Appendices for "An empirical and conceptual categorization of value-based exploration methods"

A Environment specification

Antishaping si a corridor bounded in $[0, 1]$, with termination when the agent goes out of bounds. The agent begins each episode at a position uniformly sampled from $[0.45, 0.55]$, and moves left or right depending on the action according to a $N(\mu = 0.005, \sigma = 0.001)$ normal distribution each time-step. On average, by repeating the same action from the center of the environment, the terminal state can be reached in 100 time-steps. The reward signal is based on the density of a normal distribution centered at 0.5, as well as an additional large reward upon termination. Despite the relatively high reward near the center of the environment, the optimal policy is to move towards the closest environment boundary.

Hypercube is a five-dimensional hypercube with edge length 20 that is centered at the start state. The agent can move either direction in any dimension according to a $N(\mu = 1, \sigma = 0.15)$ random variable. If the agent reaches an edge, any further movement past the edge is ignored. To guide exploration, we provide a shaping reward based on the number of edges that the agent touches. Moving to a state that touches an additional edge produces a reward that is higher than the gamma-discounted return the agent would receive if it had remained in the previous state forever. If the agent moves to a state adjacent to five edges, the episode terminates. Therefore, the optimal policy is to go towards the closest edge that the agent is not already touching until the episode terminates.

VarianceWorld is a corridor bounded in $[0, 1]$, with termination when the agent goes out of bounds. If the agent moves to a state less than 0, it receives a reward of 0.02. If the agent moves to a state greater than 1, the reward is uniformly drawn from $\{1, -0.1, 0.1, -2, 2\}$. Episodes start uniformly in $[0.45, 0.55]$, and the agent can move left or right by an amount drawn from a $N(\mu = 0.05, \sigma = 0.01)$ distribution. Since the average reward from the rightmost terminal state, 1, is higher than 0.02, the optimal policy is to always select the 'right' action.

Discrete RiverSwim has six states, which we emulate by bounding the world in $[0, 1]$ and considerings movement by 0.2 to be similar to moving from one state to another in discrete RiverSwim. Movement is similar to the original RiverSwim, the downstream action causes the agent to go downstream according to a $N(\mu = 0.2, \sigma = 0.02)$ distribution. The upstream action is a bit more complicated. In discrete RiverSwim, the upstream action has a chance to move downstream, stay in the same state, or move upstream. In continuous RiverSwim, the upstream action is drawn from a Gamma distribution with mean 0.05 and 0.4 probability to move downstream by some amount. The agent starts uniformly in $[0.2, 0.4]$, but there is no episode termination. When an action would move the agent to a state downstream of 0, it is moved to state 0 and given a 0.0005 reward. Similarly, when the agent moves upstream of 1, it is moved to state 1 and given a reward of 1. While these rewards are at a different scale than the discrete version of RiverSwim, their relative magnitudes are the same. We observed similar results under the two reward structures.

B Experimental setup

To represent the state we use tilecoding with 3 tiles in each dimension and 32 tilings. This setup provides both broad generalization, from the small number of tiles, and fine discretization, from the large number of tilings. Our results are aggregated over 360 runs for each combination of algorithm and environment.

Agent	Parameters
Softmax	learning rate
State counts	learning rate, count bonus multiplier
State-action counts	learning rate, count bonus multiplier
Optimistic initialization	learning rate, initial weight value
Epsilon greedy	learning rate, epsilon
IEQL+	learning rate, confidence interval width, max standard deviation
Softmax Actor Critic	learning rate (separately for policy and value update)
Next-state prediction	learning rate (separately for value and next-state functions), error bonus multiplier
Bootstrap Q-learning	learning rate
Noisy networks	learning rate

Table 1: List of parameters swept for each agent.

Parameter	Values
Learning rate	$\left\{\frac{1}{2^5}, \frac{1}{2^6}, \frac{1}{2^7}, \frac{1}{2^8}, \frac{1}{2^9}, \frac{1}{2^{10}}\right\}$
Count bonus multiplier	$\{0.005, 0.01, 0.05, 0.1, 0.5\}$
Initial weight value	$\{1, 10, 50, 100, 200\}$
Epsilon	$\left\{\frac{1}{2^1}, \frac{1}{2^2}, \frac{1}{2^3}, \frac{1}{2^4}, \frac{1}{2^5}, \frac{1}{2^6}\right\}$
Confidence interval width	$\{0.5, 0.75, 0.9, 0.95, 0.99\}$
Max standard deviation	$\{0.001, 0.05, 0.01, 0.015\}$
Error bonus multiplier	$\{1, 10, 100, 1000\}$

Table 2: Values swept for each parameter

B.1 Parameter sweep

Up to three parameters are coarsely swept for each agent, using 360 runs for each parameter combination. The parameters swept for each agent are listed in Table 1, while the parameter values are listed in Table 2.

B.2 Evaluation

We freeze each agent's behaviour policy before starting the evaluation phase. The frozen behaviour policies take the greedy action with respect to the learned value function. In the cases with randomized value functions, we freeze the value function distributions and continue to sample value functions from those distributions. We take the same approach with actor-critic, which is an on-policy method. Actor critic's target policy is represented by the frozen action preferences that are learned in the 500000 time-step learning phase.