Deep Exploration Bonuses for Episodic MDPs

Juan Duque
Princeton University
jduque@princeton.edu

John Li
Princeton University
johnli@princeton.edu

1 Background

1.1 Tabular Q-Learning:

In the classical reinforcement learning problem statement, an agent interacts with an environment by taking an action a in a state s at some time t and the environment returns a reward r and a subsequent state s' at time t'. We want to find the optimal Q function [7], Q^*, which maps the state s and the action a to its expected reward outcome by maximizing the subsequent actions

$$\begin{aligned} Q^*(s,a) &= \max_a E[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'} | s_t = s, a_t = a] \\ &= \max_a E[r_t + \gamma \max_{a'} Q^*(s', a') | s_t = s, a_t = a] \end{aligned}$$

To do so, we compute an estimate of the optimal Q function, $\hat{Q}_i$, which we update iteratively using the temporal difference error and a learning rate η

$$\hat{Q}_{h+1}(s,a) = \hat{Q}_h(s,a) + \eta(r_t + \gamma \max_{a'} \hat{Q}_h(s', a') - \hat{Q}_h(s,a))$$

The optimal policy, π^*, can easily be recovered from the Q function by taking the action with the biggest Q-value at each state.

1.2 Deep Q-learning

As described in the paper *Human-level control through deep reinforcement learning* [4], the table used in regular Q-learning can be replaced by a deep neural network, parameterized by θ. We can then minimize the square of the temporal difference error to train the network as follows

$$L(\theta) = E_{(s,a,r,s') \sim D}[(r + \gamma \max_{a'} \hat{Q}_h(s', a'; \theta)) - \hat{Q}_h(s, a; \theta)]^2$$

The gradient of the Loss L with respect to the parameters [6] θ is given by

$$\nabla_{\theta,h} L(\theta) = E_{(s,a,r,s') \sim D}[((r + \gamma \max_{a'} \hat{Q}_h(s', a'; \theta)) - \hat{Q}_h(s, a; \theta) \nabla_{\theta,h} \hat{Q}_h(s, a; \theta))]$$

2 Model Description

We aim to improve on existing methods that leverage state-action counts ([2], [5]) to improve exploration in reinforcement learning. In addition to bonuses that reward less visited state-action pairs, we introduce a depth based exploration bonus that rewards state-action pairs that are "far" from the starting state of the environment. More formally, we define the depth of a state-action pair, $D(x', a')$, as the minimum number of state-action pairs that precede (x', a'). To compute the depth we initialize $D(x, a) = H$ for all states x and actions a, and then at each step of the episode we update as follows

$$D(x', a') = \min(D(x', a'),\ D(x, a) + 1)$$

Where (x, a) is the state-action pair selected at the previous step. Now, we can add a depth-based bonus to the temporal difference error

$$\hat{Q}_{h+1}(s, a) = \hat{Q}_h(s, a) + \eta(r_t + \gamma \max_{a'} \hat{Q}_h(s', a') + b_D + b_N - \hat{Q}_h(s, a)) \tag{1}$$

$$L(\theta) = E_{(s,a,r,s')\sim D}[(r + \gamma \max_{a'} \hat{Q}_h(s', a'; \theta) + b_D + b_N) - \hat{Q}_h(s, a; \theta)]^2 \tag{2}$$

Equation (1) provides the update rule in the tabular setting, whereas equation (2) is the loss for the deep learning setting, with

$$b_D = \rho\sqrt{D(x, a)},\ b_N = \frac{\beta}{\sqrt{N(x, a)}}$$

Here ρ and β are constants and $N(x, a)$ is the number of times the state-action pair (x, a) has been visited. The bonus b_N is chosen with the settings described in [5]. We now construct a variation of the Q-learning algorithm described by [2], that takes advantage of both of these bonuses.

Algorithm 1: UCB Q-learning with depth bonuses

initialize $Q_h(x, a)$ randomly, $N_h(x, a) \leftarrow 0$, $D_h(x, a) \leftarrow H$ for all $(x, a, h) \in \mathcal{S} \times \mathcal{A} \times [H]$
for episode $k = 1, \ldots, K$ **do**
  receive x_1
  $D(x_1, 0) \leftarrow 0$
  $a_{h-1} \leftarrow 0$
  for episode $k = 1, \ldots, K$ **do**
    Take action $a_h \leftarrow \arg\max_{a'} Q_h(x_h, a')$ and observe x_{h+1}
    $t = N_h(x_h, a_h) \leftarrow N_h(x_h, a_h) + 1$
    $d = D(x_{h+1}, a_h) \leftarrow \min(D(x_{h+1}, a_h),\ D(x_h, a_{h-1}) + 1)$
    $b_N \leftarrow \frac{\beta}{\sqrt{t}}$; $b_D \leftarrow \rho\sqrt{d}$
    $Q_h(x_h, a_h) \leftarrow (1 - \alpha_t)Q_h(x_h,\ a_h) + \alpha_t[r_h(x_h, a_h) + V_{h+1}(x_{h+1}) + b_N + b_D]$
    $V_h(x_h) \leftarrow \max_{a' \in \mathcal{A}} Q_h(x_h, a')$
    $a_{h-1} \leftarrow a_h$
return a_t

Where α_t is the learning rate given by $\frac{H+1}{H+t}$. Intuitively, we would like the algorithm to explore state-action pairs with higher depth first, following the principle of optimistic exploration. In such way, the agent is more likely to find the terminal states in MDPs with long term or delayed rewards. By finding these terminal states earlier and their corresponding trajectories, the algorithm is likely to converge faster to more optimal policies.

3 Theoretical Results

Regret bounds on model-free RL algorithms with episodic MDP			
Algorithm	**Regret**	**Time**	**Space**
Q-learning ϵ-greedy [3]	$\Omega(\min\{T, A^{H/2}\})$	$O(T)$	$O(SAH)$
Delayed Q-learning [1]	$O_{S,A,H}(T^{4/5})$		
Q-learning (UCB-H) [2]	$O(\sqrt{H^4SAT})$		
Q-learning (UCB-B) [2]	$O(\sqrt{H^3SAT})$		
Lower bound	$O(\sqrt{H^2SAT})$	-	-

Jin et al. [2] bounds the total regret over K training episodes as $O(\sqrt{H^4SAT\iota})$ with probability $1-p$, where S is the size of the state space, A is the size of the action space, H is the maximum number of steps in each training episode, and $\iota = \log(SAT/p)$. We show that our depth-based exploration bonus achieves the same asymptotic regret bound. Our proof is largely based on the proof presented in that paper, with only slight modifications to lemma 4.3 and the proof of the main theorem.

First we strengthen lemma 4.3 by allowing b_t to include an additional $O(\sqrt{H/K})$ exploration bonus.

Lemma 3.1. *There exists an absolute constant $c > 0$ such that, for any $p \in (0,1)$, letting $b_t = c\sqrt{H^3\iota/t} + \rho\sqrt{H}$ with $\rho = c/K$, we have $\beta_t = 2\sum_{i=1}^{t} \alpha_t^i b_t \le 4c\sqrt{(H^3\iota + H)/t}$ and, with probability at least $1-p$, the following holds simultaneously for all $(x,a,h,k) \in \mathcal{S}\times\mathcal{A}\times[H]\times[K]$:*

$$0 \le (Q_h^k - Q_h^\star)(x,a) \le \alpha_t^0 H + \sum_{i=1}^{t} \alpha_t^i (V_{h+1}^{k_i} - V_{h+1}^\star)(x_{h+1}^{k_i}) + \beta_{t,d}$$

where $t = N_h^k(x,a)$ and $k_1, \ldots, k_t < k$ are the episodes where (x,a) was taken at step h.

Proof. The proof is mostly the same as the proof of lemma 4.3 in Jin et al. [2]. It suffices to show that the new bound on β_t holds when $b_t = c\sqrt{H^3\iota/t} + \rho\sqrt{H}$. This is straightforward, as $t = N_h^k(x,a) \le K$ (the state-action pair (x,a) can appear at step h at most K times—once in each of the K episodes) and $\rho = c/K$:

$$\begin{aligned}
\beta_{t,d}/2 &= \sum_{i=1}^{t} \alpha_t^i b_i \\
&= \sum_{i=1}^{t} \alpha_t^i (c\sqrt{H^3\iota/i} + \rho\sqrt{H}) \\
&= \sum_{i=1}^{t} \alpha_t^i (c\sqrt{H^3\iota/i} + c\frac{\sqrt{H}}{K}) \\
&\le \sum_{i=1}^{t} \alpha_t^i (c\sqrt{H^3\iota/i} + c\sqrt{H/i}) \text{ (because } t \le K) \\
&= \sum_{i=1}^{t} \alpha_t^i (c\sqrt{(H^3\iota + H)/i}) \\
&\in [c\sqrt{(H^3\iota + H)/t}, 2c\sqrt{(H^3\iota + H)/t}] \text{ by lemma 4.1.a in the paper}
\end{aligned}$$

□

Using this lemma, we obtain Jin et al. [2]'s regret bound for our modified exploration strategy so long as $\rho \leq c$.

Theorem 3.2. *There exists an absolute constant $c \geq \rho > 0$ such that, for any $p \in (0,1)$, if we choose $b_N = c\sqrt{H^3\iota/t}$ and $b_D = \rho\sqrt{H}$, then with probability $1-p$, the total regret of UCB Q-learning with depth bonuses (our Algorithm 1) is at most $O(\sqrt{H^4SAT\iota})$, where $\iota := \log(SAT/p)$.*

Proof. Again the proof is largely the same as in Jin et al. [2]. We follow the proof in the paper up to the following point, where $\delta_h^k := (V_h^k - V_h^{\pi_k})(x_h^k)$ and $\phi_h^k := (V_h^k - V_h^\star)(x_h^k)$:

$$\text{Regret}(K) = \sum_{k=1}^{K}(V_1^\star - V_1^{\pi_k})(x_1^k) \leq \sum_{k=1}^{K}(V_1^k - V_1^{\pi_k})(x_1^k) = \sum_{k=1}^{K}\delta_1^k$$

The paper then establishes the following bound on δ_h^k, using lemma 4.3 along with a number of other lemmas:

$$\delta_h^k \leq \alpha_t^0 H + \sum_{i=1}^{t}\alpha_t^i\phi_{h+1}^{k_i} + \beta_t - \phi_{h+1}^k + \delta_{h+1}^k + \xi_{h+1}^k \tag{3}$$

We have checked that each of the other lemmas used to establish this bound still hold in our setting. This allows us to use the same argument to establish the following modified bound, where we have applied our lemma 3.1 in place of [2]'s lemma 4.3:

$$\delta_h^k \leq \alpha_t^0 H + \sum_{i=1}^{t}\alpha_t^i\phi_{h+1}^{k_i} + \beta_{t,d} - \phi_{h+1}^k + \delta_{h+1}^k + \xi_{h+1}^k \tag{4}$$

Where the only difference is the $\beta_{t,d}$ term. Our lemma 3.1 is applicable because our depth-based exploration bonus is $\rho\sqrt{d} = \sqrt{d}/K \leq O(\sqrt{H/K})$. We use the fact that $d \leq H$, which holds because $D(x,a) = H$ initially and any reachable state-action pair can be at most H steps away from the initial state.

Jin et al. [2] proceeds to bound each of the terms in the right-hand side of (3), and then puts these bounds together to get a bound for the total regret $\sum_{k=1}^{K}\delta_1^k$. Using the same argument, we obtain a regret bound for our modified exploration strategy; the only difference is that our argument accounts for an extra $O(KH)$ term resulting from depth-based bonuses. Letting $T = KH$, we have:

$$\begin{aligned}
\text{Regret}(K) = \sum_{k=1}^{K}\delta_h^k &\leq O(\sqrt{H^4SAT\iota}) + \beta_{t,d} \text{ (by argument in the paper and (4))} \\
&\leq O(\sqrt{H^4SAT\iota}) + \sum_{k=1}^{K}\beta_{n_h^k,d} \\
&= O(\sqrt{H^4SAT\iota}) + O(1)\cdot\sum_{x,a}\sum_{n=1}^{N_h^K(x,a)}\sqrt{\frac{H^3\iota + H}{n}} \\
&\leq O(\sqrt{H^4SAT\iota}) + O(\sqrt{H^3SAK\iota}) \\
&\leq O(\sqrt{H^4SAT\iota}) + O(\sqrt{H^2SAT\iota}) \\
&= O(\sqrt{H^4SAT\iota})
\end{aligned}$$

□

Therefore our exploration bonus does not affect the regret bound proved by Jin et al. [2]. We now proceed to test the depth exploration bonus strategy empirically in different MDPs with their own reward structures.

4 Experiments

4.1 Acrobot

Description: The first environment described by Sutton (1996) consists of a system composed of two joints and two links where the joint between the links has an actuator. At the beginning of the episode, the links hang down and the agent must use the actuator to swing the lower joint to a desired height. The actuator, however is not strong enough to raise the joints in a single try, so the agent must accumulate velocity by swinging forwards and backwards. We use an OpenAI gym implementation with discrete actions (push left, no push, push forward).

Setup: We allow the agent to run for 100 episodes, where each episode finalizes after a maximum number of steps 1000 or when the goal is reached. The hyperparameters chosen are as follows: $\epsilon_0 = 0.9$, $\epsilon_\infty = 0.05$, $H = 1000$, $\beta = 0.1$, $\rho = 0.01$.

4.2 Cartpole

Description: In the Cartpole environment, a cart is attached to a pole through a joint. The cart moves in a friction-less track and must prevent the pole from reaching an angle greater than 15 degrees with respect to the vertical axis. The cart should not exceed certain x-axis positions. The episode ends if any of the constraints are violated and the agent receives a reward of 1 for each time-step it remains within the constraints. We use an OpenAI gym implementation with discrete actions (push left, no push, push forward), and a state tuple composed by cart velocity, pole velocity and the position values of the cart and the pole.

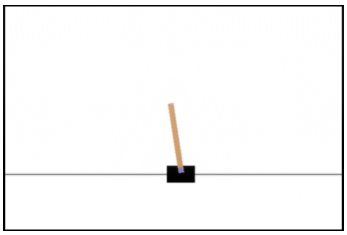


Figure 1: A frame from the Cartpole environment

Setup: We allow the agent to run for 200 episodes, where each episode finalizes after a maximum number of steps 300 or when the goal is reached. The hyperparameters chosen are as follows: $\epsilon_0 = 0.9$, $\epsilon_\infty = 0.05$, $H = 300$, $\beta = 0.1$, $\rho = 0.01$.

4.3 Mountain Car

Description: We test our approach on "Mountain Car"as described by Andrew Moore: an environment in which a car located in a valley between two mountains receives a reward if it reaches the top of the mountain at its right. However, the car's engine is not strong enough to climb the mountain in a single pass. Thus, the car must accumulate enough momentum, by swinging between the two mountains, in order to reach its objective. We use an OpenAI gym implementation with discrete actions (push left, no push, push forward) and a state tuple composed by velocity and position values. This environment is well known in the reinforcement learning literature for requiring an agent to explore the environment thoroughly.

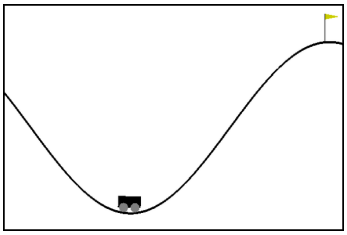
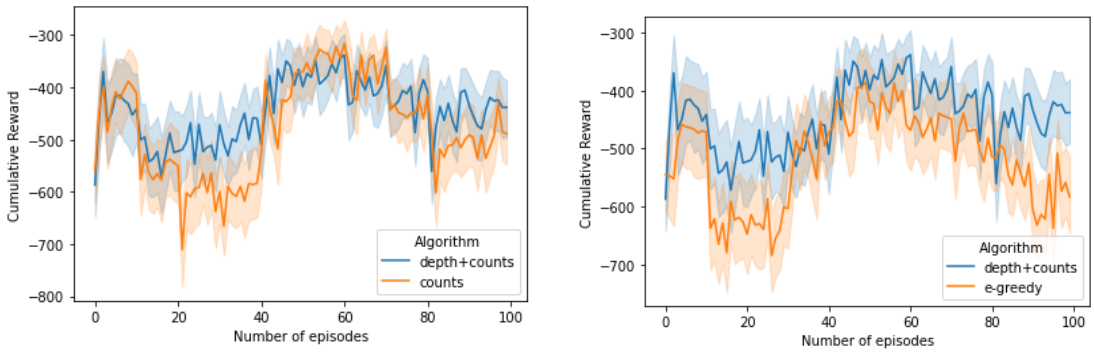


Figure 2: A frame from the Mountain Car environment

Setup: We allow the agent to run for 100 episodes, where each episode finalizes after a maximum number of steps 10000 or when the goal is reached. The hyperparameters chosen are as follows: $\epsilon_0 = 0.9$, $\epsilon_\infty = 0.05$, $H = 10000$, $\beta = 0.1$, $\rho = 0.01$.

5 Experimental Results

5.1 Acrobot

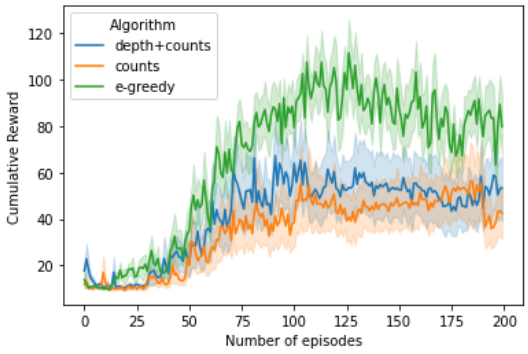


(a) (b)

Figure 3: (a) Compares the average cumulative reward our algorithm to the variant with only counts bonuses. (b) Compares the average cumulative reward our algorithm to ϵ-greedy. Each trial (run of total number of episodes) was done 30 times to get statistical significance. The hard colored line shows the average, whereas the light colored region shows a 75% confidence interval.

Acrobot is an environment with delayed rewards (only given at the end of the episode), this is showcased in Figure 3: (b) where both algorithms confidently outperform epsilon greedy. Epsilon greedy is known to have worst case exponential performance in environments with delayed rewards [2]. The difference between exploring with counts only bonuses vs counts and depths is almost negligible, although using the depth bonuses seems to have an edge in earlier episodes.

5.2 Cartpole

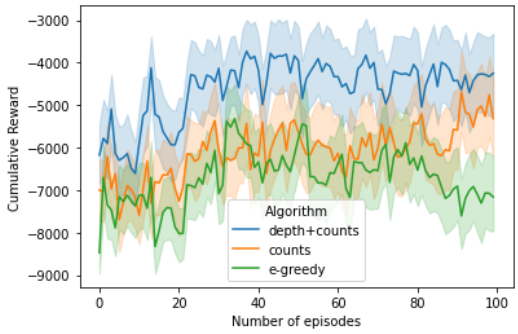


(a)

Figure 4: Compares the average cumulative reward of the three exploration algorithms in the Cartpole environment. Each trial (run of total number of episodes) was done 30 times to get statistical significance. The hard colored line shows the average, whereas the light colored region shows a 75% confidence interval.

In Cartpole, rewards are immediate and state-action pairs tend to be repeated multiple times in the optimal policy. It is no surprise then that epsilon greedy outperforms both bonus based methods. From the two methods, including the depths bonus is marginally better which makes sense intuitively: it is better to explore state-action pairs with depth as these are likely to have higher cumulative reward.

5.3 Mountain Car



(a)

Figure 5: Compares the average cumulative reward of the three exploration algorithms in the Mountain Car environment. Each trial (run of total number of episodes) was done 30 times to get statistical significance. The hard colored line shows the average, whereas the light colored region shows a 75% confidence interval.

Our algorithm confidently outperforms the other exploration strategies in the Mountain Car environment. The rewards in this environment are very delayed compared to the other ones, thus thorough exploration is necessary to find the optimal policies. By rewarding deep state action pairs the agent is more likely to find the terminal states at earlier episodes, ultimately leading to faster convergence.

6 Conclusion

Deep exploration bonuses have been shown to be particularly effective in MDPs that have delayed rewards when compared to the naive exploration strategy (ϵ-greedy) and the count based bonuses. The theoretical bounds on the regret for this exploration strategy have been supported empirically on three different control tasks. We hope that bonus functions that make use of a depth heuristic can be further explored to achieve even better results, as it is not yet clear which one of them is more effective.

References

hwan Oh, M., & Iyengar, G. (2018). *Directed exploration in pac model-free reinforcement learning.*

Jin, C., Allen-Zhu, Z., Bubeck, S., & Jordan, M. I. (2018). Is q-learning provably efficient? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2018/file/d3b1fb02964aa64e257f9f26a31f72cf-Paper.pdf

Kearns, M., & Kaelbling, L. (2002). Near-optimal reinforcement learning in polynomial time. In *Machine learning* (pp. 209–232).

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... others (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529.

Rashid, T., Peng, B., Böhmer, W., & Whiteson, S. (2020). *Optimistic exploration even with a pessimistic initialisation.*

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... others (2016). Mastering the game of go with deep neural networks and tree search. *nature*, *529*(7587), 484.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (Second ed.). The MIT Press. Retrieved from http://incompleteideas.net/book/the-book-2nd.html