A Competitive Strategy for Function Approximation in Q-Learning

Alejandro Agostini and Enric Celaya
Institut de Robòtica i Informàtica Industrial (CSIC-UPC)
Barcelona, Spain
{agostini,celaya}@iri.upc.edu

Abstract

In this work we propose an approach for generalization in continuous domain Reinforcement Learning that, instead of using a single function approximator, tries many different function approximators in parallel, each one defined in a different region of the domain. Associated with each approximator is a relevance function that locally quantifies the quality of its approximation, so that, at each input point, the approximator with highest relevance can be selected. The relevance function is defined using parametric estimations of the variance of the q-values and the density of samples in the input space, which are used to quantify the accuracy and the confidence in the approximation, respectively. These parametric estimations are obtained from a probability density distribution represented as a Gaussian Mixture Model embedded in the input-output space of each approximator. In our experiments, the proposed approach required a lesser number of experiences for learning and produced more stable convergence profiles than when using a single function approximator.

1 Introduction

It is known that generalization in Reinforcement Learning (RL) is mandatory in applications with large domains [Sutton and Barto, 1998]. Generalization usually involves some kind of Function Approximation (FA) and this is a very challenging problem in RL since, during the learning process, the current estimation of the value function, varies not only as a consequence of the iterative policy evaluation process, where the value function is estimated for the current policy, but also as a consequence of the policy improvement process, which progressively changes the policy towards the optimal one. On the other hand, in incremental RL, data arrive sequentially, following trajectories in the domain dictated by the dynamics of the environment and the current action selection strategy. This causes sampling to be very biased to certain regions, posing a serious difficulty to the FA method to preserve the estimation at sparsely sampled, though also relevant, regions. The non-stationarity problem, and the biased sampling problem usually appear in different degrees at different regions of the domain, and at different stages of the learning process [Boyan and Moore, 1995]. All these problems make the selection of the generalization method a fundamental issue for successful learning.

One way to address the generalization problems is through *fitted value iteration* [Gordon, 1995], which consists in adjusting the approximation so that the error is minimized on a finite number of preselected domain points that represent the problem sufficiently well. Fitted value iteration takes the data for the given points and tries to fit the function to them by iteratively processing the whole set in a batch process, so that all points get uniformly re-sampled, avoiding the biased sampling problem. Though, in principle, fitted value iteration cannot be considered as an incremental RL method, on-line versions have been proposed like the Neural Fitted Q Iteration (NFQ) [Riedmiller, 2005], in which the set of points is periodically redefined by collecting new samples along trajectories obtained using the policy arrived at so far. Note, indeed, that learning will succeed only if the set of points eventually cover all the relevant parts of the domain, what may be hard to achieve when the dynamics of the problem is not well known. NFQ uses a neural net for function approximation. Other techniques that fall into the fitted value iteration class use Gaussian Processes (GP) for function approximation [Rottmann and Burgard, 2009; Deisenroth *et al.*, 2009; Engel *et al.*, 2005]. They have the advantage of being non-parametric, what makes the class of functions that can be represented much more flexible. In addition, besides providing the expected value of the function, GP also provide its variance, what allows to quantify the uncertainty of the predicted value. As pointed out in [Engel *et al.*, 2005], this information may be very useful to direct the exploration in RL. Similarly, [Agostini and Celaya, 2010] use a Gaussian Mixture Model (GMM) to represent the sample density in the joint input-output space of the function, from which the expected values and their variances can be also obtained. This algorithm, which is not of the fitted value iteration class, uses an on-line Expectation-Maximization algorithm that makes the approach incremental, and tries to solve the biased sampling and the local non-stationarity problems by making the effect of each update sufficiently local.

A more drastic way of keeping the updating local, is that of variable resolution techniques [Moore, 1991], in which the domain is partitioned into regions that are updated indepen-

dently of each other, and may be further subdivided when their resolution proves to be insufficient. Main drawbacks of this approach are the sometimes unnecessary proliferation of small regions, and the lack of generalization between nearby regions once they have been partitioned.

In this work, we propose the use of many competing function approximators in parallel, each one with a restricted domain, but in which, unlike variable resolution, their domains may overlap in different ways, so that each point is covered by multiple approximators. Trying different approximators in parallel increases the chances of having a good approximator among the competing ones, avoiding the need of adjusting a global approximator complex enough to represent the function in the whole domain. Having approximators with domains of different sizes increases the opportunities to end with the right granularity for generalization, so that the effect of an update reaches distant enough related points, while keeping far away unrelated regions unaffected. An important aspect of this approach is the selection of the competitor that will be used to determine the function value at each point of the domain. For this, a relevance function is computed for each competitor, taking into account its expected accuracy and confidence at each point. The competitor that has the highest relevance in a given point is used to provide the value at this point. Using this approach for Q-learning would require, in principle, the estimation of three functions for each competitor: one for the q-value, one for the accuracy, and a third one for the confidence at each state-action point. Fortunately, we can obtain estimations of all these functions from a unique probability density function in the joint space of states, actions, and values. Thus, from the joint probability density, we can obtain the q-value function simply as the mean of the q-value conditioned to a given state-action; the accuracy will be obtained using the conditioned variance of this q-value; and the confidence will be obtained using the sample density in the state-action space. According to this, we will approximate a single joint density function for each competitor by means of a GMM, for which an efficient updating EM algorithm can be used. In the next Section we formally define the competitor system and the relevance function. In Section 3, the implementation of the system using Gaussian Mixture Models is presented, and in Section 4 its application to Q-learning is explained. Section 5 describes the experiments carried out to demonstrate the validity of the approach, and Section 6 closes the paper with some conclusions.

2 Competitive Function Approximation

This section presents the formalization of a function representation using a competitive strategy that will be used for the approximation of an arbitrary target function.

We define a *competitor* function, $\Phi_i(x, \xi_i)$, as a parametric function with parameters ξ_i, defined on values x of a D-dimensional continuous domain X_i. Associated with each competitor Φ_i, there is a *relevance* function, $\Gamma_i(x, \nu_i)$, which is a parametric function with parameters ν_i, also defined in the domain X_i.

Given a set of competitors $\mathbf{\Phi} = \{\Phi_1, \Phi_2, \ldots, \Phi_{|\mathbf{\Phi}|}\}$ and the set of their corresponding relevance functions $\mathbf{\Gamma} = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_{|\mathbf{\Phi}|}\}$, we consider the function, $y = \mathcal{F}(x, \mathbf{\Phi}, \mathbf{\Gamma})$, for $x \in X = \bigcup_{i=1}^{|\mathbf{\Phi}|} X_i$, defined as follows: First, a *winner* competitor, Φ_w, is selected such that

$$w = \underset{i \in I_x}{\operatorname{argmax}} \Gamma_i(x, \nu_i), \tag{1}$$

where $I_x = \{i | x \in X_i\}$ references the set of *active* competitors for x, $\mathbf{\Phi}_x = \{\Phi_i | x \in X_i\}$. Then the value y is obtained as,

$$\begin{aligned} y &= \mathcal{F}(x, \mathbf{\Phi}, \mathbf{\Gamma}) \\ &= \Phi_w(x, \xi_w). \end{aligned} \tag{2}$$

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 online (i.e., incremental), memory-free function approximation method using a function $\mathcal{F}(x, \mathbf{\Phi}, \mathbf{\Gamma})$ as defined before. To this end we need to specify the functions that will be used as competitors $\Phi_i(x, \xi_i)$, their respective relevance functions $\Gamma_i(x, \nu_i)$, and the method that will be used to update the parameters ξ_i and ν_i in order to get a good approximation of the target function, so that:

$$f(x) \approx \mathcal{F}(x, \mathbf{\Phi}, \mathbf{\Gamma}). \tag{3}$$

We call a method for function approximation that uses a function $\mathcal{F}(x, \mathbf{\Phi}, \mathbf{\Gamma})$ a *competitive function approximation* (CFA) method.

For the competitors $\Phi_i(x, \xi_i)$ of a CFA system, we can use any set of arbitrarily complex parametric functions and, in principle, they 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: $\Phi_i(x, \xi_i) = \Phi(x, \xi_i)$. It will be convenient to choose as $\Phi(x, \xi_i)$ some function for which an efficient incremental updating method for function approximation is available, so that we can use it for updating the parameters ξ_i. The type of function selected and the extension of the domains of each competitor will influence the possibilities of generalization 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 will 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.

2.1 Relevance Function

To make the exposition clear we assume first that the competitor and the relevance functions are constant-valued functions of x. They can be seen as uniparametric functions where the parameter ξ_i corresponds to the constant value assigned by the function. In this case, the sample mean of the observed values in each domain is a clear choice for the parameter of the corresponding approximator,

$$\Phi_i(x, \xi_i) = \xi_i = \bar{Y}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} y_j, \tag{4}$$

where n_i is the total number of samples observed so far in X_i. To define the relevance function, note that the variance of f in X_i, $\sigma_i^2 = E_{X_i}\left[(f(x) - E_{X_i}\left[f(x)\right])^2\right]$, is a good indicator of how well $E_{X_i}\left[f(x)\right] (\approx \bar{Y}_i)$ approximates f in x. Hence, the inverse of the variance σ_i^2 would be a good value for the relevance. However, as we only have a limited amount of samples of $f(x)$, its variance can not be determined precisely and should be estimated. We estimate the variance using the (unbiased) sample variance S_i^2:

$$S_i^2 = \frac{1}{n_i - 1} \sum_{j=1}^{n_i} \left(y_j - \bar{Y}_i\right)^2. \tag{5}$$

The uncertainty of this estimation depends on the number of samples n_i and to take into account this uncertainty we use the upper bound of the confidence interval for the variance [Blom, 1989]:

$$\frac{(n_i - 1)\, S_i^2}{\chi_\alpha^2\,(n_i - 1)}, \tag{6}$$

where $\chi_\alpha^2\,(n_i - 1)$ is the α-quantile of a χ^2 distribution, with $n_i - 1$ degrees of freedom. This upper bound represents the highest possible value (with α confidence) of the actual unknown variance given the current uncertainties in the estimation. Less confident estimations will have higher upper bounds for the variance. The value (6) provides a measure of the quality in the approximation that balances the estimation accuracy and the confidence in that estimation. Hence, we can adopt the inverse of the upper bound of the confidence interval for the variance as the relevance of a competitor Φ_i:

$$\Gamma_i(x, \nu_i) = \frac{\chi_\alpha^2\,(n_i - 1)}{(n_i - 1)\, S_i^2}. \tag{7}$$

This definition of the relevance prevents competitors with estimations obtained from a few samples, but showing low variances, to be favoured in front of those with slightly higher variances but with much more confident estimations.

Until now, we have assumed constant-valued estimations for the competitor and the relevance functions but, in general, they will depend on x. For the relevance to be x-dependent, the estimation of the variance and its confidence must also be x-dependent. Thus, assuming that the sample variance is approximated by some parametric function $S_i^2(x, \tau_i)$, we can estimate the sample variance $\tilde{S}_i^2$ for samples experienced in a region $\tilde{X}_i \subset X_i$ as

$$\tilde{S}_i^2 \approx \frac{\int_{\tilde{X}_i} S_i^2(x, \tau_i)\, p_i(x, \varsigma_i) dx}{\int_{\tilde{X}_i} p_i(x, \varsigma_i) dx}, \tag{8}$$

where $p_i(x, \varsigma_i)$ is a parametric estimation of the sample probability density function in X_i. In a similar way, we can estimate the number of samples in $\tilde{X}_i$ as:

$$\tilde{n}_i \approx n_i \int_{\tilde{X}_i} p_i(x, \varsigma_i) dx. \tag{9}$$

Now, replacing the sample variance and the number of samples in X_i in (7) by their respective counterparts in $\tilde{X}_i$, we can express the relevance function of competitor i in a point $x \in \tilde{X}_i$ as

$$\Gamma_i(x, \nu_i) = \frac{\chi_\alpha^2\,(\tilde{n}_i - 1)}{(\tilde{n}_i - 1)\, \tilde{S}_i^2}. \tag{10}$$

With this relevance formula we can regulate the precision of the relevance of a competitor at a point x by adjusting the size of the region $\tilde{X}_i$ to an appropriate value depending on the nature of the generalization performed and the regularity of the variance function. In order to simplify the calculations of the integrals in (8) and (9), we assume that the region $\tilde{X}_i$ is small enough for the values for the probability density function $p_i(x, \varsigma_i)$ and the variance function $S_i^2(x, \tau_i)$ to be nearly constant, which permits to estimate the relevance (10) as

$$\Gamma_i(x, \nu_i) \approx \frac{\chi_\alpha^2\left(\tilde{V}_i\, n_i\, p_i(x, \varsigma_i) - 1\right)}{\left(\tilde{V}_i\, n_i\, p_i(x, \varsigma_i) - 1\right)\; S_i^2(x, \tau_i)} \tag{11}$$

where $\tilde{V}_i$ is the volume of $\tilde{X}_i$. In practice, the volume $\tilde{V}_i$ is defined empirically for each particular problem.

3 Competitive Function Approximation with Gaussian Mixture Models

In this section we present the specific instantiation of the method for CFA that we will use for generalization in RL. We need to specify the competitor functions $\Phi_i(x, \xi_i)$, as well as the variance functions $S_i^2(x, \tau_i)$, and the density functions $p_i(x, \varsigma_i)$ needed to calculate the relevance (11). In addition, we also need to provide the mechanisms to update all the parameters involved. Next, we just give a general description of the approach. A more detailed explanation can be found in [Agostini and Celaya, 2010].

For the instantiation of each competitor function we will use a Gaussian Mixture Model (GMM) defined in the input-output joint space,

$$p(z; \mathbf{\Theta}) = \sum_{j=1}^{K} \alpha_i \mathcal{N}(z; \mu_j, \mathbf{\Sigma}_j), \tag{12}$$

where $z = (x, y)$ is the input-output variable; K is the number of Gaussians of the mixture; α_j, usually denoted as the mixing parameter, is the prior probability, $P(j)$, of Gaussian j to generate a sample; $\mathcal{N}(z; \mu_j, \mathbf{\Sigma}_j)$ is the multidimensional Gaussian function with mean vector μ_j and covariance matrix $\mathbf{\Sigma}_j$; and $\mathbf{\Theta} = \{\{\alpha_1, \mu_1, \mathbf{\Sigma}_1\}, ..., \{\alpha_K, \mu_K, \mathbf{\Sigma}_K\}\}$ is the whole set of parameters of the mixture. This representation has the virtue of providing an approximation of the value function as well as all the quantities required in (11) from a single density model. Thus, by embedding a probability density model $p_i(z; \mathbf{\Theta}_i)$ at each competitor Φ_i, we can easily derive, through the conditional probabilities, a point-dependent estimation of the sample mean,

$$\Phi_i(x, \xi_i) = \mu_i(y|x; \mathbf{\Theta}_i), \tag{13}$$

as well as a point-dependent estimation of the sample variance,

$$S_i^2(x, \tau_i) = \sigma_i^2(y|x; \mathbf{\Theta}_i). \tag{14}$$

In addition, from the density model we can estimate the number of samples involved in the estimation of the mean and variance at each point,

$$\tilde{n}_i \approx \tilde{V}_i \, n_i \, p_i(x; \mathbf{\Theta}_i), \tag{15}$$

where $p_i(x; \mathbf{\Theta}_i)$ is the probability density estimation in the input space. The parameters $\mathbf{\Theta}_i$ are updated from each incoming experience (x_t, y_t), using the incremental version of the Expectation-Maximization algorithm proposed in [Agostini and Celaya, 2010].

3.1 Competitor Management

The set of competitors of a CFA must be defined so that their domains X_i form a covering of the whole domain X. Since it is not possible, in general, to determine beforehand the best way to define the number and sizes of the X_i's, we start with an initial covering with a relatively small number of competitors and, in the course of learning, new competitors are generated as required by the need of a better approximation. This makes the approach non-parametric and, in principle, able to approach any arbitrary function. We would like to note that it is also possible to make the approach non-parametric by changing not only the number of competitors but also the number of Gaussians at each competitor. However, we will keep the number of Gaussians fixed, varying only the number of competitors.

Our criteria for competitor generation involves two user-defined thresholds: n_g, which sets the minimum number of samples that should be collected by a competitor before its estimations can be considered confident, and e_g, which sets the maximum error allowed in the approximation. New competitors will be generated when, after observing a new sample (x, y), the approximation error is too large,

$$(f(x) - \mathcal{F}(x, \mathbf{\Phi}, \mathbf{\Gamma}))^2 = (y - \mu_w(y|x))^2 > e_g, \tag{16}$$

and all the active competitors for x have reached the minimum confidence threshold:

$$n_i \geq n_g, \, \forall i \, | x \in X_i. \tag{17}$$

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, a new competitor domain is generated by intersecting that of the winner competitor at x, X_w, with the domain of the competitor that showed the least prediction error at the sampled point. If this domain is different from all the already existing X_i, a new competitor is generated with it. In the case that a competitor with such domain already exists, it makes no sense to generate it again, so that, in this case, 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 along the dimension d along which the dispersion of samples of X_w is the largest (i.e., the dimension whose marginal variance is maximum for the winner competitor). Generation via splitting complements the generation via intersection of existing domains, since it permits to increase the resolution indefinitely, which is not possible by intersections alone.

Each new competitor is initialized with a Gaussian Mixture Model with K Gaussians with initial mean vectors $(x, y) = (x_r, \Phi_w(x_r, \xi_w))$, where the input points x_r are selected at random from the domain of the competitor. The covariance matrices are initialized to $diag\{d_1^2, ..., d_D^2, S_w^2(x_r, \tau_w)\}$, where d_j is a fixed percentage of the length of the domain in the j dimension, and D is the dimension of the input space. In the cases in which the new domain contains the experienced input x_t, it is taken as one of the x_r, and the sample point (x_t, y_t) is taken as the mean of the corresponding Gaussian.

Elimination of Competitors

The competitors that become less useful for the system can be eliminated. We remove those competitors that are redundant with others. We say that a competitor Φ_i is redundant with a competitor Φ_j when

$$X_i \cap X_j = X_i, \tag{18}$$

and the normalized differences in their values and relevances are both below some thresholds for all $x \in X_i$:

$$\frac{|\Phi_i(x) - \Phi_j(x)|}{\max(|\Phi_i(x)|, |\Phi_j(x)|)} < \mathrm{thr}_v, \tag{19}$$

and

$$\frac{|\Gamma_i(x) - \Gamma_j(x)|}{\max(|\Gamma_i(x)|, |\Gamma_j(x)|)} < \mathrm{thr}_r. \tag{20}$$

In practice, these conditions are tested only in a finite number of points of X_i. These tests are performed between overlapping competitors once every fixed number of samples.

4 Q-Learning with a CFA

Next, we explain how the CFA just described can be used with Q-learning in continuous state and action spaces. The function to be approximated is the utility function $Q(s, a)$. The input points x correspond to state-action pairs (s, a), and the function values y will be the current estimation $q(s, a)$ of the real $Q(s, a)$ function, obtained from the sampled version of the Bellman equation:

$$q(s, a) = r(s, a) + \gamma \max_{a^*}(\hat{Q}(s', a')), \tag{21}$$

where $r(s, a)$ is the immediate reward obtained after executing action a in state s; s' is the state observed after the action execution; and $\hat{Q}(s', a')$ is the approximation of the action-value function at (s', a') provided by the CFA system: $\hat{Q}(s', a') = \mathcal{F}((s, a), \mathbf{\Phi}, \mathbf{\Gamma})) = \Phi_w(s', a')$.

To compute the value $q(s, a)$ we need to solve the maximization problem $\max_{a^*}(\hat{Q}(s', a'))$. We adopt the strategy of computing the values $\hat{Q}(s', a')$ for a finite number of actions, and then taking the value of the action that provides the largest Q as the approximated maximum.

For action exploration we follow the strategy proposed in [Agostini and Celaya, 2010], assigning to each (s, a) to be evaluated a value calculated randomly as,

$$Q_{rnd}(s, a) = \hat{Q}(s, a) + \Delta_Q(\sigma^2(q|s, a)), \tag{22}$$

where $\Delta_Q(\sigma^2(q|s,a))$ is a value taken at random from a normal probability distribution with 0 mean and variance $\sigma^2(q|s,a) = S_w^2((s,a),\tau_w)$. Once a $Q_{rnd}(s,a)$ value has been assigned for every evaluation (s,a), the action to be executed is selected using the greedy policy as,

$$a = \underset{a^z}{\operatorname{argmax}} \hat{Q}_{rnd}(s,a'). \tag{23}$$

5 Experiments

To demonstrate the performance of the CFA in Q-Learning we use two different benchmarks: the pendulum swing-up and stabilization [Doya, 2000], and the cart-pole balancing [Sutton and Barto, 1998]. The first task consists in swinging the pendulum until reaching the upright position and then stay there indefinitely. The pendulum model is the same used in [Doya, 2000] with a reward given by the height of the pendulum. Training and test episodes consist in 50 seconds of simulated time, with an action interval of 0.1 seconds. The second task, the cart-pole balancing, consists in a pole mounted on cart that has to be stabilized inside an acceptable region by the motions of the cart. The cart is free to move in an acceptable range within a 1-dimensional bounded track, and the pole moves in the vertical plane parallel to this track. The cart-pole benchmark is of interest since it involves a 5-dimensional state-action space, two more than the inverted pendulum, and serves to illustrate the scalability of the algorithm. The set-up for the experiment is the same used in [Peters and Schaal, 2008], with a reward function that varies linearly from 0 at the bound of the acceptable angle for the pole, to 1 at the vertical position. Reward is -1 when either the pole or the cart leave their acceptable regions.

For both benchmarks, we give results using three different function approximation techniques: the GMM with variable number of Gaussians of [Agostini and Celaya, 2010], the CFA introduced here, and Variable Resolution, to provide a general reference. Table 1 shows the parameters for CFA used in both benchmarks.

Parameter	pendulum	cart-pole
# initial competitors	10	16
# Gs. per competitor	10	20
$\tilde{V}$	1	50
e_g	0.5	0.2
n_g	200	50
thr_v	0.2	0.2
thr_r	0.2	0.2

Table 1: Parameters for the experiments with CFA

5.1 Results

Figures 1 and 2 show the results of the experiments, that in all cases are the average of 15 independent runs of each algorithm. In both benchmarks, the GMM and CFA approaches are clearly superior to Variable Resolution. For the pendulum swing-up task, we show the performance of the GMM approach with two different initializations: One with 10 Gaussians, that reported the best results in [Agostini and Celaya, 2010], and another with 100 Gaussians, what corresponds to the same total number of initial Gaussians of our approach with CFA. Figure 1 shows that CFA has a faster and more stable convergence than both instantiations of the GMM. For the case of 10 initial Gaussians, GMM reaches a stable value in about 45 episodes, while CFA does the same in about 23. For the GMM with 100 initial Gaussians the performance is worst, likely due to the fact that the system must simultaneously balance a large number of parameters (in fact, 100 Gaussians is more than required, since the GMM with 10 initial Gaussians reaches convergence with an average of about 45 Gaussians). Increasing the number of competitors, and therefore the total number of Gaussains, has not the same effect in CFA, since each competitor must only balance their own parameters, independently of those of other competitors (convergence is achieved with 185 competitors in average). To give a comparison of these results with the state of the art on this benchmark, we recall the results given by [Riedmiller, 2005] for this task, where, using a batch NFQ algorithm, reports convergence after $100.000 \times D$ updates (D = number of samples), while GMM takes about $45 \times 500 = 22.500$ updates, and CFA only $23 \times 500 = 11.500$.

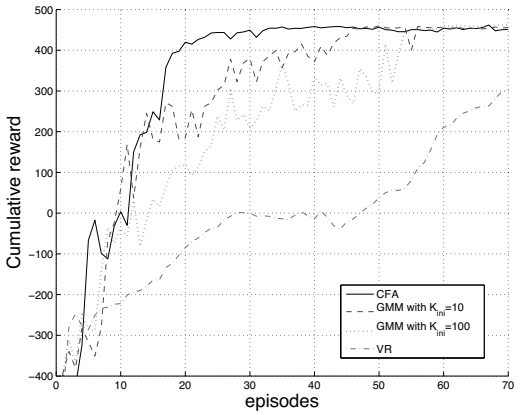

Figure 1: Comparison of the performance of the CFA, GMM, and VR methods in the inverted pendulum task. The curves show the sum of rewards obtained in test episodes, performed after each training episode, starting from the hang-down position.

In the case of the cart-pole, the different performance between the three approaches is even larger, as shown in figure 2. We attribute the superiority of GMM and CFA over VR to their better generalization capabilities, which become more important as the dimensionality of the state-space increases. To interpret the much better performance of CFA over GMM in this benchmark, we note that GMM required, in average, a final amount of 190 Gaussians to reach convergence, while CFA required, in average, about 445 competitors with only 20 Gaussians each. Thus, even if the total number of Gaussians in CFA is much larger, many competitors are updated in parallel at each iteration, and each of them has to adjust a much smaller set of parameters. To compare these results with the

state of the art, we observe that CFA reaches a good performance after about 3 minutes of simulated time and converges in less that 5 minutes, while [Peters and Schaal, 2008], using a Natural Actor-Critic approach, reported convergence after 10 minutes.

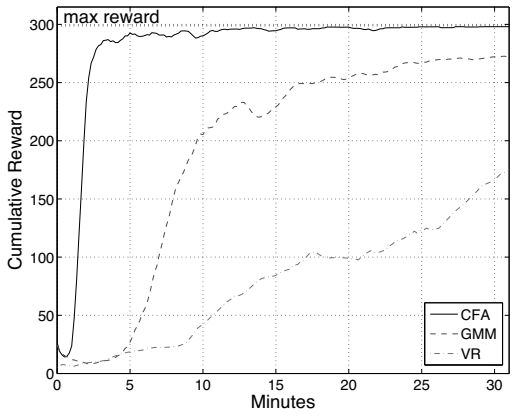

Figure 2: Comparison of the performance of the CFA, GMM, and VR methods in the cart-pole balancing task. The curves show the sum of rewards obtained in test episodes performed every 10 seconds of simulated time. Each test episode consists in averaging the sum of rewards obtained during 5 seconds of simulated time, starting at four different positions of the pole: $-10\,^\circ, -5\,^\circ, 5\,^\circ, 10\,^\circ$, with the cart centred on the track.

6 Conclusions

The results obtained in this work support our claim that it is advantageous to use a collection of function approximators defined on overlapping regions of the whole domain that compete between them to provide the best estimation at each point, instead of using a single global approximator of arbitrary complexity. There are several reasons that may explain this. One is locality: since each approximator is only updated with those samples that fall in its domain, it is not influenced by farther away samples, thus avoiding most of the overfitting problems caused by the biased sampling usually found in reinforcement learning. Another reason is that covering the domain with regions of different sizes, it is possible to find the right one where the complexity of the target function matches the approximation capabilities of the competitors. An optimal match between the complexity of the target function and the approximator gives rise to a maximally efficient learning and accurate generalization. Being able to choose the more plausible (or relevant) competitor at each point makes possible to keep inaccurate competitors in the system without distorting the accuracy of the actual approximation.

Still another reason that makes the competitive approach attractive in reinforcement learning is its capability to deal with non-stationary target functions: when a drastic 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 before the output becomes accurate; instead, a different competitor that managed to be more accurate at that point will provide the output as soon as its relevance gets larger there.

As expected, the computational cost of the CFA method is higher than that of the single global GMM method. However, in the presented experiments, the increase in computation time is less than one order of magnitude.

Acknowledgments

This research was partially supported by Consolider Ingenio 2010, project CSD2007-00018.

References

[Agostini and Celaya, 2010] A. Agostini and E. Celaya. Reinforcement learning with a Gaussian mixture model. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN'10). (Barcelona, Spain)*, pages 3485–3492, 2010.

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

[Boyan and Moore, 1995] 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.

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

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

[Engel *et al.*, 2005] Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. In *Proc. of the 22nd Int. Conf. on Machine learning*, pages 201–208, 2005.

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

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

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

[Riedmiller, 2005] M. Riedmiller. Neural Reinforcement Learning to Swing-up and Balance a Real Pole. In *Proc. of the Int. Conf. on Systems, Man and Cybernetics*, volume 4, pages 3191–3196, 2005.

[Rottmann and Burgard, 2009] A. Rottmann and W. Burgard. Adaptive autonomous control using online value iteration with Gaussian processes. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 3033–3038, 2009.

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