Meta-Learning of Exploration and Exploitation Parameters with Replacing Eligibility Traces

Michel Tokic[1,2], Friedhelm Schwenker[1], 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

When developing autonomous learning agents, the performance depends crucially on the selection of reasonable learning parameters, for example learning rates or exploration parameters. In this work we investigate meta-learning of exploration parameters by using the *"REINFORCE exploration control"* (REC) framework, and combine REC with replacing eligibility traces, which are a basic mechanism for tackling the problem of delayed rewards in reinforcement learning. We show empirically for a robot example and the mountain-car problem with two goals how the proposed combination can help to improve learning performance. Furthermore, we also observe that the setting of time constant λ is not straightforward, because it is intimately interrelated with the learning rate α.

1 Introduction

Controlling exploration and exploitation is one of the main challenges when developing autonomous learning agents. In general, exploratory actions lead to an increase in knowledge about the long-term utility of actions (long-term optimization), but often may cause the income of negative reward due to randomly selected bad actions. However, exploiting knowledge (short-term optimization) may also lead to sub-optimal action selections if the utility of an optimal action is underestimated. As a consequence, the *dilemma between exploration and exploitation* arises [1].

Several approaches exist to tackle this problem in reinforcement learning. Using action counters for determining confidence intervals is a popular approach in the domain of machine learning [2, 3, 4]. In contrast, neurobiologically inspired models utilize the immediate reward [5] or the temporal-difference error [6] to adapt the amount of exploration, e.g. by the meta-parameter τ of Softmax action selection. In a more recent approach, we proposed to use the *value difference* (the product between the temporal-difference error and the learning rate) as an indicator for the *uncertainty of knowledge about the environment* [7]. This indicator is used for controlling the action-selection policy between Greedy and Softmax, which aims at robustness with regard to stochastic rewards and even non-stationary environments [8].

In this paper we consider the problem of adapting the amount of exploration and exploitation in model-free reinforcement learning. We combine our recently

proposed *"REINFORCE exploration control"* (REC) policies [9, 10] with replacing eligibility traces [11], for tackling the problem of delayed rewards. We investigate the proposed algorithm on a reward model of a crawling robot that learns to walk forward through sensorimotor interactions, and also on the mountain-car problem with two goals.

2 Methods

We investigate the problem of maximizing an agent's cumulative reward over time, which in our experiments can be described as learning in a Markovian Decision Process (MDP) in discrete time $t \in \{0, 1, 2, \dots\}$ [1]. In general, an MDP consists of a finite set of states $\mathcal{S}$ and a finite set of possible actions in each state, $\mathcal{A}(s) \in \mathcal{A}, \forall s \in \mathcal{S}$. A transition function $P(s, a, s')$ 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. After the selection of an action, a reward $r \in \mathbb{R}$ is received from the environment and the agent finds itself in a successor state $s' \in \mathcal{S}$. The choice of action a is significant, and therefore the general goal is to find an optimal action-selection policy, $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$, that maximizes the cumulative reward.

2.1 Action-value functions

Action-selection policies can either be learned using model-based techniques (the model of rewards and dynamics are approximated separately) or model-free techniques (only the value function is learned) [1]. Both require learning a value function for the prediction of future reward. In the following, we are particularly interested in optimizing model-free techniques, for the reason of being closely related to reinforcement learning in the brain [12, 13].

Action-selection policies can be derived from value functions representing so far learned knowledge about the future reward [1]. An action-value function, $Q(s, a)$, approximates the cumulative 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 $E_\pi\{\cdot\}$ which depends on the action-selection policy π. The parameter γ is allowed to take on the value of 1 only in episodic learning problems, i.e. an episode must terminate after a maximum of T steps, which prevents $Q(s, a)$ from growing to an infinite sum. In case a fixed horizon does not exist, γ must to be chosen < 1.

2.2 Q-learning with replacing eligibility traces

The action-value function is sampled from interactions with the environment. For this we use Watkin's Q-learning algorithm [14], which adapts reward estimates according to:

$$b^* \leftarrow \arg\max_{b \in \mathcal{A}(s')} Q(s', b) \tag{2}$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha \underbrace{\left(r + \gamma Q(s', b^*) - Q(s,a)\right)}_{\text{Temporal-Difference Error } \Delta} \tag{3}$$

where $0 < \alpha \leq 1$ denotes a step-size parameter [15]. For an optimal action-value function, $Q^*(s,a)$, from which the optimal (greedy) policy π^* can be derived, the temporal-difference error is zero for all observation tuples (s, a, r, s').

As proposed by Singh and Sutton [1, 11], we also combine Q-learning with *replacing eligibility traces*, which is known as $Q(\lambda)$-learning as shown in Algorithm 1. The advantage is that rewards are propagated faster to previously taken actions, which tackles the general problem of accounting delayed rewards in reinforcement learning. Visited state-actions pairs $e(s_e, a_e)$ are memorized in an *eligibility trace list*, where each entry in this list is associated with an additional memory variable, the *eligibility trace*, used for weighting the current temporal-difference error Δ also in the action value $Q(s_e, a_e)$ of previously taken actions. The trace for the last taken action is set to $e(s,a) \leftarrow 1$, which means that the temporal-difference error is fully credited.

All traces are decayed by $\gamma\lambda$ after taking an action. In this sense $0 \leq \lambda \leq 1$ denotes a time constant. For $\lambda = 0$, credit is only assigned to the last taken action, where Eq. (3) is only computed for the most recent observation tuple (s, a, r, s'). On the other hand, a time constant of $\lambda = 1$ refers to Monte-Carlo backups. Eligibility traces $e(s_e, a_e)$ are removed from the list as soon as their value becomes lower than a small threshold Θ, and thus the parameter λ does implicitly control over the amount of actions the current temporal-difference error is propagated into the past. The list of eligibility traces is also cleared in case an exploratory action is selected, because Q-learning learns the action-value function independently of the actual policy (called off-policy learning), assuming to follow a greedy policy in the limit for $t \to \infty$. Therefore, crediting the reward of exploratory actions to previously taken actions has no necessary relationship to a greedy policy [1].

3 Exploration and exploitation

In the following we describe two basic strategies for deriving action-selection probabilities $\pi(s,a)$ from the action-value function, and afterwards we show how to combine them for meta-parameter learning.

The ε-Greedy policy selects a uniform randomly distributed action with probability $0 \leq \varepsilon < 1$ [1]. With probability $1 - \varepsilon$, a greedy action from the set $\mathcal{A}^*(s)$

Algorithm 1 Robot control: $Q(\lambda)$-learning with local REC adaption of ε-Greedy

Initialize Q arbitrarily, e.g. $Q(s,a)=0$ for all s,a
Initialize start state, e.g. $s \leftarrow \{g_x = g_y = 0\}$

repeat
    EXPLORATION / EXPLOITATION:
    $\xi \leftarrow$ random number from the interval $(0,1)$
    draw $\varepsilon(s)$ from a Gaussian distribution: $\varepsilon(s) \sim \mathcal{N}(\mu(s), \sigma(s))$

    if $\xi < \varepsilon(s)$ **then**
        $a \leftarrow$ random action from the set $\mathcal{A}(s)$
    else
        $a \leftarrow \arg\max_{b \in \mathcal{A}(s)} Q(s,b)$
    end if

    take action a
    observe reward r and successor state s'

    COMPUTE TEMPORAL-DIFFERENCE ERROR:
    $\Delta \leftarrow r + \gamma \max_{b \in \mathcal{A}(s')} Q(s',b) - Q(s,a)$

    ELIGIBILITY-TRACE DECAY / CLEANUP:
    $e(s,a) \leftarrow 1$
    $V(s) \leftarrow \max_{b \in \mathcal{A}(s)} Q(s,b)$
    if $V(s) == Q(s,a)$ **then**
        for all (s_e, a_e) in e-trace list: **do**
            $Q(s_e,a_e) \leftarrow Q(s_e,a_e) + \alpha \Delta e(s_e,a_e)$
            $e(s_e,a_e) \leftarrow \lambda \gamma e(s_e,a_e)$
            if $e(s_e,a_e) < \Theta$ **then**
                mark $e(s_e,a_e)$ for deletion
            end if
        end for
        delete marked eligibility traces
    else
        $Q(s,a) \leftarrow Q(s,a) + \alpha \Delta$
        clear e-trace list
    end if

    LOCAL REC ADAPTATION:
    $\rho \leftarrow r + \gamma \max_{b \in \mathcal{A}(s')} Q(s',b)$
    $\mu(s) \leftarrow bound\left[\mu(s) + \alpha(\rho - \bar{\rho}(s))\,(\varepsilon(s) - \mu(s))\right]$
    $\sigma(s) \leftarrow bound\left[\sigma(s) + \alpha(\rho - \bar{\rho}(s))\,\frac{(\varepsilon(s)-\mu(s))^2 - \sigma(s)^2}{\sigma(s)}\right]$
    $\bar{\rho}(s) \leftarrow \bar{\rho}(s) + \alpha(\rho - \bar{\rho}(s))$

    $s \leftarrow s'$
until robot is switched off

of so far estimated optimal actions in state s is selected:

$$\mathcal{A}^*(s) = \arg\max_a Q(s,a)$$

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

Note that in any state s all selection probabilities $\pi_{\mathrm{EG}}(s,a)$ sum up to 1. A drawback of ε-Greedy is the choice of uniformly selected random actions, which might cause the income of negative reward due to bad actions. However, the ε-Greedy policy is reported for being hard to beat when a proper exploration parameter ε is configured [16].

The second policy we investigate is the Softmax policy, which selects an action according to its weighting in a Boltzmann distribution [1]:

$$\pi_{\mathrm{SM}}(s,a) = \frac{\exp\left(\frac{Q(s,a)}{\tau}\right)}{\sum_b \exp\left(\frac{Q(s,b)}{\tau}\right)} \quad . \tag{5}$$

This policy utilizes a positive parameter τ, called temperature, which controls between exploration and exploitation. High values of τ lead to equally distributed random actions, however low values to greedy actions. Again, in any state s all selection probabilities $\pi_{\mathrm{SM}}(s,a)$ sum up to 1.

In general, it is a problem to define global constants $\tau > 0$ and $\varepsilon \in [0,1]$ for achieving reasonable performance. Therefore, these parameters are typically initialized with a high value at the beginning of an experiment, being decreased over time. Especially for large state-action spaces, such fine-tuning is most often a time-consuming process.

3.1 Meta-learning of exploration/exploitation parameters

A drawback of ε-Greedy and Softmax is the *optimism in the face of uncertainty*. For this we recently proposed *"Value-Difference Based Exploration with Softmax"* (VDBE-Softmax) [7], which controls exploration and exploitation between Greedy and Softmax in a meta-learning fashion. A local exploration rate $\varepsilon(s)$, initialized by 1, is assigned to each state in the state space, which denotes the probability of selecting an exploratory action in state s. The selection probabilities $\pi(s,a)$ are adapted in dependence of fluctuations in the action-value function, which are considered to be a measure of the uncertainty in knowledge about the environment. Furthermore and in contrast to the ε-Greedy policy, exploratory actions are not selected equally distributed, but weighted according to the Softmax rule, which prevents selecting of very bad actions due to their weighting in the Boltzmann distribution. This idea of combining both policies was introduced by Wiering [17], called *Max Boltzmann Exploration* (MBE), but without learning of the policy parameters. Instead he configures ε and τ (globally) by hand.

The general idea of VDBE-Softmax is that high fluctuations of the action-value function should lead to a high degree of exploration, i.e. $\varepsilon(s) \to 1$, because the observation is insufficiently approximated by the prediction. On the other hand, when the prediction about future reward is well approximated, so far learned knowledge should be exploited, i.e. $\varepsilon(s) \to 0$. In this sense the corresponding exploration rate in state s is updated after each value-function backup

for any action a in state s according to:

$$\varepsilon(s) \leftarrow \varepsilon(s) + \delta \left[1 - \exp\left(\frac{-|\alpha \Delta|}{\phi} \right) - \varepsilon(s) \right] \quad , \tag{6}$$

where ϕ is a positive parameter for the *sensibility* with regard to the absolute value difference $|\alpha \Delta|$ [7]. In case an exploratory action should be selected, all Q-values in state s are scaled into the interval $[-1, 1]$, and the action is selected according to Eq. (5) using $\tau = 1$. As proposed in [18, 7] the learning rate δ can be determined online by the inverse of the number of actions, i.e. $\delta = \frac{1}{\mathcal{A}(s_t)}$ in the current state s_t.

Kobayashi and colleagues [6] proposed a similar approach (using the temporal-difference error Δ), but adapting the parameter τ of Softmax in a global manner instead. In contrast, VDBE-Softmax is a state-based strategy, which has the advantage of selecting exploratory actions only in states where the observation is insufficiently approximated by the action-value function.

The extension of VDBE-Softmax for $Q(\lambda)$-learning is straightforward. We simply need to apply the learning rule right after each action-value update (line 19 of Algorithm 1), but additionally including the eligibility trace for state s_e and action a_e. Therefore Eq. (6) is slightly modified to:

$$\varepsilon(s_e) \leftarrow \varepsilon(s_e) + \delta \left[1 - \exp\left(\frac{-|\alpha \Delta e(s_e, a_e)|}{\phi} \right) - \varepsilon(s_e) \right] \quad . \tag{7}$$

3.2 REINFORCE exploration control

We recently proposed a further alternative for meta-learning of exploration parameters called *REINFORCE Exploration Control* (REC) [9, 10]. The general idea is to control the exploration parameter of any above mentioned policies[3] by a gradient-following algorithm. The heart of REC is Williams *"REINFORCE with multiparameter distributions"* algorithm [19] for reinforcement learning in continuous action spaces.

In REC the exploration parameter of a policy is considered to be an continuous action. In this sense, we proposed two variants: (1) the global (episodic) variant selecting an exploration parameter for the duration of an learning episode, and (2) the local (stepwise) variant selecting a state-based exploration parameter before actually selecting an action by one of the above policies. In any of both cases, the exploration parameter is drawn according to a Gaussian distribution with parameters μ (mean) and σ (standard deviation). For example, in the local variant the parameter of ε-Greedy is selected according to:

$$\varepsilon(s) \sim \mathit{bound}\left[\mathcal{N}\left(\mu(s), \sigma(s) \right) \right] \quad , \tag{8}$$

where $\mathit{bound}[\cdot]$ ensures that $\varepsilon(s)$ stays within the interval $[0, 1]$. After taking action a according to the policy, and after observing the successor state s' and

[3] ε-Greedy: ε; Softmax: τ; MBE: ε; VDBE-Softmax: ϕ.

reward r, the local distribution parameters $\mu(s)$ and $\sigma(s)$ are adapted according to a reinforcement comparison scheme:

$$\mu(s) \leftarrow bound\left[\mu(s) + \alpha_R(\rho - \bar{\rho}(s))\frac{\varepsilon(s) - \mu(s)}{\sigma(s)^2}\right] \tag{9}$$

$$\sigma(s) \leftarrow bound\left[\sigma(s) + \alpha_R(\rho - \bar{\rho}(s))\frac{(\varepsilon(s) - \mu(s))^2 - \sigma(s)^2}{\sigma(s)^3}\right] \tag{10}$$

using performance measure

$$\rho = r + \max_b Q(s', b) \tag{11}$$

and its baseline

$$\bar{\rho}(s) \leftarrow \bar{\rho}(s) + \alpha(\rho - \bar{\rho}(s)) \ . \tag{12}$$

The learning rate α_R has to be chosen appropriately, e.g. as a small positive constant, $\alpha_R = \alpha\sigma^2$, as proposed by Williams [19]. Furthermore, all REC parameters must be bounded, e.g. $0 \leq \varepsilon(s), \mu(s) \leq 1$ and $0.001 \leq \sigma(s) \leq 5$. The bounds for Softmax, MBE and VDBE-Softmax can be taken according to [10]. In this paper, the performance measure ρ slightly differs from [10], which yielded to improved results in the experiments (especially for Softmax).

In contrast, the global variant of REC draws the exploration parameter at the beginning of an episode, for which reason the distribution parameters and baseline need only to be approximated for starting states [9, 10]. When updating the distribution parameters at the end of episode i, the sum of rewards is taken as performance measure ρ_i, i.e.

$$\rho_i = \sum_{t=1}^{T} r_t \ . \tag{13}$$

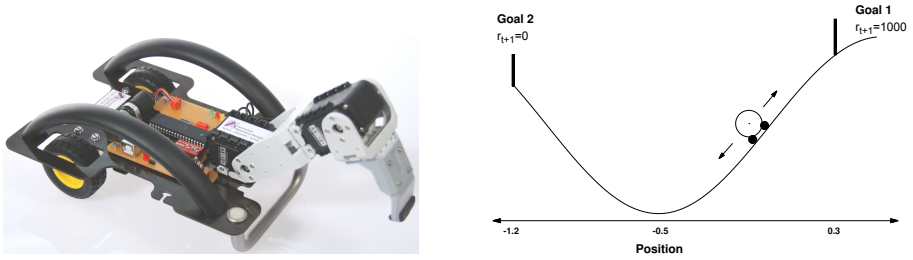
In case a learning problem has only one starting state (such as in the investigated mountain-car problem with two goals), a stateless (global) approximation of μ, σ and $\bar{\rho}$ can be used.

4 Experiments

In this section, the two learning problems shown in Fig. 1 are investigated using $Q(\lambda)$-learning with meta-parameter learning of exploration parameters. First, a little crawling robot is investigated, which is a non-episodic learning problem on which the local variant of REC is applied. Second, the mountain-car problem with two goals is investigated, which is an episodic learning problem, and therefore the global variant of REC is applied.

4.1 The Robot

We investigate a little crawling robot whose architecture was inspired from [21]. Fig. 1(a) shows the corresponding hardware robot. The general aim is learning



(a) (b)

Fig. 1. Investigated learning problems: (a) the crawling robot [20], and (b) the mountain-car problem with two goals [9].

to crawl forward through sensorimotor interactions, which is achieved by a cyclic policy representing movements of the two joints g_x and g_y. The components of the corresponding MDP are defined as follows:

States: At each time step, the state $s \in \mathcal{S}$ consists of the arm's discrete joint positions at present, i. e. the state is fully described by $s = \{g_x, g_y\}$. Due to the small onboard memory of size 2kB, each joint is discretized into 5 equidistant state positions resulting in total to 25 states as shown in [20].

Actions: The set of possible actions in each state $s \in \mathcal{S}$ consists of the four cardinal directions in the state-space model:

$$\mathcal{A}(s) \in \{\text{UP}, \text{DOWN}, \text{LEFT}, \text{RIGHT}\} \quad .$$

Rewards: Performing an action a in the environment leads to a reward, $r \in \mathbb{R}$, which is measured as the number of accumulated wheel-encoder tics while repositioning the arm. The encoder tics trigger the external interrupt of the microcontroller. An interrupt service routine accumulates positive and negative encoder ticks for the reward signal r delivered to the learning algorithm.

For simplicity we do not model transition probabilities on the crawling robot, because actions (movements of the joints) always transition with probability 1 to the corresponding neighbor state.

We performed simulation experiments with a reward model sampled from the robot as shown in [20]. For simulating sensor noise, the reward from the model is perturbed with a Gaussian noise (mean 0 and variance 1). Fig. 2 shows the results of our study, which are averages over 1000 runs, and using a discounting factor of $\gamma = 0.95$. In general it's observable that local REC adaptation behaves very robust independently of the policy and its parameter to be adapted. Using low learning rates of $\alpha = 0.1$, eligibility traces significantly improve the results the higher λ was chosen, which improves the standard Q-learning algorithm ($\lambda = 0$) from Eq. (3). In contrast, high learning rates lead in general to better results, with little effect of using eligibility traces. From the results with high learning rates we further observe that the VDBE-Softmax policy is in general a bit better, but which is on cost of additionally memorizing $\varepsilon(s)$.

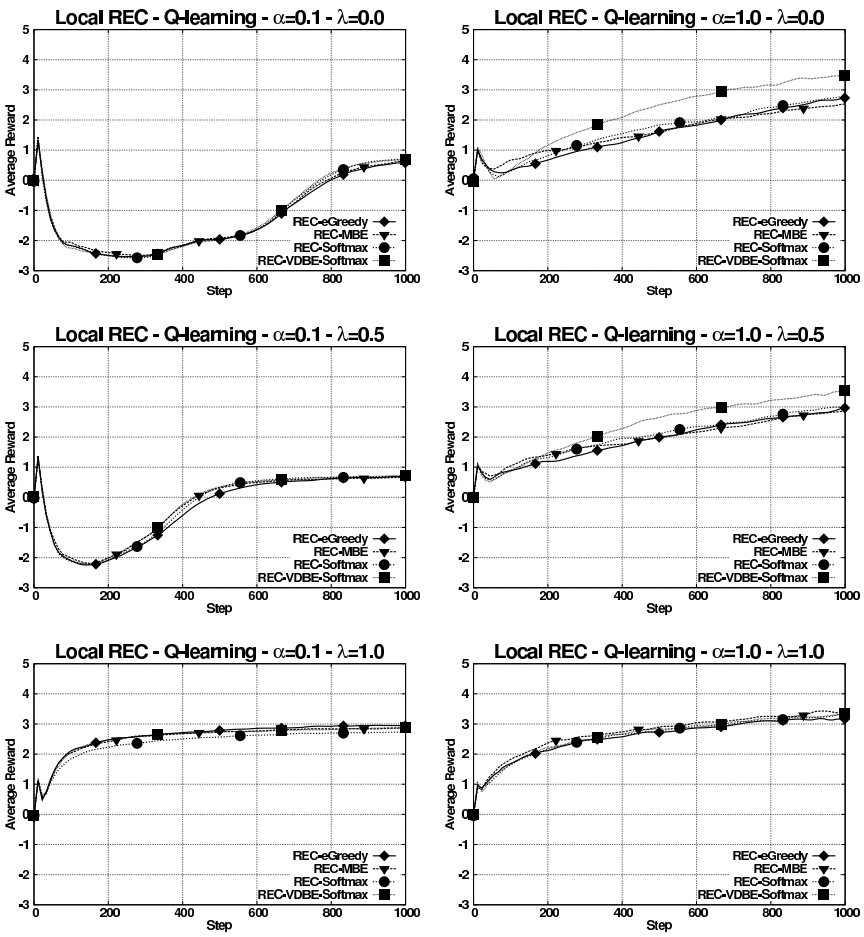


Fig. 2. The crawling robot: results for low learning rates of $\alpha = 0.1$ are on the left, high learning rates of $\alpha = 1.0$ are on the right. Results are smoothed and averaged over 1000 runs.

4.2 The mountain-car problem with two goals

We investigate $Q(\lambda)$-learning on the *mountain-car problem with two goals* as proposed in [9, 10] and depicted in Fig. 1(b). This learning problem is an extension of the originally proposed mountain-car problem [11], but having two goal states and an enlarged action set.

Our results shown in Fig. 3 reveal an interesting effect. The higher λ was chosen, the more policies tend to behave greedily, with the result of terminating an episode more likely at the left goal. For MBE eligibility traces seem to be contra productive. In contrast, the VDBE-Softmax results are improved in the range of $0.3 < \lambda < 0.5$, but also with degrading performance the more λ approaches 1.

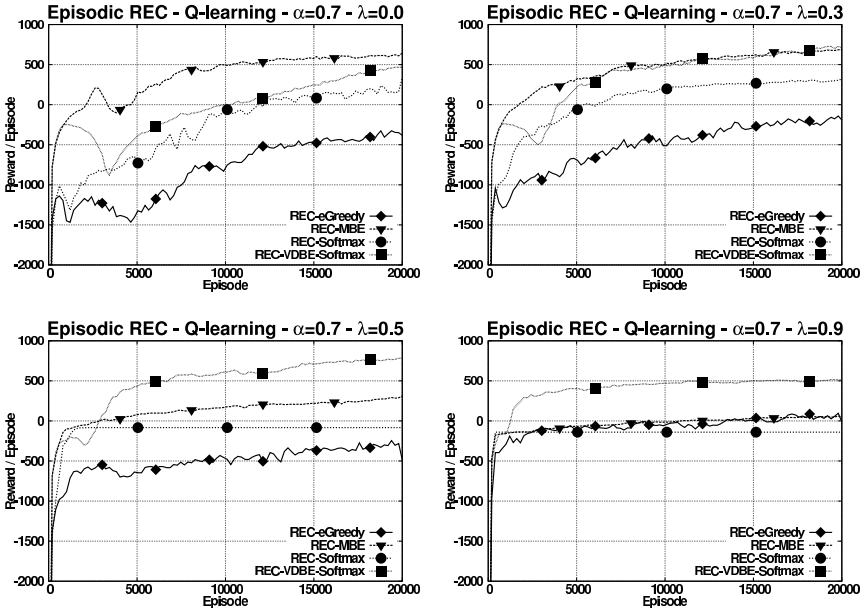


Fig. 3. The mountain-car problem with two goals: smoothed results for various λ; each averaged over 200 runs.

5 Summary & Conclusion

We showed that replacing eligibility traces can improve the reward of an agent, which is consistent with the results shown by Singh and Sutton [11]. As shown on the robot example, learning improves the more λ approaches 1, but this is not the case for the mountain-car problem with two goals. Therefore it is definitely not straightforward to decide on the choice of time constant λ. The reason for this effect is that oversampling of actual performed actions can also lead to underestimating actions not selected so far (which might be better, than their current action-value predicts). As a result, it is more likely for $\lambda \to 1$ that a greedy behavior arises, as shown in the results of the mountain-car problem with two goals. Therefore, the optimal choice of λ is dependent on the learning rate α used for sampling the action-value function, which was also shown in the results of Singh and Sutton [11]. As a conclusion, meta-learning of λ and α in combination remains to be an interesting direction of further research.

Finally, it seems reasonable to initiate a discussion of the idea that reinforcement learning should also be considered as part of the relatively new research paradigm of *partially-supervised learning*, which by now had its emphasis on combinations of unsupervised and supervised learning. Our opinion is that the general framework of developing reinforcement learning agents fits well into this paradigm due to the following reasons: (1) reinforcement learning is utilized for sampling estimates about the future reward that are usually approximated by

value functions [1], which (2) are most often learned using supervised learning algorithms, e.g. in a neural network fashion [22, 23]. In previous research we successfully showed this relationship in the development of learning agents for board games [24, 25]. Also in the context of neurobiology all three paradigms apparently interact with each other [26, 27].

References

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

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

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

[4] Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In Fürnkranz, J., Scheffer, T., Spiliopoulou, M., eds.: Machine Learning: ECML 2006. Volume 4212 of LNCS. Springer Berlin / Heidelberg, Berlin, Heidelberg (2006) 282–293

[5] Schweighofer, N., Doya, K.: Meta-learning in reinforcement learning. Neural Networks **16**(1) (2003) 5–9

[6] Kobayashi, K., Mizoue, H., Kuremoto, T., Obayashi, M.: A meta-learning method based on temporal difference error. In Leung, C.S., Lee, M., Chan, J.H., eds.: Neural Information Processing (ICONIP 2009) . Number 5863 in Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2009) 530–537

[7] Tokic, M., Palm, G.: Value-difference based exploration: Adaptive exploration between epsilon-greedy and softmax. In: KI 2011: Advances in Artificial Intelligence. Springer Berlin / Heidelberg (2011) 335–346

[8] 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 (2012) 660–665

[9] Tokic, M., Palm, G.: Adaptive exploration using stochastic neurons. In Villa, A., Duch, W., Érdi, P., Masulli, F., Palm, G., eds.: Artificial Neural Networks and Machine Learning – ICANN 2012. Springer Berlin / Heidelberg (2012) 42–49

[10] Tokic, M., Palm, G.: Gradient algorithms for exploration/exploitation trade-offs: Global and local variants. In Mana, N., Schwenker, F., Trentin, E., eds.: Artificial Neural Networks in Pattern Recognition. Springer Berlin / Heidelberg (2012) 60–71

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

[12] Niv, Y., Daw, N.D., Dayan, P.: Choice values. Nature Neuroscience **9**(8) (2006) 987–988

[13] Niv, Y.: Reinforcement learning in the brain. Journal of Mathematical Psychology **53**(3) (2009) 139–154

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

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

[16] Vermorel, J., Mohri, M.: Multi-armed bandit algorithms and empirical evaluation. In: ECML. Volume 3720 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2005) 437–448
[17] Wiering, M.: Explorations in Efficient Reinforcement Learning. PhD thesis, University of Amsterdam, Amsterdam (1999)
[18] Tokic, M.: Adaptive e-Greedy exploration in reinforcement learning based on value differences. In Dillmann, R., Beyerer, J., Hanebeck, U., Schultz, T., eds.: KI 2010: Advances in Artificial Intelligence. Volume 6359 of Lecture Notes in Artificial Intelligence. Springer Berlin / Heidelberg (2010) 203–210
[19] Williams, R.J.: Simple statistical Gradient-Following algorithms for connectionist reinforcement learning. Machine Learning **8** (1992) 229–256
[20] Tokic, M., Bou Ammar, H.: Teaching reinforcement learning using a physical robot. In: Proceedings of the Workshop on Teaching Machine Learning at the 29th International Conference on Machine Learning, Edinburgh, UK (2012) 1–4
[21] Kimura, H., Miyazaki, K., Kobayashi, S.: Reinforcement learning in POMDPs with function approximation. In: Proceedings of the 14th International Conference on Machine Learning, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1997) 152–160
[22] Riedmiller, M.: Neural fitted q iteration - first experiences with a data efficient neural reinforcement learning method. In: ECML, Springer Berlin / Heidelberg (2005) 317–328
[23] Riedmiller, M., Montemerlo, M., Dahlkamp, H.: Learning to drive a real car in 20 minutes. In: Proceedings of the FBIT 2007 Conference, Jeju, Korea. Special Track on autonomous robots. (2007)
[24] 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, IEEE Computer Society (2010) 2925–2928
[25] Faußer, S., Schwenker, F.: Neural approximation of monte carlo policy evaluation deployed in connect four. In: Proceedings of the 3rd IAPR workshop on Artificial Neural Networks in Pattern Recognition, Springer Berlin / Heidelberg (2008) 90–100
[26] Doya, K.: What are the computations of the cerebellum, the basal ganglia and the cerebral cortex? Neural Networks **12**(7–8) (1999) 961–974
[27] Bostan, A.C., Dum, R.P., Strick, P.L.: The basal ganglia communicate with the cerebellum. Proceedings of the National Academy of Sciences **107**(18) (2010) 8452–8456