Gradient Algorithms for Exploration/Exploitation Trade-Offs: Global and Local Variants

Michel Tokic[1,2] and Günther Palm[1]

[1] Institute of Neural Information Processing, University of Ulm, Germany
[2] Institute of Applied Research, University of Applied Sciences
Ravensburg-Weingarten, Germany

Abstract. Gradient-following algorithms are deployed for efficient adaptation of exploration parameters in temporal-difference learning with discrete action spaces. Global and local variants are evaluated in discrete and continuous state spaces. The global variant is memory efficient in terms of requiring exploratory data only for starting states. In contrast, the local variant requires exploratory data for each state of the state space, but produces exploratory behavior only in states with improvement potential. Our results suggest that gradient-based exploration can be efficiently used in combination with off- and on-policy algorithms such as Q-learning and Sarsa.

Keywords: reinforcement learning, exploration/exploitation.

1 Introduction

In reinforcement learning (RL), one of the most challenging tasks is balancing the amount of exploration and exploitation [1]. If the behavior of an agent is purely exploratory, the outcome of random actions prevents from maximizing short-term reward. In contrast, if an agent is purely exploitative, the selection of sub-optimal actions possibly prevents from maximizing long-term reward, because of underestimating the outcome of optimal actions. Conclusively, the optimal balance is somewhere in between, dependent on many parameters such as the learning rate, discounting factor, learning progress, and of course on the learning problem itself.

Many different approaches exist for tackling the trade-off between exploration and exploitation. Based on a single exploration parameter, some basic policies select random actions either equally distributed (ε-Greedy) or value sensitively (Softmax) [1], or by a combination of both [2], with the advantage of not requiring to memorize any exploratory data. In contrast, other approaches utilize counters for every state (exploration bonuses) direct the exploration process towards finding the optimal action-selection strategy in polynomial time under certain circumstances [3, 4]. Nevertheless, basic policies such as ε-Greedy and Softmax are known to be very effective having a proper exploration parameter configured, which has been successfully shown for example in board games with

N. Mana, F. Schwenker, and E. Trentin (Eds.): ANNPR 2012, LNAI 7477, pp. 60–71, 2012.

huge discrete state spaces like Othello [5] or English Draughts [6]. In such state spaces, utility functions are hard to approximate and experiments for determining a proper exploration parameter can be time consuming. A non-convergent counter function is even harder to approximate than a convergent value function [7]. Interestingly, Daw et al. revealed in biologically-motivated studies on exploratory decisions in humans that there is *[...] no evidence to justify the introduction of an extra parameter that allowed exploration to be directed towards uncertainty (softmax with an uncertainty bonus): at optimal fit, the bonus was negligible, making the model equivalent to the simpler softmax* [8]. However, the search for an appropriate exploration parameter for such a policy remains as a challenging pattern-recognition task based on sensorimotor observations.

In the following, stochastic neurons are deployed for adapting exploration parameters by gradient-following algorithms. Adaptation of such parameters is with regard to the learning progress, instead of being tuned by hand in advance. We evaluate the presented approach in discrete and continuous state spaces using variants of the cliff-walking and mountain-car problems. The global variant of the presented algorithm was recently introduced in former research work [9]. Therefore, the contribution of this paper is an extended version in more detail also considering the local variant.

2 Methods

The learning problems considered in this paper can be described as Markov Decision Processes (MDP) that basically consist of a set of states, $\mathcal{S}$, and a set of possible actions within each state, $\mathcal{A}(s) \in \mathcal{A}, \forall s \in \mathcal{S}$ [1]. A stochastic transition function $\mathcal{P}(s, a, s')$ describes the (stochastic) behavior of the environment, i.e. the probability of reaching successor state s' after selecting action $a \in \mathcal{A}(s)$ in state s. The selection of an action is rewarded by a numerical signal from the environment, $r \in \Re$, used for evaluating the utility of the selected action. The goal of an agent is finding an optimal policy, $\pi^* : \mathcal{S} \to \mathcal{A}$, maximizing the cumulative reward. In the following, it is allowed for $\mathcal{S}$ to be continuous, but assumed that $\mathcal{A}$ is a finite set of actions. Action-selection decisions are taken at regular time steps, $t \in \{1, 2, \ldots, T\}$, until a maximum number of T actions is exceeded or a terminal state is reached.

2.1 Learning Algorithms

In temporal-difference learning, policies can be derived from utility functions representing so far learned knowledge [1]. An action-value function, $Q(s, a)$, denotes the cumulative and discounted reward for following policy π, when starting in state s and taking action a

$$Q(s,a) = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right\} \quad , \tag{1}$$

where $0 < \gamma \leq 1$ is a discounting factor used for weighting future rewards in $Q(s,a)$. Since $Q(s,a)$ depends on rewards received in the future, the cumulative reward is considered to be an expected value $E_\pi\left\{\cdot\right\}$.

Action-value function are learned from sensorimotor interactions of an agent with its environment. Two commonly used algorithms for learning $Q(s,a)$ are Sarsa for *on-policy* learning [7]

$$\begin{aligned} \varDelta_{\text{Sarsa}} &\leftarrow \left[r_{t+1} + \gamma Q\left(s_{t+1}, a_{t+1}\right) - Q\left(s_t, a_t\right)\right] \\ Q(s_t, a_t) &\leftarrow Q(s_t, a_t) + \alpha \varDelta_{\text{Sarsa}} \ , \end{aligned} \tag{2}$$

and Q-learning for *off-policy* learning [10]

$$\begin{aligned} b^* &\leftarrow \operatorname*{arg\,max}_{b \in \mathcal{A}(s_{t+1})} Q(s_{t+1}, b) \\ \varDelta_{\text{Qlearning}} &\leftarrow \left[r_{t+1} + \gamma Q\left(s_{t+1}, b^*\right) - Q\left(s_t, a_t\right)\right] \\ Q(s_t, a_t) &\leftarrow Q(s_t, a_t) + \alpha \varDelta_{\text{Qlearning}} \ , \end{aligned} \tag{3}$$

where α denotes a stepsize parameter [11]. The technical difference between both algorithms is the inclusion of successor-state information used for evaluating action a_t in state s_t. Sarsa includes the discounted value of the actual selected action in the successor state, $Q(s_{t+1}, a_{t+1})$, for which reason the algorithm belongs to the family of on-policy algorithms. In contrast, Q-learning includes the discounted value of the estimated optimal action in the successor state, $Q(s_{t+1}, b^*)$, for which reason it is an off-policy algorithm. On-policy algorithms have the advantage of including into $Q(s,a)$ respective costs from stochastic action-selection policies, but have in turn no convergence guarantee, except when the policy has a greedy behavior [1].

2.2 Basic Exploration Policies

ε-Greedy. One basic policy for trading-off exploration/exploitation is selecting in state s an equally-distributed random action with probability $0 \leq \varepsilon \leq 1$, which is called an ε-Greedy policy. With probability $1-\varepsilon$, a greedy action from the set of so far estimated optimal actions, $\mathcal{A}^*(s) = \operatorname{arg\,max}_{a \in \mathcal{A}(s)} Q(s,a)$, is selected

$$\pi(\varepsilon, s, a) = \begin{cases} \frac{1-\varepsilon}{|\mathcal{A}^*(s)|} + \frac{\varepsilon}{|\mathcal{A}(s)|} & \text{if } a \in \mathcal{A}^*(s) \\ \frac{\varepsilon}{|\mathcal{A}(s)|} & \text{otherwise} \ . \end{cases} \tag{4}$$

Softmax. A disadvantage of ε-Greedy is that exploration actions are selected equally distributed among all possible actions, which may cause the income of high negative rewards from several bad actions, even if their true utility is correctly estimated. Therefore, an alternative is determining the selection probabilities according to a Boltzmann distribution (the Softmax policy), which also takes so far estimated utility into account

$$\pi(\tau, s, a) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_b e^{\frac{Q(s,b)}{\tau}}} \ . \tag{5}$$

Low settings of the exploration parameter τ (temperature) cause greediness, however high settings cause randomness.

MBE. A known problem of Softmax is that it *[...] has large problems of focusing on the best actions while still being able to sometimes deviate from them* [2]. This issue can be improved by combining Softmax with ε-Greedy into the *Max-Boltzmann Exploration* (MBE) rule [2], which selects exploration actions according to Softmax instead of being equally distributed

$$\pi(\varepsilon,\tau,s,a) = \begin{cases} \text{random action from } \mathcal{A}^*(s) & \text{with probability } 1-\varepsilon \\ \text{Softmax action } \pi(\tau,s,a) & \text{with probability } \varepsilon \end{cases} \tag{6}$$

VDBE-Softmax. A drawback of the above policies is that an appropriate exploration parameter (ε or τ, or both for MBE) needs to be found for optimizing the cumulative reward. Such parameter varies dependently on learning progress, and typically an exploratory behavior is desired at the beginning of the learning process or in cases non-stationary environment responses are received. For this, some approaches make use of a decreasing exploration rate [12, 13], but which is known as to be inefficient for non-stationary environments. As a solution, we proposed the VDBE-Softmax policy in former research [14, 15], which adapts the exploration rate of MBE, state dependently based on fluctuations in the utility function

$$\begin{aligned} f(s,a,\phi) &= 1 - e^{\frac{-|Q^n(s,a)-Q(s,a)|}{\phi}} = 1 - e^{\frac{-|\alpha\cdot\Delta|}{\phi}} \\ \varepsilon_{t+1}(s) &= \delta\cdot f(s_t,a_t,\phi) + (1-\delta)\cdot\varepsilon_t(s)\ , \end{aligned} \tag{7}$$

where ϕ is a positive constant called *inverse sensitivity* and $\delta \in [0,1)$ a parameter determining the effect of the selected action on the exploration rate. The second parameter (temperature) is set constantly to the value of $\tau = 1$, using normalized Q-values within the interval $[-1,1]$.

3 Exploration-Parameter Control

Finding an appropriate exploration parameter by hand can be time consuming, and conclusively it is desired having algorithms adapting this parameter based on current operating conditions. The proposed solution for this problem is adapting the exploration parameter a_e of an action-selection policy, $\pi(a_e,\cdot,\cdot)$, towards improving the outcome of π based on some reasonable performance measure ρ. For maximizing ρ in the future, we deploy Williams' *"REINFORCE with multiparameter distributions"* algorithm using a stochastic neuron model [16], originally designed for reinforcement learning in continuous action spaces. The input of such neuron is a weighted parameter vector θ, from which the neuron determines the adaptable stochastic scalar as its output (the continuous-valued action). However, here we use discrete actions, thus the algorithm is applied for adapting the continuous-valued exploration parameter, i.e. REINFORCE Exploration-Parameter Control (REC). For example, if $\pi(a_e, \cdot, \cdot)$ is an ε-Greedy policy, the exploration parameter a_e refers to the exploration rate ε, i.e. $a_e \equiv \varepsilon$. In connectionist networks, such stochastic neurons can easily be integrated with compatibility to backpropagation [16]. The observed state s of the environment is typically the input to such network, which determines the parameter vector θ as its output to be processed by REC. However, in the following we use a tabular approximation of θ for the reason of measuring unbiased results.

At first, we show how a_e can be globally adapted with regard to the episodic cumulative reward, which is of interest for episodic learning problems consisting of a small set of starting states. Thereafter, a local variant is presented for determining a_e state dependently, aiming at producing exploratory behavior only in regions with improvement potential. However, the simple episodically version is an interesting variant because of being computational and memory efficient.

3.1 Global Episodic Control

In the following formulation, a single starting state s_s is assumed (for better readability), but which can easily be extended to multiple starting states as it will be discussed afterwards. The REC algorithm determines at each time step a continuous-valued action from a multiparameter distribution [16], representing the exploration parameter a_e. For this purpose, we use a normal (Gaussian) distribution with parameters μ (mean) and σ (standard deviation), which are given to the neuron as inputs.

At the beginning of each learning episode, the exploration parameter, being valid over the whole episode, is determined from the distribution (i.e. the activation function of the stochastic neuron), $a_e \sim \mathcal{N}(\mu, \sigma)$, whose density function g is given by

$$g(a_e, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(a_e-\mu)^2/2\sigma^2} \ . \tag{8}$$

Let θ denote the vector of adaptable parameters consisting of

$$\theta = \begin{pmatrix} \mu \\ \sigma \end{pmatrix} \ . \tag{9}$$

Our goal is adapting the components of θ at the end of episode i towards the gradient

$$\theta_{i+1} \approx \theta_i + \alpha \nabla_\theta \rho \ . \tag{10}$$

For improving the future performance of $\pi(a_e, \cdot, \cdot)$, the policies outcome can be measured as the cumulative reward in episode i

$$\rho_i = E\{r_1 + r_2 + \cdots + r_T | \pi_i(a_e, \cdot, \cdot)\} \ . \tag{11}$$

Next, the characteristic eligibility of each component of θ is estimated by

$$\frac{\partial \ln g(a_e,\mu,\sigma)}{\partial \mu} = \frac{a_e-\mu}{\sigma^2} \tag{12}$$

$$\frac{\partial \ln g(a_e,\mu,\sigma)}{\partial \sigma} = \frac{(a_e-\mu)^2-\sigma^2}{\sigma^3} \quad , \tag{13}$$

and a reasonable algorithm for adapting μ and σ has the following form

$$\Delta\mu = \alpha_R(\rho-\bar{\rho})\frac{a_e-\mu}{\sigma^2} \tag{14}$$

$$\Delta\sigma = \alpha_R(\rho-\bar{\rho})\frac{(a_e-\mu)^2-\sigma^2}{\sigma^3} \quad , \tag{15}$$

being applied at the end of each learning episode. The learning rate α_R has to be chosen appropriately, e.g. as a small positive constant, $\alpha_R = \alpha\sigma^2$, [16]. The baseline $\bar{\rho}$ is adapted by a simple reinforcement-comparison scheme

$$\bar{\rho} = \bar{\rho} + \alpha(\rho-\bar{\rho}) \ . \tag{16}$$

Analytically, in Equation 14 the mean μ is shifted towards a_e in case of $\rho \geq \bar{\rho}$. On the contrary, μ is shifted towards the opposite direction if ρ is less than $\bar{\rho}$. Similarly, in Equation 15 the standard deviation σ is adapted in a way that the occurrence of a_e is increased if $\rho \geq \bar{\rho}$, and decreased otherwise (see proof in [16]). In simple words, the standard deviation controls exploration in the space of a_e.

A proper functioning of the algorithm depends on some requirements. The exploration parameter, mean, and standard deviation need to be bounded for obtaining reasonable performance (see Table 1). Furthermore, if the learning problem consists of more than one starting state, the parameters μ, σ and $\bar{\rho}$ must be associated to each occurring starting state, i.e. $\mu \rightarrow \mu(s), \sigma \rightarrow \sigma(s)$ and $\bar{\rho} \rightarrow \bar{\rho}(s)$, because way costs might affect ρ unevenly. However, if a learning problem consists of just one starting state, all utilized parameters can be considered as being global parameters.

Table 1. Parameter bounds for determining a_e

Policy	$\mu_{\min}$; $a_e^{\min}$	$\mu_{\max}$; $a_e^{\max}$	$\sigma_{\min}$	$\sigma_{\max}$
REC ε-Greedy: $\pi(\varepsilon,\cdot,\cdot)$	0.0	1.0	0.001	5.0
REC MBE: $\pi(\varepsilon,\cdot,\cdot)$	0.0	1.0	0.001	5.0
REC Softmax: $\pi(\tau,\cdot,\cdot)$	0.001	1000.0	0.1	5000.0
REC VDBE-Softmax: $\pi(\phi,\cdot,\cdot)$	0.001	1000.0	0.1	5000.0

3.2 Local Step-Wise Control

The results from the previous section can easily be extended for obtaining a local variant, aiming at producing exploratory behavior only in states with improvement potential. In general, all utilized parameters become local parameters associated to each state, i.e. $\mu \rightarrow \mu(s), \sigma \rightarrow \sigma(s)$ and $\bar{\rho} \rightarrow \bar{\rho}(s)$. The mean and standard deviation are readapted after evaluating action a performed in state s (by Q-learning or Sarsa). In prior to an action selection in state s, an exploration parameter a_e is determined based on $\mu(s)$ and $\sigma(s)$

$$a_e \sim \mathcal{N}(\mu(s), \sigma(s)) \ . \tag{17}$$

For evaluating the utility of a_e, the estimated utility of the actual taken action, $Q_{t+1}(s_t, a_t)$, is considered

$$\rho = Q_{t+1}(s_t, a_t) \ . \tag{18}$$

The following equations now readapt the distribution parameters from state s_t based on the policies outcome using its current parameter a_e

$$\Delta\mu(s_t) = \alpha_R(\rho - \bar{\rho}(s_t)) \frac{a_e - \mu(s_t)}{\sigma(s_t)^2} \tag{19}$$

$$\Delta\sigma(s_t) = \alpha_R(\rho - \bar{\rho}(s_t)) \frac{(a_e - \mu(s_t))^2 - \sigma(s_t)^2}{\sigma(s_t)^3} \ . \tag{20}$$

Finally, the baseline $\bar{\rho}(s)$ is readapted analogously to Equation 16

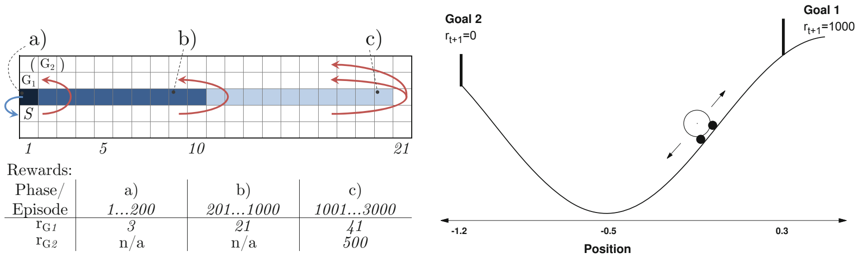
$$\bar{\rho}(s_t) = \bar{\rho}(s_t) + \alpha(\rho - \bar{\rho}(s_t)) \ . \tag{21}$$

4 Experiments

The presented policies are evaluated in two environments using *off-policy* Q-learning and *on-policy* Sarsa learning. First, a variation of the cliff-walking problem [1] is proposed as the *non-stationary cliff-walking problem* comprising a non-stationary environment. Second, a variation of the mountain-car problem is investigated comprising a continuous-valued state space approximated by a table, which causes partial observability of the actual coordinates. Investigated basic exploration policies are ε-Greedy, Softmax, MBE and VDBE-Softmax, using REC adaptation with parameter bounds according to Table 1. Since MBE requires two parameters to be set (ε and τ), we only adapt ε of this policy, while setting the temperature parameter constantly to the value of $\tau = 1$, and normalizing all Q-values in state s into the interval $[-1, 1]$. For the VDBE-Softmax policy, the inverse sensitivity parameter ϕ is adapted.

4.1 The Non-stationary Cliff-Walking Problem

The non-stationary cliff-walking problem is a modification of the cliff-walking problem presented by Sutton and Barto [1], which additionally comprises non-stationary responses of the environment. The goal for the agent is learning a path from start state S to goal state G_1, which is rewarded with the absolute



Rewards:

Phase/ Episode	a) *1...200*	b) *201...1000*	c) *1001...3000*
r_{G1}	*3*	*21*	*41*
r_{G2}	n/a	n/a	*500*

(a) (b)

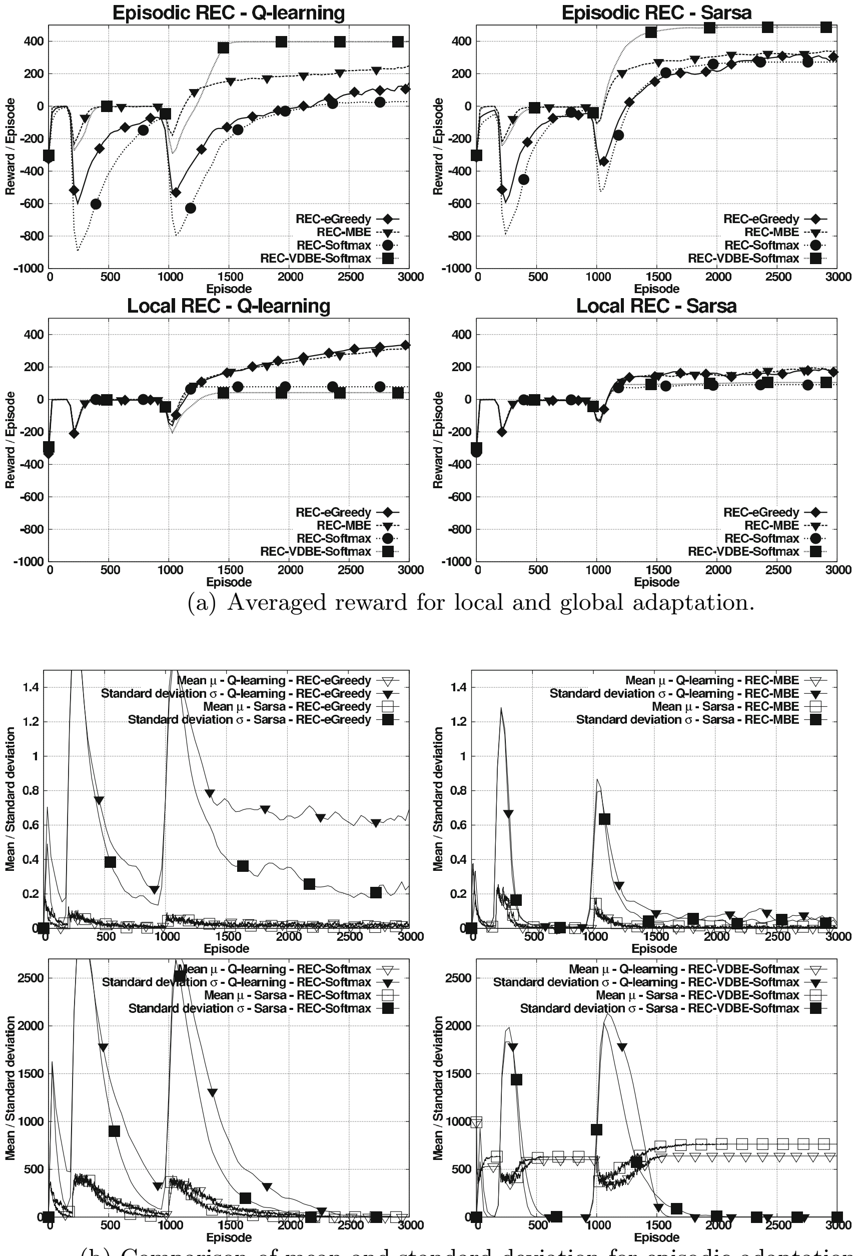
Fig. 1. The non-stationary cliff-walking problem (a) and the mountain-car problem with two goals (b)

costs of the shortest path minus 1 if successful (see Figure 1(a)). The reward for each action is defined as $r_{\text{step}} = -1$ (way costs). The environment also comprises unsafe *cliff states*, which lead to a high negative reward of $r_{\text{cliff}} = -100$, and also reset the agent back to the starting state S.

At the beginning of an experiment, learning takes place in phase (a) having one cliff state at left border. After 200 learning episodes, the grid world changes to phase (b), now comprising 10 cliff states. This change requires adapting the already learned behavior for circumventing the additional cliffs. After additional 800 episodes, the problem is tightened as shown in phase (c), where the number of cliffs is increased to 20. An alternative goal state $G2$ also appears, which is much higher rewarded with $r_{\text{G2}} = 500$ when entered.

Each episode begins in the starting state S, and terminates when either a goal state is entered or a maximum number of $T_{\max} = 200$ actions is exceeded. Throughout the experiment, the step-size parameter α is constantly set to the value of $\alpha = 0.2$. Since the learning problem is episodic, no discounting ($\gamma = 1$) is used. Finally, all action values are optimistically initialized with $Q_{t=0}(s, a) = 0$.

Results. Figure 2(a) shows the reward per episode averaged over 500 experiments. Averages of the mean and standard deviation for episodic policies are shown in Figure 2(b). It is observable that VDBE-Softmax maximizes the reward/episode in the episodic case. MBE shows best performance of the remaining three basic policies with the advantage of not requiring to memorize any further exploratory data such as utilized by VDBE-Softmax. The Sarsa algorithm shows better results for all four investigated policies when using episodic adaptation, but which is not the case for local adaptation. This discrepancy is for the reason that local adaptation tends to fast to become greedy, thus finding the second goal state in phase (c) more seldom. For comparison, all REC policies are better in episode 3000 compared to when using a pure greedy policy, which always converges to a reward per episode of 0, and which is only the optimal policy within the first 1000 episodes. In contrast, a pure random policy converges to -2750 respectively.



(a) Averaged reward for local and global adaptation.

(b) Comparison of mean and standard deviation for episodic adaptation.

Fig. 2. The non-stationary cliff-walking problem: Averaged results (smoothed) for investigated REC policies using Q-learning and Sarsa. Note the dynamics of exploration for non-stationary environment responses.

4.2 The Mountain-Car Problem with Two Goals

In the mountain-car problem [17], the goal is driving an underpowered car up a mountain road, by initially standing in the valley between two mountains. The problem is that gravity is stronger than the car's engine, thus reaching the mountain top by full throttle only is not possible. Instead the car has to swing up for collecting enough inertia for overcoming gravity. In the here presented modification of the original learning problem, two goal states are utilized as depicted in Figure 1(b), which are rewarded differently upon arrival.

Once the car reaches one of the two goals, an episode terminates and the car is instantly set back to the middle of the valley. The idea of utilizing two differently valued goals is for the reason of measuring performance improvements achieved by various exploration policies, because a simple greedy policy (ε-Greedy with $\varepsilon = 0$) leads to optimal performance in the original description of the learning problem. The state variables are continuously valued consisting of the position of the car, $-1.2 \leq x \leq 0.3$, and its velocity, $-0.07 \leq \dot{x} \leq 0.07$. The dynamics of the environment are described by differential equations

$$\begin{aligned} x_{t+1} &= bound\big[x_t + \dot{x}_{t+1}\big] \\ \dot{x}_{t+1} &= bound\big[\dot{x}_t + 0.001 a_t - 0.0025 \cos(3x_t)\big] \ . \end{aligned} \tag{22}$$

At each discrete time step, the agent can chose between one of seven actions, $a_t \in \{-1.0, -0.66, -0.33, 0, 0.33, 0.66, 1.0\}$, each rewarded by $r_{t+1} = -1$, except for reaching the right goal that is rewarded by $r_{t+1} = 1000$. An episode terminates when either one of the two goals has been arrived or when a maximum number of actions, $T_{\max} = 10000$, is exceeded. At the beginning of each episode the car is positioned in the valley at position $x = -0.5$ with initial velocity $\dot{x} = 0.0$. The state space is approximated by a 100×100 matrix, which causes the actual positions to be partially observable. Throughout the experiment, the step-size parameter α is constantly set to the value of $\alpha = 0.7$. Since the learning problem is episodic, no discounting ($\gamma = 1$) is used. Finally, all action values are optimistically initialized with $Q_{t=0}(s, a) = -200$.

Results. The results are averaged over 200 runs as shown in Figure 3. In general, best results are achieved for any policy using Q-learning in combination with local adaptation. Episodic adaptation of MBE outperforms ε-Greedy and Softmax. Furthermore, the Sarsa algorithm shows only to be advantageous in combination with ε-Greedy and Softmax using episodic adaptation, in contrast to MBE and VDBE-Softmax behaving more efficiently in combination with Q-learning. In the first phase of learning a degradation of performance is recognizable for episodic MBE and VDBE-Softmax. This is for the reason of learning at first a path to the left goal, and thereafter finding the right (better) goal. For comparison, a greedy policy converges to an average reward per episode of 114, in contrast to a pure random policy converging to -5440.

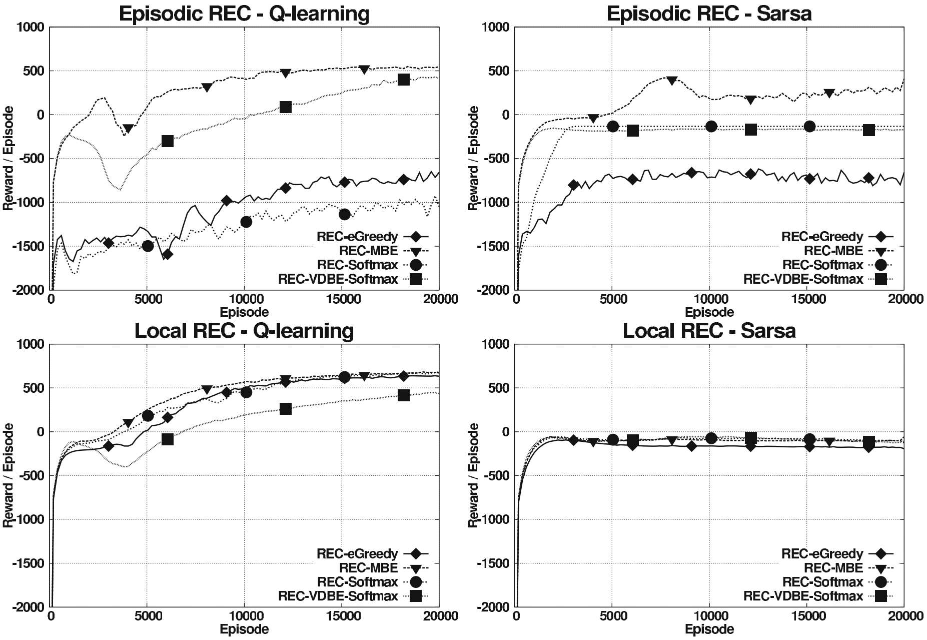


Fig. 3. The mountain-car problem with two goals: Averaged reward (smoothed) for investigated REC policies using Q-learning and Sarsa

5 Discussion and Conclusions

In this paper, new exploration policies are proposed for adapting the exploration parameter of basic exploration policies by gradient-following algorithms. Local and global variants have been evaluated in two different learning problems requiring to properly trade off exploration and exploitation. Results from the non-stationary cliff-walking problem show how the exploration parameter is readapted based on learning progress when non-stationary environment responses are received. For basic exploration policies in combination with REC, MBE shows to be most efficient in terms of performance. However, additional exploratory data might improve results as shown using episodic VDBE-Softmax in the non-stationary cliff-walking problem, but which is not always the case as it is observable in the mountain-car problem with two goals. When comparing local and global adaptation, the performance shows to be nearly the same for MBE, suggesting to use the memory and computational efficient global variant in episodic learning problems. Finally, the presented results show that gradient-following algorithms can be effectively used for balancing the exploration/exploitation dilemma inherent to reinforcement learning.

Acknowledgements. Michel Tokic received funding by the collaborative center for applied research *ZAFH-Servicerobotik*. The authors gratefully acknowledge the research grants of the federal state Baden-Württemberg and the European Union.

References

[1] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)

[2] Wiering, M.: Explorations in Efficient Reinforcement Learning. PhD thesis, University of Amsterdam, Amsterdam (1999)

[3] Thrun, S.B.: Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, Carnegie Mellon University, Pittsburgh, PA, USA (1992)

[4] Auer, P.: Using confidence bounds for exploitation-exploration trade-offs. The Journal of Machine Learning Research 3, 397–422 (2002)

[5] van Eck, N.J., van Wezel, M.: Application of reinforcement learning to the game of Othello. Computers and Operations Research 35, 1999–2017 (2008)

[6] Faußer, S., Schwenker, F.: Learning a strategy with neural approximated temporal-difference methods in english draughts. In: Proceedings of the 20th International Conference on Pattern Recognition, ICPR 2010, pp. 2925–2928. IEEE Computer Society (2010)

[7] Rummery, G.A., Niranjan, M.: On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University (1994)

[8] Daw, N.D., O'Doherty, J.P., Dayan, P., Seymour, B., Dolan, R.J.: Cortical substrates for exploratory decisions in humans. Nature 441(7095), 876–879 (2006)

[9] Tokic, M., Palm, G.: Adaptive exploration using stochastic neurons. In: Proceedings of the 22nd International Conference on Artificial Neural Networks, Lausanne, Switzerland. Springer (to appear, 2012)

[10] Watkins, C.: Learning from Delayed Rewards. PhD thesis, University of Cambridge, England (1989)

[11] George, A.P., Powell, W.B.: Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. Machine Learning 65(1), 167–198 (2006)

[12] Grzes, M., Kudenko, D.: Online learning of shaping rewards in reinforcement learning. Neural Networks 23(4), 541–550 (2010)

[13] Nouri, A., Littman, M.L.: Multi-resolution exploration in continuous spaces. In: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems, vol. 21, pp. 1209–1216 (2009)

[14] Tokic, M., Palm, G.: Value-Difference Based Exploration: Adaptive Control between Epsilon-Greedy and Softmax. In: Bach, J., Edelkamp, S. (eds.) KI 2011. LNCS, vol. 7006, pp. 335–346. Springer, Heidelberg (2011)

[15] Tokic, M., Ertle, P., Palm, G., Söffker, D., Voos, H.: Robust Exploration/Exploitation Trade-Offs in Safety-Critical applications. In: Proceedings of the 8th International Symposium on Fault Detection, Supervision and Safety of Technical Processes, Mexico City, Mexico. IFAC (to appear, 2012)

[16] Williams, R.J.: Simple statistical Gradient-Following algorithms for connectionist reinforcement learning. Machine Learning 8, 229–256 (1992)

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