REPRODUCING: DORA THE EXPLORER: DIRECTED OUTREACHING REINFORCEMENT ACTION-SELECTION

Jiaxuan Wang, Tianyang Pan, Drew Davis
Department of Electrical and Computer Engineering
University of Michigan
Ann Arbor, MI 48109, USA
{typan, drewdavi, jiaxuan}@umich.edu

ABSTRACT

The Directed Outreaching Reinforcement Action-Selection (DORA) (Anonymous (2017)) paper had five primary experiments and we replicated all five. Additionally, we replicated an experiment located in the appendix of the paper to further investigate how E-values compared to optimistic algorithms. For each of these experiments, the authors did not provide code or additional resources beyond the originally submitted paper. Beyond the experiments found in the original submission, we performed two additional experiments under function approximation settings: one in the bridge environment and one in the cart pole environment. We found that tabular environments are reproducible while the function approximation task is brittle. We perform additional experiments and identified misreported hyperparameter as the primary cause of failure in continuous state space.

1 INTRODUCTION

The Directed Outreaching Reinforcement Action-Selection (DORA) paper introduces the concept of E-values. The paper claims that E-values enable a more effective exploration vs exploitation balance than previous methods by considering the uncertainty of all future paths. E-values are modeled as a second MDP, which represents uncertainty and runs parallel to the original MDP. E-values are initialized to 1 for each state and their values decrease every time they are visited. This optimistic bias results in directed exploration (Brafman & Tennenholtz (2002); Sutton & Barto (1998)).

E-values act as generalized visit counters; therefore, they can be plugged into any counter based approach. Not only are they easy to implement in a tabular setting, they extend naturally to a function approximation setup by learning a separate value function for uncertainty alongside the regular MDP. This idea is formalized by considering two MDPs: $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ and $M_E = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathbf{0}, \gamma_E)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition probability from state action to next state, $\mathcal{R}$ is a reward distribution conditioned on current state and action, and γ is the discount factor. Since the reward is always 0 in M_E, the true value function, V_E , is 0 for all states. We denote $E(s, a)$ as an estimate of $\mathbb{E}_{s'|s,a}(V_E(s'))$ and Q^* as the optimal action value for M. The DORA paper provides convergence proof in the tabular setting and is achieved with the LLL determination rule.

2 EXPERIMENTS

The DORA paper reports results on six experiments. We replicate them all, with our results shown on the left and their plots shown on the right in Figure 1. For all the details on code implementation of the experiments and environments, refer to Section 4.7. We also report further experimental results in the appendix.

2.1 TABULAR ENVIRONMENT

The first set of experiments are done with the bridge environment (Section 4.1). All results with randomness are averaged over 50 trials. The first experiment (Figure 1(a) and (b)) uses an ϵ-greedy agent with an exploration bonus of $\frac{1}{log_{1-\alpha}E}$. We vary γ_E from 0 to 0.9. The length of the bridge envirnoment is set to 5. Our results show a similar trend to the original paper. The agents with larger γ_E converge faster. The difference is our agent with no bonus converges faster than the one reported in the paper, which could be an artifact of their unspecified parameters.

The second experiment (Figure 1(c) and (d)) compares DORA to common action selection baselines, on a bridge with length 15. As γ_E increases, the LLL algorithm converges faster. Notice that the difference between our replication and the original work is that the agents converge in different episodes. For example, our softmax agent converges in about 2000 episodes, while the same agent converges in about 1700 episodes in their work.

The third and the fourth experiment together test E-values as a measure of uncertainty. Figure 1(e) (left: standard counter; right: generalized counter) shows that uncertainty (as measured by relative difference with Q^*) decreases with the number of visits, on the bridge environment with length 5. With regular counters, for different state-action pairs, the convergence levels vary significantly. In contrast, the generalized counters perform much more uniformly for the chosen pairs. Notice that the original figure is incomplete because the measurement doesn't reach zero at the end. Our plot shows that the measurement converges to zero when the value of the counter is larger than 30.

In addition to these experiments on the bridge environment, we also compared the LLL algorithm with delayed Q-learning (Strehl et al. (2006)), which initializes the values optimistically. Please refer to Section 4.6 for a detailed discussion.

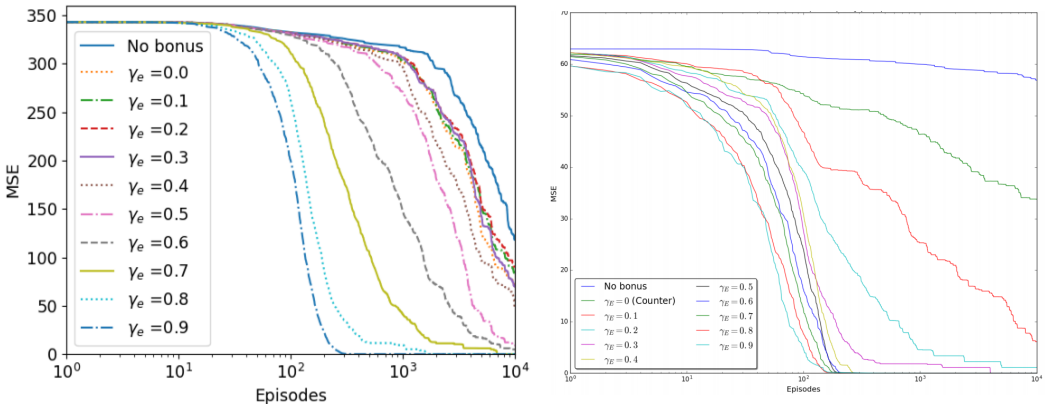
2.2 FUNCTION APPROXIMATION

Beyond the tabular setting, we use DQN-based system (Mnih et al. (2015)) with an exploration bonus computed by the E network to solve the mountain car environment. The E network differs from the Q network in two ways. First, the E network has the final layer weights initialized to 0 and uses a sigmoid activation function. Second, the E network is learned using the SARSA algorithm while the Q network is learned using deep Q learning (implementation detail in Section 4.7). Figure 1(g) and (f) depict the difference between our replication and the paper's result averaged over 10 runs (values are smoothed by a window of 100 episodes). While using the same set up as reported in the paper, we are not able to reproduce its result. Since the DORA paper ignores the LLL ϵ-greedy settings in function approximation, we added them in (red and blue lines in Figure 1(g)). Figure 1(g) shows that none of the DORA methods worked as claimed. We then applied the exact same algorithm to the Cart Pole environment and succeeded (Section 4.2). We further verified that our implementation of DQN is working in the Mountain Car environment by changing the hyper parameters, which provided a much better result (Figure 6). We conclude that the hyper parameters given in this experiment are inaccurate. For details of our investigation, please refer to the Appendix.

3 CONCLUSION

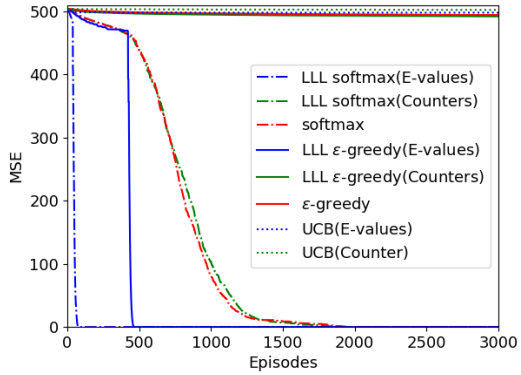
Our attempts to replicate the results shown in the DORA paper were met with varying levels of success. We were able to recreate the experiments presented in the tabular setting and obtained similar results to the ones presented in the paper. This success further validates the tabular setting claims made in the DORA paper and shows promise that E-values can successfully be used as generalized counters.

In contrast, our attempt to reproduce results in the mountain car environment failed. As discussed in Section 2.2, the most likely cause of the discrepancy in results is misreported hyperparameters. In addition, the DQN baseline in DORA is not a fair comparison as we found another set of hyperparameters that solves the mountain car environment much faster than the reported (Section 4.4). This discrepancy does not disprove the potential of E-values in function approximation settings. In fact, our additional experiments in the cart pole environment (Section 4.2) and the bridge environment (Section 4.3) showed promise. However, it does raise questions about the claims and reproducibility of the DORA paper.

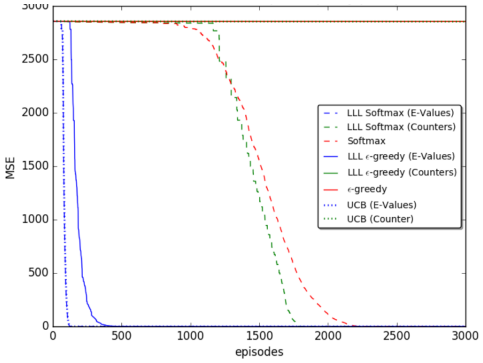


(a) ours

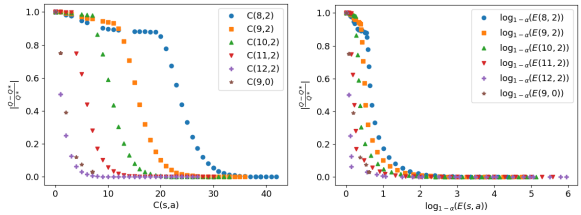
(b) original



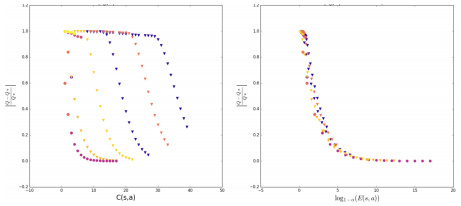
(c) ours



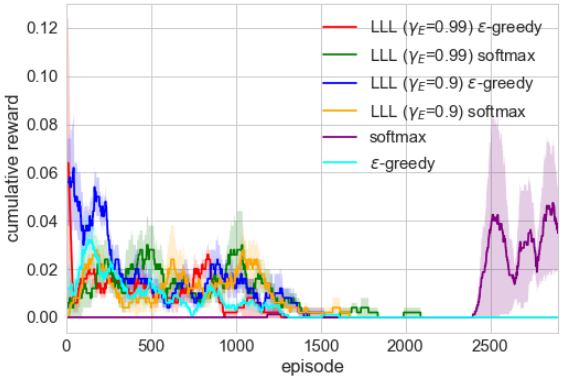
(d) original



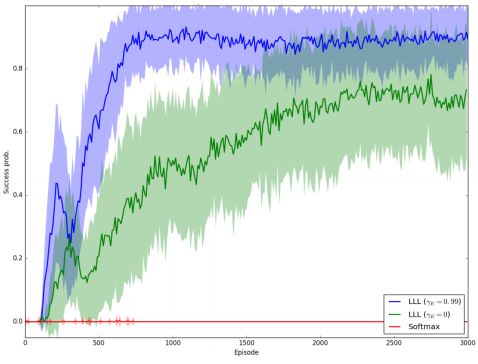
(e) ours



(f) original



(g) ours



(h) original

Figure 1: Note: original plots are difficult to read due to low resolution of DORA paper. **(a)** Convergence to true value under ϵ-greedy for the Bridge environment (k=5) with different γ_E **(c)** Comparison of baselines on the Bridge environment (k=15); E value converges the fastest. **(e)** For a given state action pair (s, a), generalized counter $\log_{1-\alpha}(E(s, a))$ better measure convergence to the true value than counter $C(s, a)$ **(g)** We challenge the paper's claim that DORA outperforms DQN in mountain car under their chosen hyper parameters

REFERENCES

Anonymous. Dora the explorer: Directed outreaching reinforcement action-selection. *ICLR submission*, 2017.

Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Alexander L Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L Littman. Pac model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 881–888. ACM, 2006.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

4 APPENDIX

4.1 THE BRIDGE ENVIRONMENT

In the bridge environment, the agent starts at a state (the blue point in Figure 2) next to a terminal state with reward 1. The goal of the agent is to cross a length k bridge to reach the terminal state with reward 10. During the trip, any up or down movement would cause the agent to jump off the bridge, incurring a reward of -100.

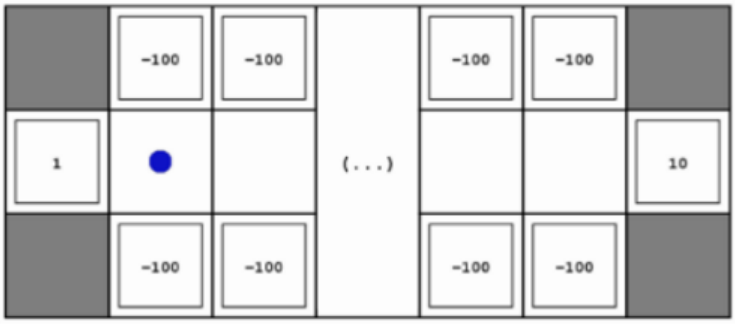


Figure 2: Bridge environment. Start state is the blue point. (This figure is from the DORA paper.)

4.2 CARTPOLE

One of the strengths of E-values is that they can be extended to function approximation settings. This is key for utilizing E-values on many of the current challenges in reinforcement learning. Therefore, to further test the potential of E-values in function approximation settings we set up an experiment in the cart pole environment. In this environment, the agent attempts to balance a pole on a cart that can be moved left or right. A reward of 1 is earned for every time step that the pole remains balanced. Once the balance is lost or the car has moved too far from its origin, the episode ends and the environment is reset. The score is capped at 200: after reaching a total score of 200, the environment is reset and the next episode begins. The agent learns directly from the visual display of the environment. The performance of E-value systems was compared against DQN systems. The hyper parameters used are shown in Section 4.5. The performance of LLL softmax and LLL ϵ-greedy with γ_E=0.900 were also evaluated in the cart pole environment; however, to improve the readability of Figure 3 these results are hidden. The plots shown are a running average of episode score over 10 episodes.

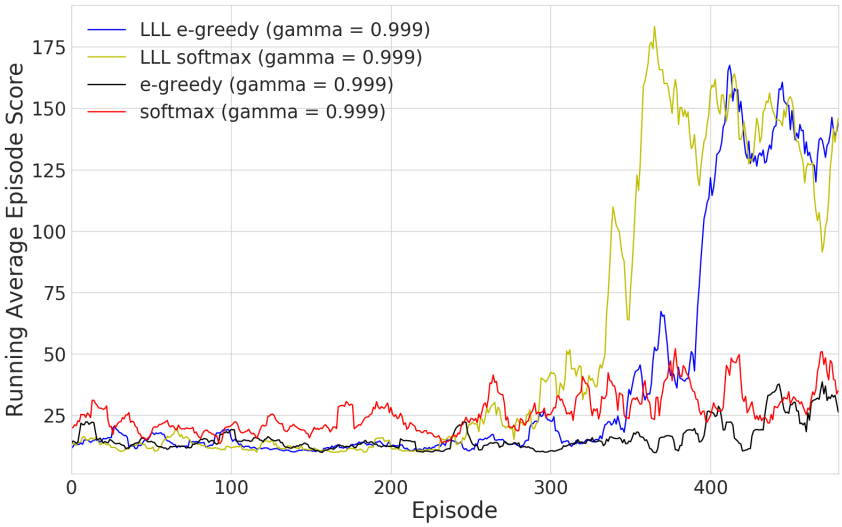


Figure 3: Running average of episode score for E-value and DQN systems on cart pole environment (computed by averaging over 10 subsequent episodes).

The results shown in Figure 3 show promise for E-value based systems. Both E-value based systems were able to improve their performance more quickly than the baseline DQN systems. This shows that there is potential for E-values to have a positive impact on learning in function approximation. However, due to computational constraints, an extensive hyperparameter search was not able to be performed and it is not clear exactly how much of a benefit E-values can offer over other systems.

4.3 DQN ON BRIDGE

While it is easy to see that E-values represent uncertainty in the tabular setting, it is much harder to visualize its utility in function approximation. Part of the reason is that E-values are no longer guaranteed to monotonically decrease with the number of visits when approximated by a neural network. In this experiment, we investigate how robust E-values are using function approximation on the bridge environment. We use a tabular environment so that we can count the number of visits to a state action pair and then compare to function approximator's result. Figure 4 (left) shows a strong positive trend between time (how many times the state action pair is visited) and the generalized counter value for 6 state action pairs (selected by filtering pairs that are eventually visited for more than 10 times). Though they appear noisy in the beginning, these 6 lines overlap together and monotonically increases after 50 visits. This is a surprising, but positive, result because function approximation has no guarantee of this monotonicity. Even more striking is the fact that the generalized counters converge at a very similar rate, as shown in Figure 4 (right), which closely resembles the result shown in Figure 1(e). Given more time, we would like to examine this behavior more carefully. The model and its parameters are listed in Section 4.5.3

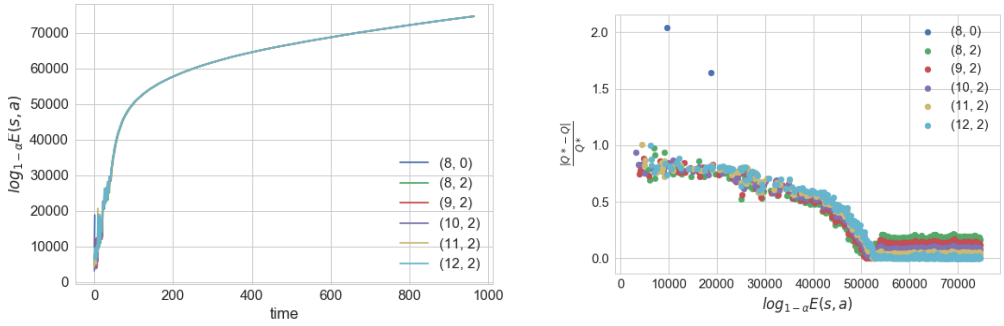


Figure 4: **(left)** Positive trend between time visited and E values with function approximation **(right)** E values as generalized counters, using function approximation on the bridge environment, converge uniformly for different state action pairs.

4.4 Mountain Car with better hyperparameters

In replicating DORA's result, we found that its function approximation experiment was not reproducible (Figure 1(g)). To identify the cause of this issue, we confirmed that our pytorch DQN implementation was correct by comparing our result on mountain car to OpenAI's implementation `https://github.com/openai/baselines`. Figure 6 shows that our DQN reached high reward in less than 500 episodes, which is comparable to OpenAI's result with the same parameters. In this setting, we obtained the exact opposite result shown in the DORA paper (ϵ-greedy DQN significantly outperforms DORA). The model and the hyperparameters used are reported in Section 4.5.1. Based on this result, we are confident that a) the DORA paper did not carefully choose their hyperparameters and b) its reported hyperparameters may well be incorrect.

4.5 Hyperparameters

We report the hyperparameters used in each experiment here.

4.5.1 Mountain Car

We trained two models on the mountain car environment, one for the reproduction of DORA, the other for the reproduction of OpenAI baseline. The model used for DORA is a neural network with 2 hidden layers with 64 and 128 neurons respectively. Between each layer, there is a $tanh$ activation.

For setup in reproducing the OpenAI result, we used only one hidden layer of size 64 and the ReLU activation function. The hyperparameters used are shown in Table 1 and Table 2 respectively.

4.5.2 Cart Pole

Since we used a variant of the cart pole environment based on RGB image input (see Section 4.7 to get the environment from GitHub), we use CNN here. The architecture consists of 3 convolutional layers each with kernel size of 5 and stride of 2. Between each convolutional layers, we have batch normalization and ReLU activation to speed up training and adding non-linearity. While the first convolutional layer has 16 feature maps, the other two have 32 feature maps. The convolutional layers are followed by a fully connected layer that maps input size 448 to 2 action value output. The hyperparameters used are reported in Table 3.

4.5.3 Bridge

We use a fully connected neural network with 2 hidden layers in this experiment. The first hidden layer has size 64, while the second has size 128. Each of the layers is followed by ReLU activation function. The hyperparameters used are reported in Figure 4.

Table 1: Mountain Car reported hyperparameters in DORA

Hyperparameter	Value	Description
minibatch size	32	Number of training cases over which value function update is computed
replay memory size	50000	value function updates are sampled from this number of most recent frame
Q update frequency	4	The number of actions selected by the agent between successive updates to the value function network
E update frequency	4	The number of actions selected by the agent between successive updates to the E function network
Q target network update frequency	10000	The frequency with which the Q target network is updated
E target network update frequency	1000	The frequency with which the E target network is updated
Q-value discount factor	0.99	Discount factor gamma used in the Q-value learning update
E-value discount factor	0.90 or 0.99	Discount factor gamma used in the E-value learning update
learning rate	0.0001	The learning rate used (α)

Table 2: MountainCar DQN working Hyperparameters

Hyperparameter	Value	Description
minibatch size	32	Number of training cases over which value function update is computed
replay memory size	50000	value function updates are sampled from this number of most recent frame
Q update frequency	1	The number of actions selected by the agent between successive updates to the value function network
E update frequency	1	The number of actions selected by the agent between successive updates to the E function network
Q target network update frequency	500	The frequency with which the Q target network is updated
E target network update frequency	500	The frequency with which the E target network is updated
Q-value discount factor	1	Discount factor gamma used in the Q-value learning update
E-value discount factor	1	Discount factor gamma used in the E-value learning update
learning rate	0.001	The learning rate used (α)
initial exploration	0.9	Initial value of ϵ in ϵ-greedy
final exploration	0.05	Final value of ϵ in ϵ-greedy
decay rate	0.0005	Decay constant in exponential decay of gamma

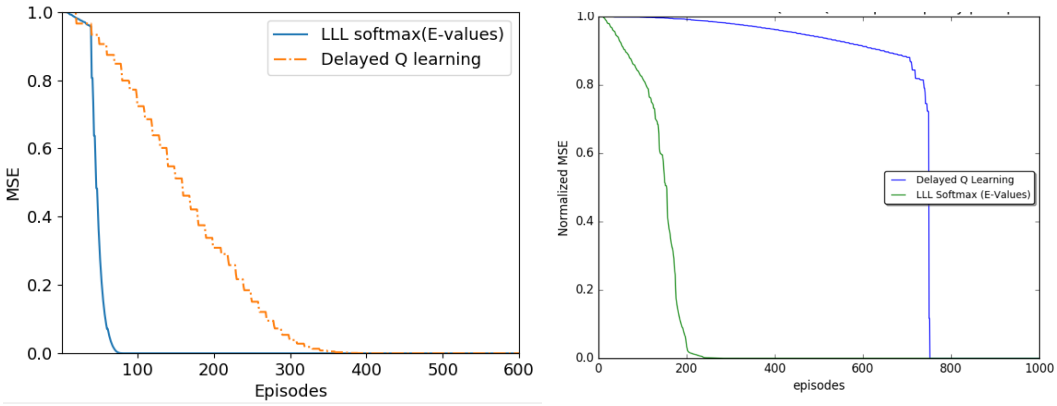
4.6 Optimism under uncertainty

In addition to main results in the paper, we also replicated an experiment in the Appendix to show how DORA compares to delayed Q learning. Delayed Q-learning starts exploration and exploitation with optimistic initial values. As the mean squared error of Q-values would be different in these two algorithms, we normalized it to $[0, 1]$, as mentioned in the DORA paper. Figure 5 shows similar trends between our result and DORA's, with our implementation converging faster. The difference

Table 3: CartPole Hyperparameters

Hyperparameter	Value	Description
minibatch size	32	Number of training cases over which value function update is computed
replay memory size	50000	value function updates are sampled from this number of most recent frame
Q update frequency	4	The number of actions selected by the agent between successive updates to the value function network
E update frequency	4	The number of actions selected by the agent between successive updates to the E function network
Q target network update frequency	1000	The frequency with which the Q target network is updated
E target network update frequency	1000	The frequency with which the E target network is updated
Q-value discount factor	0.999	Discount factor gamma used in the Q-value learning update
E-value discount factor	0.999	Discount factor gamma used in the E-value learning update
learning rate	0.0001	The learning rate used (α)
initial exploration	0.9	Initial value of ϵ in ϵ-greedy
final exploration	0.05	Final value of ϵ in ϵ-greedy
decay rate	0.005	Decay constant in exponential decay of gamma

should come from the unspecified hyperparameters used in DORA. We tuned γ to 0.9 and ϵ_1 to 0.01 to reach our result.



(a) ours

(b) original

Figure 5: Verified that LLL algorithm converges faster than delayed Q-learning.

4.7 CODE IMPLEMENTATION

To reproduce results from this work, we started from scratch as authors of DORA did not release any code. We write all the algorithms in tabular setting on our own from the pseudo code, and tune the parameters to make the agents perform best.

For the function approximation experiments, the DQN is a substantially modified version of pytorch tutorial found `http://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html`. Our implementation differs from the tutorial as a) the tutorial does not have

Table 4: Bridge Hyperparameters

Hyperparameter	Value	Description
minibatch size	30	Number of training cases over which value function update is computed
replay memory size	10000	value function updates are sampled from this number of most recent frame
Q net update frequency	1	The number of actions selected by the agent between successive updates to the value function network
E net update frequency	1	The number of actions selected by the agent between successive updates to the E function network
Q target network update frequency	20	The frequency with which the target network for Q net is updated
E target network update frequency	20	The frequency with which the target network for E net is updated
Q-value discount factor	0.99	Discount factor gamma used in the Q-value learning update
E-value discount factor	0.99	Discount factor gamma used in the E-value learning update
learning rate	0.01	The learning rate used (α)
initial exploration	0.9	Initial value of ϵ in ϵ-greedy
final exploration	0.05	Final value of ϵ in ϵ-greedy
decay rate	0.005	Decay constant in exponential decay of gamma

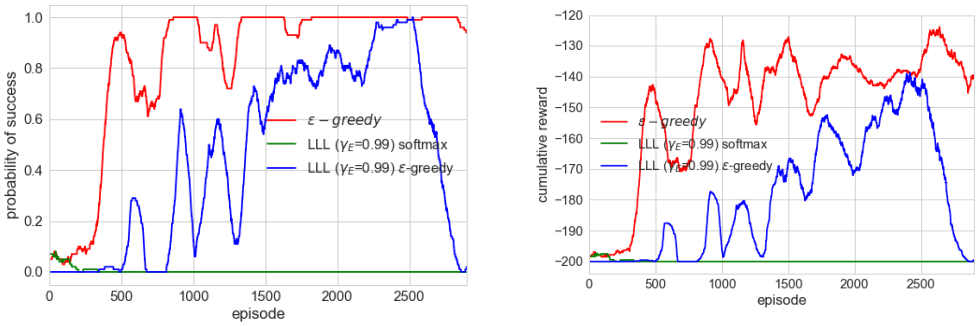


Figure 6: Verification of our pytorch DQN net on the mountain car environment

a target network, b) the tutorial is flawed when the next actions in the minibatch are all empty (e.g., the sampled transitions are the final transitions, as commonly encountered in DORA), and c) we immensely refactored the code to make it an reusable library. We verified our implementation by reproducing the results in openAI baseline on the mountain car environment (results shown in Figure 6 and hyperparameters in Table 2).

The mountain car and cart pole environments used came from the OpenAI gym (`https://github.com/openai/gym`). The bridge environment is a variant on the Frozen lake. Visit our code repository on github for detailed setup instruction.

All code and replication instructions are available at `https://github.com/nathanwang000/deep_exploration_with_E_network.git`.