LIGHT-WEIGHT REINFORCEMENT LEARNING WITH FUNCTION APPROXIMATION FOR REAL-LIFE CONTROL TASKS

Kary Främling
Helsinki University of Technology, PL 5500, FI-02015, Finland
Kary.Framling@hut.fi

Keywords: Reinforcement learning, function approximation, normalised radial basis function network, eligibility trace, mountain-car, cart-pole, pendulum.

Abstract: Despite the impressive achievements of reinforcement learning (RL) in playing Backgammon already in the beginning of the 90's, relatively few successful real-world applications of RL have been reported since then. This could be due to the tendency of RL research to focus on discrete Markov Decision Processes that make it difficult to handle tasks with continuous-valued features. Another reason could be a tendency to develop continuously more complex mathematical RL models that are difficult to implement and operate. Both of these issues are addressed in this paper by using the gradient-descent Sarsa(λ) method together with a Normalised Radial Basis Function neural net. The experimental results on three typical benchmark control tasks show that these methods outperform most previously reported results on these tasks, while remaining computationally feasible to implement even as embedded software. Therefore the presented results can serve as a reference both regarding learning performance and computational applicability of RL for real-life applications.

1 Introduction

A Reinforcement Learning (RL) agent can sense the state of itself and its environment, take actions that may change that state in some way and improve its behaviour based on a reward signal that expresses if one or several actions (or their result) was good or bad and to what extent. RL is therefore similar to the way that most animals learn to handle new or changing situations, e.g. babies learning how to walk and grab things, children learning how to ride a bicycle or adults learning how to drive a car. Despite this relationship with human and animal learning in the real world, it is surprising how few successful uses of RL for real-world control applications have been reported. The most well-known success of RL so far might be the TD-Gammon system (Tesauro, 1995) that learned to play Backgammon on world-champion level. However, Backgammon is not a control task and the methods and results are not easy to extrapolate to control tasks. A recent success on a real-world RL control task is the helicopter flight control reported in (Abbeel et al., 2007). What is common to both of these successful applications of RL (especially in helicopter flight) is that they are the result of rather complex calculations that involve several different learning phases and extensive hand-tuning by human designers. Even in experiments with standard control tasks such as those used in this paper, the RL methods employed tend to be complex and require extensive hand-tuning by human designers in order to make the learning task feasible. This complexity makes it difficult to re-use successful RL models in new application areas. Such models also tend to be too complex to provide plausible models of animal learning.

Both TD-Gammon and the helicopter controller use function approximation in order to be able to handle the huge state spaces involved. Especially continuous-valued control tasks tend to be problematic for RL because of their infinite state space. Many popular RL methods assume that the task to learn can be represented as a Markov Decision Process (MDP). In tasks with continuous-valued inputs it becomes necessary to discretize the state space in order to handle it as an MDP but the discretisation easily leads to an explosion of the state space. This contradiction between 'provably converging' but 'hard-to-use in real-life' MDP-based methods may also be a reason for the lack of reported successful uses of RL in real-world control tasks.

In this paper we will show how function approximation and one of the 'lightest' RL methods, i.e. model-free action-value learning with gradient-descent Sarsa(λ), can be used for successful learning of three benchmark control tasks. Less hand-tuning is needed and learning is significantly faster than in previously reported experiments. These small memory and computation requirements and the rapid learning should make it possible to use RL even in embedded systems that could adapt to their environment. These characteristics also make the learning biologically plausible. The experimental tasks used are the *Mountain-Car*, *Swinging up pendulum with limited torque* and *Cart-Pole* tasks.

After this introduction, section 2 describes the background and theory of gradient-descent Sarsa(λ), then section 3 presents the Normalized Radial Basis Function approximation technique used, followed by experimental results in section 4 and conclusions.

2 Action-value learning with function approximation

Action-value learning is needed in control tasks where no model of the controlled system is available. In the case of continuous-valued function approximation, *gradient-descent Sarsa(λ)* (Sutton and Barto, 1998, p. 211) can be used:

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha\left[r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)\right]\vec{e}_t, \tag{1}$$

where $\vec{\theta}_t$ is the parameter vector of the function approximator, r_{t+1} is the reward received upon entering a new state, α is a learning rate, γ is the discount rate and $Q_t(s_t, a_t)$ and $Q_t(s_{t+1}, a_{t+1})$ are the action-value estimates for the current and next state, respectively. In the case of an accumulating trace, the trace $\vec{e}_t$ is updated according to:

$$\vec{e}_t = \gamma\lambda\vec{e}_{t-1} + \nabla_{\vec{\theta}_t} Q_t(s_t, a_t) \tag{2}$$

with $\vec{e}_0 = \vec{0}$. $\nabla_{\vec{\theta}_t} f\left(\vec{\theta}_t\right)$, for any function f denotes the vector of partial derivatives (Sutton and Barto, 1998, p. 197):

$$\left(\frac{\partial f\left(\vec{\theta}_t\right)}{\partial\theta_t(1)}, \frac{\partial f\left(\vec{\theta}_t\right)}{\partial\theta_t(2)}, \ldots, \frac{\partial f\left(\vec{\theta}_t\right)}{\partial\theta_t(n)}\right)^T, \tag{3}$$

where $\vec{\phi}$ is *feature vector* that consist of the state variables of the task (continuous-valued or not). Eligibility traces can speed up learning significantly by improving temporal credit assignment. They are inspired from the behaviour of biological neurons that reach maximum eligibility for learning a short time after their activation and were mentioned in the context of Machine Learning at least as early as 1972 and used for action-value learning at least as early as 1983 (Barto et al., 1983), where the accumulating eligibility trace for discrete state tasks (MDP) was proposed:

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & \texttt{if } s = s_t \texttt{ and } a = a_t; \\ \gamma\lambda e_{t-1}(s,a) & \texttt{otherwise.} \end{cases} \tag{4}$$

for all s, a. λ is a trace decay parameter. s corresponds to the feature vector $\vec{\phi}$ in 3 but is restricted to binary values. Equation 4 is a special case of equation 2 when using binary state representations because then $\nabla_{\vec{\theta}_t} Q_t(s_t, a_t) = 1$ when $s = s_t$ and $a = a_t$ and zero otherwise. In (Singh and Sutton, 1996) it was proposed to use a replacing eligibility trace instead of the accumulating eligibility trace:

$$e_t(s,a) = \begin{cases} 1 & \texttt{if } s = s_t \texttt{ and } a = a_t; \\ 0 & \texttt{if } s = s_t \texttt{ and } a \neq a_t; \\ \gamma\lambda e_{t-1}(s,a) & \texttt{if } s \neq s_t. \end{cases} \tag{5}$$

The replacing eligibility trace outperformed the accumulating eligibility trace in the Mountain-Car task as reported in (Singh and Sutton, 1996). This is the main reason why replacing traces are considered to perform better than accumulating traces. In (Främling, 2007b) a generalisation was proposed for the replacing trace that makes it possible to use with continuous-valued function approximation:

$$\vec{e}_t = \max(\gamma\lambda\vec{e}_{t-1}, \nabla_{\vec{\theta}_t} Q_t(s_t, a_t)). \tag{6}$$

As for the accumulating trace, equation 5 is identical to equation 6 in the case of binary features, except for resetting the trace of unused actions of the current state to zero in equation 5. Even though the results in (Främling, 2007b) seem to confirm the advantage of replacing traces, especially when not resetting the traces of unused actions, there is still a lack of evidence on whether this is also true for other tasks. The experiments in this paper are performed using both kinds of eligibility traces as an attempt to provide increased insight into this issue.

3 Normalised Radial Basis Function Network applied to Action-Value learning

When function approximation with neural nets is used for learning an action-value function, an eligibility trace value is associated with every weight of the neural net (Barto et al., 1983). Then the eligibility trace value reflects to what extent and how long ago the neurons connected by the weight have been activated. The use of function approximation rather than binary lookup-tables signifies that more than one feature value ϕ can be different from zero. This is also the case with the binary CMAC (Cerebellar Model Articulation Controller) neural network (Albus, 1975) used in (Singh and Sutton, 1996). When using CMAC, there are as many active features as there are overlapping layers in the CMAC, which could lead to excessive weight changes in Equation 1. In order to avoid this, (Singh and Sutton, 1996) divide every value $e_t(s,a)$ by the sum of all feature values in order to avoid learning divergence as a result of excessive weight changes. No justification was given for this operation in (Singh and Sutton, 1996) but there seems to be a connection with the well-known Normalised Least Mean Squares (NLMS) method, where α in Equation 1 is replaced by α_{norm}:

$$\alpha_{norm} = \alpha / \sum \vec{e}_t. \tag{7}$$

The use of NLMS for RL in a real-world robotics task has been studied e.g. in (Främling, 2004) and (Främling, 2005). When using continuous-valued function approximation, most terms $\nabla_{\vec{\theta}_t} Q_t(s_t,a_t)$ in equation 2 will be different both from zero and one, which is the main source of incompatibility between the discrete eligibility traces (Equations 4 and 5) and their continuous-valued counterparts (Equations 2 and 6). In order to avoid weight divergence in equation 1, it can be assumed that an NLMS-type normalisation would be useful also with continuous-valued features. Because NLMS does not seem to have been used in neural nets, another kind of normalisation than NLMS is used here, i.e. the Normalised Radial Basis Function network (NRBF).

The continuous feature values are calculated by an RBF network as:

$$\phi_i(\vec{s}_t) = \exp\left(-\frac{(\vec{s}_t - \vec{c})^2}{r^2}\right), \tag{8}$$

where $\vec{c}$ is the *centroid* vector of the RBF neuron and r^2 is the *spread* parameter. The action-value estimate of action a is:

$$Q_t(\vec{s}_t, a) = \sum_{i=1}^{N} w_{ia} \left(\phi_i / \sum_{k=1}^{N} \phi_k\right), \tag{9}$$

where w_{ia} is the weight between the 'action' neuron a and the RBF neuron i. The division by $\sum_{k=1}^{N} \phi_k$, where N is the number of RBF neurons, is the normalisation step that ensures that the sum of all ϕ will be exactly one. Every parameter w_{ia}, has its corresponding eligibility trace value. For the accumulating trace, equation 2 was used in all experiments, while equation 6 was used for the replacing trace. In all experiments, an affine transformation mapped the actual state values from the respective value ranges into the range [0,1] that was used as $\vec{s}$. This transformation makes it easier to adjust the spread parameter ($r^2 = 0.01$ is used in all experiments).

4 Experiments

The Mountain-Car task (Figure 1) was used in (Singh and Sutton, 1996) and the results presented there seem to be the main reason why replacing eligibility traces are considered to perform better than accumulating traces. The second task consists in swinging up a pendulum with limited torque and keeping it in upright position (Figure 4). The pendulum task is surprisingly difficult for ordinary learning methods such as Sarsa(λ) with binary representations (lookup-table, CMAC) due to the non-linear control function required for passing from swing-up behaviour to balance-keeping behaviour. For the Cart-Pole task (Figure 7), successful results were reported using actor-critic and lookup-tables already in (Barto et al., 1983).

For each task, results are first shown by graphs that allow comparison between results shown here and those reported in existing literature. A supplementary graph may be used to show the results in a homogeneous way between the three tasks that allows to evaluate both learning speed and final performance after learning, as well as how well the accumulating and the replacing traces perform in each task. These graphs are also intended to provide a uniform benchmark framework for future experiments using the same tasks.

4.1 Mountain-Car

One of the first presentations of this task seems to be in (Moore, 1991), where a model of the environment was learned. The task description used here is identical to the one in (Singh and Sutton, 1996). The task

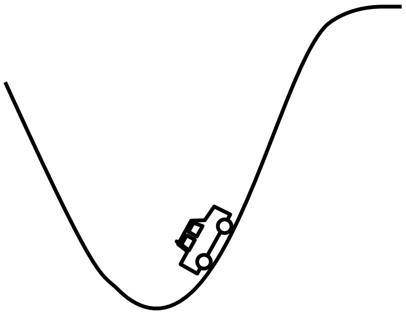

Figure 1: Mountain-Car task.

consists in accelerating an under-powered car up a hill, which is only possible if the car first gains enough inertia by backing away from the hill. The task has a continuous-valued state vector $\vec{s}_t = (x_t, v_t)$, i.e. the position x_t and the velocity v_t. At the beginning of each trial these are initialized randomly, uniformly from the range $x \in [-1.2, 0.5]$ and $v \in [-0.07, 0.07]$. The altitude is $\sin(3x)$. 8×8 RBF neurons were used with regularly spaced centroids. The agent chooses from three actions $\{+1, 0, -1\}$ corresponding to forward thrust, no thrust and reverse. The physics of the task are:

$$\begin{aligned} v_{t+1} &= \mathrm{bound}\left(v_t + 0.001 a_t + g\cos(3x_t)\right) \\ x_{t+1} &= \max\left\{x_t + v_{t+1}, -1.2\right\} \end{aligned},$$

where $g = 0.0025$ is the force of gravity and the bound operation places the variable within its allowed range. If x_{t+1} is clipped by the max-operator, then v_{t+1} is reset to 0. The terminal state is any position with $x_{t+1} > 0.5$. The reward function is the *cost-to-go* function as in (Singh and Sutton, 1996), i.e. giving -1 reward for every step except when reaching the goal, where zero reward is given. The weights of the neural net were set to zero before every new run so the initial action-value estimates are zero. The discount rate γ was one.

Figure 2 shows the average number of steps per episode for the best α values (4.9, 4.3, 4.3, 4.7, 3.9, 3.3 and 2.5) as a function of λ, where the numbers are averaged for 30 agent runs with 20 episodes each as in (Singh and Sutton, 1996). For some reason, the results reported here indicate 50% less steps than in most other papers (except e.g. (Smart and Kaelbling, 2000) who have similar numbers as here), which should be taken into consideration when comparing results. Taking this difference into consideration, the CMAC results shown in Figure 2 are consistent with those in (Singh and Sutton, 1996). The results using gradient-descent Sarsa(λ) and NRBF are significantly better than those obtained with CMAC.

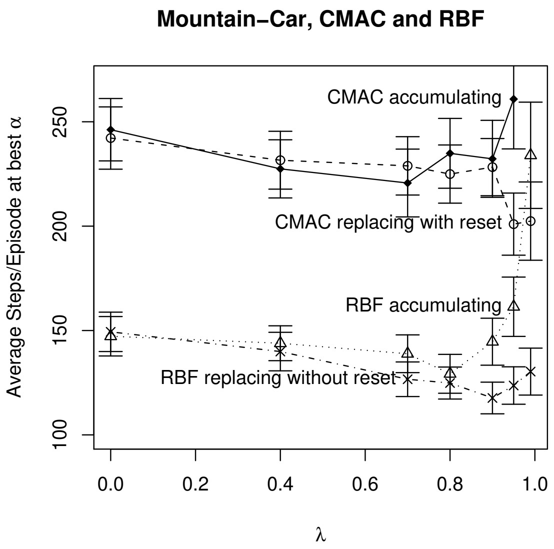

Figure 2: Average number of steps per episode for 30 agents running 20 episodes as in (Singh and Sutton, 1996).

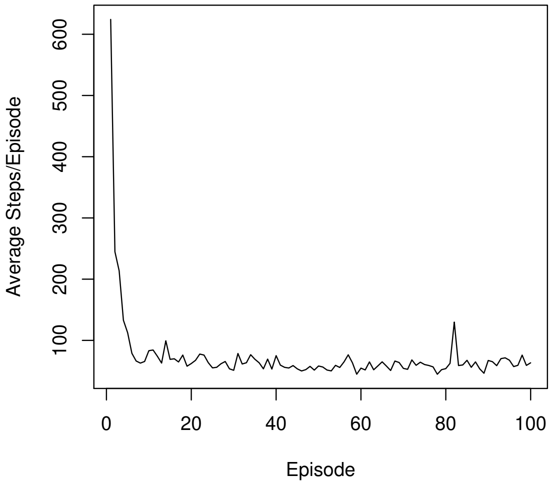

Figure 3: Average number of steps per episode for best-performing agent (replacing trace, $\alpha = 3.9$ and $\lambda = 0.9$).

Figure 3 shows the average number of steps per episode for the best-performing agent with $\alpha = 3.9$ and $\lambda = 0.9$. Convergence to an average of 60 steps per episode is achieved after about 40 episodes. This is better than the 'less than 100 episodes' indicated in (Abramson et al., 2003). The optimal number of steps indicated in (Smart and Kaelbling, 2000) is 56 steps so 60 steps per episode can be considered optimal due

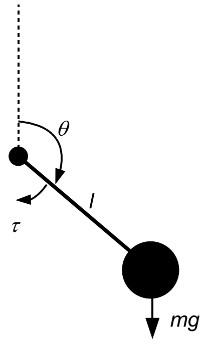

Figure 4: Pendulum swing-up with limited torque task.

to the continuing exploration caused by the -1 reward on every step. Therefore the converged learning result is also as good as in (Abramson et al., 2003) and (Schaefer et al., 2007). (Abramson et al., 2003) used Learning Vector Quantization together with Sarsa instead of using NRBF so the complexity of the learning algorithm is similar to the one used in this paper. The reward function provided more guidance towards the goal than the one used here and a fixed starting point was used instead of a random one. The neural net used in (Schaefer et al., 2007) is more complex than the one used here while a reward of +1 was given at the goal and zero otherwise. (Smart and Kaelbling, 2000) use an RBF-like neural network together with Sarsa but no eligibility traces and a reward that is inversely proportional to the velocity at goal (one if zero velocity) and zero otherwise. Results are indicated as a function of training steps rather than episodes but a rough estimate would indicate that convergence occurs after over 2000 episodes. The experimental settings and the ways in which results are reported in (Strens and Moore, 2002) do not make it possible to compare results in a coherent way. In (Mahadevan and Maggioni, 2007) convergence happens after about 100 episodes, which is much slower than in Figure 3 despite the use of much more mathematically and computationally complex methods. The results in (Främling, 2007a) are clearly superior to all others but they are obtained by having a parallel, hand-coded controller guiding the initial exploration, which could also be combined with gradient-descent Sarsa(λ) and NRBF. These differences make it difficult to compare results but the results shown in this paper are systematically better than comparable results reported in literature.

4.2 Pendulum swing-up with limited torque

Swinging up a pendulum to the upright position and keeping it there despite an under-powered motor is a task that has been used e.g in (Doya, 2000), (Schaal, 1997) and (Santamaría et al., 1998). The task description used here is similar to the one in (Doya, 2000). There are two continuous state variables: the angle θ and the angular speed $\dot{\theta}$. The agent chooses from the two actions $\pm 5\mathtt{Nm}$ corresponding to clockwise and anti-clockwise torque. The discount rate γ was set to one. The system dynamics are defined as follows:

$$ml^2\ddot{\theta} = -\pi\dot{\theta} + mgl\sin\theta + \tau, \theta \in [-\pi, \pi]$$
$$m = l = 1, g = 9.81, \mu = 0.01, \tau_{max} = 5Nm$$

At the beginning of each episode θ is initialized to a random value from $[-\pi, \pi]$ and $\dot{\theta} = 0$. Every episode lasted for 60 seconds with a time step $\Delta_t = 0.02$ using Euler's method. The performance measure used was the time that $|\theta| \leq \frac{pi}{2}$. The reward was $r_1 = \cos\theta - 0.7$, except when $|\theta| \leq \frac{pi}{5}$ where it was $r_2 = r_1 - |\dot{\theta}|$. Such a reward function encourages exploration when the pendulum cannot be taken directly to the upright position by the available torque. It also provides some guidance to that zero is the ideal speed in the upright position. This shaping reward remains less explicit than the one used e.g. in (Schaal, 1997) and (Santamaría et al., 1998). This learning task is more difficult than in (Schaal, 1997) and (Doya, 2000) where a model of the system dynamics either has been used directly or learned by a model. 10×10 NRBF neurons were used with regularly spaced centroids. An affine transformation mapped the actual state values from the intervals $[-\pi, \pi]$ and $[-1.5, 1.5]$ (however, angular speeds outside this interval are also allowed) into the interval [0,1] that was used as $\vec{s}$.

Figure 5 shows the time that the pendulum remains in the upright position for the best-performing agent (replacing trace, $\alpha = 1.5$ and $\lambda = 0.4$). An uptime over 40 seconds signifies that the pendulum is successfully balanced until the end, which occurs after about 10 episodes. In (Doya, 2000) a continuous actor-critic model was used that is computationally more complex than what is used in this paper. A model-free agent in (Doya, 2000) started performing successfully after about 50 episodes but never achieved a stable performance. The best results were achieved by an agent with a complete model of the system dynamics, which performed successfully after about 10 episodes. However, such model-based results should not be compared with model-free methods like the one used in this paper. In (Schaal, 1997)

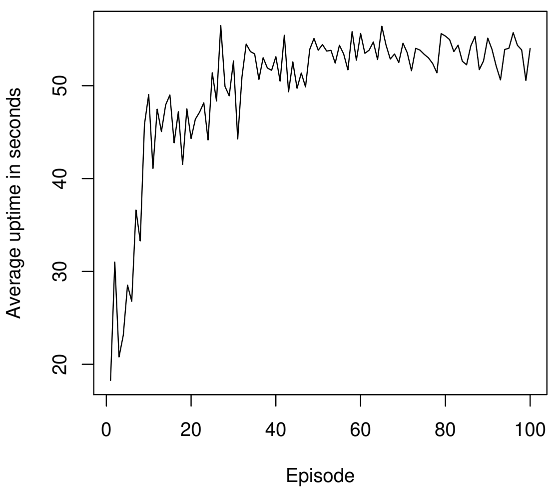

Figure 5: Average up-time for the best-performing agent in seconds as a function of episode for pendulum task.

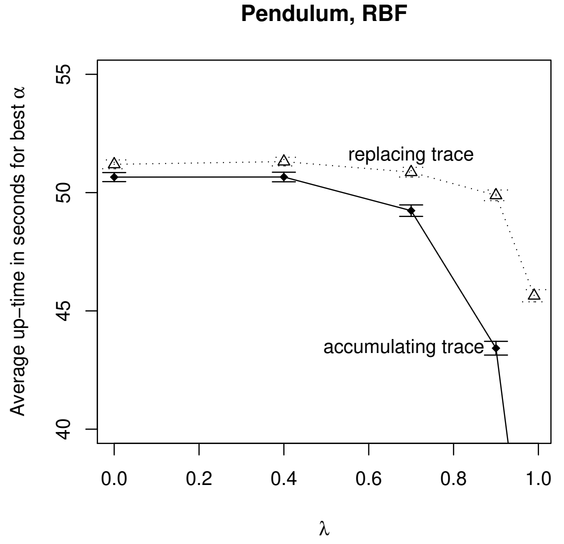

Figure 6: Average up-time in seconds for pendulum task with best α value as a function of λ. Error bars indicate one standard error.

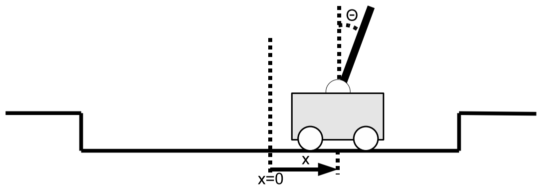

Figure 7: Cart-Pole task.

a model-based TD(λ) method was used that performed successfully after about 100 episodes, so despite the model-based approach the results are not as good as those in Figure 5. Gradient-descent Sarsa(λ) is used together with RBF networks in (Santamaría et al., 1998) also for the case of continuous actions. However, they used a reward function that guided learning much more than the one used here. Their instance-based agent that is computationally on the same level as the NRBF net used here did not manage to learn successfully, while the more complex case-based agent did so. The case-based agents managed to balance the pendulum after about 30 episodes but due to a cut-off in the top of the result-showing graphs it is difficult to assess the performance after that. In any case, the results shown in this paper are systematically better than those compared with here, while remaining simpler to use and computationally lighter.

As shown in Figure 6, the replacing trace consistently gives slightly better performance than the accumulating trace for this task. Learning results are also much more sensitive to the value of the learning rate for the accumulating trace than for the replacing trace.

4.3 Cart-Pole

One of the first uses of RL for the Cart-Pole task was reported in (Barto et al., 1983). It is unknown whether successful learning has been achieved with model-free Sarsa(λ) on this task. An actor-critic architecture was used in (Barto et al., 1983), (Kimura and Kobayashi, 1998) and (Schneegaß et al., 2007). Memory-based architectures were used in (Whitehead and Lin, 1995) while (Schaal, 1997) uses initial supervised learning for six episodes in order to provide an approximately correct initial action-value function. Some kind of reward shaping is also employed in order to simplify the learning task.

The task description used here is from (Barto et al., 1983). There are four continuous state variables: the angle θ and angular speed $\dot{\theta}$ of the pole and the position x and velocity $\dot{x}$ of the cart. The agent chooses from the two actions $\pm 10\mathtt{N}$ corresponding to right or left acceleration. The discount rate γ was set to one and ε-greedy exploration with $\varepsilon = 0.1$ was used. The system dynamics are:

$$\ddot{\theta} = \frac{g \sin\theta_t + \cos\theta_t \left[\frac{-F_t - ml\dot{\theta}_t^2 \sin\theta_t + \mu_c \operatorname{sgn}(\dot{x}_t)}{m_c + m} \right] - \frac{\mu_p \dot{\theta}_t}{ml}}{l \left[\frac{4}{3} - \frac{m \cos^2\theta_t}{m_c + m} \right]}$$

$$\ddot{x}_t = \frac{F_t + ml\left[\dot{\theta}_t^2 \sin\theta_t - \ddot{\theta}_t \cos\theta_t\right] - \mu_c \operatorname{sgn}(\dot{x}_t)}{m_c + m}$$

This paper	100
(Barto et al., 1983)	80
(Whitehead and Lin, 1995)	200
(Kimura and Kobayashi, 1998)	120
(Schneegaß et al., 2007)	100-150
(Lagoudakis and Parr, 2003)	100
(Mahadevan and Maggioni, 2007)	20

Table 1: Approximate average episode when successful balancing occurs in different sources, indicated where available

$$m_c = 1.0\texttt{kg}, m = 0.1\texttt{kg}, l = 0.5\texttt{m}, g = 9.81 m/s^2,$$
$$\mu_c = 0.0005, \mu_p = 0.000002, F_t = \pm 10\texttt{N}$$

Every episode started with $\theta = 0$, $\dot{\theta} = 0$, $x = 0$, $\dot{x} = 0$ and lasted for 240 seconds or until the pole tipped over ($\theta > 12°$) with a time step $\Delta_t = 0.02$ using Euler's method. The reward was zero all the time except when the pole tipped over, where a -1 'reward' was given. The NRBF network used $6 \times 6 \times 6 \times 6$ RBF neurons with regularly spaced centroids. An affine transformation mapped the actual state values into the interval [0,1] as for the other tasks.

Table 1 indicates approximate numbers for how many episodes were required before most agents have succesfully learned to balance the pole. Comparison between the results presented in different sources is particularly difficult for this task due to the great variation between the worst and the best performing agents. Anyway, the results presented here are at least as good as in the literature listed here despite the use of conceptually and computationally significantly simpler methods in this paper. The only exception is (Mahadevan and Maggioni, 2007) that performs well in this task, most probably due to a considerably more efficient use of kernels in the parts of the state space that are actually visited, rather than the regularly spaced $6 \times 6 \times 6 \times 6$ RBF neurons used here, of which a majority is probably never used. However, the kernel allocation in (Mahadevan and Maggioni, 2007) requires initial random walks in order to obtain samples from the task that allow offline calculation of corresponding Eigen vectors and allocating 'Proto-Value Functions' from them.

A challenge in interpreting the results of the experiments performed for this paper is that the agent 'un-learns' after a certain number of successful episodes where no reward signal at all is received. As pointed out in (Barto et al., 1983), this lack of reward input slowly deteriorates the already learned action-value function in critic-only architectures such as Sarsa(λ). The results shown in Figure 8 include this phenomenon, i.e. the average values are calculated for 200 episodes during which the task is successfully learned but also possibly un-learned and re-learned again.

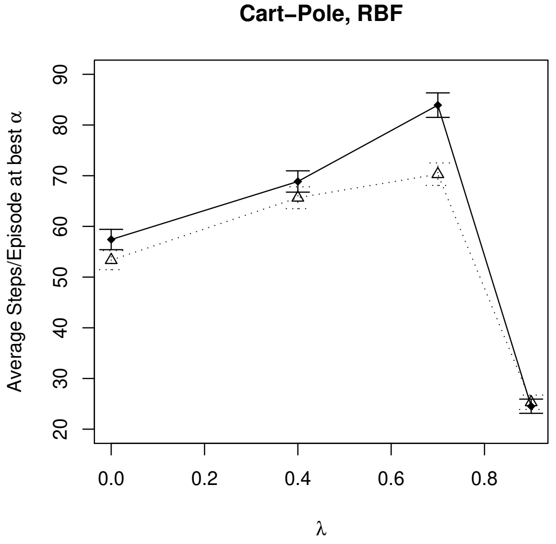

Figure 8: Average up-time in seconds for Cart-Pole task with best α value as a function of λ. Error bars indicate one standard error.

The results shown in Figure 8 differ from those of the previous tasks by the fact that the accumulating trace performs slightly better than the replacing trace and that the parameter sensitivity is quite similar for both trace types in this task. This difference compared to the two previous tasks is probably due to the delayed (and rare) reward that allows the eligibility trace to decline before receiving reward, thus avoiding weight divergence in equation 1. Further empirical results from tasks with delayed reward are still needed before declaring which eligibility trace type is better.

5 Conclusions

Gradient-descent Sarsa(λ) is not a new method for action-value learning with continuous-valued function approximation. Therefore it is surprising how little empirical knowledge exists that would allow practitioners to assess its usability compared with more complex RL methods. The only paper of those cited here that uses gradient-descent Sarsa(λ) is (Santamaría et al., 1998). The results obtained in this paper show that at least for the three well-known benchmark tests used, gradient-descent Sarsa(λ) together with NRBF function approximation tends to outperform all other methods while being the conceptually and computationally most simple-to-use method. The results presented here should provide a useful benchmark for future experiments because they seem to outperform most previously reported results, even those where a previously collected training set or a model of the system dynamics was used directly or learned.

These results are obtained due to the linking of different pieces together in an operational way. Important elements are for instance the choice of using NRBF and the normalisation of state values into the interval [0,1] that simplifies finding suitable learning rates and r^2 values. The replacing eligibility trace presented in (Främling, 2007b) is also more stable against learning parameter variations than the accumulating trace while giving slightly better performance. Future directions of this work could consists in performing experiments with more challenging tasks in order to see the limits of pure gradient-descent Sarsa(λ) learning. In tasks where gradient-descent Sarsa(λ) is not sufficient, combining it with pre-existing knowledge as in (Främling, 2007a) could be a solution for applying RL to new real-world control tasks.

REFERENCES

Abbeel, P., Coates, A., Quigley, M., and Ng, A. Y. (2007). An application of reinforcement learning to aerobatic helicopter flight. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*, pages 1–8, Cambridge, MA. MIT Press.

Abramson, M., Pachowicz, P., and Wechsler, H. (2003). Competitive reinforcement learning in continuous control tasks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN), Portland, OR*, volume 3, pages 1909–1914.

Albus, J. S. (1975). Data storage in the cerebellar model articulation controller (cmac). *Journal of Dynamic Systems, Measurement and Control*, September:228–233.

Barto, A., Sutton, R., and Anderson, C. (1983). Neuron-like adaptive elements that can solve difficult learning control problems. *IEEE Trans. on Systems, Man, and Cybernetics*, 13:835–846.

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

Främling, K. (2004). Scaled gradient descent learning rate - reinforcement learning with light-seeking robot. In *Proceedings of ICINCO'2004 conference, 25-28 August 2004, Setubal, Spain*, pages 3–11.

Främling, K. (2005). Adaptive robot learning in a non-stationary environment. In *Proceedings of the* 13^{th} *European Symposium on Artificial Neural Networks, April 27-29, Bruges, Belgium*, pages 381–386.

Främling, K. (2007a). Guiding exploration by pre-existing knowledge without modifying reward. *Neural Networks*, 20:736–747.

Främling, K. (2007b). Replacing eligibility trace for action-value learning with function approximation. In *Proceedings of the* 15^{th} *European Symposium on Artificial Neural Networks, April 25-27, Bruges, Belgium*, pages 313–318.

Kimura, H. and Kobayashi, S. (1998). An analysis of actor/critic algorithms using eligibility traces: Reinforcement learning with imperfect value functions. In *Proceedings of the* 15^{th} *Int. Conf. on Machine Learning*, pages 278–286.

Lagoudakis, M. G. and Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149.

Mahadevan, S. and Maggioni, M. (2007). Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *J. Mach. Learn. Res.*, 8:2169–2231.

Moore, A. (1991). Variable resolution dynamic programming. efficiently learning action maps in multivariate real-valued state-spaces. In *Machine Learning: Proceedings of the Eight International Workshop, San Mateo, CA.*, pages 333–337. Morgan-Kaufmann.

Santamaría, J., Sutton, R., and Ram, A. (1998). Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior*, 6:163–217.

Schaal, S. (1997). Learning from demonstration. In *Advances in Neural Information Processing Systems (NIPS)*, volume 9, pages 1040–1046. MIT Press.

Schaefer, A. M., Udluft, S., and Zimmermann, H.-G. (2007). The recurrent control neural network. In *Proceedings of* 15^{th} *European Symposium on Artificial Neural Networks, Bruges, Belgium, 25-27 April 2007*, pages 319–324. D-Side.

Schneegaß, D., Udluft, S., and Martinetz, T. (2007). Neural rewards regression for near-optimal policy identification in markovian and partial observable environments. In *Proceedings of* 15^{th} *European Symposium on Artificial Neural Networks, Bruges, Belgium, 25-27 April 2007*, pages 301–306. D-Side.

Singh, S. and Sutton, R. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158.

Smart, W. D. and Kaelbling, L. P. (2000). Practical reinforcement learning in continuous spaces. In *Proceedings of the Seventeenth* 17^{th} *International Conference on Machine Learning*, pages 903–910. Morgan Kaufmann.

Strens, M. J. and Moore, A. W. (2002). Policy search using paired comparisons. *Journal of Machine Learning Research*, 3:921–950.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning*. MIT Press, Cambridge, MA.

Tesauro, G. (1995). Temporal difference learning and td-gammon. *Communications of the ACM*, 38:58–68.

Whitehead, S. and Lin, L.-J. (1995). Reinforcement learning of non-markov decision processes. *Artificial Intelligence*, 73:271–306.