Action Selection Methods in a Robotic Reinforcement Learning Scenario

Francisco Cruz [1,2], Peter Wüppen [1], Alvin Fazrie [1], Cornelius Weber [1], and Stefan Wermter [1]
[1]Knowledge Technology, Department of Informatics, Universität Hamburg, Hamburg, Germany
[2]Escuela de Computación e Informática, Facultad de Ingeniería, Universidad Central de Chile, Santiago, Chile.
Email: {cruz, 5wueppen, 4fazrie, weber, wermter}@informatik.uni-hamburg.de

***Abstract*—Reinforcement learning allows an agent to learn a new task while autonomously exploring its environment. For this aim, the agent chooses an action to perform among the available ones for a certain state. Nonetheless, a common problem for a reinforcement learning agent is to find a proper balance between exploration and exploitation of actions in order to achieve an optimal behavior. This paper compares multiple approaches to the exploration/exploitation dilemma in reinforcement learning and, moreover, it implements an exemplary reinforcement learning task within the domain of domestic robotics to show the performance of different exploration policies on it. We perform the domestic task using ϵ-greedy, softmax, VDBE, and VDBE-Softmax with online and offline temporal-difference learning. The obtained results show that the agent is able to collect larger and faster reward by using the VDBE-Softmax exploration strategy with both Q-learning and SARSA.**

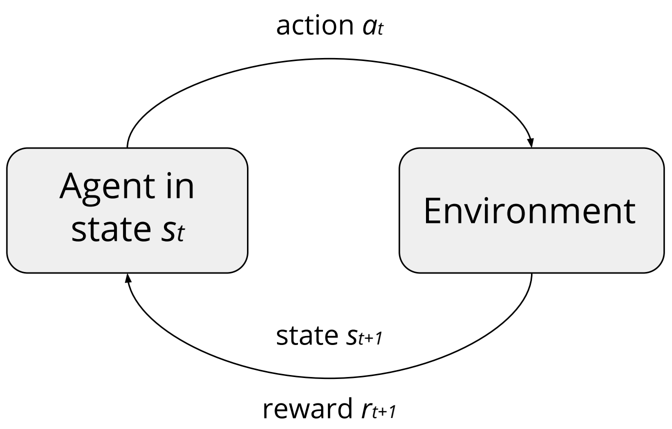


Fig. 1. The reinforcement learning loop showing the agent's interaction with the environment. Every time step from the state s_t, the agent selects an action a_t to be performed in the environment obtaining a new state s_{t+1} and a reward r_{t+1}.

I. Introduction

Autonomous learning, from a human perspective, is an activity where past experiences influence the decision-making on current actions based on associations made with those actions previously [1]. The same principle is replicated by Reinforcement Learning (RL) [2]. RL is a class of learning mechanisms where an agent autonomously executes actions and receives a reward from its environment. Once the agent is faced with a similar condition, it will try to make decisions according to rewards which were obtained earlier.

In our daily life, we need to enhance the learning process in order to maximize the benefits. One crucial dilemma is the balance between exploration and exploitation [3], such as when we face a certain problem in our daily activities and we need to decide whether to utilize a known strategy to solve the problem or to look for a different, possibly better solution. Similarly, this also becomes a challenging task in RL to enhance the performance by balancing the proportion between exploration and exploitation. Imbalance could lead to adverse effects on learning performance [2], [3]. The domination of exploration would obstruct the agent to maximize short-term reward since the exploration approach may generate negative reward from the environment. On the other hand, the domination of exploitation may block an agent from maximizing long-term reward because the chosen actions may stay suboptimal.

One of the most common approaches to balance the ratio of exploration and exploitation is by implementing the ϵ-greedy method [4]. This simple method often leads to successful outcomes, but one of the issues with ϵ-greedy is that its setting is global and may not accommodate to state-specific requirements. Besides ϵ-greedy, many other successful methods have been introduced in an attempt to mitigate weaknesses of traditional approaches, such as softmax [5], VDBE [6], and VDBE-Softmax [7]. In this paper, we will describe and test them using off-policy and on-policy learning, i.e. Q-learning and SARSA, in a domestic robot scenario.

This paper is organized as follows: in the second section, we present the RL basics along with temporal-difference learning algorithms Q-learning and SARSA. The third section presents the exploration strategies used in this work: ϵ-greedy, softmax, VDBE, and VDBE-Softmax. In the fourth section, we show a domestic robotic scenario which is described as a Markov decision process. The fifth section exposes the main findings along with a discussion of the obtained results. Finally, in the sixth section, we draw conclusions from this work.

II. Temporal-difference Learning

Reinforcement learning represents a framework where an agent is able to learn a task by interacting with the environment [8]. A graphical representation of the basic concept of RL can be seen in Figure 1.

The core of the RL is formed by a Markov Decision Process (MDP) [9]. An MDP characterizes several components for RL which model a problem with $\{S, A, r, \delta, \pi, V^{\pi}\}$ parameters [10]. A state space, denoted by S, is a discrete set of environment states; an action space, denoted by A, is a discrete set of actions from the environment's agent; a state transition function, denoted by $\delta : S \times A \to S$, is a function which gives the potential state s' when the action a is conducted; a reward function, denoted by $r : S \times A \to \mathbb{R}$, is a function which turns each transition for a given state into a scalar value. Other components are the policy and the state value function where policy, denoted by π, is a function which specifies the agent's behavior. It maps the action to be taken for each given state. i.e., $\pi_t : S \to A$. The state value function, denoted by $V^{\pi} : S \to \mathbb{R}$, is a function that will be used to obtain the highest reward as the agent will always try to learn. It specifies the value for each state and maps the state to the reward that an agent can expect to accumulate.

Initially, an agent observes a certain state $s_t \in S$ in each time step and decides on a possible action a_t to perform, $a_t \in A(s_t)$, where $A(s_t)$ is the set of all possible actions in state s at timestep t. Once the action is performed by the agent, the environment gives a reward r_{t+1} accordingly, which could be positive or negative. When the agent receives the reward, it uses it to update its action selection policy in order to maximize its obtained cumulative reward [10].

The policy π is used to map a state to an action through a function $\pi_t(s_t, a_t)$. The learning process changes the policy in order to obtain experience from the given environment and to maximize the accumulated reward. The main aim of this RL model is to achieve an optimal behavior by performing the best action for each state to get the maximum observed rewards for the agent [2], [11].

Therefore, the agent is expected to acquire the maximal total reward from the action taken in each step as the main objective. The estimation of the total reward that an agent could attain it is expressed by the following equation:

$$Q^{\pi}(s, a) = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_t + k + 1 | s_t = s, a_t = a\}.$$

where $Q^{\pi}(s, a)$ is the state-action value, r is the reward following the policy π in the state s to select action a. Moreover, γ is the discount rate of future rewards for which $0 < \gamma \leq 1$ for periodic learning and $0 < \gamma < 1$ for continuous learning tasks [12].

Updates to the state-action value function are learned by observing the interaction between the agent and its environment. There are two common algorithms from the branch of temporal-difference learning, namely SARSA for on-policy control [13] and Q-learning for off-policy control [4], [14]. Both of them are characterized by three parameters which influence their behavior. Firstly, the learning rate α determines to which degree the most recent information will modify the previous information. Secondly, the discount factor γ decides the importance of future reward. If γ is 0, the agent will only consider the current rewards, and if the factor is 1, the agent will attempt a long-term high reward. Lastly, the initial condition $Q(s_0, a_0)$ is required since RL is an iterative algorithm.

Although Q-learning and SARSA algorithms technically are pretty similar, they differ under some circumstances [7]. The difference between both algorithms from the technical point of view is the requirement to involve successor-state information. On the one hand, Q-learning acquires the best policy even when actions are performed based on more exploratory or even random policy. Q-learning uses the discounted value from the optimal action in the successor state $Q(s_{t+1}, b^*)$ [4], [14]:

$$b^* \leftarrow argmax_{b \in A(s_{t+1})} Q(s_{t+1}, b),$$

$$\Delta_{Qlearning} \leftarrow [r_{t+1} + \gamma Q(s_{t+1}, b^*) - Q(s_t, a_t)],$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \Delta_{Qlearning}.$$

On the other hand, SARSA's name originates from the quintuple $Q(s, a, r', s', a')$, where s, a are the original state and action, r' is the reward observed in the following state and s', a' are the new state-action pair. It uses the discounted value from the action selected according to the used policy in the successor state $Q(s_{t+1}, a_{t+1})$ [13]:

$$\Delta_{SARSA} \leftarrow [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)],$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \Delta_{SARSA}.$$

The major difference between SARSA and Q-learning is that the optimal reward for the next state is not necessarily utilized to update the Q-values. Instead, a new action, and therefore reward, is chosen using the same policy that determined the original action. In essence, SARSA respects the fact that future action selection may not be perfect and in the case of the existence of highly undesirable states, it converges to a safer behavior that attempts to circumvent these states. Q-learning in the same scenario would disregard the risk of a misstep and converge under the assumption that the agent will select its actions solely based on the Q-values of a state. A comprehensive example of the different behaviors of the two algorithms can be found in their application on the cliff-walking task [2].

III. Exploration Strategies in Reinforcement Learning

In this section, we briefly describe the different exploration strategies which are used in this work: ϵ-greedy, softmax, VDBE, and VDBE-Softmax.

A. ϵ-greedy

According to Sutton and Barto, the ϵ-greedy method is the most chosen approach to balance exploration and exploitation in RL [2]. The parameter ϵ controls the amount of exploration and defines the randomness of action selections [4]. An advantage of ϵ-greedy is that exploration specific data such as counters [15] or confidence bounds [16] are not required to be set. The agent chooses a random action with the probability $0 \leq \epsilon \leq 1$ and otherwise chooses greedily one of the optimal actions which have been learned in respect to the Q-function:

$$\pi(s) = \begin{cases} \text{random action from } A(s) & \text{if } \xi < \epsilon \\ \text{argmax}_{a \in A(s)} Q(s,a) & otherwise, \end{cases}$$

where ξ is a random number drawn at each time step from a uniform distribution between 0 and 1.

B. Softmax

The softmax approach is used to convert state-action values into action probabilities using a Boltzmann distribution [5]:

$$\pi(a|s) = Pr\left\{a_t = a | s_t = s\right\} = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_b e^{\frac{Q(s,b)}{\tau}}},$$

where the parameter τ, also called temperature, decides how much Q-values influence the action selection. Low temperatures lead to greedy action selection with regards to Q, whereas high temperatures cause all actions to have more similar chances of being chosen.

C. VDBE

Classic ϵ-greedy and softmax exploration assume similar exploration behavior for all states. Often, however, especially in episodic tasks, states are naturally being explored unequally, with some initial states reached far more often than others throughout episodes. To reduce unnecessary exploration once knowledge about these initial states has been sufficiently established, Tokic proposes the notion of a Value-Difference Based Exploration (VDBE) [6]. The key part of it is the introduction of a state-dependent exploration probability $\epsilon(s)$ as opposed to a global parameter ϵ. The exploration probability of a state s is updated every time an action a is taken in that state, where the nature of the change depends on the difference in a Boltzmann distribution between the old and the updated value of $Q(s,a)$. Larger differences in these Q-values lead to larger values of f and in turn to larger values of ϵ, i.e. more exploration. The concrete update steps for $\epsilon(s)$ are as follows:

$$f(s,a,\sigma) =$$

$$\left| \frac{e^{\frac{Q_t(s,a)}{\sigma}}}{e^{\frac{Q_t(s,a)}{\sigma}} + e^{\frac{Q_{t+1}(s,a)}{\sigma}}} - \frac{e^{\frac{Q_{t+1}(s,a)}{\sigma}}}{e^{\frac{Q_t(s,a)}{\sigma}} + e^{\frac{Q_{t+1}(s,a)}{\sigma}}} \right|$$

$$= \frac{1 - e^{\frac{-|Q_{t+1}(s,a) - Q_t(s,a)|}{\sigma}}}{1 + e^{\frac{-|Q_{t+1}(s,a) - Q_t(s,a)|}{\sigma}}},$$

$$\epsilon_{t+1}(s) = \delta * f(s,a,\sigma) + (1-\delta) * \epsilon_t(s).$$

The introduced parameters are σ, the so-called inverse sensitivity, and δ. The parameter σ gets its name from the property that low values cause the process to be very sensitive to changes in the Q-value, therefore, even small changes will make future exploration much more likely. High values of σ, however, mean that very large differences are needed to make exploration likely. The parameter δ, on the other hand, denotes the influence of a single action on the ϵ-value of a state. This has to be considered as the changes are made following an already executed action and none of the other possible actions or the certainty about their benefit have any influence on this particular update step. Tokic thus suggests choosing δ roughly as the multiplicative inverse of the number of actions in the state so that each action may have an equal contribution to the exploration likelihood update.

D. VDBE-Softmax

An additional adaptation to the previously introduced VDBE is to include the softmax behavior in the exploration strategy [17] resulting in the so-called VDBE-Softmax [7]. This means that just like for basic VDBE, a state dependent exploration probability is maintained and evaluated when to choose whether to explore or not.

$$\pi(s) = \begin{cases} \text{softmax action} & \text{if } \xi < \epsilon(s) \\ \text{argmax}_{a \in A(s)} Q(s,a) & otherwise. \end{cases}$$

In the former case, the executed action is not chosen by a uniform distribution like in VDBE or ϵ-greedy but by applying the softmax rule. This is meant to help in cases where some actions yield strongly negative rewards, which in combination with Q-value oscillations could mean an unnecessary amount of exploration of (clearly bad) actions.

IV. Implemented Robotic Scenario

To test the various exploration strategies, we implemented a robotic domestic scenario introduced in [18] which was originally constructed to test the influence of external interaction during the reinforcement learning task. Likewise, it can also be used to examine the influence of the previously described exploration approaches and the corresponding parameterization.

In the scenario, an autonomous robot is faced with the task of cleaning a table in front of it. There are three defined areas: *left* and *right* corresponding to either half of the table surface, as well as *home* corresponding to an additional storage position. Two interactable objects are part of the scenario, namely the *cup* and the *sponge*. Every state s_t is represented as a vector as follows:

$$s_t = < handObj, handPos, cupPos, sideCond[] >,$$

where *handObj* is the object which is currently in the robot's hand, *handPos* the position of the robot's arm, *cupPos* the position of the cup on the table, and *sideCondition[]* a 2-tuple with the current condition of every side of the table surface, that is, whether the table location has already been cleaned or not.

Initially, the *sponge* is in the *home* position and the cup is located on either the *left* or the *right* side of the table. Therefore, the initial state vector s_0 is described as:

$$s_0 = < free, home, left|right, [dirty, dirty] > .$$

The robot has seven different actions available to execute with its arm: *get*, *drop*, *go left*, *go right*, *go home*, *clean*, and *abort*. The action *get* picks up an item available at the current position of the arm, *drop* puts down the currently held object (if any) at the location that the arm is at, *go left*, *go right*, and *go home* move the robot's arm to the indicated position, *clean* makes the robot attempt to clean the current position with whatever object it happens to hold at that point of time, and *abort* cancels the current episode of learning and finishes the task returning to the initial state.

The goal for the robot is to end up with its arm in the *home* position not holding any objects and with both sides of the table cleaned. There are plenty of action sequences to achieve this result, with the shortest ones consisting of 15 actions. The final state s_f can be represented as:

$$s_f = < free, home, left|right, [clean, clean] > .$$

A number of actions can also lead to immediate failure of the episode when executed at the wrong moment, like attempting to *clean* a location that the *cup* is currently at or while it is being held. We name these states as failed-states. The reward function accordingly rewards 1 for the goal state, -1 for all failed-states and -0.01 for all other states (to discourage the robot from executing redundant actions), resulting in a maximum reward of 0.86 that the robot can possibly achieve. The reward function is summarized as follows:

$$r(s) = \begin{cases} 1 & \text{if } s \text{ is a goal state} \\ -1 & \text{if } s \text{ is a failed-state} \\ -0.01 & \text{otherwise.} \end{cases}$$

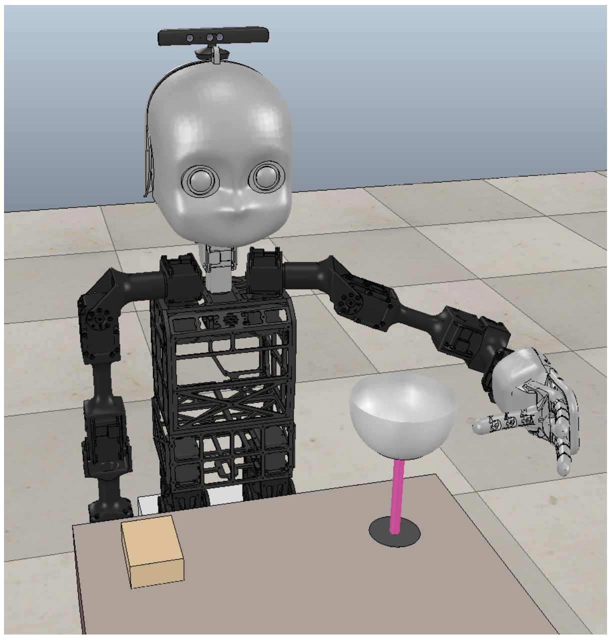


Fig. 2. Simulation of the table-cleaning scenario which comprises three locations, two objects, and seven actions.

Fig. 2 shows the implemented robotic scenario in a robot simulator.

V. Results and Discussion

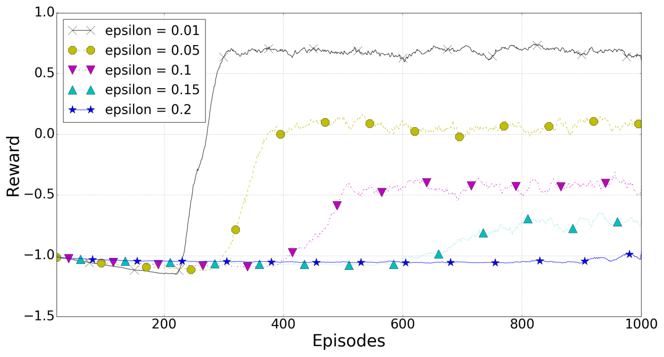
Using the described robotic scenario, we trained 20 independent agents each for 1000 episodes using different exploration strategies for both Q-learning and SARSA with differing parameterization according to each strategy. As general parameters, we chose the learning rate $\alpha = 0.8$ and the discount factor $\gamma = 0.9$. For the VDBE-based strategies, we chose $\delta = 0.1$ roughly as the inverse of the number of actions the robot is considering in each state.

We track the average reward that the agent obtains for each method and parameterization instance as can be seen in Figure 3. The curves are smoothed out using a convolution of 20 neighbors for clarity.

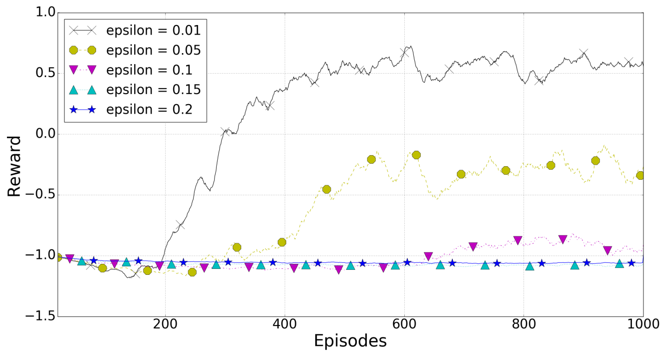
By using ϵ-greedy we perform the scenario using ϵ values as $\epsilon \in \{0.01, 0.05, 0.1, 0.15, 0.2\}$ for both Q-learning and SARSA. In both cases, the two best performances are obtained with $\epsilon = 0.1$ and $\epsilon = 0.05$ which suggests that low exploration values benefit the implemented scenario.

With softmax exploration, we use τ values as $\tau \in \{0.0001, 0.001, 0.01, 0.1, 1.0, 10.0\}$. Low values of the temperature parameters obtain better results, e.g., $\tau = 0.001$ and $\tau = 0.0001$.

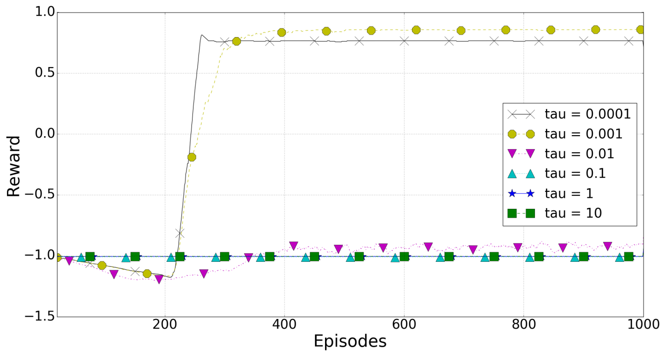
In the case of VDBE, we use $\tau = 0.001$ and σ values as $\sigma \in \{0.001, 0.01, 0.1, 1.0, 10.0, 100.0\}$. In case of Q-learning the best performance is obtained with $\sigma = 0.1$, $\sigma = 1.0$, $\sigma = 10.0$, and $\sigma = 100.0$, i.e., $\sigma > 0.01$. When SARSA is used the best performance is achieved for $\sigma = 1.0$, $\sigma = 10.0$, and $\sigma = 100.0$, i.e., $\sigma > 0.1$.



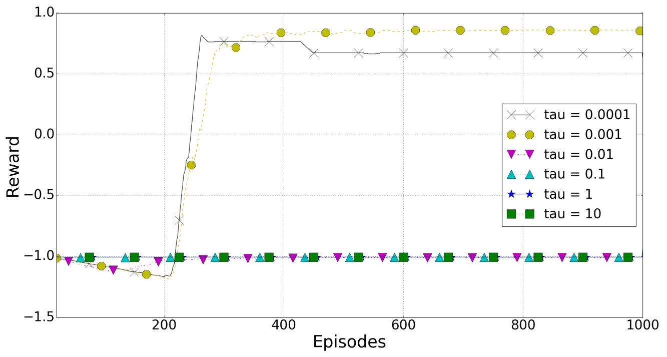
(a) Q-learning + ϵ-greedy.



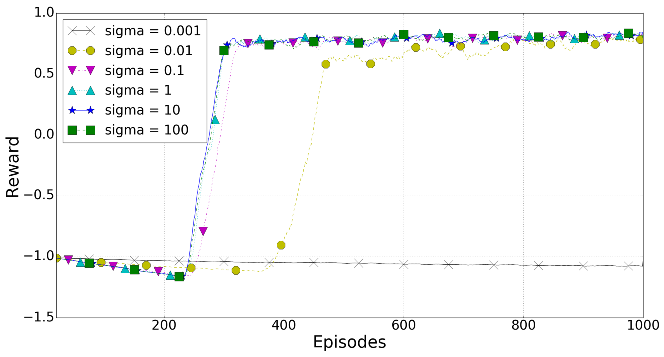
(b) SARSA + ϵ-greedy.



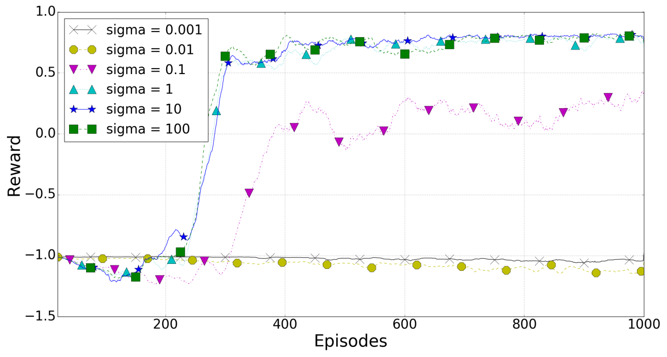
(c) Q-learning + softmax.



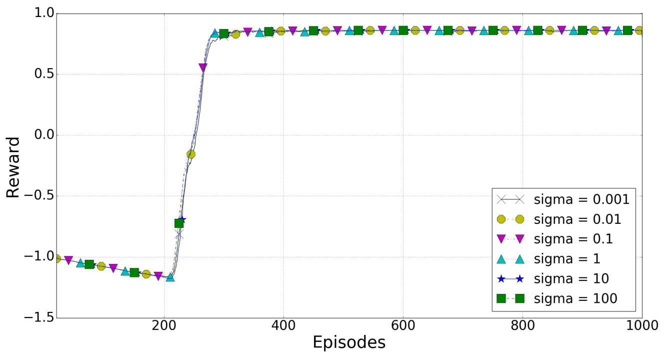
(d) SARSA + softmax.



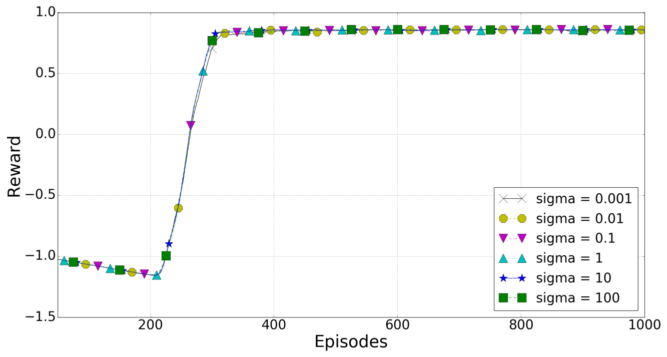
(e) Q-learning + VDBE.



(f) SARSA + VDBE.



(g) Q-learning + VDBE-Softmax



(h) SARSA + VDBE-Softmax

Fig. 3. Accumulated reward by the RL agent performing the table-cleaning task using ϵ-greedy, softmax, VDBE, and VDBE-Softmax as exploration strategies. Each method is performed with Q-learning and SARSA.

For VDBE-Softmax, we use $\tau = 0.001$ and σ values as $\sigma \in \{0.001, 0.01, 0.1, 1.0, 10.0, 100.0\}$. In this case, regardless of the temporal-difference learning, the agent obtains a good performance for all the values of σ used.

The obtained learning performance in our scenario seems to be better when using relatively greedy parametrization, e.g. low ϵ for the ϵ-greedy approach, low temperatures for softmax, and high inverse sensitivity values for VDBE and VDBE-Softmax respectively. The quality of the final solutions using the optimal parameters does not differ a whole lot except for ϵ-greedy which even for an extremely low ϵ does not manage to produce a better average cumulative reward than 0.6 and 0.7 for SARSA and Q-learning, respectively. Meanwhile, all other approaches on average end up almost or entirely at the best possible cumulative reward of 0.86.

In terms of convergence speed, VDBE-Softmax outperforms the other approaches by acquiring its optimal average reward after around 300 episodes for SARSA and 280 episodes for Q-learning. Only Q-learning with an ϵ-greedy strategy converges comparably faster but ends up at a significantly worse average reward. The other approaches reach their final solutions at around 350 episodes.

In the case of VDBE-Softmax, our implemented scenario is robust towards suboptimal parametrization as compared to simple VDBE. When choosing σ too low, especially in combination with SARSA, a cycle of exploration, oscillating Q-values and constantly large ϵ-values causes the VDBE approach to converge only very slowly or not at all. Furthermore, parametrization seems not to influence significantly VDBE-Softmax, where any choice within the range of our tests yielded almost the same, consistent results.

VI. Conclusion

We have shown VDBE-Softmax to be an exploration strategy to make learning very consistent and reliable as well as robust to parameter changes. This suggests that making exploration probabilities dependent on Q-value changes is indeed a valid approach for successful learning. Even though ϵ-greedy is the most used exploration/exploitation method due to its simplicity and which regularly works well, it is very slow and in many occasions does not enable the agent to achieve the optimal behavior. Furthermore, although softmax and VDBE improve the performance in comparison with the ϵ-greedy method in terms of accumulated reward, VDBE-Softmax outperforms all of them in the implemented scenario collecting greater reward and obtaining faster convergence.

The scenario we discussed in this paper simplifies to make comparisons easy and meaningful, however, there exists a plethora of other learning problems that are of different nature. Testing these methods on them may help as well to acquire a deeper understanding of effective exploration methods. Particularly, testing the method's performance on problems with larger state-spaces and more complex solutions may yield interesting insights and better comparisons between their expected quality.

Acknowledgment

The authors gratefully acknowledge partial support by CONICYT scholarship 5043, the German Research Foundation DFG under project CML (TRR 169), the European Union under project SECURE (No 642667), and the Hamburg Landesforschungsförderungsprojekt CROSS.

References

[1] Y. Niv, "Reinforcement learning in the brain," *Journal of Mathematical Psychology*, vol. 53, pp. 139–154, 2009.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: Bradford Book, 1998.

[3] J. D. Cohen, S. M. McClure, and A. J. Yu, "Should I stay or should I go? how the human brain manages the trade-off between exploitation and exploration," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 362, pp. 933–942, 2007.

[4] C. Watkins, "Learning from Delayed Rewards," Ph.D. dissertation, University of Cambridge, England, 1989.

[5] J. S. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimates of parameters," in *Proceedings of the 1989 Conference on Advances in Neural Information Processing Systems NIPS*. Morgan Kaufmann, San Mateo, CA, 1990, pp. 211–217.

[6] M. Tokic, "Adaptive ϵ-greedy exploration in reinforcement learning based on value differences," in *Proceedings of Annual Conference on Artificial Intelligence*. Heidelberg, Germany: Springer, 2010, pp. 203–210.

[7] M. Tokic and G. Palm, "Value-difference based exploration: Adaptive control between ϵ-greedy and softmax," in *Proceedings of 34th Annual German conference on Advances in artificial intelligence*. Heidelberg, Germany: Springer, 2011, pp. 335–346.

[8] C. Szepesvári, *Algorithms for Reinforcement Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool, 2010.

[9] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

[10] V. Rieser and O. Lemon, *Reinforcement Learning for Adaptive Dialogue Systems*. Heidelberg, Germany: Springer, 2011.

[11] S. Marsland, *Machine Learning: An Algorithmic Perspective*. CRC press, 2015.

[12] H. V. Hasselt and M. Wiering, "Reinforcement learning in continuous action spaces," in *Proceedings of the IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning ADPRL*. IEEE, 2007, pp. 272–279.

[13] G. A. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," *Technical Report CUED/F-INFENG/TR166*, 1994.

[14] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[15] S. B. Thrun, "Efficient exploration in reinforcement learning," *Technical Report EER/865072*, 1992.

[16] P. Auer, "Using confidence bounds for exploitation-exploration trade-offs," *Journal of Machine Learning Research*, vol. 3, pp. 397–422, 2003.

[17] M. Wiering, "Explorations in Efficient Reinforcement Learning," Ph.D. dissertation, University of Amsterdam, The Netherlands, 1999.

[18] F. Cruz, S. Magg, C. Weber, and S. Wermter, "Training agents with interactive reinforcement learning and contextual affordances," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 8, pp. 271–284, 2016.