neural Information Processing Systems, Vol. 5, Denver, CO, USA, 1993, pp. 271–278.
[12] C. Anderson, S. Crawford-Hines, Multigrid q-learning, Technical Report CS-94-121, Colorado State University, Fort Collins, 1994.
[13] S.P. Singh, T. Jaakkola, M.I. Jordan, Reinforcement learning with soft state aggregation, in: G. Tesauro, D. Touretzky, T. Leen (Eds.), Advances in Neural Information Processing Systems, Denver, CO, USA, 1995, pp. 361–368.
[14] R.S. Sutton, Generalization in reinforcement learning: successful examples using sparse coarse coding, in: D.S. Touretzky, M. Mozer, M.E. Hasselmo (Eds.), Advances in Neural Information Processing Systems, Vol. 8, Denver, CO, USA, 1995, pp. 1038–1044.
[15] D. Broomhead, D. Lowe, Multivariable functional interpolation and adaptive networks, Complex Systems 2 (1988) 321–355.
[16] M. Powell, Radial basis functions for multivariate interpolation: a review, Algorithms for Approximation (1988) 143–167.
[17] J.D.R. Millan, D. Posenato, E. Dedieu, Continuous-action q-learning, Machine Learning 49 (2002) 247–265.
[18] A.A. Sherstov, P. Stone, Function approximation via tile coding: automating parameter choice, in: J.-D. Zucker, I. Saitta (Eds.), Sixth Internat. Symp. Abstraction, Reformulation and Approximation (SARA), Lecture Notes in Artificial Intelligence, Vol. 3607, Airth Castle, Scotland, UK, 2005, pp. 194–205.

[19] S. ten Hagen, B. Krase, Q-learning for systems with continuous state and action spaces, in: 10th Belgian–Dutch Conf. on Machine Learning, 2000.
[20] A. Bonarini, Evolutionary learning, reinforcement learning, and fuzzy rules for knowledge acquisition in agent-based systems, Proceedings of the IEEE 89 (9) (2001) 1334–1346.
[21] R.E. Bellman, L.A. Zadeh, Decision-making in a fuzzy environment, Management Science 17-B (4) (1970) 141–164.
[22] H.-S. Seo, S.-J. Youn, K.-W. Oh, A fuzzy reinforcement function for the intelligent agent to process vague goals, in: Fuzzy Information Processing Society, Atlanta, GA, USA, 2000, pp. 29–33.
[23] J. Kacprzyk, J. Kacprzyk, Multistage Fuzzy Control: A Prescriptive Approach, Wiley, New York, NY, USA, 1997.
[24] P.Y. Glorennec, L. Jouffe, Fuzzy Q-learning, in: Proc. Sixth Internat. Conf. on Fuzzy Systems (Fuzz-Ieee'97), Barcelona, Spain, 1997, pp. 659–662.
[25] L. Jouffe, Fuzzy inference systems learning by reinforcement methods: application to a pig house atmosphere control, Ph.D. Thesis, Computer Science Department, Universitè de Rennes I, Rennes, France, 1997.
[26] H. Berenji, A reinforcement learning-based architecture for fuzzy logic control, International Journal of Approximate Reasoning 6 (2) (1992) 267–292.
[27] H. Beom, H. Cho, A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning, IEEE Transactions on Systems, Man and Cybernetics, Vol. 25, 1995, pp. 464–477.
[28] D. Gu, H. Hu, Accuracy based fuzzy q-learning for robot behaviours, in: Proc. IEEE Internat. Conf. on Fuzzy Systems, Budapest, Hungary, 2004, pp. 1455–1460.
[29] T. Horiuchi, O. Katai, Fuzzy interpolation-based q-learning with continuous states and actions, in: Proc. Fifth IEEE Internat. Conf. on Fuzzy Systems, New Orleans, LA, USA, 1996, pp. 594–600.
[30] D. Gu, H. Hu, L. Spacek, Learning fuzzy logic controller for reactive robot behaviours, in: IEEE/ASME Internat. Conf. on Advanced Intelligent Mechatronics, 2003, pp. 46–51.
[31] X. Dai, C. Li, A. Rad, An approach to tune fuzzy controllers based on reinforcement learning, in: The 12th IEEE Internat. Conf. on Fuzzy Systems (FUZZ '03), 2003, pp. 517–522.
[32] C. Deng, M.J. Er, Efficient implementation of dynamic fuzzy q-learning, in: Proc. 2003 Jt. Conf. of the Fourth Internat. Conf. on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conf. on Multimedia, 2003, pp. 1854–1858.
[33] C. Deng, M.J. Er, Automatic generation of fuzzy inference systems by dynamic fuzzy q-learning, in: IEEE Internat. Conf. on Systems, Man and Cybernetics, 2003, pp. 3206–3211.
[34] C.D. Meng Joo Er, Online tuning of fuzzy inference systems using dynamic fuzzy q-learning, IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol. 34, 2004, pp. 1478–1489.
[35] B. Ster, A. Dobnikar, Adaptive radial basis decomposition by learning vector quantization, Neural Processing Letters 18 (1) (2003) 17–27.
[36] R. Bellman, Dynamic Programming, Princeton University Press, Princeton, NJ, 1957.
[37] D. Bertsekas, J. Tsitsiklis, Neural Dynamic Programming, Athena Scientific, Belmont, MA, 1996.
[38] C. Watkins, Learning from delayed rewards, Ph.D. Thesis, King's College, Cambridge, UK, 1989.
[39] H. Robbins, S. Monro, A stochastic approximation method, Annals of Mathematical Statistics 22 (1951) 400–407.
[40] B. Ratitch, On characteristics of Markov decision processes and reinforcement learning in large domains, Ph.D. Thesis, School of Computer Science, McGill University, 2005.
[41] S.I. Reynolds, Reinforcement learning with exploration, Ph.D. Thesis, University of Birmingham, December 2002.
[42] G. Tesauro, Temporal difference learning and td-gammon, Communications of the ACM 38 (3) (1995) 58–68.
[43] J.A. Boyan, A.W. Moore, Generalization in reinforcement learning: safely approximating the value function, in: G. Tesauro, D.S. Touretzky, T.K. Leen (Eds.), Advances in Neural Information Processing Systems, Denver, CO, USA, 1995, pp. 369–376.
[44] L. Baird, Residual algorithms: reinforcement learning with function approximation, in: Internat. Conf. on Machine Learning, Tahoe City, CA, USA, 1995, pp. 30–37.
[45] G.J. Gordon, Reinforcement learning with function approximation converges to a region, in: T.K. Leen, T.G. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems, Denver, CO, USA, 2000, pp. 1040–1046.
[46] J. Tsitsiklis, B.V. Roy, An analysis of temporal difference learning with function approximation, IEEE Transactions on Automatic Control 42 (1997) 670–690.