A Visualization of Q-Landscape

Figure 5 shows the visualization of learned policies (actions given different states) and Q values in TD3 during training in the Pendulum-v0 environment, where the state space is 3-dim and action space is 1-dim. The red lines indicates the selected action by the current policy. The learned Q function are always **non-convex** and **locally convex**. As a consequence, in many states the TD3 is not able to find globally optimal solution and local gradient information may be misleading in finding actions with the highest Q values.

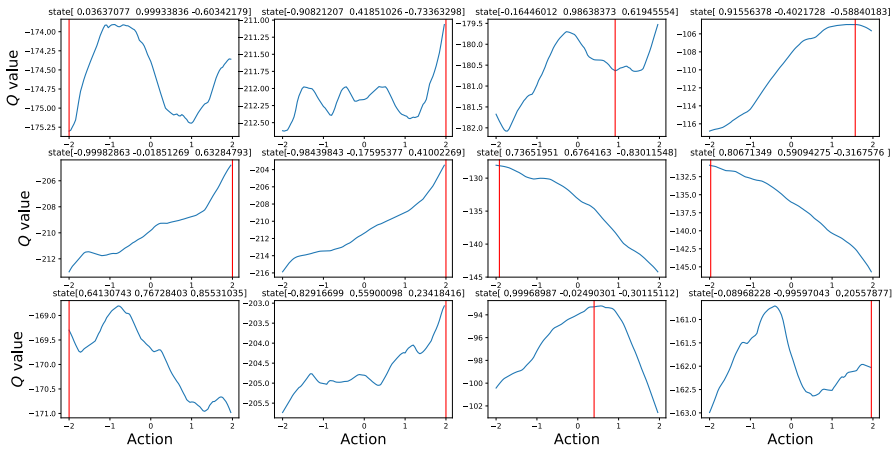


Figure 5: Landscape of learned value function of TD3 in the Pendulum-v0 environment (control task with 1-dim action space so that the state-action values can be easily visualized.)

B Missing Proofs in Section 4.2

Proof of Lemma 1. By the properties of Euclidean metric, we have the following lemma.

Lemma 2. Let $\mathcal{A} \subset \mathbb{R}^d$ with Euclidean metric. Any $a \in \mathcal{A}$, $\|x\|_2 \leq k$. Then there exists a set $\{a_1, \ldots, a_N\}$ that is a $\frac{2k\sqrt{d}}{N^{1/d}}$-covering of N. In other words, for all $a \in \mathcal{A}$, $\exists i \in [N]$,

$$\|a - a_i\|_2 \leq \frac{2k\sqrt{d}}{N^{1/d}}.$$

Define $\mathcal{A}_i$ as the set of all the actions that is closest to a_i:

$$\mathcal{A}_i \triangleq \left\{ a \in \mathcal{A} : i = \min\{\arg\min_{j \in [N]} \|a_j - a\|_2\} \right\}.$$

Let $P(\cdot)$ be the probability measure of uniform distribution over $\mathcal{A}$ We have $P(\mathcal{A}_i) \geq 1/(2N)$. Now since we have n uniform samples from set $\mathcal{A}$. Let $N = n/\log^2(n/\delta)$.

Lemma 3 (Coupon Collector's problem)**.** It takes $O(N \log^2(N/\delta))$ rounds of random sampling to see all N distinct options with a probability at least $1 - \delta$.

Proof. Consider a general sampling problem: for any finite set $\mathcal{N}$ with $|\mathcal{N}| = N$. For any n, whose sampling probability is $p(c)$, with a probability at least $1 - \delta$, it requires at most

$$\frac{\log(1/\delta)}{\log(1 + \frac{p(n)}{1-p(n)})} \text{ for } n \text{ to be sampled.}$$

Since $\log(1+x) \geq x - \frac{1}{2}x^2$ for all $x > 0$, we have

$$\frac{\log(1/\delta)}{\log(1+\frac{p(n)}{1-p(n)})} \leq \log(1/\delta)\frac{1}{\frac{p(n)}{1-p(n)} - \frac{p(n)^2}{2(1-p(n))^2}} = O(\log(1/\delta)\frac{1-p(n)}{p(n)}).$$

Searching the whole space $\mathcal{N}$ with each new element being found with probability $\frac{N-i}{N}$ at round i, it requires at most

$$O(\sum_{i=1}^{N} \log(\frac{N}{\delta})\frac{N}{N-i}) = O(\log^2(\frac{N}{\delta})N),$$

with a probability at most $1-\delta$.

By Lemma 3 We have with a probability at least $1-\delta$, there exists a sample in each $\mathcal{A}_i$ described above. To proceed, we apply the Lipschitz of the Q function, Lemma 1 follows.

Proof of Theorem 1. Now we proceed to show Theorem 1. We denote $\frac{2k\sqrt{d}L\log^{1/d}(n/\delta)}{n^{1/d}}$ by σ^2. Our global policy network gives a prediction from a linear model. Let our dataset be $\{s_t, a_t^+\}_{t=1}^T$. Let $\epsilon_t = a_t^+ - a_t^*$. Let $\boldsymbol{S} = (s_1, \ldots, s_T)^T$ and $\boldsymbol{a}^+, \boldsymbol{a}^*, \boldsymbol{\epsilon}$ be the corresponding vector for $(a_t^+)_{t=1}^T, (a_t^*)_{t=1}^T, (\epsilon_t)_{t=1}^T$. We have any ϵ_i, ϵ_j are independent for $i \neq j$. We further make an assumption that $\mathbb{E}[\epsilon_t] = 0$. Then we immediately have $\mathbb{E}[\epsilon_i\epsilon_j] = 0$ as well. This assumption is just for simplifying the proof, we can show similar results without assume the unbiasedness as the bias can be bounded.

To proceed, let $\theta^* = \arg\min_\theta \mathbb{E}_s\|s^T\theta - a^*(s)\|_2^2$.

Since the ERM solution is simply OLS (ordinary least square). We have the estimate

$$\hat{\theta} = (\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{a}^+.$$

We have

$$\hat{\theta} - \theta^* = (\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T(\boldsymbol{a}^* - \boldsymbol{S}\theta^*) + (\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{\epsilon}.$$

Since $\|\epsilon_t\|^2 \leq \sigma^2$, we have $\mathrm{Var}(\epsilon_t) \leq \sigma^2$. We observe that

$$\mathrm{Var}((\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{\epsilon}) \leq \sigma^2((\boldsymbol{S}^T\boldsymbol{S})^{-1}) = \mathcal{O}(\sigma^2 p/T).$$

The generalization error is given by

$$\begin{aligned}
&\mathbb{E}_s\|s^T\hat{\theta} - a^*(s)\|_2^2 \\
&\leq \mathbb{E}_s\|s^T\hat{\theta} - s^T\theta^*\|_2^2 + \mathbb{E}_s\|s^T\theta^* - a^*(s)\|_2^2 \\
&= \mathcal{O}\left(\frac{\sigma^2 p}{T} + \mathbb{E}_s\|s^T(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T(\boldsymbol{a}^* - \boldsymbol{S}\theta^*)\|_2^2 + \mathbb{E}_s\|s^T\theta^* - a^*(s)\|_2^2\right) \\
&= \tilde{\mathcal{O}}\left(\frac{\sigma^2 p}{T} + \mathbb{E}_s\|s^T\theta^* - a^*(s)\|_2^2\right).
\end{aligned}$$

C One-step Zeroth-Order Optimization with Consistent Iteration

Algorithm 3 One-step Zeroth-Order Optimization with Consistent Iteration

Require
Objective function Q, domain $\mathcal{A}$, current point a_0, number of local samples n_1, number of global samples n_2, local scale $\eta > 0$ and step size h, number of steps m.
for $t = 1, \dots n_2$ **do**
  Globally sampling
  Sample a point uniformly in the entire space by

$$a_{t0} \sim \mathcal{U}_{\mathcal{A}}$$

  where $\mathcal{U}_{\mathcal{A}}$ is the uniform distribution over $\mathcal{A}$.
  for $i = 1, \dots, m$ **do**
    Locally sampling
    Sample n_1 points around $a_{t,i-1}$ by

$$\tilde{a}_j = a_{t,i-1} + \mu e_j \text{ for } e_j \sim \mathcal{N}(0, I_d), j = 1, \dots n_1,$$

    where $\mathcal{N}(0, I_d)$ is the standard normal distribution centered at 0.
    Update
    Set $a_{t,i} = a_{t,i-1} + h(\arg\max_{a \in \{\tilde{a}_j\}} Q(a) - a_{t,i-1})$
  end for
end for
return $\max_{a \in \{a_t m\}_{t=1}^{n_2}} Q(a)$.

D Implementation Details

D.1 Network Structure And Hyper-Params

In our experiments, we follow [8] to use a 3-layer MLP with 256 hidden units for both critic and actor networks. We also follow [8] to use 25000 timesteps for worm-up and use a batch-size of $256 : 1$ training-interaction proportion during training.

D.2 Mixture Density Networks

Our implementation of ZOSPI with MDN is based on neural network with multiple outputs. For MDN with K Gaussian mixture outputs, the neural network has $3 \times K$-dim output. The first K-dim units are normalized with softmax activation as the probability of selecting the Gaussians, the following K-dim units are corresponding K mean values of the Gaussians, and the last K-dim units are standard deviation of the K Gaussians. In our experiments, we use Diracs instead of Gaussians for parameterization. Therefore, our networks output $2 \times K$-dim units for each action dimension, where the first K-dimensions denote the probabilities and the latter K-dimensions denote the mean values. In our experiments, we use $K = 1$ for Hopper, HalfCheetah and Ant, $K = 5$ for Walker2d, and $K = 5$ for Humanoid. ($K = 5$ and $K = 10$ achieve on-par performance for the Humanoid environment, though the $K = 10$ setting spend roughly one more time computational expense).

More implementation details are provided with the code in the supplementary material.

D.3 Running Time of ZOSPI

We conduct our experiments with 8 GTX TITAN X GPUs and 32 Intel(R) Xeon(R) E5-2640 v3 @ 2.60GHz CPUs. The wall clock time of our proposed method is roughly 3-times slower than running TD3, without application of MDNs (i.e., NoD $= 1$). It takes roughly 20 hours to train Hopper with 10 seeds, and takes about 120 hours to train Humanoid when NoD $= 10$ with 10 seeds.

E Better Exploration with Bootstrapped Networks

Sample efficient RL requires algorithms to balance exploration and exploitation. One of the most popular way to achieve this is called optimism in face of uncertainty (OFU) [15, 44–46], which gives an upper bound on Q estimates and applies the optimal action corresponding to the upper bound. The optimal action a_t is given by the following optimization problem:

$$\arg\max_a Q^+(s_t, a), \tag{6}$$

where Q^+ is the upper confidence bound on the optimal Q function. A guaranteed exploration performance requires both a good solution for (6) and a valid upper confidence bound.

While it is trivial to solve (6) in the tabular setting, the problem can be intractable in a continuous action space. Therefore, as shown in the previous section, ZOSPI adopts a local set to approximate policy gradient descent methods in the local region and further applies a global sampling scheme to increase the potential chance of finding a better maxima.

As for the requirement of a valid upper confidence bound, we use bootstrapped Q networks to address the uncertainty of Q estimates as in [47–50, 14]. Specifically, we keep K estimates of Q, namely $Q_1, \dots Q_K$ with bootstrapped samples from the replay buffer. Let $\overline{Q} = \frac{1}{K}\sum_k Q_k(s,a)$. An upper bound Q^+ is

$$Q^+(s,a) = \overline{Q} + \phi\sqrt{\frac{1}{K}\sum_k [Q_k(s,a) - \overline{Q}]^2}, \tag{7}$$

where ϕ is the hyper-parameter controlling the failure rate of the upper bound. Another issue is on the update of bootstrapped Q networks. Previous methods [49] usually update each Q network with the following target $r_t + \gamma Q_k\left(s_{t+1}, \pi_{\theta_t}(s_{t+1})\right)$, which violates the Bellman equation as π_{θ_t} is designed to be the optimal policy for Q^+ rather than Q_k. Using π_{θ_t} also introduces extra dependencies among the K estimates. We instead employ a global random sampling method to correct the violation as

$$r_t + \gamma \max_{i=1,\dots n} Q_k\left(s_{t+1}, a_i\right), \quad a_1, \dots a_n \sim \mathcal{U}_{\mathcal{A}}.$$

The correction also reinforces the argument that a global random sampling method yields a good approximation to the solution of the optimization problem (6). The detailed algorithm is provided in Algorithm 4 in Appendix E.1.

E.1 Algorithm 4: ZOSPI with Bootstrapped Q networks

F Gaussian Processes for Continuous Control

Different from previous policy gradient methods, the self-supervised learning paradigm of ZOSPI permits it to learn both its actor and critic with a regression formulation. Such a property enables the learning of actor in ZOSPI to be implemented with either parametric models like neural networks or non-parametric models like Gaussian Processes (GP). Although plenty of previous works have discussed the application of GP in RL by virtue of its natural uncertainty capture ability, most of these works are limited to model-based methods or discrete action spaces for value estimation [51–56]. On the other hand, ZOSPI formulates the policy optimization in continuous control tasks as a regression objective, therefore empowers the usage of GP policy in continuous control tasks.

As a first attempt of applying GP policies in continuous control tasks, we simply alter the actor network with a GP to interact with the environment and collect data, while the value approximator is still parameterized by a neural network. We leave the investigation of better consolidation design in future work.

G Experiments on the Four-Solution-Maze.

The Four-Solution-Maze (FSM) environment is a diagnostic environment where four positive reward regions with a unit side length are placed in the middle points of 4 edges of a $N \times N$ map. An agent starts from a uniformly initialized position in the map and can then move in the map by taking actions according to the location observations (current coordinates x and y). Valid actions are limited to

Algorithm 4 ZOSPI with UCB Exploration

Require

- The number of epochs M, the size of mini-batch N, momentum $\tau > 0$ and the number of Bootstrapped Q-networks K.
- Random initialized policy network π_{θ_1}, target policy network $\pi_{\theta'_1}$, $\theta'_1 \leftarrow \theta_1$.
- K random initialized Q networks, and corresponding target networks, parameterized by $w_{k,1}, w'_{k,1}$, $w'_{k,1} \leftarrow w_{k,1}$ for $k = 1, \ldots, K$.

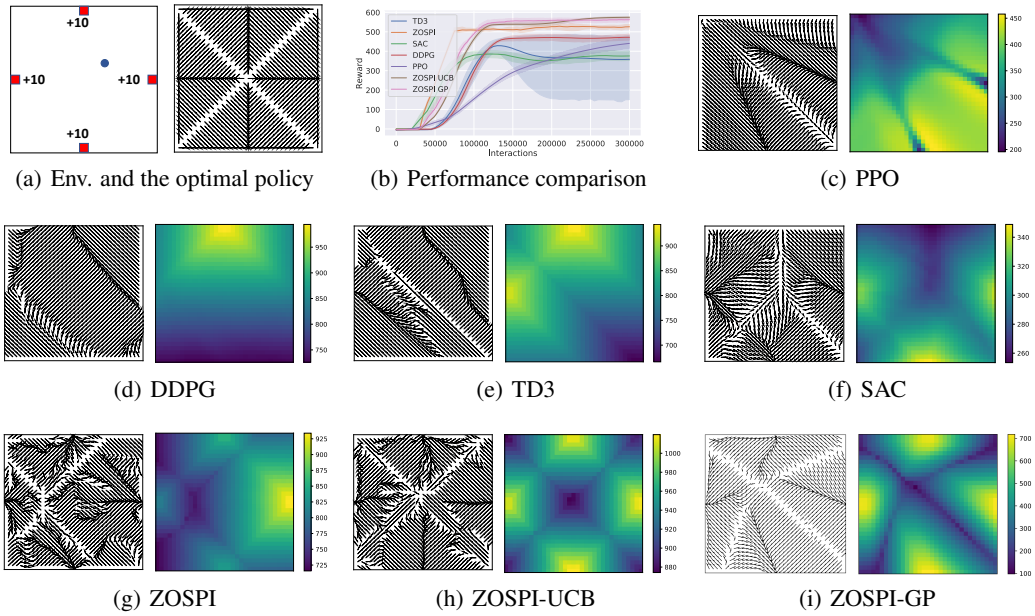
for iteration $= 1, 2, \ldots$ **do**
  for t $= 1, 2, ..., T$ **do**
    # Interaction
    Run policy $\pi_{\theta'_t}$, and collect transition tuples $(s_t, a_t, s'_t, r_t, m_t)$.
    for epoch $j = 1, 2, ..., M$ **do**
      Sample a mini-batch of transition tuples $\mathcal{D}_j = \{(s, a, s', r, m)_i\}_{i=1}^N$.
      # Update Q
      for $k = 1, 2, ..., K$ **do**
        Calculate the k-th target Q value $y_{ki} = r_i + \max_l Q_{w'_{k,t}}(s'_i, a'_l)$, where $a'_l \sim \mathcal{U}_\mathcal{A}$.
        Update $w_{k,t}$ with loss $\sum_{i=1}^N m_{ik}(y_{ki} - Q_{w'_{k,t}}(s_i, a_i))^2$.
      end for
      # Update π
      Calculate the predicted action $a_0 = \pi_{\theta'_t}(s_i)$
      Sample actions $a_l \sim \mathcal{U}_\mathcal{A}$
      Select $a^+ \in \{a_l\} \cup \{a_0\}$ as the action with maximal $Q^+(s_t, a)$ defined in (7).
      Update policy network with Eq.(2).
    end for
    $\theta'_{t+1} \leftarrow \tau\theta_t + (1-\tau)\theta'_t$.
    $w'_{k,t+1} \leftarrow \tau w_{k,t} + (1-\tau)w'_{k,t}$.
    $w_{k,t+1} \leftarrow w_{k,t}$; $\theta_{t+1} \leftarrow \theta_t$.
  end for
end for

$[-1, 1]$ for both x and y axes. Each game consists of $2N$ timesteps for the agent to navigate in the map and collect rewards. In each timestep, the agent will receive a $+10$ reward if it is inside one of the 4 reward regions or a tiny penalty otherwise. For simplicity, there are no obstacles in the map, the optimal policy thus will find the nearest reward region, directly move towards it, and stay in the region till the end. Figure 6(a) visualizes the environment and the ground-truth optimal solution.

Although the environment is simple, we found it extremely challenging due to existence of multiple sub-optimal policies that only find some but not all four reward regions. We do not conduct grid search on hyper-parameters of the algorithms compared in our experiments but set them to default setting across all experiments. Though elaborated hyper-parameter tuning may benefit for certain environment.

On this environment we compare ZOSPI to on-policy and off-policy SOTA policy gradient methods in terms of the learning curves, each of which is averaged by 5 runs. The results are presented in Figure 6(b). And learned policies from different methods are visualized in Figure 6(c)-6(i). For each method we plot the predicted behaviors of its learned policy at grid points using arrows (although the environment is continuous in the state space), and show the corresponding value function of its learned policy with a colored map. All policies and value functions are learned with 0.3M interactions except for SAC whose figures are learned with 1.2M interactions as it can find 3 out of 4 target regions when more interactions are provided.

We use 4 bootstrapped Q networks for the upper bound estimation in consideration of both better value estimation and computational cost for ZOSPI with UCB. And in ZOSPI with GP, a GP model is used to replace the actor network in data-collection, $i.e.$,exploration. The sample efficiency of ZOSPI is much higher than that of other methods. Noticeably ZOSPI with UCB exploration is the only method that can find the optimal solution, *i.e.*, a policy directs to the nearest region with a positive reward. All other methods get trapped in sub-optimal solutions by moving to only part of reward regions they find instead of moving toward the nearest one.



(a) Env. and the optimal policy (b) Performance comparison (c) PPO

(d) DDPG (e) TD3 (f) SAC

(g) ZOSPI (h) ZOSPI-UCB (i) ZOSPI-GP

Figure 6: Visualization of learned policies on the FSM environment. (a) the FSM environment and its optimal solution, where the policy should find the nearest reward region and move toward it; (b) learning curves of different approaches; (c)-(i) visualize the learned policies and corresponding value functions. We run multiple repeat experiments and show the most representative and well-performing learned value function and policy of each method.