2009 International Conference on Machine Learning and Applications

Exploiting Domain Symmetries in Reinforcement Learning with Continuous State and Action Spaces

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

***Abstract*—A central problem in Reinforcement Learning is how to deal with large state and action spaces. When the problem domain presents intrinsic symmetries, exploiting them can be key to achieve good performance. We analyze the gains that can be effectively achieved by exploiting different kinds of symmetries, and the effect of combining them, in a test case: the stand-up and stabilization of an inverted pendulum.**

***Keywords*-Reinforcement learning; function approximation; domain symmetries.**

I. Introduction

Reinforcement Learning in large state-action spaces suffers from the "curse of dimensionality" problem, which means that the number of states that the agent has to visit for learning to succeed soon grows too large, and the learning task becomes unfeasible. Using function approximation techniques to represent the Value or the Q function should attenuate the problem due to their capability of generalization between similar situations, but, in any case, when the state space is large, using an appropriate representation is crucial in order to allow efficient learning. It is often possible to simplify the task when the domain is known to present some symmetry since it allows transferring the observations made in one situation to other situations symmetric to it.

In [1], the notion of symmetry in Markov decision processes is examined. Two types of symmetry are considered, that they call "adherence to an equivalence relation" and "invariance under a group of transformations". The last one is introduced to deal with multiagent systems, that we will not consider here. The "adherence to an equivalence relation" type of symmetry is one of those which we are interested in, and corresponds, in essence, to identifying symmetric states s and $\overline{s}$ such that, for each possible action a in state s leading to state s', there is a symmetric action $\overline{a}$ in state $\overline{s}$ that leads to a state $\overline{s}'$, symmetric to s', with the property that both give rise to exactly the same reward. The identification of symmetric states can be incorporated in the definition of the state, thus, effectively reducing the actual size of the space on which learning takes place.

However, the above definition does not include all possible kinds of symmetry. In this paper we present an example showing a "temporal inversion" symmetry, which does not correspond to the previous type. In this kind of symmetry, experiences can be transferred between symmetric situations for which the obtained reward, however, is not the same. Due to the special nature of the "temporal inversion" type of symmetry, it is not possible to identify symmetric situations as equivalent, and thus, they must be represented as different states and updated in separate operations. This fact makes impossible to use a representation of the state that results in a reduction of the size of the state space.

Returning to the "adherence to an equivalence relation", we observe that there are two alternative ways of exploiting this type of symmetry: we can define the state as suggested above, so that equivalent situations get represented by a single state, or we can represent symmetric situations as different states and perform multiple updates for each experience (one for each symmetric state), as must be done in the case of "temporal inversion" type of symmetry. The last is certainly computationally less efficient, but as we will discuss later, it could avoid certain problems that may appear with the former approach.

Both kinds of symmetries are in principle equally applicable not matter if the Q function is represented with entries in a table or through function approximation. However, the gains obtained by symmetries may be different in both cases. Thus, for example, if we follow the approach of multiple updates for symmetric states using a tabular representation, all symmetric states will be updated with the same experiences, and we will end with an exact replication of the Q estimations for each symmetric state (provided they are initialized with the same values). The result in this case will not be different from what would be obtained following the approach of identifying symmetric states. But this is not necessarily true with function approximation, since each update modifies the estimation in an extended region of the domain, so that, in general, the Q estimation at a given point will be updated two times per experience, and this may have the effect of accelerating learning. Additionally, since in general the final result of performing the two successive updates depends on the order in which they are done, a perfect symmetry in the estimation function will be lost.

In this paper we test the effect of exploiting different kinds of symmetries in Reinforcement Learning with function

978-0-7695-3926-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ICMLA.2009.41

approximation. For this, we use a Q-learning algorithm recently developed by us [2], which approximates the Q-function through the estimation of a probability density function in the state-action-Q-value space, but similar results should be expected with any other method of function approximation.

The rest of the paper is organized as follows: Section II introduces our approach to RL with probability density estimation, Section III defines the test problem and the symmetries applied, Section IV shows the results of the experiments, and Section V concludes the paper.

II. Q-Learning with Density Estimation

In the Reinforcement Learning paradigm, an agent must improve its performance by selecting actions that maximize the accumulation of rewards provided by the environment [3]. At each time step, the agent observes the state s_t and chooses an action a_t according to its policy $u(s)$. The environment changes to state s_{t+1} in response to this action, and produces an instantaneous reward r_t. One of the most popular algorithms used in RL is Q-Learning [4], which uses an action-value function $Q(s,a)$ to evaluate the expectation of the maximum future cumulative reward that will be obtained from the execution of action a in the situation s. Q-learning uses a sampled version of the Bellman optimality equations [5] to estimate instantaneous q values,

$$q(s,a) = r(s,a) + \gamma \max_{a'} Q(s',a') \tag{1}$$

where $r(s,a)$ is the immediate reward obtained from executing action a in situation s, $\max_{a'} Q(s',a')$ is the maximum estimated cumulative reward in the next observed situation s', and $\gamma \in [0,1]$ is a discount factor that regulates the importance of future rewards. At a given stage of the learning, the current policy is derived from the learned Q-values as,

$$u(s) = \operatorname*{argmax}_a Q(s,a) \tag{2}$$

The basic formulation of Q-learning assumes discrete state-action spaces and the Q function is stored in a tabular representation. For continuous domains a function approximation is required to represent the Q function and to generalize between similar situations. Boyan and Moore [6] showed that simply replacing a lookup table by a function approximator may cause learning to fail. They attribute this, in part, to the fact that, when the function approximation model is not general enough, learning may be impossible if the model cannot correctly fit the transient function estimations, even if the target function can be exactly represented by the parametric model. This problem may be overcome, at least in principle, using a non-parametric function approximation model with universal approximation capabilities. In our approach, instead of directly approximating the Q function, we approximate the probability density function of the observed experience samples. This probability is defined in the joint space of the state, action and Q-values, and from this distribution we obtain the probability distribution of Q for any given state and action combination. We represent the probability density with a Gaussian Mixture Model, which is known to be a general approximator provided the number of Gaussians is not limited [7]. The information obtained in this way is richer than what it is possible to represent by directly approximating the Q function, and its adjustment can be achieved by means of the EM algorithm, which is conceptually simple and well understood.

Next we present our proposal for function approximation using density estimations.

A. The Gaussian Mixture Model

We approximate the probability density with a mixture of multivariate Gaussians, or Gaussian Mixture Model [8]:

$$p(\mathbf{x}_t|\theta) = \sum_{i=1}^{K} \alpha_i \mathrm{N}(\mathbf{x}_t|\theta_i) \tag{3}$$

where α_i, usually denoted as the mixing parameter, is the prior probability $P(i)$ that Gaussian i generates a sample $\mathbf{x}_t$. $\theta_i = \{\mu_i, \sum_i\}$ are the parameters of Gaussian i, and $\theta = \{\{\alpha_1, \mu_1, \sum_1\}, ..., \{\alpha_K, \mu_K, \sum_K\}\}$ is the whole set of parameters for the mixture. The parameters of the model can be estimated using a maximum-likelihood estimator (MLE). Given a set of samples $\mathbf{X}$, the likelihood function is given by:

$$L[X;\theta] = \prod_{t=1}^{N} p(\mathbf{x}_t|\theta) \tag{4}$$

For the formulation of RL, a sample corresponds to an instance in the joint space, $\mathbf{x}_t = (\mathbf{s}, a, q)$. The maximum-likelihood estimation of the model parameters is that maximizing (4). Direct computation of the MLE requires complete information about which mixture component generated which sample. Since this information is missing, the EM algorithm [9], described in the next section, is usually employed.

B. The Expectation-Maximization algorithm

To maximize (4), the (EM) algorithm first produces an estimation of the expected values of the missing variables using initial values of the parameters to be estimated (E step), and then computes the MLE of the parameters given the expected values of the missing variables (M step). This process is iterated until a convergence criterion is fulfilled.

Next we briefly describe how EM is applied for the case of a GMM. The E step consists in the calculation of the probability $P(i|\mathbf{x}_t)$ for each component i of generating sample $\mathbf{x}_t$ that we denote by $w_{t,i}$:

$$w_{t,i} = \frac{P(i)p(\mathbf{x}_t|i)}{\sum_{j=1}^{K} P(j)p(\mathbf{x}_t|j)} = \frac{\alpha_i N(\mathbf{x}_t|\mu_i, \sum_i)}{\sum_{j=1}^{K} \alpha_j N(\mathbf{x}_t|\mu_j, \sum_j)} \tag{5}$$

where $t = 1, .., n$, the number of samples, and $i = 1, .., K$. The M step consists in computing the MLE using the estimated $w_{t,i}$. It can be shown [10] that the mixing parameters, means, and covariances are given by:

$$\alpha_i = \frac{1}{n} \sum_{t=1}^{n} w_{t,i} \tag{6}$$

$$\mu_i = \frac{\sum_{t=1}^{n} w_{t,i} \mathbf{x}_t}{\sum_{t=1}^{n} w_{t,i}} \tag{7}$$

$$\Sigma_i = \frac{\sum_{t=1}^{n} w_{t,i} (\mathbf{x}_t - \mu_i)(\mathbf{x}_t - \mu_i)'}{\sum_{t=1}^{n} w_{t,i}} \tag{8}$$

C. On-line EM

Estimating a probability density distribution by means of the EM algorithm involves the iteration of E and M steps on the complete set of data. However, in RL, sample data are not all available at once: they arrive sequentially as learning proceeds. This prevents using the raw EM algorithm, and requires an incremental version of it, several of which have been proposed for the Gaussians Mixture Model [11], [12].

Here, we adopt an algorithm based on that developed in [13] for the NGnet, but adapted to the case of GMM. It consists in performing an E-M step after the observation of each sample. The E step does not differ from the batch version (equation 5), though it is only computed for the new sample. For the M step, the parameters of all mixture components are updated with the new sample. For this, we define the following time-discounted weighted sums:

$$W_{t,i} = [[1]]_{t,i}, \tag{9}$$

$$X_{t,i} = [[\mathbf{x}]]_{t,i}, \tag{10}$$

$$(XX)_{t,i} = [[(\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)']]_{t,i} \tag{11}$$

where we have used the notation:

$$[[f]]_{T,i} = \sum_{t=1}^{T} \left(\prod_{s=t+1}^{T} \lambda_s \right) (f_t w_{t,i}) \tag{12}$$

where $\lambda_t, \in [0, 1]$, is a time dependent discount factor introduced for forgetting the effect of old, less accurate values.

When a new sample $\mathbf{x}_t$ arrives, these accumulators are updated with the step-wise incremental formula,

$$[[f]]_{t,i} = \lambda_t^{w_{t,i}} [[f]]_{t-1,i} + f_{t,i} w_{t,i} \tag{13}$$

where the power $w_{t,i}$ of λ_t is introduced to make the influence of each update more local, reducing the updating of Gaussians which are not responsible of generating the observed values. Then, the GMM estimators can be obtained as:

$$\alpha_{t,i} = \frac{W_{t,i}}{\sum_{j=1}^{K} W_{t,i}} \tag{14}$$

$$\mu_{t,i} = \frac{X_{t,i}}{W_{t,i}} \tag{15}$$

$$\Sigma_{t,i} = \frac{(XX)_{t,i}}{W_{t,i}} \tag{16}$$

D. Action Selection

Given the GMM

$$p(\mathbf{s}, a, q) = \sum_{i=1}^{K} \alpha_i \mathrm{N}(\mathbf{s}, a, q | \mu_i, \Sigma_i), \tag{17}$$

the probability distribution for q is obtained as:

$$p(q|\mathbf{s}, a) = \sum_{i=1}^{K} \beta_i(\mathbf{s}, a) \mathrm{N}\left(q | \mu_i(q|\mathbf{s}, a), \sigma_i(q)\right) \tag{18}$$

where,

$$\mu_i(q|\mathbf{s}, a) = \mu_i^q + \Sigma_i^{q,(\mathbf{s},a)} \left(\Sigma_i^{(\mathbf{s},a)(\mathbf{s},a)} \right)^{-1} \left((\mathbf{s}, a) - \mu_i^{(\mathbf{s},a)} \right) \tag{19}$$

$$\sigma_i^2(q) = \Sigma_i^{qq} - \Sigma_i^{q,(\mathbf{s},a)} \left(\Sigma_i^{(\mathbf{s},a)(\mathbf{s},a)} \right)^{-1} \Sigma_i^{(\mathbf{s},a),q} \tag{20}$$

$$\beta_i(\mathbf{s}, a) = \frac{\alpha_i p(\mathbf{s}, a | \mu_i^{(\mathbf{s},a)}, \Sigma_i^{(\mathbf{s},a)(\mathbf{s},a)})}{\sum_{j=1}^{K} \alpha_j p(\mathbf{s}, a | \mu_j^{(\mathbf{s},a)}, \Sigma_j^{(\mathbf{s},a)(\mathbf{s},a)})} \tag{21}$$

¿From (18) we can obtain the conditional mean and covariance, $\mu(q|\mathbf{s}, a)$ and $\sigma^2(q|\mathbf{s}, a)$ respectively,

$$\mu(q|\mathbf{s}, a) = \sum_{i=1}^{K} \beta_i(\mathbf{s}, a) \mu_i(q|\mathbf{s}, a) \tag{22}$$

$$\sigma^2(q|\mathbf{s}, a) = \sum_{i=1}^{K} \beta_i(\mathbf{s}, a) (\sigma_i^2(q) + (\mu_i(q|\mathbf{s}, a) - \mu(q|\mathbf{s}, a))^2) \tag{23}$$

To select an action we use the policy function (2) where the maximum is taken form a finite set of action values, a_n, regularly sampled along its range. Each $Q(s, a_n)$ is obtained stochastically from a normal distribution with mean (22) and variance (23).

E. Model Updating

To update the density model with the new experience we must provide the sample $\mathbf{x}_t = (\mathbf{s}_t, a_t, q_{t+1}(s, a))$, where $q_{t+1}(s, a)$ is given by (1), which involves again the estimation of the maximum value $\max_a Q(s, a)$. We proceed in a similar way as in action selection, but in this case, we just take as $Q(s, a)$ the expected value given by (22).

The approximation capabilities of a GMM depend on the number K of Gaussians of the mixture. Since we can

not determine the most appropriate number beforehand, new Gaussians are generated in two situations: When the estimation error of the q values is larger than a given δ, and when the density of samples in the given point of the joint space is below a threshold ρ.

III. Test application: The inverted pendulum

For our experiments we selected the benchmark application of an inverted pendulum with limited torque [14], [15]. The task consists in swinging up the pendulum until reaching its upright position, and stabilize it there. This task is not trivial due to the torque limitation, which forces the controller to swing the pendulum several times until its kinetic energy is large enough to reach the top position. We assume a frictionless pendulum for the temporal symmetry to hold.

The state of the system is determined by the angle θ of the pendulum to the top position, and its temporal derivative: $\mathbf{s} = (\theta, \dot{\theta})$. Thus, the density is estimated in a four-dimensional joint space $(\theta, \dot{\theta}, a, q)$. As the reward signal we simply take the height of the pendulum $h = cos(\theta)$ which ranges in the interval $[-1, 1]$. The discount coefficient γ in equation (1) is set to 0.99. We consider two kinds of symmetry in this problem: spatial and temporal.

A. Spatial symmetry

The spatial symmetry corresponds to the fact that the behavior of the pendulum is the same if we look at it reflected in a mirror. This allows identifying a given state $(\theta, \dot{\theta})$ with its spatial inversion $(-\theta, -\dot{\theta})$, substituting at the same time each action a by its opposite $-a$. This is a symmetry of the "adherence to an equivalence relation" type discussed before. We consider two ways of exploiting this symmetry:

1. Representing symmetric situations with the same state. We consider only states with positive values of θ by means of the equivalence $(-\theta, \dot{\theta}, a) \sim (\theta, -\dot{\theta}, -a)$. Then, if σ_θ is the sign of θ, we represent situation $(\theta, \dot{\theta})$ by the state $(\theta', \dot{\theta}') = (\sigma_\theta\theta, \sigma_\theta\dot{\theta})$. When an action a' is selected, $a = \sigma_\theta a'$ is executed, and the result is transformed into its state representation in order to perform the update.

2. Representing symmetric situations with separate states, but using the result obtained in each situation to update the corresponding state and also its symmetric one, as if it had been also experienced.

B. Temporal symmetry

While perhaps less intuitive, the symmetry obtained by temporal inversion is as fundamental as that of spatial inversion. It applies to conservative systems, that is, systems with no loss of mechanical energy into heat (as would occur with a friction term $\neq 0$), and establishes that the behavior of a mechanical system is symmetric if observed backwards in time. Changing the direction of time implies a change of sign in all velocity vectors (since they involve time linearly), but does not affect positions nor accelerations (since they involve time squared), and consequently, it does not affect the sign of forces. For the pendulum, this symmetry implies that, if we execute action a and observe the transition $(\theta_1, \dot{\theta}_1) \rightarrow (\theta_2, \dot{\theta}_2)$ with reward r_1, it is as if (executing the same action a) we also observed the inverse transition $(\theta_2, -\dot{\theta}_2) \rightarrow (\theta_1, -\dot{\theta}_1)$ with reward r_2 corresponding to the initial situation. Note that this symmetry cannot be used to reduce the state space by identifying symmetric situations: due to the inversion in time and the difference in rewards, both transitions are different and the only possibility to exploit this symmetry is by successive updates.

IV. Experiments

In order to analyze the gains provided by each kind of symmetry, we will compare them against a basic approach in which all symmetries are ignored. The settings of the system have been tuned for the non-symmetric case and held fixed for all the experiments. We provide the system with 20 initial Gaussians. The components of the mean μ_i of Gaussian i are selected randomly, except for the q variable that is initialized to the maximum possible value to favor exploration of unvisited regions. The initial covariance matrices Σ_i are diagonal and the variance for each variable is set to the range of the variable. The initial number of samples for each Gaussian, W_i, is set to 0.1. The discount coefficient λ_t of Eq. (12) is made to vary according to:

$$\lambda_t = 1 - 1/(at + b) \tag{24}$$

with $a = 0.001$ and $b = 10$. We performed the experiments using episodes of 7 seconds with torque actions issued every 0.1 seconds. The evolution of the system is simulated with the Euler integration method using an integration step of 0.001 seconds. At the beginning of each episode the state of the pendulum is initialized at random. To accelerate convergence, the angle is initialized in an interval around the upright position that is gradually increased with each episode. In all experiments, we let the system learn for 50 episodes. At the end of each episode a test is performed, consisting in 7 seconds of simulation using the policy learned so far. Since we are interested in whether the system is able to swing up and stabilize the pendulum, we take the reward accumulated during the last second of the simulation as the result of each test, ignoring the initial phase of the test since this transient process greatly depends on the random initial configuration. An accumulated reward close to 100 means that the system reached a high position and stayed there during this period. Each experiment is repeated 10 times and the results are averaged. Figure 1 shows the results obtained for the basic approach. In average, the pendulum is reliably stabilized after about 25 episodes. These results compare well with the state of the art, for example, [15] reports that, with a similar approach but using

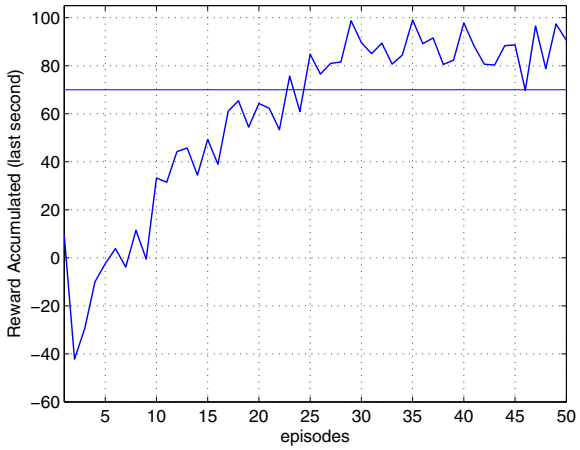

Figure 1. Reference test: no symmetries.

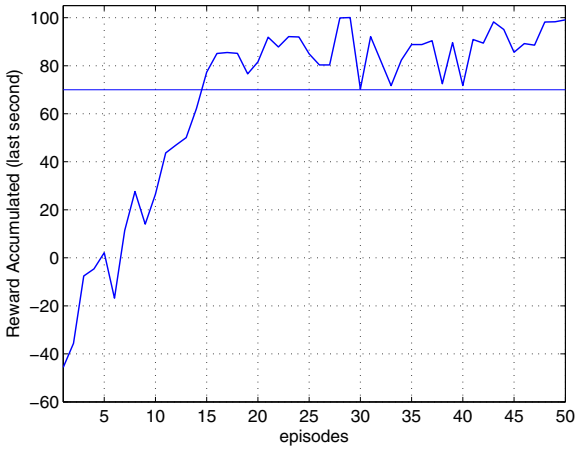

Figure 3. Spatial symmetry (approach 2).

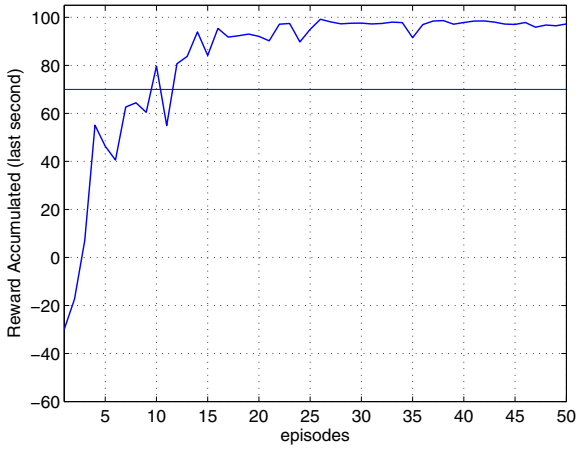

Figure 2. Spatial symmetry (approach 1).

Normalized Gaussian nets for function approximation and a more elaborated reward function, can roughly achieve the task ("stabilize the pendulum at the upright position from almost all initial states") after 40 episodes of the same length and similar settings.

Figures 2 and 3 show the results obtained with the two ways of exploiting the spatial symmetry. We observe that approach 1 is more efficient than approach 2. This could be expected because, with approach 1, the Q function has to be learnt in a domain that is a half in size, making the resulting function simpler. This is so despite the fact that, with the symmetric definition of state, the upright position lies exactly at the boundary of the resulting workspace, which makes more difficult to achieve a good approximation at this point by means of a mixture of Gaussians.

In order to compare quantitatively the performance of each approach, we define the convergence time T_x of each approach as the number of episodes required, in the average, to reliably obtain an accumulated reward equivalent to keeping the pendulum inside an angle of 45° from the top position (marked in the figures by the horizontal line). From the graphs, we derive the value $T_0 = 25$ for the basic approach, and $T_{S1} = 12$, and $T_{S2} = 15$, for approaches 1 and 2, respectively. This corresponds to an improvement in efficiency by a factor of roughly 2, what agrees with what could be expected from the use of the symmetry.

Figure 4 shows the performance of using the temporal symmetry. Though the graphic shows a performance that seems to lay in between the two approaches for spatial symmetry, the quantitative measure we chose gives the slightly worst result of $T_T = 18$. In figure 5, the two symmetries are applied at the same time (using approach 1 for the spatial symmetry). In this case, $T_{ST} = 5$. This result is somehow surprising, since the final gain is by a factor 5 with respect to the basic approach, which is better than combining the gains obtained from applying each symmetry in isolation. We suggest that this result can be due to the fact that, with the double updating of the temporal symmetry, situations lying near the upright position are updated twice for each experience, what should accelerate the convergence in this region.

V. Conclusions

Reinforcement learning in continuous domains can benefit from the exploitation of different kinds of symmetries. Using a function approximation based on the estimation of the probability density in the joint space of states, actions, and Q-values, we have shown that different symmetries can be applied, individually or simultaneously, even if each one requires a different approach, resulting in clear efficiency gains in all cases. Results suggest that, by applying two

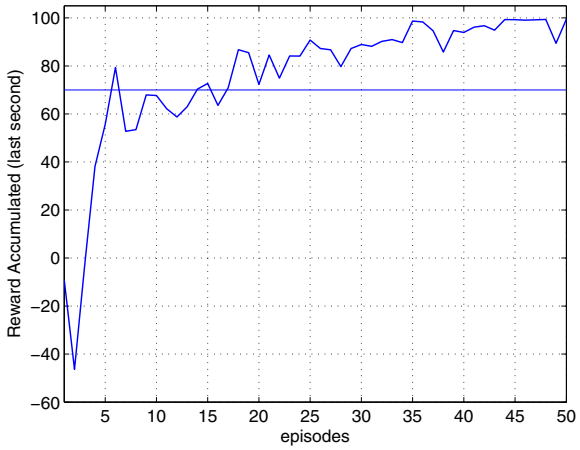

Figure 4. Temporal symmetry.

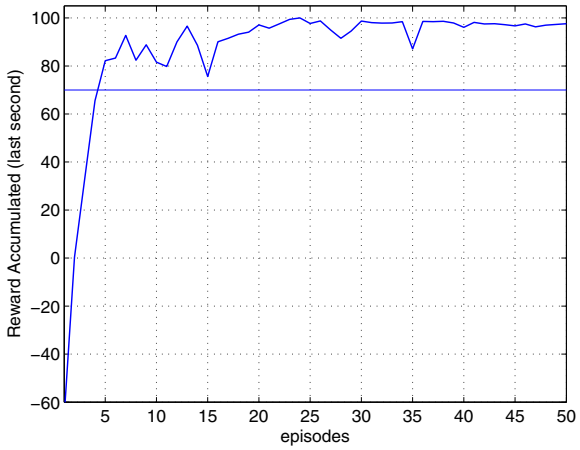

Figure 5. Temporal plus spatial symmetry, approach 1

symmetries together, the total gain is higher than combining the gains individually achieved by each symmetry alone. However, this conclusion should be further investigated, since it can depend to a great extent on the exact way in which the performance comparison is done.

ACKNOWLEDGMENT

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

REFERENCES

[1] M. Zinkevich and T. Balch, "Symmetry in Markov decision processes and its implications for single agent and multiagent learning," in *Proceedings of the 18th International Conference on Machine Learning (ICML-2001)*. Morgan Kaufmann, 2001, pp. 632–640.

[2] A. Agostini and E. Celaya, "Probability density estimation of the Q function for reinforcement learning," Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Barcelona, Tech. Rep. IRI-TR-06-09, 2009.

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

[4] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992. [Online]. Available: http://jmvidal.cse.sc.edu/library/watkins92a.pdf

[5] R. Bellman and S. Dreyfus, *Applied Dynamic Programming*. Princeton, New Jersey: Princeton University Press, 1962.

[6] J. A. Boyan and A. W. Moore, "Generalization in reinforcement learning: Safely approximating the value function," in *Advances in Neural Information Processing Systems 7*. MIT Press, 1995, pp. 369–376.

[7] M. Figueiredo, "On Gaussian radial basis function approximations: Interpretation, extensions, and learning strategies," *International Conf. on Pattern Recognition*, vol. 2, pp. 618–621, 2000.

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

[9] A. Dempster, N. Laird, D. Rubin *et al.*, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

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

[11] R. Neal and G. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in *Proceedings of the NATO Advanced Study Institute on Learning in graphical models*. Norwell, MA, USA: Kluwer Academic Publishers, 1998, pp. 355–368.

[12] M. Song and H. Wang, "Highly efficient incremental estimation of Gaussian mixture models for online data stream clustering," in *Proceedings of SPIE: Intelligent Computing: Theory and Applications III*, Orlando, FL, USA, 2005, pp. 174–183.

[13] M.-A. Sato and S. Ishii, "On-line EM algorithm for the normalized Gaussian network," *Neural Comput.*, vol. 12, no. 2, pp. 407–432, 2000.

[14] K. Doya, "Reinforcement learning in continuous time and space," *Neural Comput.*, vol. 12, no. 1, pp. 219–245, 2000.

[15] M.-A. Sato and S. Ishii, "Reinforcement learning based on on-line EM algorithm," in *Proceedings of the 1998 conference on Advances in neural information processing systems (NIPS'99)*. Cambridge, MA, USA: MIT Press, 1999, pp. 1052–1058.