Technical Report

IRI-TR-14-05

Competitive Function Approximation for Reinforcement Learning

IRI Technical Report

Alejandro Agostini
Enric Celaya

November, 2014

Institut de Robòtica i Informàtica Industrial

Abstract

The application of reinforcement learning to problems with continuous domains requires representing the value function by means of function approximation. We identify two aspects of reinforcement learning that make the function approximation process hard: non-stationarity of the target function and biased sampling. Non-stationarity is the result of the bootstrapping nature of dynamic programming where the value function is estimated using its current approximation. Biased sampling occurs when some regions of the state space are visited too often, causing a reiterated updating with similar values which fade out the occasional updates of infrequently sampled regions.

We propose a competitive approach for function approximation where many different local approximators are available at a given input and the one with expectedly best approximation is selected by means of a relevance function. The local nature of the approximators allows their fast adaptation to non-stationary changes and mitigates the biased sampling problem. The coexistence of multiple approximators updated and tried in parallel permits obtaining a good estimation much faster than would be possible with a single approximator. Experiments in different benchmark problems show that the competitive strategy provides a faster and more stable learning than non-competitive approaches.

Institut de Robòtica i Informàtica Industrial (IRI)
Consejo Superior de Investigaciones Científicas (CSIC)
Universitat Politècnica de Catalunya (UPC)
Llorens i Artigas 4-6, 08028, Barcelona, Spain
Tel (fax): +34 93 401 5750 (5751)
http://www.iri.upc.edu

Corresponding author:
A. Agostini
email:aagosti@gwdg.de
Bernstein Center for Computational
Neuroscience 37077 Goettingen, Germany

1 Introduction

It is well known that generalization is mandatory in applications of reinforcement learning (RL) involving large domains [29]. Only the most basic formulation of RL, which assumes a discrete state-space, can avoid the use of generalization by explicitly storing the value (or the action-value) of each state (or state-action) in a tabular representation. But the lack of generalization implies that, in order to grant finding the optimal policy, all the possible actions in all the states must be experienced at least once, if the system is deterministic, or many times in general. This is impractical in applications where the number of states and actions is very large, and completely impossible when it is infinite. Unfortunately, many of the interesting applications of RL involve very large or continuous domains, and the only way to deal with them is to generalize the result of experiences obtained for a given state and action to others similar to them.

Generalization techniques have been widely explored in the field of machine learning [20] and pattern classification [14, 6]. In off-line, batch approaches, a finite set of input-output samples, corresponding to an unknown function commonly referred as the *target function*, are available, and some method for *function approximation* (FA) [11] is used to build an approximating function that permits inferring values of the target function in points of the domain whose output is actually unknown. A usual approach to FA consists in defining a parameterized family of functions that is expected to contain a good enough approximation of the target function, and trying to adjust the parameters to reproduce the known samples with optimal accuracy. On-line incremental approaches are similar, except that the samples are not all available from the start but arrive sequentially, and the approximating function is progressively adapted according to the new incoming information.

In principle, many different machine learning methods for FA can be used in RL, but not all of them will be equally suitable. Due to the bootstrapping nature of the value estimations [29], the function that the FA process is trying to approximate becomes *non-stationary*. During the learning process, this non-stationarity is usually different at different regions [16] and the intermediate value function approximations at the different stages of the learning usually have different structures, and could be even more complex than the optimal one [9]. This demands the FA method to be flexible enough to cope with the local non-stationarity and complexity variations. Another aspect to take into account in on-line RL is that samples can only be obtained from the interaction with the environment, and they are presented along the trajectories produced by the current action policy and the dynamics of the environment. This makes the sampling of the target function to be *highly biased*, with some regions much more densely sampled than others. Not all techniques for on-line FA are robust against biased sampling, since many of them will tend to forget the information provided by the weakly sampled regions as more and more inputs are processed from the more frequently sampled ones.

In this work we present a new approach to FA that addresses the non-stationarity and the biased sampling problems associated with on-line RL. The main idea consists in building multiple approximating functions in parallel, defined on different, overlapping regions of the input (state or state-action) space, so that, at each point, several approximations will be available. Among them, the system will select the one with the best expected accuracy, as measured by a relevance function estimated from the observed performance of each approximator. Since each approximation is restricted to a limited region and independent of the other ones, there is more flexibility in the overall global function that can be represented. Since each approximation is only updated with samples falling in its domain, they can adapt more easily to local changes of the target function, and oversampling in one region does not affect at all the approximation existing in a different region.

This paper is organized as follows. Section 2 reviews existing local approaches to FA and analyses how a competitive approach may improve their performance. Section 3 introduces

the proposed approach to FA and presents the ICFA algorithm. Section 4 proposes a specific implementation based on Gaussian mixture models, and Section 5 specifies how ICFA can be used with RL, introducing the algorithm ICFARL. Next, in Section 6, some experiments with the algorithm in standard RL problems are presented, and Section 7 closes the paper with some conclusions.

2 Local Approaches for Function Approximation

A feasible way to deal with the non-stationarity and the biased sampling problems typical of on-line RL consists in, instead of using a single function to approximate the target function in the whole domain, partitioning the input space in smaller regions and try to approximate the function locally with an independent FA process in each region. By doing this, the approximation in a region is not affected by the eventual changes that may take place in other regions, nor by how unbalanced are the sample rates in different regions, since each local approximator will only take into account the samples falling within its own domain.

Moreover, in general, if the domain of a complex function is partitioned in smaller regions, the complexity of the local functions required to approximate the target function will be lesser than that required for a global function to provide an equivalent accuracy. As a consequence of this, FA methods that use local function approximators may converge much faster to the desired accuracy than those trying to make the approximation with a single global function.

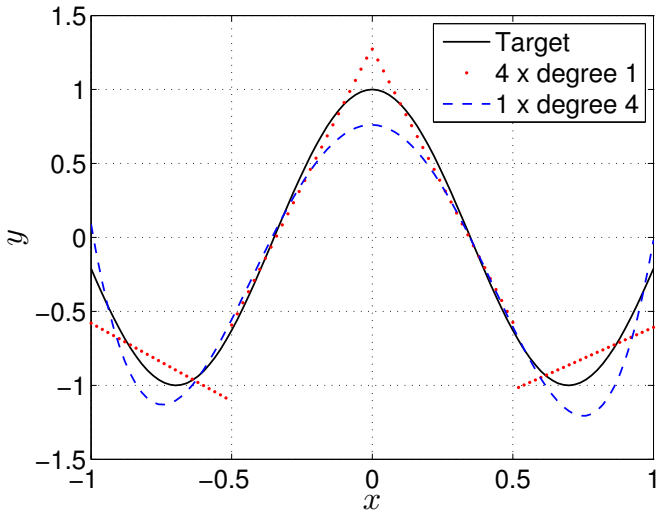
2.1 Efficiency of Local versus Global Approximations

To compare the efficiency of using local and global approximators, we consider the example of Figure 1(a). In this example, a sinusoidal target function defined in the interval [-1, 1] is approximated, on the one hand, globally by means of a fourth degree polynomial and, on the other hand, locally in 4 intervals of length 1/2 by means of 4 linear functions (polynomials of degree 1). For a fair comparison, the functions have been chosen so that none of the approximations can represent the target function perfectly, but the optimal solutions provide similar accuracies in both cases. In the experiment, all polynomials are initialized with random coefficients, and a set of randomly generated input-output pairs are used to sequentially update the coefficients of both approximations using a gradient descent method.

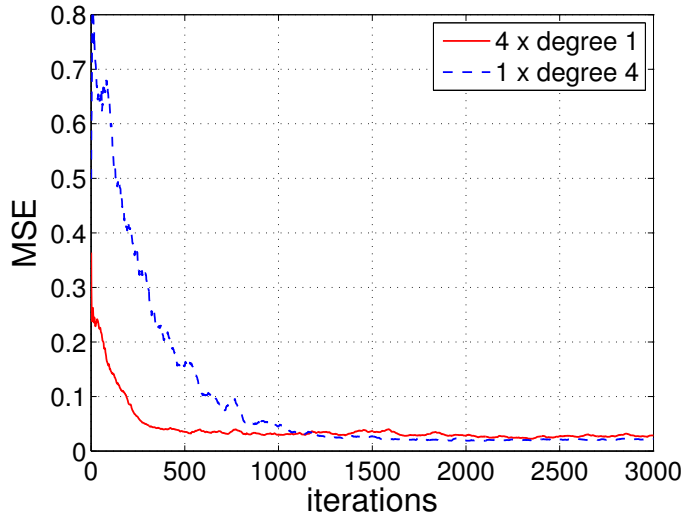
As shown in Figure 1(b), the local function approximation converges much faster than the global one, and both stabilize around a similar error value in the long term. Note that the faster convergence of the local functions occurs despite the fact that they involve a total of 8 parameters, while the global function involves only 5. This result may be justified by the fact that the parameters of each linear function are independent of those of the remaining functions, so that only two parameters must be adjusted for each approximator, while in the global function all parameters are interdependent and must be balanced altogether. Note also that each linear approximator receives, in average, only one fourth of the training samples but, even so, their adaptation is faster than that of the fourth degree polynomial which is updated with all samples.

2.2 Searching for Proper Local Domains: Partitions versus Coverings

One common approach in continuous-domain RL that performs local FA is *state aggregation*, in which the state space is discretized into cells with some resolution, and each cell is considered as a single state and treated as in the tabular RL approach. In this case each local approximator consists in a simple constant-value function, with its only parameter being the value that will be assigned to all the states aggregated in the cell. But, though simply partitioning the domain to define local approximators may improve the accuracy and efficiency of FA, such improvement



(a) Target function approximations.



(b) Convergence profiles.

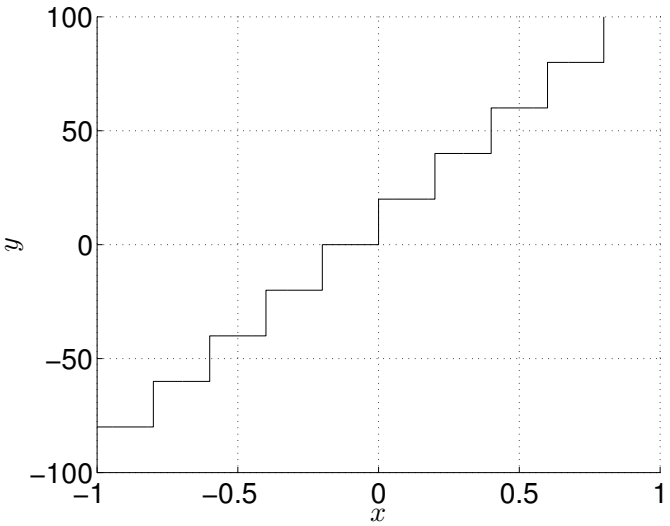
Figure 1: Comparison between a global FA with a 4^{th} deg. polynomial vs. 4 local linear FA.

may vary largely depending on how the partition is made. For instance, in the example of Figure 1(a), it is clear that the accuracy of the approximation could be improved if the local domains had been chosen in accordance with the shape of the target function, so that the bounds of the intervals would coincide with the extrema of the function. However, since the target function is unknown, a good partition can only be found through search.

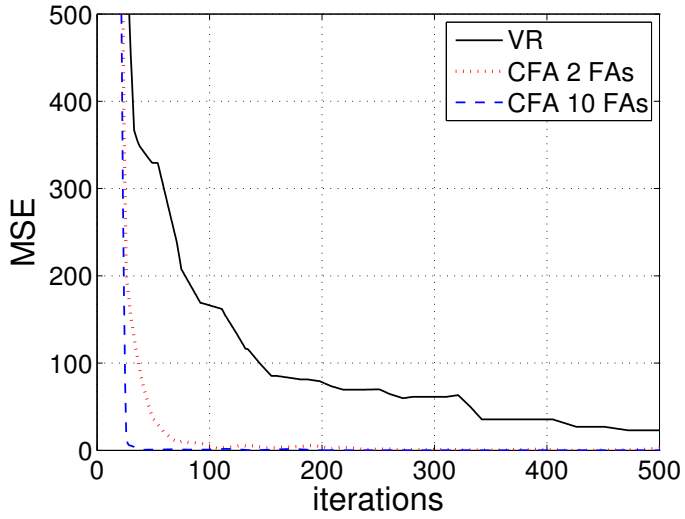
A search strategy typically used with state aggregation is that of variable resolution (VR) [22, 23], which consists in defining an initial coarse partition of the domain, and recursively split those regions that can not be approximated with the desired accuracy with a constant value. For example, in [26] a VR technique is used to approximate the action-value function in a continuous-state, discrete-action space. In this case, the partition is represented with a *kd*-tree [15], where each node defines a region of the state space, and children nodes correspond to subsequent splittings of their parents. The leaves of the tree configure a partition where a cumulative reward value for each action is associated to each part (leaf). The depth of the tree is increased at those regions where a finer resolution is required to select the optimal action. A related approach is proposed in the *kd*-*Q*-Learning method [30, 31], which also uses a *kd*-tree representation but where the values for each action are stored not only at the leaves of the tree but also at the interior nodes at all levels of resolution. This allows selecting, for each action, the value from the resolution level with highest confidence, which is given by the number of updates. In this case, instead of increasing the resolution on-line by adding new leaves to the tree, the maximum resolution at all parts of the state space is fixed from the very beginning by a parameter of the system. Similar strategies can be found with multi-partition approaches, as in the Adaptive Tile Coding [33], where the resolution of different partitions is increased at regions with large approximation error. Some other VR approaches not only change the resolution at the state space but also at the action space, allowing for generalization along the action dimensions [17, 21].

A limitation of such VR techniques is that the improvement of the partition only takes place by splitting already existing regions, what invariably increases its number and impoverishes generalization. An alternative way to deal with this issue is, instead of forming a partition of the domain in disjoint regions, allow regions to overlap to form an arbitrary covering. In this case, when a new region needs to be generated, it can be defined at will and independently of the already existing regions, thus increasing the opportunities of finding a region in which the function can be appropriately approximated without losing generalization.

To illustrate this idea, we perform a simple experiment in which the function of Figure 2(a) must be approximated with local constant-value functions. Note that, since the target function is piecewise constant, it can be exactly represented with state aggregation, provided that the right partition is found. As in the previous experiment, samples are randomly generated and used to update the values of the approximations of the local functions, which in this case are taken as the mean of the values observed within the domain of each approximator. Each time a sample arrives for which the approximation error is above a given threshold, new approximators are generated to improve the approximation. In a first test, we use a classical VR approach, in which the inaccurate approximator is replaced by two new ones obtained by splitting its domain in two halves, so that a partition of the domain is always obtained. In a second test, the inaccurate approximator is maintained, and new approximators are generated with intervals containing the input point but with bounds chosen randomly in the whole input range, so that the result is a covering of the domain with overlapped regions. Since the purpose of this experiment is to test the efficiency of the covering approach in generating the right intervals, we assume that, in this case, the value attributed to the function approximation is that of the approximator that is closest to the target function value. Of course, in a real situation, the target function is unknown, and we should specify some way to select it. How this can be done will be discussed later (see Section 3.2). For a fair comparison with the VR approach, two new approximators



(a) Piecewise constant target function.



(b) Convergence profiles.

Figure 2: Evolution of the MSE of the partition versus the covering approaches.

are generated when the approximation at the current point is inaccurate, but nothing prevents us from generating an arbitrary number of them. In order to see the influence of the generation rate, we performed a third test in which the number of competing approximators generated each time is increased to 10.

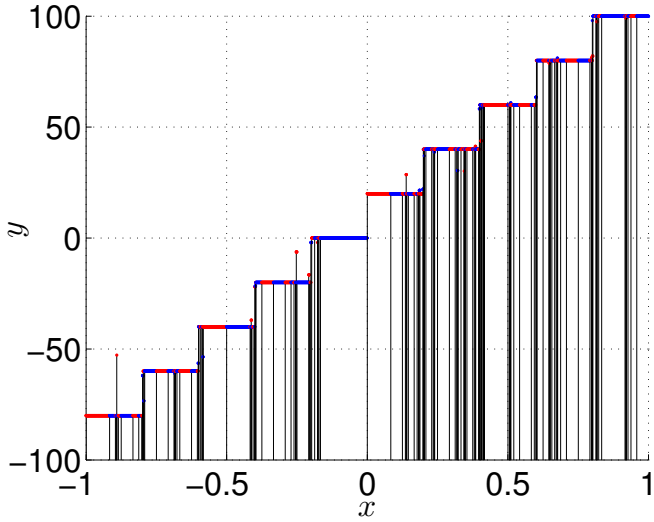
Figure 2(b) shows the evolution, for each test, of the mean squared error of the approximation with the number of samples. It can be seen that the strategy that generates a random covering (labelled as CFA) produces a set of regions able to approximate the target function much faster than the VR approach which simply refines the partition. As could be expected, increasing the generation rate allows reducing the approximation error much faster. To better interpret this result, Figure 3 shows the approximators generated by the two first tests at the end of the experiment. We observe that the partition generated by the VR approach has been unnecessarily fragmented due to the bad fitting of the binary subdivisions with the constant-valued intervals of the target function (Figure 3(a)). The final partition consists of 173 intervals, and it can be observed that some of them have remained with a tiny domain and a wrong value. These are just the remnants of the initial intervals that have been cut off by successive splittings and, with the simple approach adopted here, they will stay there forever since, even if a sample turns out to fall into one of these intervals, it will be split in two, and only the half containing the point will use its value. In contrast, the covering obtained by generating the two intervals at random is able to give an accurate account of the target function with only 26 approximators, assuming that the best one can be selected at any point of the domain (in fact, due to overlapping of some of these 26 approximators, they configure an effective partition of the domain in 33 different intervals).

In this example, each local approximator is reduced to a constant value for all points in its domain, but to better fit the data, we can use any other form of FA, e.g., polynomials of some degree, as in the example of figure 1. Whatever is the choice of the FA used for each local approximator, a similar search strategy can be applied in order to find the regions of the state-space whose function values fit best with such form of FA.

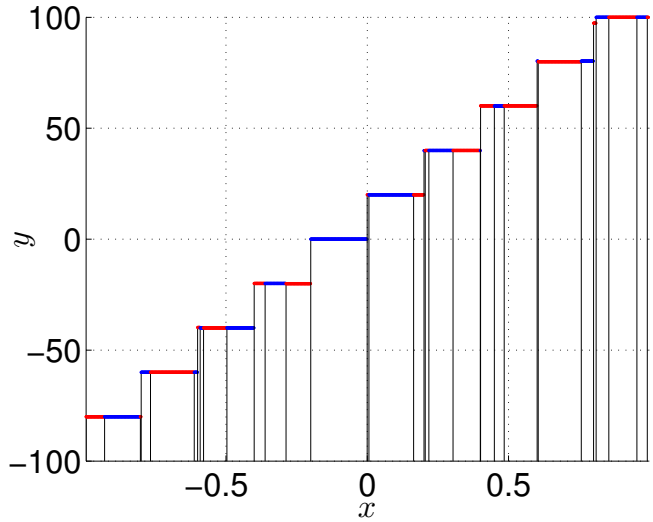
3 Function Approximation by Competition of Local Approximators

In the previous section we have seen that using local approximators with overlapping domains is potentially more advantageous than using a single global approximator. However, in most existing FA approaches that involve functions with overlapping domains, instead of taking the value of just one of them, the value at each point is computed as a linear combination of the values provided by the different functions, with weights that may depend on the distance to the center of the respective domains [10]. In some of these approaches, the combined functions are constant, as in some multi-partition schemes [33, 8], while in others they consist in some form of Radial Basis Functions, which can be defined either in the input-space [24, 19], or in the input-output space, as in [1]. It is usual that the number or scope of the functions is adjusted on-line in order to improve the accuracy of the approximation. For example, [5] use a set of spline-like kernels as basis functions, and their density is progressively increased by splitting them when required, in a general to specific way. An alternative approach is followed in [19], where the area of influence of the basis functions is modified by adapting the parameters defining their center and dispersion with a gradient-based algorithm.

There is a crucial difference between combining overlapped functions and taking the value of one of them. In the first case, none of the combined functions is completely responsible of approximating the function at any point, since it will be combined with all the other functions having some influence at this point. This means that the value taken by each function can not be set independently of the others, and such a dependency may make the system difficult to stabilize. On the other hand, when the domains of the combined functions are too wide,



(a) Approximation obtained with the partition of the VR approach.



(b) Approximation obtained with the partition of the VR approach.

Figure 3: Comparison of the partition versus the covering approximations. Vertical lines delimit each part of the partition configured by the approximations.

the isolation from different regions of the domain is lost, and the system becomes prone to be perturbed by the non-stationarity and the biased-sampling effects. In contrast, none of these problems appear when the value is given by a single local approximator, since each of them tries to make its best approximation in its domain that is not affected by how good or wrong are performing other competing approximators. If eventually, one approximator gets its approximation degraded, there may still be some other approximator that performs acceptably well there, and it will be selected as soon as it proves to be more accurate, allowing the system to keep a good overall performance and producing a more stable behavior.

The counterpart of using local independent overlapping approximators is that a mechanism is needed to ensure that, at each input point, the best or, at least, a good approximator is selected among those that include the point in its domain. Next, we propose a competitive FA approach with overlapping approximators for which a relevance function will be defined to estimate their relative expected accuracies. Some of the contributions presented next are extensions of our previous work [2].

3.1 Competitive Function Approximation

Let X be a connected D-dimensional set, and $\mathbf{\Phi} = \{\Phi_i(x, \xi_i)|i = 1...N\}$ a set of parametric functions $\Phi_i : X_i \rightarrow \Re$ with parameter vectors ξ_i, such that $X = \bigcup_{i=1}^{N} X_i$, i.e., that the domains of the functions form a covering of X. For a given $x \in X$, we say that a function Φ_i is *active* if $x \in X_i$. The functions Φ_i will be called *competitor functions* since, given an input point x for which more than one Φ_i is active, each of them will provide its own value in competition with the other ones. For the sake of the competition, we will associate to each competitor a *relevance function*, $\Gamma_i(x, \nu_i)$, which is a parametric function with parameters ν_i, defined in the same domain X_i as the competitor. If $\mathbf{\Gamma} = \{\Gamma_i(x, \nu_i)|i = 1...N\}$ is the set of relevance functions associated to $\mathbf{\Phi}$, we define a *competitive function* $\mathcal{F}(x, \mathbf{\Phi}, \mathbf{\Gamma})$, with competitors set $\mathbf{\Phi}$ and relevance functions $\mathbf{\Gamma}$, as a function $\mathcal{F} : X \rightarrow \Re$ that assigns to each x the value provided by the active competitor with highest relevance at x. More formally, let $I_x = \{i|x \in X_i\}$ be the set of indexes of all active competitors for x. We say that Φ_ω is the *winner competitor* at x, if

$$w = \underset{i \in I_x}{\operatorname{argmax}} \Gamma_i(x, \nu_i).$$

Then, the value provided by the competitive function is the value given by the winner competitor

$$\mathcal{F}(x, \mathbf{\Phi}, \mathbf{\Gamma}) = \Phi_w(x, \xi_w).$$

Now, assume that there is an arbitrary unknown target function $f(x)$ defined in X that should be approximated, but from which we can only observe its values at particular points obtained sequentially, (x_t, y_t), with $y_t = f(x_t)$, where t accounts for the t-th time step. Our aim is to devise an on-line (i.e., incremental), memory-free function approximation method using a competitive function $\mathcal{F}(x, \mathbf{\Phi}, \mathbf{\Gamma})$ as defined before. We will call any method for function approximation that uses a competitive function $\mathcal{F}(x, \mathbf{\Phi}, \mathbf{\Gamma})$ a *competitive function approximation* (CFA) method, and in the specific case in which the updates are done incrementally after each observation we will call it *incremental competitive function approximation* (ICFA). For each competitor $\Phi_i(x, \xi_i)$ of a ICFA method we can use any arbitrarily complex parametric function appropriate for the problem at hand for which an incremental FA method is available. In principle, the FA's of the competitors do not need to be all the same. However, in the following, we will assume that all competitors implement the same function of its parameters, i.e., $\Phi_i(x, \xi_i) = \Phi(x, \xi_i)$, and all of them are updated using the same procedures. Of course, the specific FA method selected for the competitors and the extent of the domain of each competitor will influence the allowed accuracy and performance of the CFA. However, if the relevance

function is not able to provide a correct distinction between good and bad approximations, the performance of the CFA system may be poor, no matter what competitor function is used. Hence, a right definition of the relevance function is crucial for the good performance of the CFA system. The next section is focused on this aspect.

3.2 The Relevance Function

To make the exposition clear, we will first assume that the competitor and the relevance functions are constant-valued functions. A sensible choice for the value of a competitor function would be the mean (i.e., the expected value $E_{X_i}[f(x)]$) of the target function in the domain of the competitor, which can be estimated by the sample mean $\bar{Y}_i$ of the values of the samples observed in this domain [7],

$$\Phi_i(x, \xi_i) = \bar{Y}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} y_{\mathbf{ti}_j},$$

where n_i is the total number of samples observed so far in X_i and $\mathbf{ti}_j$ is the j-th element of the list of time steps at which competitor i was active. To define the relevance function, note that the variance of the target function f in X_i, $\sigma_i^2 = E_{X_i}\left[(f(x) - E_{X_i}[f(x)])^2\right]$, is a good account of how well $E_{X_i}[f(x)]\,(\approx \bar{Y}_i)$ approximates f at a point $x \in X_i$. Hence, the inverse of the variance σ_i^2 would be a reasonable value for the relevance. However, as we only have a limited amount of samples of f, the variance can not be determined precisely and should be estimated. We estimate the variance using the (unbiased) sample variance S_i^2 [7],

$$S_i^2 = \frac{1}{n_i - 1} \sum_{j=1}^{n_i} \left(y_{\mathbf{ti}_j} - \bar{Y}_i\right)^2. \tag{1}$$

Since the variance is estimated using a limited amount of information, its estimation carries uncertainties that vary with the number of samples n_i. To take into account these uncertainties we use a known result from statistics [7],

$$\frac{1}{\sigma_i^2} \sum_{j=1}^{n_i} \left(y_{\mathbf{ti}_j} - \bar{Y}_i\right)^2 \sim \chi^2\,(n_i - 1)\,. \tag{2}$$

Equation (2) states that the expression on the left has a χ^2 distribution with $n_i - 1$ degrees of freedom. Using (1) and (2) we obtain

$$\frac{(n_i - 1)\, S_i^2}{\sigma_i^2} \sim \chi^2\,(n_i - 1)\,.$$

This allows us to build a confidence interval for the true variance σ_i^2 that only depends on the sample variance and the number of samples experienced in the competitor domain[1]. For this, we use the definition of the α-quantile $\chi_\alpha^2(h)$ given by

$$P(Z > \chi_\alpha^2(h)) = \alpha. \tag{3}$$

Equation (3) means that, if a random variable Z has a χ^2 distribution with h degrees of freedom, the probability that Z takes values greater than $\chi_\alpha^2(h)$ is α. Fixing α to a convenient value, for

[1] In principle, the method to calculate the confidence interval for the variance through the χ^2 distribution assumes normally distributed samples. However, it is known [7] that this method is also valid when samples are obtained from any other distribution, different from the normal. In this case, the values provided would indicate approximated confidences, rather than accurate ones, that would become more precise as more samples are considered.

instance $\alpha = 0.95$ to make things concrete, we can obtain an interval for Z that is 95 % certain to include its true value. Taking $Z = \frac{(n_i-1)S_i^2}{\sigma_i^2}$ we have,

$$P\left(\frac{(n_i-1)\,S_i^2}{\sigma_i^2} > \chi_\alpha^2\,(n_i-1)\right) = P\left(\sigma_i^2 < \frac{(n_i-1)\,S_i^2}{\chi_\alpha^2\,(n_i-1)}\right) = \alpha.$$

The upper bound of this interval,

$$\overline{\sigma_i^2} = \frac{(n_i-1)\,S_i^2}{\chi_\alpha^2\,(n_i-1)},$$

represents the highest possible value (with α confidence) of the actual unknown variance σ_i^2 given the current uncertainties in the estimation. Identical sample variances obtained with a lesser number of samples will have higher upper bounds. The value $\overline{\sigma_i^2}$ provides a measure of the quality in the approximation that balances the estimation accuracy and the confidence in that estimation. Hence, we adopt the inverse of $\overline{\sigma_i^2}$ as the relevance of a constant-valued competitor Φ_i,

$$\Gamma_i(x,\nu_i) \equiv \left(\overline{\sigma_i^2}\right)^{-1} = \frac{\chi_\alpha^2\,(n_i-1)}{(n_i-1)\,S_i^2}. \tag{4}$$

This definition of the relevance avoids that estimations obtained from few observations that eventually show low sample variances, can be preferred to those with slightly higher sample variances but with much more confident estimations. To illustrate the behavior of the relevance function (4) with the accumulation of samples we note that

$$\lim_{n\to\infty} \frac{\chi_\alpha^2\,(n_i-1)}{n_i-1} = 1$$

and

$$\lim_{n\to\infty} \frac{1}{S_i^2} = \frac{1}{\sigma_i^2}.$$

Hence,

$$\lim_{n\to\infty} \Gamma_i(x,\nu_i) = \lim_{n\to\infty} \frac{\chi_\alpha^2\,(n_i-1)}{(n_i-1)\,S_i^2} = \frac{1}{\sigma_i^2},$$

which shows that, as the number of samples tends to infinite, the relevance function tends to the inverse of the actual unknown variance.

For the definition of relevance in Eq. (4) we have assumed that both, the competitor and the relevance functions are constant in their domains. Now we can extend this definition to the general case of point-dependent competitor and relevance functions. In this case, no matter what FA method is used for the competitor function, it is expected to approximate the mean value of the target function at each point and, consequently, the relevance will be associated to the variance of the target function at each point. A point-dependent estimation of the variance for each competitor $S_i^2(x,\tau_i)$ can be obtained with a further FA process, similar to that used for the competitor function, but whose samples involve the squared differences between the observed value and the current approximation, $(y_t - \Phi_i(x_t,\xi_i))^2$. Note that in many FA problems, the target function is not deterministic but stochastic, so that the variance at a given point does not only reflect the inaccuracy in the estimation of the target function at this point, but also its variability due to the non-determinism. However, since this additional source of variability is essentially associated to a given point x, it will affect in a similar measure all the competitors covering it, and will not influence in the competition. Thus, in the general case of a

point-dependent competitor function, definition (4) is extended to a point-dependent relevance function that can be approximated by

$$\Gamma_i(x, \nu_i) \equiv \frac{\chi^2_\alpha \left(h_i(x, \varsigma_i)\right)}{h_i(x, \varsigma_i) S_i^2(x, \tau_i)}, \tag{5}$$

were $h_i(x, \varsigma_i)$ plays the role of $n_i - 1$ in (4), but in this case is a point-dependent function that estimates the number of samples used in the computation of the variance at each point. Such estimation must take into account that, due to the generalization carried out by the FA process, each sample is not used just to update the approximation at the input point, but has an influence on its neighborhood in a way that depends on the specific FA approach and updating method. This may make the estimation of $h_i(x, \varsigma_i)$ rather complex and, in order to simplify it, we will assume that the amount of samples used at each point is proportional to the sample density at this point. Thus, if $p_i(x, \varsigma_i)$ is a parametric approximation of the sample probability density for the competitor, $h_i(x, \varsigma_i)$ can be approximated by

$$h_i(x, \varsigma_i) \approx V n_i \, p_i(x, \varsigma_i) - 1, \tag{6}$$

where n_i is the total number of samples observed in X_i and the proportionality factor V can be thought of as a measure of the equivalent volume from which each point is influenced with the specific updating method, and must be set empirically for each particular problem. Note that $\chi^2_\alpha \left(h_i(x, \varsigma_i)\right)$ is not defined for negative values of $h_i(x, \varsigma_i)$, so that in such situation a small positive value should be used for it.

3.3 Competitor Management

As pointed out in the introduction, during the process of FA in RL, the intermediate value functions obtained at the different stages usually have different structures, and could be even more complex than the optimal one [9]. Since the complexity of the intermediate value functions, and of the final optimal value function, is usually unknown in advance, they cannot be expected to fit into a predefined parametric model. In general, non-parametric methods deal better with these complexity variations than parametric ones.

In order to make our approach non-parametric so as to be able to approximate an arbitrary function to any desired precision, new competitors will be created on demand for a better approximation. The idea is to find, through generating competitors, the regions of the domain in which the target function can be well approximated with the parametric function used by each competitor. In principle, any generation mechanism can be employed for this purpose and even a random generation strategy could work, as shown in the example of Section 2.2. However, in this work we propose the use of a more informed generation strategy, as we explain next.

New competitors will be generated when, after observing a new sample (x, y), the approximation error is larger than a threshold e_c corresponding to the maximum error allowed in the approximation, and all the active competitors for x have reached a minimum user-defined confidence threshold n_c,

$$\left(f(x) - \mathcal{F}(x, \mathbf{\Phi}, \mathbf{\Gamma})\right)^2 = \left(y - \Phi_w(x, \xi_w)\right)^2 > e_c,$$

$$n_i \geq n_c, \, \forall i \, | x \in X_i.$$

This last condition is necessary to avoid the generation of further competitors in a region where one has already been generated but has not yet been updated enough to adjust its parameters.

When these two criteria are fulfilled, new competitor domains are generated by two different mechanisms. On the one hand, a new domain is generated by intersecting that of the winner competitor, X_w, with the domain of the competitor that showed the least prediction error at the

sampled point, assuming that they do not coincide. On the other hand, new domains are built by splitting that of the winner, X_w, in three overlapping subdomains, each one of half the size of X_w. Each subdomain is obtained by appropriately cutting X_w in the dimension d corresponding to the longest side of the domain. Generation via splitting complements the generation via intersection of existing domains, since it permits increasing the resolution indefinitely, which is not possible by intersections alone.

Algorithm 1: ICFA (Incremental Competitive Function Approximation)

Initialize a set of N competitors Φ_i and their corresponding relevance functions Γ_i so that their domains form a covering of the domain of the target function.
$n_i \leftarrow 0, \forall i \in \{1...N\}$
$t \leftarrow 1$
loop
  Get new sample (x_t, y_t)
  $I_{x_t} \leftarrow$ set of indexes of active competitors for x_t
  $w \leftarrow$ index of the winner competitor for x_t
  for $i \in I_{x_t}$ **do**
    $e_i(x_t) \leftarrow (y_t - \Phi_i(x_t, \xi_i))^2$ %compute the approximation error of competitor Φ_i
    Update the parameters ξ_i of competitor Φ_i using sample (x_t, y_t)
    Update the relevance function Γ_i by updating $S_i^2(x, \tau_i)$ with sample $(x_t, e_i(x_t))$ and $p_i(x, \varsigma_i)$ with sample x_t
    $n_i \leftarrow n_i + 1$
  end for
  if $(e_w(x_t) > e_c$ and $n_i \geq n_c, \forall i \in I_{x_t})$ **then**
    Generate new competitors by combination
    Generate new competitors by splitting
  end if
  if (t modulo it_{elim}) $= 1$ **then**
    Eliminate useless competitors
  end if
  $t \leftarrow t + 1$
end loop

3.3.1 Elimination of Competitors

The generation process will produce competitors that eventually may prove to be of little use for the approximation. Despite a large amount of competitors improves the chances of generalization, the elimination of the less useful competitors may be convenient to avoid an excessive computational cost in storing and processing them. The elimination of a competitor will not be based on its inaccuracy since, even if a competitor has a large approximation error, it may be the best the system has for some points of the domain. Instead, the competitors eliminated are those which have not won in the last n_a times they have been active, since this means that they are not taking part in the approximation. Once every a certain number of iterations it_{elim}, we eliminate all the competitors that fulfill the elimination criterion.

3.4 The ICFA Algorithm

The complete algorithm for the incremental updating of a CFA is presented in Algorithm 1. Note that, in this algorithm, the initialization and updating of each individual competitor and

relevance functions is performed according to the specific FA approach chosen for them, which is independent of the ICFA algorithm itself. In the following we will assume that an incremental, memory-free FA method is used in all cases.

4 ICFA with Gaussian Mixture Models

As presented above, the ICFA system requires maintaining, for each competitor, three function approximations: one for the estimation of the target function given by $\Phi_i(x, \xi_i)$, a second one for its variance $S_i^2(x, \tau_i)$, and a third one for the sample probability density $p_i(x, \varsigma_i)$, the last two, used to compute the relevance function (5). In principle, each of these estimations could be obtained from an independent FA process with its own memory requirements and updating process. To reduce this burden, we propose a simpler approach in which, instead of maintaining these three function approximations, a single approximation is maintained for the sample density distribution $p_i((x, y))$ in the input-output space of samples (x_t, y_t). From this distribution we can readily obtain the conditional distribution $p_i(y|x_t)$ for the desired input x_t, form which the mean $\mu_i(y|x_t)$ and variance $\sigma_i^2(y|x_t)$ can be obtained, which correspond respectively, to the estimated value $\Phi_i(x_t, \xi_i)$ and variance $S_i^2(x_t, \tau_i)$ of the target function at this point. On the other hand, marginalization of $p_i((x, y))$ for the x variable provides the desired estimation of the sample density in the input space $p_i(x, \varsigma_i)$. Figure 4 provides a schematic representation to illustrate this.

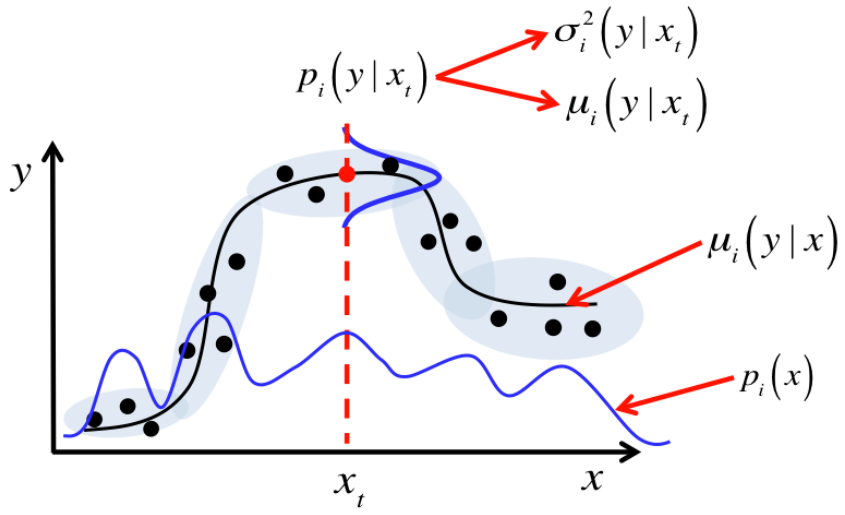


Figure 4: Estimation of the sample density in the input-output space. Dots correspond to observed samples (x_t, y_t), and shaded areas represent the sample density estimation. The conditional distribution $p_i(y|x_t)$ is represented on the vertical line passing through x_t, from which the mean value $\mu_i(y|x_t)$ and variance $\sigma_i^2(y|x_t)$ can be obtained. The curve on the x axis represents the probability density of samples in the input space obtained by marginalization.

A convenient way to approximate the sample density distribution $p_i((x, y))$ is by means of a Gaussian mixture model (GMM), which is a standard tool for such task, and for which incremental updating procedures have been developed. A GMM consists in a set of K multivariate Gaussians defined in the input-output space, and the density estimation at a given point z is

obtained as a weighted average of the values provided by the Gaussians as

$$p(z;\mathbf{\Theta}) = \sum_{j=1}^{K} \alpha_j \mathcal{N}(z;\mu_j,\Sigma_j),$$

where $z = (x, y)$ is the input-output pair; K is the number of Gaussians of the mixture; α_j, usually denoted as the mixing parameter, represents the prior probability $P(j)$ of Gaussian j to generate a sample; $\mathcal{N}(z;\mu_j,\Sigma_j)$ is the multidimensional Gaussian function with mean vector μ_j and covariance matrix Σ_j; and $\mathbf{\Theta} = \{\{\alpha_1,\mu_1,\Sigma_1\},...,\{\alpha_K,\mu_K,\Sigma_K\}\}$ is the whole set of parameters of the mixture. We omitted the subindex i referring to the competitor for clarity in the formulas.

In [1], an incremental version of the Expectation-Maximization algorithm to update the parameters of a GMM is proposed, which incorporates a progressive forgetting of old, possibly outdated observations. This permits a better adaptation to non-stationary target functions, as is the case of the Q-value estimation, and makes the approach well suited for RL, so that we adopted it in our implementation. The details of the updating algorithm will not be provided here, but the interested reader may found them in [1]. The difference between the mentioned work and the present one is that the first uses a single GMM to represent the sample density distribution in the whole domain, and new Gaussians are generated in the GMM when this is required to improve the approximation, thus making the approach non-parametric. In our case, a relatively simpler GMM is embedded in each competitor and, since the accuracy of the approximation is already improved by the addition of new competitors as described in Section 3.3, increasing the complexity of each competitor is not necessary, so that the GMM of each competitor will always stay with the same fixed number K of Gaussians.

4.1 Deriving the conditional and marginal distributions

To obtain the mean value and variance estimations for a given input point x_t, we first need to derive the conditional distribution $p(y|x_t)$. Decomposing the covariances Σ_j and means μ_j in the following way:

$$\mu_j = \begin{pmatrix} \mu_j^x \\ \mu_j^y \end{pmatrix},$$

$$\Sigma_j = \begin{pmatrix} \Sigma_j^{xx} & \Sigma_j^{xy} \\ \Sigma_j^{yx} & \Sigma_j^{yy} \end{pmatrix},$$

the probability distribution of y for the given input x_t, can then be expressed as

$$p(y|x_t) = \sum_{j=1}^{K} \beta_j(x_t)\mathcal{N}\left(y;\mu_j(y|x_t),\sigma_j^2(y)\right), \tag{7}$$

where

$$\mu_j(y|x_t) = \mu_j^y + \Sigma_j^{yx}\left(\Sigma_j^{xx}\right)^{-1}\left(x_t - \mu_j^x\right),$$

$$\sigma_j^2(y) = \Sigma_j^{yy} - \Sigma_j^{yx}\left(\Sigma_j^{xx}\right)^{-1}\Sigma_j^{xy},$$

and

$$\beta_j(x_t) = \frac{\alpha_j \mathcal{N}(x_t;\mu_j^x,\Sigma_j^{xx})}{\sum_{m=1}^{K} \alpha_m \mathcal{N}(x_t;\mu_m^x,\Sigma_m^{xx})}.$$

The competitor value function $\Phi_i(x_t, \xi_i)$ can be obtained for each i as the mean $\mu(y|x_t)$ of the corresponding conditional distribution (7), which is given by

$$\mu(y|x_t) = \sum_{j=1}^{K} \beta_j(x_t)\mu_j(y|x_t).$$

Similarly, the variance $S_i^2(x_t, \tau_i)$ is given by the corresponding $\sigma^2(y|x_t)$,

$$\sigma^2(y|x_t) = \sum_{j=1}^{K} \beta_j(x_t)(\sigma_j^2(y) + (\mu_j(y|x_t) - \mu(y|x_t))^2).$$

The density of samples in the input space $p_i(x, \varsigma_i)$ is obtained as the marginal distribution

$$p(x_t) = \sum_{j=1}^{K} \alpha_j \mathcal{N}(x_t; \mu_j^x, \Sigma_j^{xx}).$$

Note that, in this case, the parameters of the three function approximators of each competitor, i.e. ξ_i, τ_i, and ς_i, are those of the corresponding GMM, $\mathbf{\Theta}_i$.

5 Reinforcement Learning with a Competitive Function Approximation

Our purpose now is to use the ICFA approach as the function approximation method for RL in continuous state and action spaces. The specific RL paradigm we have chosen to illustrate how this can be done is Q-Learning [32, 29]. In this case, the target function to be approximated by the CFA $\mathcal{F}(x, \mathbf{\Phi}, \mathbf{\Gamma})$ is the value function $Q(s, a)$, so that the input variable x consists in the state-action pair (s, a), and each competitor function $\Phi_i((s, a), \xi_i)$ will approximate the $Q(s, a)$ values in its own domain of the state-action space.

As usual in RL, samples are obtained from the interaction with the environment, which is modeled as a Markov decision process, so that, at each time step, the system is in a given state s_t and the agent executes an action a_t, after what the system changes to state s_{t+1} and the agent receives an immediate reward r_t according to certain probability distributions determined by the environment.

Ideally, each action execution should provide an observation of the (stochastic) target function Q, thus giving rise to a training sample consisting of a pair $((s_t, a_t), q^*(s_t, a_t))$, with $q^*(s_t, a_t)$ defined according to the sampled version of the Bellman optimality equation [4] as

$$q^*(s_t, a_t) = r_t + \gamma \max_a(Q^*(s_{t+1}, a)),$$

where γ is the discount factor, and $Q^*(s_{t+1}, a)$ is the true Q-value of the state-action pair (s_{t+1}, a). However, since the true Q-function is actually unknown, the sample value must be estimated using the current value provided by the same CFA that is being learned. Thus, defining $\hat{Q}(s, a) \equiv \mathcal{F}((s, a), \mathbf{\Phi}, \mathbf{\Gamma})$, the sample value will be given by a $q(s_t, a_t)$ obtained as

$$q(s_t, a_t) = r_t + \gamma \max_a(\hat{Q}(s_{t+1}, a)).$$

To compute the value $q(s_t, a_t)$ we need to solve the maximization problem $\max_a(\hat{Q}(s_{t+1}, a))$, but, in order to avoid the complexity of solving such maximization problem when the action set is continuous, we will just estimate its value by computing $\hat{Q}(s_{t+1}, a)$ for a finite number of actions and taking the largest of them.

5.1 Action Selection

A distinguishing trait of RL is that the agent must select what action to execute at each time step, and for this, it has to counterpoise two conflicting aims: on the one hand, it has to maximize the reward received in the long term, so that it should prefer to execute the actions that were found to be more effective for this purpose in the past. On the other hand, the agent must discover what actions that have not yet been tried enough can be even more effective in producing reward, so that there is a need to execute exploratory actions whose result is uncertain. This is the well known exploration-exploitation dilemma. A pure exploitation policy, known as the greedy policy, would select, for each state, the action with the highest Q-value as estimated by the current approximation,

$$a = \underset{a'}{\operatorname{argmax}}\, \hat{Q}(s, a'). \tag{8}$$

To allow for the execution of exploratory actions, a simple strategy consists in what is known as the ϵ-greedy policy, that takes the greedy action (8) with probability $1-\epsilon$, and an exploratory random action with probability ϵ. A more sophisticated strategy is the Boltzmann exploration [29] which, in the case of a discrete action-set, selects the action to execute according to a probability given by the values $\hat{Q}(s, a)$ of each alternative action,

$$p\,(a|s) = \frac{e^{\hat{Q}(s,a)/T}}{\sum_{a'} e^{\hat{Q}(s,a')/T}},$$

where T is a positive parameter, called the temperature, that is made to decrease with time, and which regulates the influence of the values $\hat{Q}(s, a)$ in the probability: the higher the temperature the lower the influence of $\hat{Q}(s, a)$. A high temperature is preferred at the beginning of the learning process to decrease the influence of still poor estimations of $\hat{Q}(s, a)$. In contrast with the ϵ-greedy policy, which selects exploratory actions blindly, the Boltzmann exploration is biased towards executing the more promising actions more often than less promising ones, what should improve the overall performance. However, while it takes into account the estimated Q-value of each action, it ignores the confidence with which each estimation is known. To address this issue, [18] proposed an alternative action selection strategy that takes into account the uncertainty in the estimation of $\hat{Q}(s, a)$. It is based on computing a confidence interval for $\hat{Q}(s, a)$ and executing the action with the highest upper bound. By doing this, actions that have been tried a fewer number of times will have larger confidence intervals, increasing their opportunities to be executed. However, since this strategy is deterministic, it is vulnerable to sticking (a phenomenon by which a statistically unlikely series of observations makes that a good action will never be executed again), and while it favors the exploration of less executed actions, it may not exploit enough the already acquired knowledge: even if it is well known that an action produces a high reward, it will never be executed while there is some other action with so little confidence as to have a larger interval upper bound.

Probabilistic Confidence-based Action Selection

We propose an action selection strategy that, like in the Boltzmann strategy, actions are not selected in a deterministic way, but according to a probability distribution and, like in the strategy of [18], both the current estimation of $Q(s, a)$ and the confidence on this estimation are taken into account. For this, we will use a known result from statistics which states that, if μ is the expected value of a random variable, $\bar{Y}$ is the sample mean of a set of n observations of this variable, and S^2 is the sample variance, then the following random variable has a t-distribution with $n-1$ degrees of freedom [7],

$$\frac{\bar{Y}-\mu}{\sqrt{S^2/n}} \sim t(n-1). \tag{9}$$

Taking $q(s,a)$ as the sampled random variable, μ corresponds to its true (unknown) expected value $Q(s,a)$; the sample mean $\bar{Y}$ can be approximated by the estimation $\hat{Q}(s,a)$ given by the winner competitor Φ_w for (s,a); the sample variance S^2, by the point dependent variance estimation of the winner competitor, $S^2_w((s,a),\tau_w)$; and the number of samples n, by the density-based estimation (6) of the winner competitor $n_w(s,a) = h_w((s,a),\varsigma_w) + 1$, so that we can approximate Eq. (9) by

$$\frac{\hat{Q}(s,a)-Q(s,a)}{\sqrt{S^2_w((s,a),\tau_w)/n_w(s,a)}} \sim t(n_w(s,a)-1). \tag{10}$$

To select the action to execute, we consider only a finite set of m actions $a_j, j = 1...m$ and we evaluate each of them in a probabilistic way as follows: first, we take a value t_j for the random variable of the left term of (10) probabilistically, according to a t-distribution with $n_w(s,a_j)-1$ degrees of freedom. From this value, we derive the randomized value $Q_{rnd}(s,a_j)$ that will be used to evaluate action a_j as

$$Q_{rnd}(s,a_j) = \hat{Q}(s,a_j) - t_j\sqrt{\frac{S^2_w((s,a_j),\tau_w)}{n_w(s,a_j)}}.$$

Once a $Q_{rnd}(s,a_j)$ value has been assigned to every action a_j in the current state, the action with the highest randomized value is selected for execution,

$$a = \underset{a_j}{\operatorname{argmax}}\, Q_{rnd}(s,a_j).$$

With this action selection strategy, each possible action a_j has a non-zero probability of execution and, in the long run, will be selected an unbounded number of times as required for the correct convergence of any asynchronous dynamic programming algorithm. Moreover, the exploration-exploitation trade-off is solved by giving higher probabilities to those actions whose expected Q-value is larger or whose estimations are less certain, in a well balanced approach based on statistically justified grounds. Note that, as the confidences increase, the amount of exploration automatically decreases without any explicit schedule or parameter settings, as required by the Boltzmann strategy for the temperature. It must be emphasized that, while this form of action selection requires the estimations of $S^2_w((s,a),\tau_w)$ and $n_w(s,a)$, this does not represent any additional cost, since they are already available for the computation of the relevance of each competitor.

5.2 Algorithm for Q-Learning with a Competitive Function Approximation

When the ICFA is used to approximate the value function in Q-learning, we can use Algorithm 1 as is, with the particularity that the samples used to update the system are obtained through interaction with the environment following the action selection procedure just described. All the steps necessary to obtain a new sample (x_t, y_t) at each cycle of the main algorithm are summarized in Algorithm 2, which replaces the step "Get new sample (x_t, y_t)" in Algorithm 1. Algorithm 2 contains two *for* loops, the first is to evaluate each of the m possible actions to execute, and the second to evaluate the Q-value of the next state for m' actions, but note that they do not need to be necessarily the same set of actions in the two cases. The resulting algorithm that combines the ICFA method of function approximation with RL will be denoted as ICFARL.

Algorithm 2: The "Get new sample (x_t, y_t)" step of the ICFA algorithm with Q-Learning

$s_t \leftarrow$ current state
for $j = 1$ to m**do**
  $w_j \leftarrow$ index of the winner competitor for (s_t, a_j)
  $\hat{Q}(s_t, a_j) \leftarrow \Phi_{w_j}(s_t, a_j)$
  $t_j \leftarrow$ random value generated with a t-distribution with $n_{w_j}(s_t, a_j) - 1$ d.o.f.
  $Q_{rnd}(s_t, a_j) \leftarrow \hat{Q}(s_t, a_j) - t_j \sqrt{\frac{S^2_{w_j}((s_t,a_j),\tau_{w_j})}{n_{w_j}(s_t,a_j)}}$
end for
$a_t \leftarrow \underset{a_j}{\operatorname{argmax}}\, Q_{rnd}(s_t, a_j)$ % Action selection
Execute a_t and observe reward r_t and next state s_{t+1}
for $j = 1$ to m' **do**
  $w_j \leftarrow$ index of the winner competitor for (s_{t+1}, a_j)
  $\hat{Q}(s_{t+1}, a_j) \leftarrow \Phi_{w_j}(s_{t+1}, a_j)$
end for
$\hat{Q}_{s_{t+1}} \leftarrow \max_{a_j} \hat{Q}(s_{t+1}, a_j)$ % Next state evaluation
$q(s_t, a_t) \leftarrow r_t + \gamma \hat{Q}_{s_{t+1}}$
$(x_t, y_t) \leftarrow ((s_t, a_t), q(s_t, a_t))$

6 Experiments

In order to assess the generality and efficiency of the approach we will consider three different tasks that cover the basic types of difficulty in control problems [27]: *avoidance*, which consists in keeping the system into a valid region of the state space; *goal reaching*, where the system is requested to reach a goal area of the state space and finish the task when it gets there; and *regulation*, where the system is also requested to reach a goal but, instead of finishing the task when this happens, it should stay there using active control. For each of these tasks, we will use a standard benchmark of reinforcement learning: the cart-pole balancing for the avoidance task, the mountain-car for the goal reaching task, and the inverted pendulum for the regulation task. With these benchmarks, we will not only assess the generality and efficiency of the ICFARL method but, to some extent, also its scalability, since the state space of the cart-pole balancing has two more dimensions than the other two benchmarks.

For each benchmark, the performance of ICFARL will be compared with that of Q-learning using two different FA techniques: an incremental learning version of Variable Resolution (VR), and a single global Gaussian Mixture Model corresponding to the GMMRL method described in [1]. The first will serve as a basic reference with a well known standard approach, while the GMMRL method will serve to asses the difference in performance specifically attributable to the competitive approach. For both methods, the same "Get new sample (x_t, y_t)" step described in Algorithm 2 is implemented, just noting that the winner competitor must be understood as the only approximator available at each input point. In particular, action selection is performed with the probabilistic confidence-based approach described in Sec. 5.1, what in the case of the VR method implies that additional statistics must be maintained for the variance of each aggregated state. The parameters of each method are optimized through a non-exhaustive search for each experiment.

6.1 A Regulation Problem: the Inverted Pendulum

The inverted pendulum benchmark [13] is depicted in Figure 5. The task consists in swinging an under-actuated pendulum until reaching the upright position and then staying there indefinitely. The optimal policy for this problem is not trivial since, due to the limited torques available, the controller has to swing the pendulum several times until its kinetic energy is large enough to overcome the load torque and reach the upright position. Thanks to its simplicity but still challenging control task, this benchmark was widely used to test the performance of state of the art methods for FA in RL [28, 12, 10, 27].

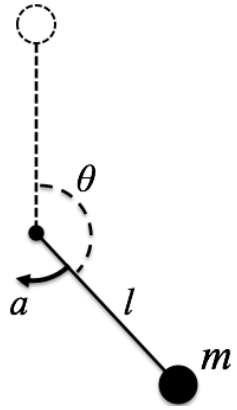


Figure 5: Inverted pendulum benchmark.

The dynamics of the inverted pendulum is modelled as

$$ml^2\ddot{\theta} = -\mu\dot{\theta} + mgl\sin\theta + a,$$

where θ is the angular position, l is the length of the link, m the mass of the pendulum (assumed to be concentrated at a point at the end of the link), g the gravity constant, μ the friction factor, and a the input torque. In our simulations we use the values $l = m = 1$, $g = 9.8$, $\mu = 0.01$, and an input torque a ranging in $[-5, 5]$. For the simulation we use the Euler method with a differential for time of $dt = 0.001$ and an actuation interval of 0.1.

The state-action space is three-dimensional, configured by the angular position θ, the angular velocity $\dot{\theta}$, and the action a. As the reward signal we simply take minus the absolute value of the angle of the pendulum from its top position: $r(\theta, \dot{\theta}) = -|\theta|$ which ranges in the interval $[-\pi, 0]$. The discount factor γ is set to 0.85.

6.1.1 Set-up for the Inverted Pendulum Experiments

Each competitor of the ICFARL algorithm is provided with a GMM with $K = 10$ Gaussians defined in the four-dimensional space of vectors $(\theta, \dot{\theta}, a, q)$. To define the initial set of competitors, we form a covering of this space that is composed of 10 subsets, one corresponding to the entire space, and three more for each of the input variables, obtained by segmentation of the whole space along three overlapping intervals of the corresponding variable. Figure 6 shows a schematic view of the domains generated in this way in the input space in the case that it is two-dimensional.

For this experiment, we implemented two instances of the GMMRL method, one with an initial number of Gaussians $K = 10$, the same as for each competitor of the ICFARL, and a second one with $K = 100$, so that its initial number of Gaussians coincides with total number

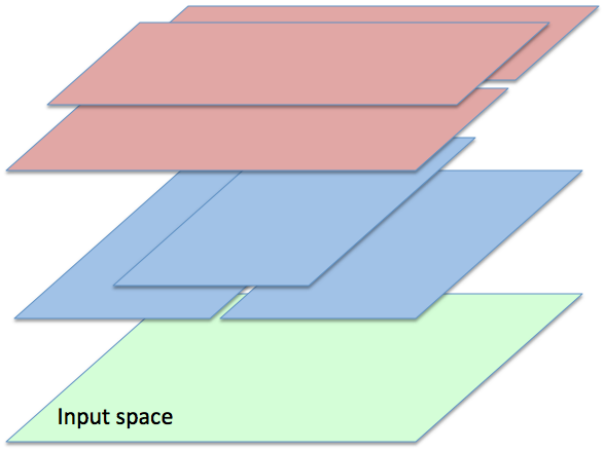


Figure 6: Example of the domains of the initial competitors in a 2D input space.

of Gaussians of the initial ICFARL competitors. For the two maximization processes involved in the action selection and next-state evaluation steps (cf. Algorithm 2), we use the same set of discrete actions, obtained by subdividing the whole action range in intervals of length 0.1 and taking their bounds. The factor V introduced in (6) is taken equal to 1.

The experiments are performed using episodes of 500 iterations. At the beginning of each episode, the pendulum is placed in the hang-down position. At the end of each episode, a test of 500 iterations is performed exploiting the policy learned so far, with no learning nor exploration. As the result of the test we take the sum of the rewards obtained at each iteration. For each experiment we show the average of 15 independent runs of 100 episodes.

6.1.2 Results

Figure 7 shows the results of the experiments comparing the performance of ICFARL, the VR approach, and the two instances of the GMMRL method. In this experiment, the ICFARL algorithm reaches a long term performace similar to that of the GMMRL with 10 initial Gaussians, though it learns significantly faster at the beginning. Notably, the GMMRL with 100 initial Gaussians shows a much poorer performance, though still outperforming the VR approach. This is likely due to the fact that, in this case, the system must simultaneously balance a large number of parameters (in fact, 100 Gaussians is far more than required, since the GMMRL with 10 initial Gaussians reaches a good performance with an average of about 45 Gaussians).

6.2 A Goal Reaching Problem: the Mountain-Car

The benchmark of the mountain-car [29] consists in driving an underpowered car up a steep mountain road (Figure 8). The difficulty of the problem is that the force exerted by gravity is larger than the force of the engine of the car. Then, in order to climb the mountain, the car needs to first move away from the mountain up to the opposite slope, and then apply maximum acceleration to accumulate enough energy to reach the tip of the mountain.

The equations modelling the dynamics of the car are

$$\begin{aligned} p_{t+1} &= \text{bound}\left[p_t + v_{t+1}\right], \\ v_{t+1} &= \text{bound}\left[v_t + 0.001a_t + (-0.0025)\cos(3p_t)\right], \end{aligned}$$

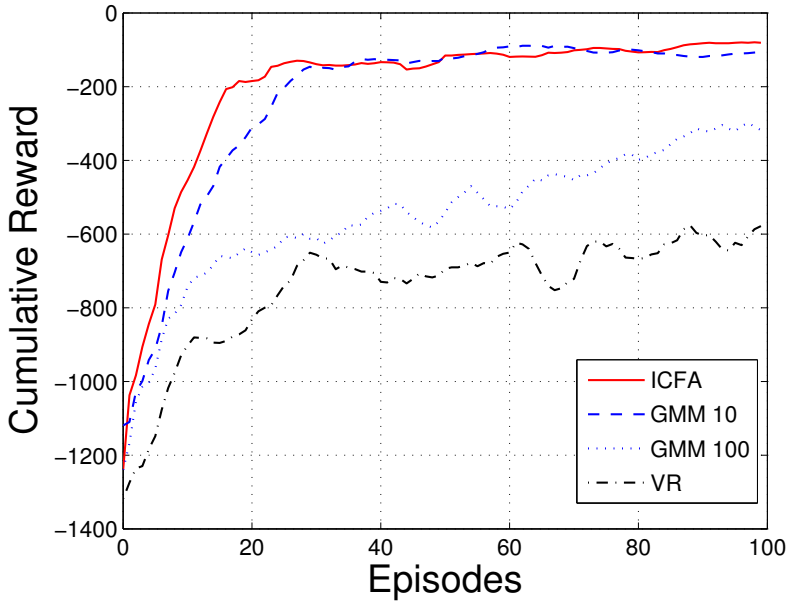


Figure 7: Comparison of performance of the different approaches on the inverted pendulum.

where p_t is the car position at time t, v_t is the car velocity, and a_t is the discrete action consisting in the car acceleration forward, $a_t = 1$, backward, $a_t = -1$, or no acceleration $a_t = 0$. The operator bound$[\cdot]$ keeps the position and velocity in the intervals $[-1.2, 0.5]$ and $[-0.07, 0.07]$, respectively. When the car reaches the left position bound, its velocity is set to 0. The car arrives at the goal condition when the position $p_t = 0.5$ is reached, in which case the reward is $r = 0$ and the episode terminates. In any other case the reward is $r = -1$. For this task the discount factor γ is set to 0.999.

6.2.1 Set-up for the Mountain-Car Experiments

Since in this problem the action set is discrete, we use an independent function approximation process of the Q function for each action $a \in \{-1, 0, 1\}$. Each one of the three ICFA's of the ICFARL system is initialized with 22 competitors whose domains are composed of 16 subsets forming a regular grid on the (p, v) space, plus 6 overlapping regions built as shown in Figure 6. Each competitor is provided with $K = 10$ Gaussians, which in this case are defined in the three-dimensional space of vectors (p, v, q). Each one of the three GMM of the GMMRL system is also provided with an initial number of $K = 10$ Gaussians. The factor V introduced in (6) is taken equal to 1.

In the experiments, we perform training episodes that run until the car reaches the goal position or until a maximum of 300 iterations have been executed. Each time 600 new iterations have been accumulated, a test episode is carried out exploiting the policy learned so far. As the result of the test we take the sum of the rewards obtained in the test episode. Both, training and test episodes are initiated in a state selected randomly. For each experiment we show the average of 15 independent runs.

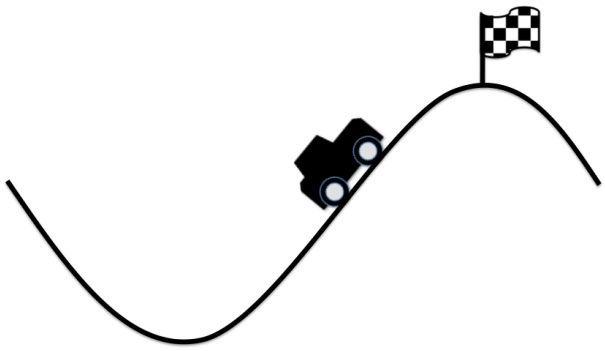


Figure 8: Mountain-car benchmark.

6.2.2 Results

The results are presented in Figure 9. It can be seen that, while the difference in the performance of the VR and GMMRL methods are not as large as in the previous experiment, the ICFARL reaches a significantly higher performance and has a more stable behavior than the VR and GMMRL methods in this problem.

6.3 An Avoidance Problem: the Cart-Pole Balancing

The Cart-Pole Balancing benchmark [3] consists in a pole mounted on a cart that has to be stabilized inside an acceptable region by the motions of the cart (Figure 10). The cart is free to move within a bounded range of a linear track, and the pole moves in a vertical plane parallel to this track.

The equations for the model of the cart-pole system are

$$\begin{aligned} \ddot{\theta} &= \frac{g\sin\theta + \cos\theta\left[-F - m_p l\dot{\theta}\sin\theta + \mu_c \mathrm{sgn}(\dot{x})\right] - \frac{\mu_p\dot{\theta}}{m_p l}}{l\left[\frac{4}{3} - \frac{m_p\cos^2\theta}{m_c+m_p}\right]}, \\ \ddot{x} &= \frac{F + m_p l\left[\dot{\theta}\sin\theta - \ddot{\theta}\cos\theta\right] - \mu_c\mathrm{sgn}(\dot{x})}{m_c+m_p}, \end{aligned}$$

where x is the cart position, θ is the angular position of the pole, m_p is the mass of the pole, l is the length of the pole, m_c is the mass of the cart, $F = a$ is the control action, consisting in the force applied to the cart, g is the gravity constant, and μ_c and μ_p are the friction coefficients of the cart and the pole, respectively. The parameters of the model are set to $m_p = 0.15$, $m_c = 1.0$, $l = 0.75$, and $g = 9.81$. In our experiments, the friction coefficients are set to 0. The acceptable region of the pole is defined by $-\pi/6 \leq \theta \leq \pi/6$, and the one corresponding to the cart is defined by $-1.5 \leq x \leq 1.5$. The action takes values within the interval $[-50, 50]$, and the actuation frequency is 60 Hz. While the cart-pole system remains in its acceptable region, the reward is a linear function of the absolute value of θ, taking the value 1 for $\theta = 0$, and 0 at the bound of its acceptable range. When either the pole or the cart leaves its acceptable region, a reward $r = -1$ is provided and the episode terminates. The discount factor is set to $\gamma = 0.95$.

The cart-pole benchmark is of interest since it involves a five-dimensional state-action space, $(s, a) = (x, \dot{x}, \theta, \dot{\theta}, a)$, two more than the inverted pendulum and mountain-car, and serves to illustrate the scalability of the algorithm.

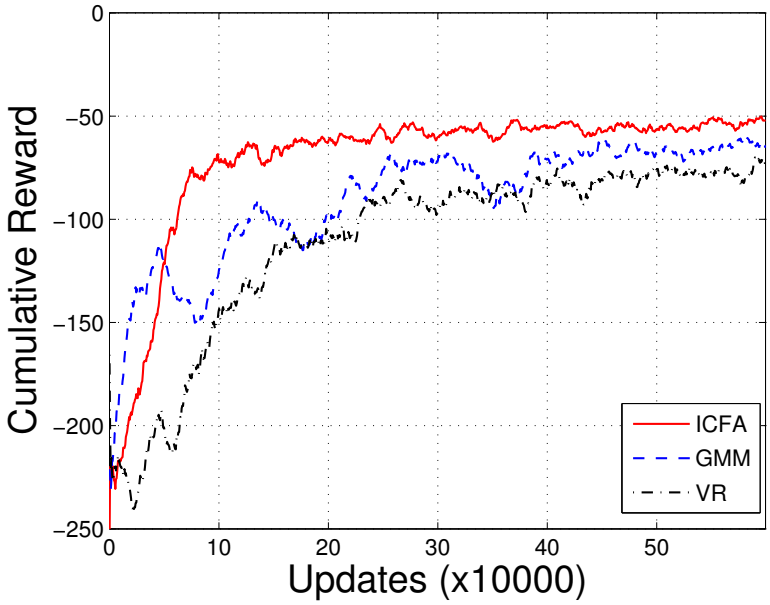


Figure 9: Performance of the ICFARL, GMMRL, and VR methods for the mountain-car.

6.3.1 Set-up for the Cart-Pole Experiments

We provide the ICFARL system with an initial set of 15 competitors generated by splitting the range of each variable in the state-action space in three overlapped segments, as illustrated in Figure 6. Each competitor is provided with $K = 20$ Gaussians defined in the six-dimensional space of vectors $(x, \dot{x}, \theta, \dot{\theta}, a, q)$. The GMMRL system is also provided with an initial number of $K = 20$ Gaussians. For the two maximization processes involved in the action selection and next state evaluation steps of Algorithm 2, we use a discrete set of actions obtained by subdividing the whole action range in intervals of length 2 and taking their bounds. The factor V introduced in (6) is taken equal to 50.

The set-up for the experiments is basically the same used in [25]. A training episode starts in a random state $s_0 = (x_0, \dot{x}_0, \theta_0, \dot{\theta}_0)$ obtained at random from a normal distribution with mean vector $\mu_0 = (0, 0, 0, 0)$ and covariance matrix $\Sigma_0 = diag(0.1, 0.1, 0.1, 0.1)$. Each training episode lasts for 5 seconds or until the system fails in keeping the pole, or the cart, in the acceptable region. After 10 seconds of simulated training time, a test is performed. Each test consists in four episodes of 5 seconds of simulated time, or until failure, starting at four different positions of the pole: $-10\,^\circ, -5\,^\circ, 5\,^\circ, 10\,^\circ$, respectively, and with the cart centered on the track. The result of the test is obtained by averaging the sums of rewards obtained in the four episodes. For each experiment we calculate the average of 15 independent runs.

6.3.2 Results

Figure 11 presents the results of the experiments where an estimation of the maximum possible reward, corresponding to the best controller obtained by exhaustive manual tuning, is also shown. Note that the ICFARL converges very fast to a near optimal policy, and keeps improving with time approaching the optimal controller asymptotically. Its performance on this problem is much better than that of the GMMRL and the VR approaches, which keep also improving with time, but at much slower rates.

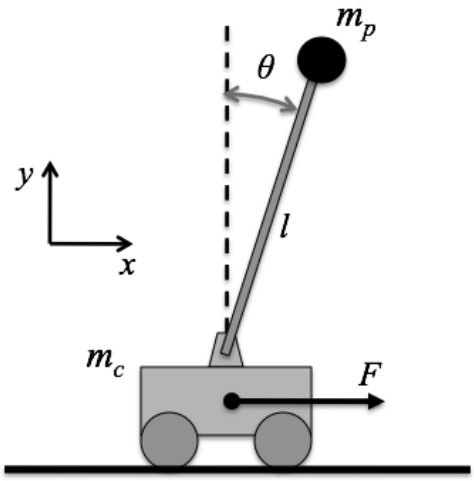


Figure 10: Cart-pole balancing benchmark.

6.4 Discussion

The results of the experiments show that, in all cases, the ICFARL method has a better performance than its non-competitive counterpart, the GMMRL method, which, in turn, outperforms the VR method used for reference. However, the differences found between the three methods vary a lot from one benchmark to another. The different characteristics shown by the three benchmarks give us the opportunity of evaluating the relative performance of the ICFARL method on different aspects that will be analyzed next.

Adaptability to Different Complexities

In the first experiment, corresponding to the inverted pendulum, the performances of ICFARL and GMMRL with 10 initial Gaussians are very similar in the long run, though ICFARL shows a better performance at the beginning, reaching a near-optimal performance significantly faster. To interpret this result we must distinguish two phases of this problem: the first phase consists in moving the pendulum back and forth to drive it to the up position. The second phase consists in, once the up position has been reached, stabilizing the pendulum there. The second phase is relatively simpler, and there is no apparent advantage of one method over the other to accomplish it. However, solving the first phase requires a precise enough determination of the correct point at which the direction of the acceleration must be inverted. This corresponds to a discontinuity in the Q function. Such a discontinuity is difficult to represent with the global approximation of GMMRL, but can be easily done with the competitive approach of ICFARL if, eventually, competitors are generated near, but not including, the discontinuity, so that they are not affected by the too dissimilar Q values found beyond the discontinuity. An alternative way, perhaps more interesting, in which ICFARL may account for a discontinuity occurs when each of two overlapping competitors, both including the discontinuity, makes a better approximation of the function at one side of the discontinuity than the other competitor. In this case their corresponding relevance functions should cross each other near the discontinuity, resulting in the selection of a different action at each side. The ability of the ICFARL method to deal with discontinuous Q functions provides a likely account of its better performance over GMMRL in the first phase of the experiment.

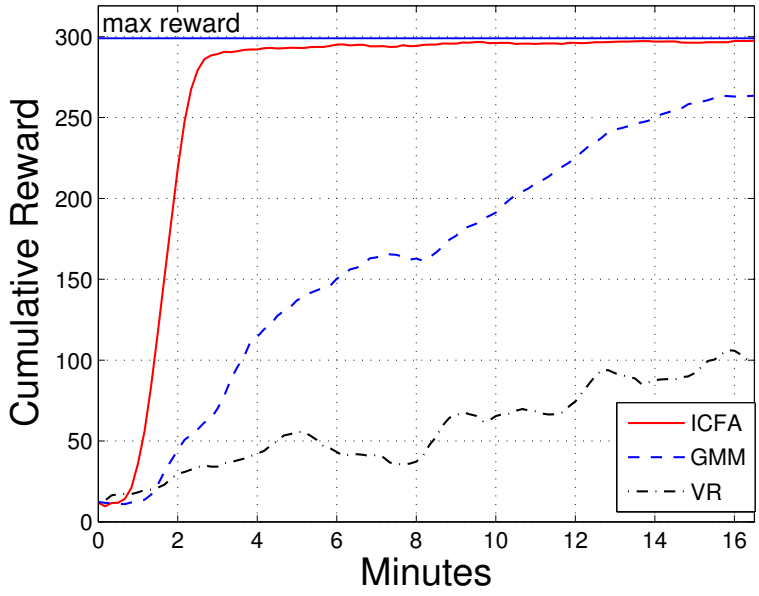


Figure 11: Performance of the ICFARL, GMMRL, and VR methods for the cart-pole.

Dealing with Biased Sampling

In the mountain-car experiment, receiving continued negative reward is only avoided at the goal state, but the system has no cue of how to approach it before reaching it for the first time, what is only possible by chaining a long series of correct actions by accident. This implies that most of the actions will occur essentially at random and most of the situations experienced by the system will be near the lower part of the road. This is a clear example of the biased sampling problem common in RL. Biased sampling causes difficulties to the GMMRL approach (and to most global FA methods), since the information provided by an eventual goal reaching will be followed by a wealth of non-goal experiences that will inevitably fade the contribution of the goal experience. This explains the instability shown by the performance curve of the GMMRL method in Figure 9, revealing that unlearning is taking place due to biased sampling. Local methods like ICFARL can deal better with biased sampling, since each local approximator is only updated by samples in its own domain, so that the information gained from an isolated goal experience can be better preserved. This fact is confirmed by the results of Figure 9 showing the better performance and more stable behavior of ICFARL. Note that locality is also a feature of the VR approach, what can explain its relatively good performance in the mountain-car benchmark in comparison with that of the other two.

Scalability

The third experiment, corresponding to the cart-pole, presents a different kind of difficulty: it involves a state space of a higher dimension. A higher dimension implies a much larger function domain, so that the complexity required by a global function approximator increases and, as argued in Section 2.1, adjusting a complex global function is harder than adjusting a set of simpler local functions covering the same domain. This is the reason why GMMRL takes much longer to learn than ICFARL. On the other hand, the subdivision strategy of the VR approach

becomes more and more inefficient as the dimension increases, and, as suggested in Section 2.2, forming a covering with overlapping competitors may be much more effective. The large increase in performance of ICFARL with respect to the other two methods demonstrates the good scalability properties of the approach.

On the Computational Cost

As could be expected, the computational cost per iteration of the ICFARL method is higher than that of the GMMRL method, essentially due to the larger total number of Gaussians used to approximate the function with the competitive approach. The average number of Gaussians required to reach a good performance were, roughly, 2000 for the inverted pendulum, 15000 for the mountain-car (considering all the actions), and 4400 for the cart-pole experiments, while the GMMRL approach required of the 45, 600, and 400 Gaussians, respectively. Despite this, the Gaussians that must be updated at each ICFARL iteration are just those of the competitors that are active in the current situation. As a consequence, in all the experiments, the effective cost per sample resulted in less than one order of magnitude higher in ICFARL than in GMMRL.

7 Conclusions

The ICFARL system has been developed to address some specific difficulties encountered by FA in the context of RL, which we identified as the non-stationarity of the target function and the biased sampling problem. To deal with the non-stationarity of the target function, the FA must be able to adapt rapidly to the new observations, and ICFARL achieves this by the use of local approximators that, thanks to its relative simplicity, may adapt much faster than a global FA for the entire domain. At the same time, the local nature of the competitors makes them much less affected by biased sampling, since no matter how many samples fall in a given region, they have no effect on those competitors whose domains do not include it.

The performance of the method is further increased by the fact that competitors have overlapped domains that are updated in parallel, and, thanks to the relevance function, the system is able to select the competitor that is more accurate and confident at each point of the domain. With this approach, when a sudden change of the target function occurs at some point of the domain, it is not necessary that the formerly winner competitor at this point completely reshape its approximation for the output to become accurate again; instead, a different competitor that manages to be more accurate at that point will provide the output as soon as its relevance gets larger there.

A key aspect in the competitive strategy is the proper definition of the relevance function that, associated to each approximator, permits selecting the best approximator in a point. We proposed a relevance function that uses a point-wise estimation of the variance as a measure of the accuracy of the approximation, and a point-wise estimation of the sample density as a measure of the confidence in the approximation. The results obtained in the experiments support the validity of our proposal and corroborates the advantage of the competitive approach for FA, which we expect that can be used in many other contexts not necessarily related with RL.

References

[1] A. Agostini and E. Celaya. Reinforcement Learning with a Gaussian Mixture Model. In *Proc. International Joint Conference on Neural Networks (IJCNN'10). Barcelona, Spain*, pages 3485–3492, 2010.

[2] A. Agostini and E. Celaya. A Competitive Strategy for Function Approximation in Q-Learning. In *Proceeding of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11). Barcelona, Spain*, pages 1146–1151, 2011.

[3] A.G. Barto, R.S. Sutton, and C.W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on systems, man, and cybernetics*, 13(5):834–846, 1983.

[4] R. Bellman. Dynamic Programming. *NJ, Princeton UP*, 1957.

[5] A. Bernstein and N. Shimkin. Adaptive-resolution reinforcement learning with polynomial exploration in deterministic domains. *Machine Learning*, 81(3):359–397, 2010.

[6] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[7] G. Blom. *Probability and statistics: theory and applications*. Springer-Verlag, 1989.

[8] A. Bonarini, A. Lazaric, and M. Restelli. Reinforcement Learning in Complex Environments Through Multiple Adaptive Partitions. *Lecture Notes in Computer Science*, 4733:531–542, 2007.

[9] J. Boyan and A.W. Moore. Generalization in reinforcement learning: Safely approximating the value function. *Advances in neural information processing systems*, pages 369–376, 1995.

[10] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming using Function Approximators*. Taylor & Francis CRC Press, 2010.

[11] E.W. Cheney. *Introduction to approximation theory*. Amer Mathematical Society, 1998.

[12] M.P. Deisenroth, C.E. Rasmussen, and J. Peters. Gaussian process dynamic programming. *Neurocomputing*, 72(7-9):1508–1524, 2009.

[13] K. Doya. Reinforcement learning in continuous time and space. *Neural Comput.*, 12(1):219–245, 2000.

[14] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley and Sons, Inc, New-York, USA, 2001.

[15] J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[16] G.J. Gordon. Stable Function Approximation in Dynamic Programming. Technical Report CS-95-130, CMU, 1995.

[17] S. Jodogne and J. Piater. Task-driven discretization of the joint space of visual percepts and continuous actions. *Machine Learning: ECML 2006*, pages 222–233, 2006.

[18] L.P. Kaelbling. *Learning in embedded systems*. The MIT Press, 1993.

[19] I. Menache, S. Mannor, and N. Shimkin. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, 134(1):215–238, 2005.

[20] T. Mitchell. *Machine Learning*. McGraw-Hill Education (ISE Editions), October 1997.

[21] CK Monson, D. Wingate, KD Seppi, and TS Peterson. Variable resolution discretization in the joint space. In *Proceedings of the 2004 International Conference on Machine Learning and Applications.*, pages 449–455. IEEE, 2004.

[22] A.W. Moore. Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 333–337, 1991.

[23] R. Munos and A. Moore. Variable resolution discretization in optimal control. *Machine learning*, 49(2):291–323, 2002.

[24] D. Ormoneit and Ś. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2):161–178, 2002.

[25] J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

[26] S.I. Reynolds. Adaptive resolution model-free reinforcement learning: Decision boundary partitioning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 783–790, 2000.

[27] M. Riedmiller. Neural fitted Q iteration-first experiences with a data efficient neural reinforcement learning method. In *Proceedings of the European Conference on Machine Learning*, pages 317–328, 2005.

[28] A. Rottmann and W. Burgard. Adaptive autonomous control using online value iteration with Gaussian processes. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 3033–3038, 2009.

[29] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[30] H. Vollbrecht. Hierarchic function approximation in kd-q-learning. In *Knowledge-Based Intelligent Engineering Systems and Allied Technologies, 2000. Proceedings. Fourth International Conference on*, volume 2, pages 466–469. IEEE, 2000.

[31] H. Vollbrecht. *Hierarchical reinforcement learning in continuous state spaces*. PhD thesis, Ph. D Dissertation. University of Ulm, Germany, 2003.

[32] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[33] S. Whiteson. *Adaptive representations for reinforcement learning*. Springer Verlag, 2010.

IRI reports

This report is in the series of IRI technical reports.
All IRI technical reports are available for download at the IRI website
`http://www.iri.upc.edu`.