Universität Hamburg
Department Informatik
Knowledge Technology, WTM

Exploration/Exploitation method studies in Reinforcement Learning

Seminar Paper

Neural Networks

Alvin Fazrie, Peter Wüppen

Matr.Nr. 6641834, 6053088

4fazrie@informatik.uni-hamburg.de, 5wueppen@informatik.uni-hamburg.de

09.06.2016

Abstract

This paper presents and compares multiple approaches to the exploration/exploitation dilemma in reinforcement learning. It describes an exemplary reinforcement learning task and shows the performance of different exploration policies on it.

Contents

1 Introduction

1.1 Motivation

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. This concept can also be applied in the field of Artificial Intelligence, where it is called Reinforcement Learning (RL). RL is a learning mechanism where an agent autonomously executes actions and receives a reward from its environment for each of them. 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 results. One crucial dilemma is the balance between exploration and exploitation, such as when we face a certain problem in our daily activities and we need to decide whether to utilize an existing method to solve the problem or to look for a different, possibly better one. Similarly, this also becomes a challenging task in Reinforcement Learning to enhance the RL performance by balancing the proportion between exploration and exploitations. If one of the approaches dominates the learning process too much, it could lead to adverse effects on learning performance [9]. 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 hinder an agent from maximizing long-term reward because the chosen actions may stay suboptimal.

The most common approach to harmonize the ratio of exploration and exploitation is by implementing the ϵ-greedy method [6]. This method often leads to very successful outcomes, but one of the issues with ϵ-greedy is that its setting is a global one and may not accommodate to state-specific requirements. Besides ϵ-greedy, many other successful methods were introduced, including ones by Tokic and Palm ([8], [9]). They did an experiment by implementing "Value-Difference Based Exploration combined with Softmax action selection" in an attempt to mitigate weaknesses of traditional approaches. In this paper, we will describe and test their methods on another scenario.

1.2 Reinforcement Learning Basics

Reinforcement Learning represents a framework where an agent is able to learn a task by interacting with the environment [6]. A graphical representations of the basic concept of Reinforcement learning can be seen in Figure 1.

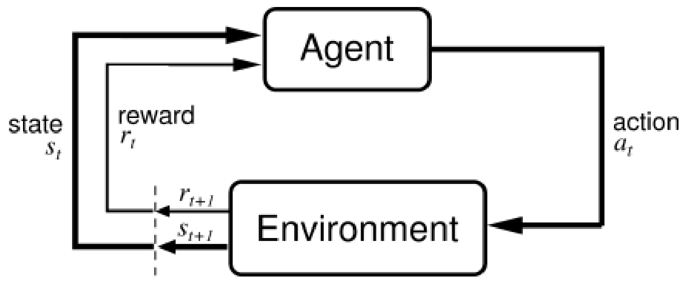


Figure 1: Reinforcement learning diagram showing an agent's interaction with its environment [6].

From the reinforcement learning diagram above, there are several basic steps of reinforcement learning model which could be emphasized. 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) 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 [6].

A policy π is used to map a state to an action through the process ($\pi_t(s_t, a_t)$). The learning process changes the policy in order to obtain experience from the given environment and maximize the accumulated reward. It is pretty obvious that the main aim of this Reinforcement Learning model is to achieve a optimal behaviour by performing the best action for each state to get the maximum observed rewards for the agent.

The core of the Reinforcement Learning is formed by a Markov Decision Process(MDP). MDP characterizes several core components for the Reinforcement Learning which models a problem with $\{s, a, r, \delta, \pi, V^{\pi}\}$ parameters [2]. A State space, denoted by $s \in S$, is a discrete set of environment states; an action space, denoted by $a \in A$, 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. Another components are Policy and State value function where policy, denoted by π, is a function which specifies the agents behavior. It maps the action to be taken for each given state. i.e., $\pi_t : S \to A$, where S is the environment state set and A is the agent action and 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 an agent can expect to accumulate. For Q learning and SARSA, instead using a state-value function they use action-value function (Q-function).

In Reinforcement Learning, the agent is expected to acquire the maximal total reward from the action taken in each step as a main objective. The estimation of total reward that an agent could attain could be figured out with the equation below [6]:

$$Q^{\pi}(s,a) = E_{\pi}\{\sum_{k=0}^{\infty}\gamma^{k}r_{t}+k+1|s_{t}=s,a_{t}=a\} \tag{1}$$

From the equation, it can be seen that $Q(s,a)$ is the state-action value, r is the reward for the following policy π in the state s to select action a . Moreover, γ is the discount rate of future actions for which holds $(0 < \gamma \leq 1)$ for periodic learning and $(0 < \gamma < 1)$ for continuous learning tasks [9].

1.3 Q-learning/SARSA

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*, SARSA for on-policy control and Q-learning for off-policy control.

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 override the previous information. When it is 0, the agent does not learn anything, but if the factor is 1, the agent will only consider the newest 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 for a long-term high reward. Lastly, the initial condition $Q(s_0,a_0)$ is required since we are dealing with iterative algorithms.

Even though Q-learning and SARSA algorithms technically are pretty similar, they differ under several circumstances [9]. The difference between both algorithms in the technical point of view is the requirement to involve successor-state information.

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 $Q(s_{t+1},b^{*})$ [10]:

$$\begin{aligned} b^{*} &\leftarrow argmax_{b\in A(s_{t+1})}Q(s_{t+1},b) \\ \Delta_{Qlearning} &\leftarrow [r_{t+1}+\gamma Q(s_{t+1},b^{*})-Q(s_t,a_t)] \\ Q(s_t,a_t) &\leftarrow Q(s_t,a_t)+\alpha\Delta_{Qlearning} \end{aligned} \tag{2}$$

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 $Q(s_{t+1},a_{t+1})$ [5]:

$$\begin{aligned} \Delta_{SARSA} &\leftarrow [r_{t+1}+\gamma Q(s_{t+1},a_{t+1})-Q(s_t,a_t)] \\ Q(s_t,a_t) &\leftarrow Q(s_t,a_t)+\alpha\Delta_{SARSA} \end{aligned} \tag{3}$$

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 case of the existence of highly undesirable states, 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 to the most optimal behavior 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 behavior of the two algorithms can be found in their application on the cliff-walking task [6].

2 Exploration strategies in RL

2.1 ϵ-greedy

According to Sutton and Barto, the ϵ-greedy method is the most chosen approach to balance exploration and exploitation in RL [6]. The parameter ϵ controls the amount of exploration/exploitation and defines the randomness of action selections. An advantage of ϵ-greedy is that exploration specific data such as counters [7] or confidence bounds [1] is not required to be set.

In ϵ-greedy, the agent chooses a random action with the probability $0 \leq \epsilon \leq 1$ and otherwise chooses greedily one of the optimal actions which has been learned in respect to the Q-function:

$$\pi(s) = \begin{cases} \text{random action from } A(s) & \text{if } \xi < \epsilon \\ \text{argmax}_{a \in A(s)} Q(s,a) & \textit{otherwise}, \end{cases} \tag{4}$$

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

2.2 Softmax

The Softmax approach is used to convert state-action values into action probabilities using a Boltzmann distribution:

$$\pi(a|s) = Pr\left\{a_t = a|s_t = s\right\} = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_b e^{\frac{Q(s,b)}{\tau}}} \tag{5}$$

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.

2.3 VDBE

Classic ϵ-greedy exploration assumes the same exploration probability for all states and in all situations. 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 and Palm propose the notion of a "Value-Difference Based Exploration" (VDBE) [8]. 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 $Q(s,a)$. The concrete update steps for $\epsilon(s)$ are as follows:

$$\begin{aligned} f(s,a,\sigma) &= \left| \frac{e^{\frac{Q_t(s,a)}{\sigma}}}{e^{\frac{Q_t(s,a)}{\sigma}} + e^{\frac{Q_{t+1}(s,a)}{\sigma}}} - \frac{e^{\frac{Q_{t+1}(s,a)}{\sigma}}}{e^{\frac{Q_t(s,a)}{\sigma}} + e^{\frac{Q_{t+1}(s,a)}{\sigma}}} \right| \\ &= \frac{1 - e^{\frac{-|Q_{t+1}(s,a)-Q_t(s,a)|}{\sigma}}}{1 + e^{\frac{-|Q_{t+1}(s,a)-Q_t(s,a)|}{\sigma}}} \\ \epsilon_{t+1}(s) &= \delta * f(s,a,\sigma) + (1-\delta) * \epsilon_t(s) \end{aligned} \tag{6}$$

The introduced parameters are σ, the so-called inverse senstivity, and δ. σ gets its name from the property that low values cause the process to be very sensitive to changes in the Q-value: 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. 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 and Palm thus suggest choosing δ roughly as the multiplicative inverse of the amount of actions in the state so that each action may have an equal contribution to the exploration likelihood update.

2.4 VDBE-Softmax

As an additional adaptation to the previously introduced VDBE, Tokic and Palm also suggest to include the Softmax behaviour in the exploration strategy, resulting in what they call VDBE-Softmax [9]. This means that just like for basic VDBE, a state dependant exploration probability is maintained and evaluated when to choose whether to explore or not. In the former case though, the executed action is not chosen by a uniform distribution like in VDBE or ϵ-greedy but by applying the Softmax-rule as presented in equation (5). According to the authors, 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.

3 Implementation

3.1 Description of implemented scenario

To test the various exploration strategies ourselves, we used a scenario introduced by Cruz et al. in [3] 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 parametrization.

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*, an external storage position. Two interactable objects are part of the scenario, namely the *cup* and the *sponge*. Initially, the *sponge* is in the *home* position and the cup is located on the *left* side of the table. The robot now has four different actions available to it to execute with its arm: It can *get*, *drop*, *go* and *clean*. *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* moves the robot's arm to another position, and *clean* makes the robot attempt to clean the current position with whatever object it happens to hold at that point in time. 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 is plenty of action sequences to achieve this result, with the shortest ones consisting of 15 actions. A number of actions can also lead to immediate failure of the episode when executed in the wrong moment, like attempting to *clean* a location that the *cup* is currently at or while it is being held. The reward function accordingly rewards 1 for the goal state, -1 for all failure 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.

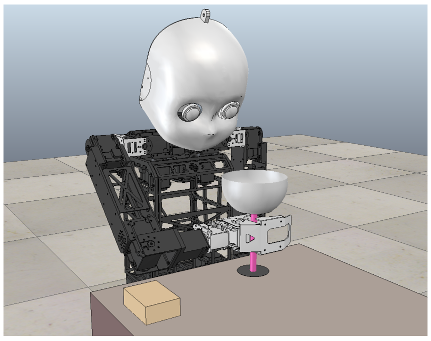


Figure 2: Simulation of the table-cleaning scenario ([4])

Using this scenario, we trained 20 independent agents each for 1000 episodes using different exploration strategies for both Q-Learning and SARSA with differing parametrization. Based on experimental results, we chose the learning rate $\alpha = 0.8$ and the discount factor $\gamma = 0.9$. Also, based on recommendation from

[8], we chose $\delta = 0.1$ for the VDBE-based strategies roughly as the inverse of the number of actions the robot is considering in each state. We tracked the average reward that the agents obtained for each method and parametrization instance as can be seen in Figure 3. The curves are smoothed out using a convolution of 20 neighbours for clarity.

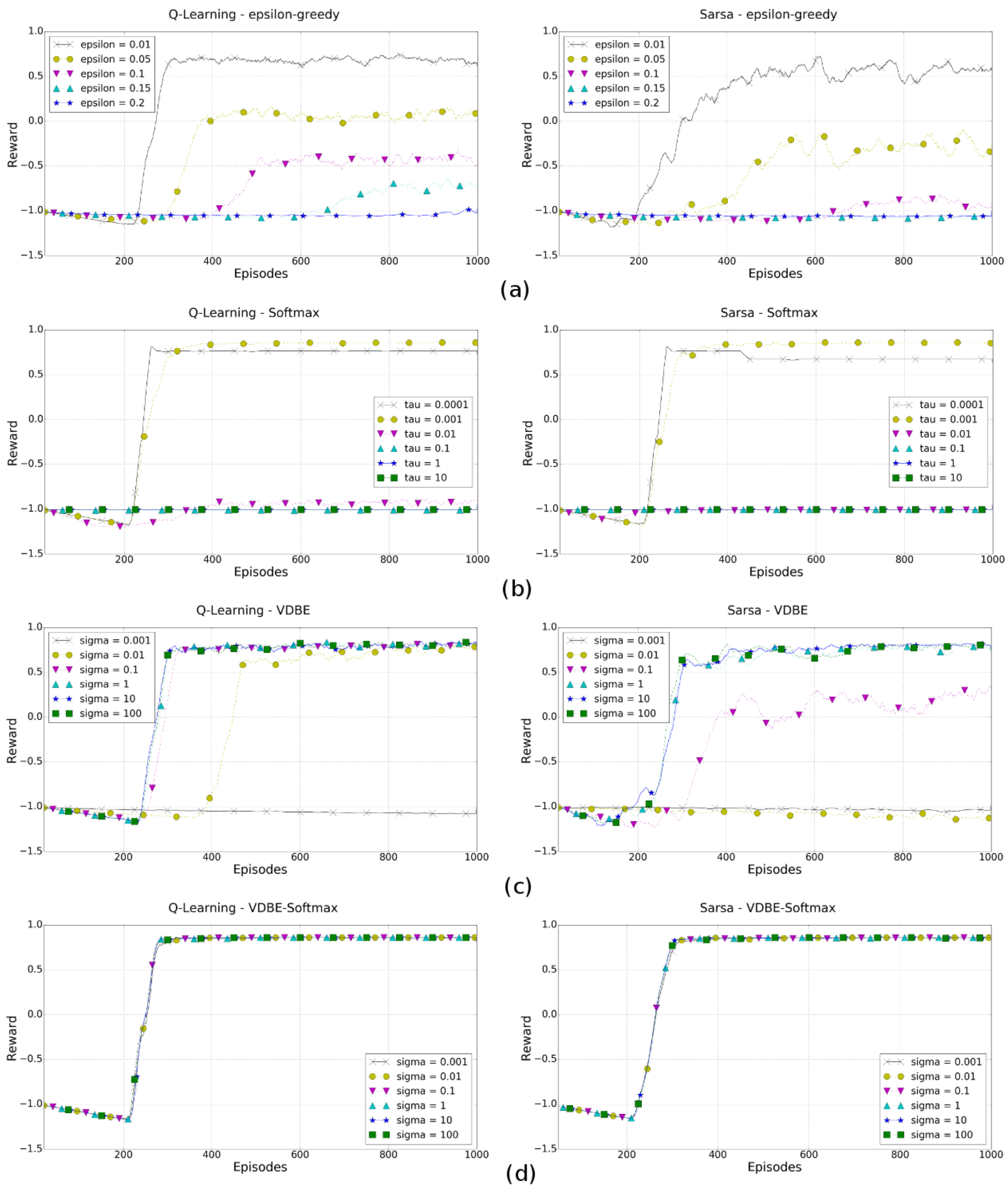
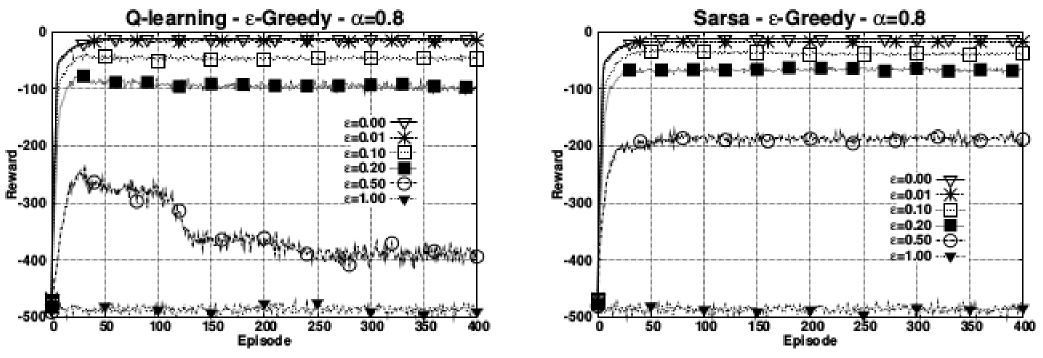
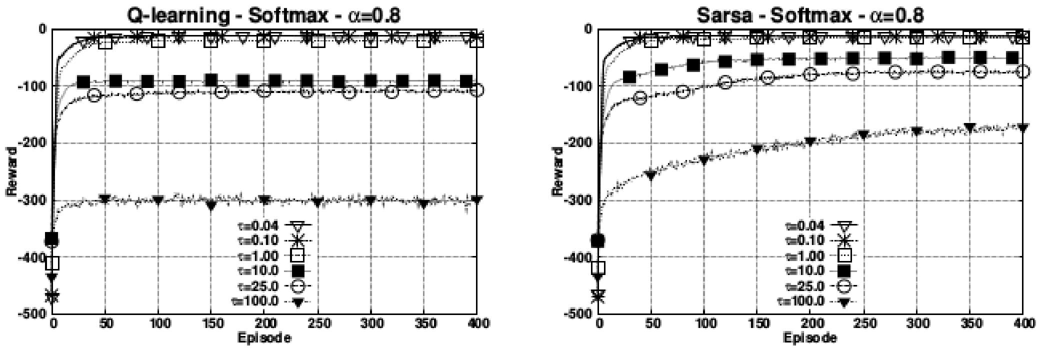


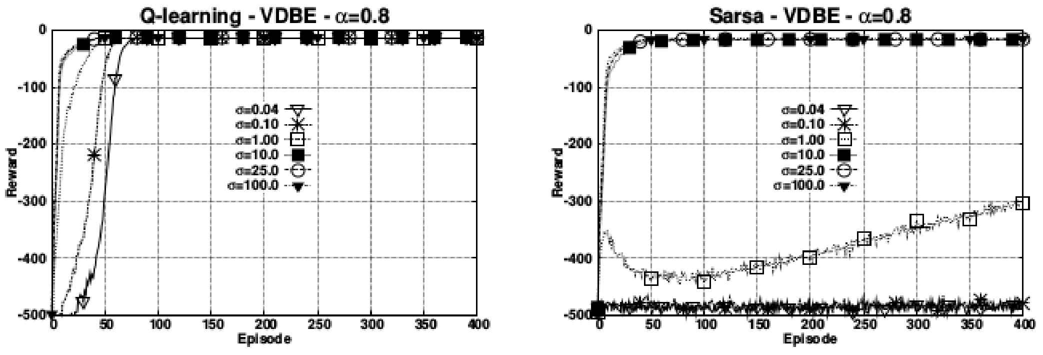
Figure 3: Learning results of the table-cleaning scenario for (a) ϵ-greedy, (b) Softmax, (c) VDBE, (d) VDBE-Softmax



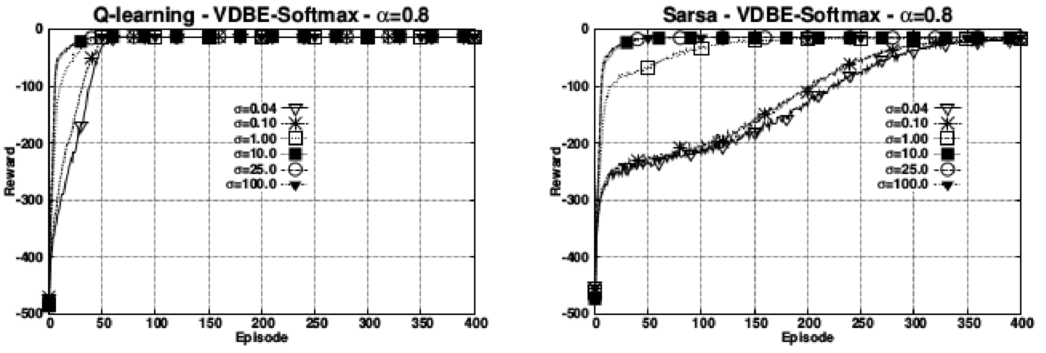
(a)



(b)



(c)



(d)

Figure 4: Learning results of the cliff-walking scenario for (a) ϵ-greedy, (b) Softmax, (c) VDBE, (d) VDBE-Softmax ([9])

3.2 Results and Discussion

In this section, we want to present our results when testing the different exploration options and set them into perspective with former results by Tokic and Palm [9].

Tokic and Palm tested their algorithm using the cliff-walking task by Sutton and Barto ([6]), a simple environment originally intended to display the different effects of on-policy (SARSA) and off-policy learning (Q-Learning), as well as on what is essentially an extension of the multi-armed bandit problem. We will focus on their results of the former scenario as it is much closer in nature to the one we tested the algorithms on. Similar as in our table-cleaning task, the cliff-walking task features certain state-action pairs that are, although not immediately ending an episode, still rewarded extremely negatively. Additionally and unlike in the multi-armed bandit task, all state transitions as well as rewards are deterministic in both scenarios.

Similar to what Tokic and Palm experienced, learning performance in our scenario seems to be best across the board 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.

Another phenomenon that is similar for both scenarios is the higher robustness towards suboptimal parametrization of VDBE-Softmax 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. On the other hand, parametrization seemed to almost not matter for VDBE-Softmax, where any choice within the range of our tests yielded almost the same, consistent results.

4 Conclusion

Our findings overall support the results obtained and conclusions made by Tokic and Palm [9]. VDBE-Softmax as an exploration strategy seems in fact to make learning very consistent and reliable as well as robust to parameter changes. This suggests that making exploration probabilities dependant on Q-value changes is indeed a valid approach for successful learning. The scenarios we discussed in this paper were of similar nature to make comparison easy and meaningful, however there exists a plethora of other learning problems that are of different nature and that testing these methods on them as well may help 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. Tokic and Palm already addressed a stochastic learning task in form of an alteration of the multi-armed bandit task with somewhat similar results, but more complex examples remain yet untouched.

References

[1] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3:397–422, March 2003.

[2] T. Baier-Lowenstein and J. Zhang. Learning to grasp everyday objects using reinforcement-learning with automatic value cut-off. In *Intelligent Robots and Systems*, page 15511556. IEEE, 2007.

[3] F. Cruz, S. Magg, C. Weber, and S. Wermter. Improving reinforcement learning with interactive feedback and affordances. In *4th International Conference on Development and Learning and on Epigenetic Robotics*, pages 165–170, Oct 2014.

[4] F. Cruz, G. I. Parisi, and S. Wermter. Learning contextual affordances with an associative neural architecture. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 665–670, 2016.

[5] G. A. Rummery and M. Niranjan. On-line q-learning using connectionist systems. Technical report, 1994.

[6] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* MIT press, 1998.

[7] S. B. Thrun. Efficient exploration in reinforcement learning. Technical report, Pittsburgh, PA, USA, 1992.

[8] M. Tokic. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In R. Dillmann, J. Beyerer, U. Hanebeck, and T. Schultz, editors, *KI 2010: Advances in Artificial Intelligence*, volume 6359 of *Lecture Notes in Artificial Intelligence*, pages 203–210. Springer Berlin / Heidelberg, 2010.

[9] M. Tokic and G. Palm. Value-difference based exploration: Adaptive control between epsilon-greedy and softmax. In J. Bach and S. Edelkamp, editors, *KI 2011: Advances in Artificial Intelligence*, volume 7006 of *Lecture Notes in Artificial Intelligence*, pages 335–346. Springer Berlin / Heidelberg, 2011.

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