Supplementary Material for Loaded DiCE: Trading off Bias and Variance in Any-Order Score Function Estimators for Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
`email`

1 Derivation of Value Function formulation

We start out with the $J_\diamond$ objective:

$$J_\diamond = \sum_{t=0}^{T} \gamma^t \Big(\boxdot(a_{\leq t}) - \boxdot(a_{<t}) \Big) R_{t+1}. \tag{1}$$

We evaluate this objective by taking an expectation over the trajectories τ as induced by the policy π. Here τ is a complete sequence of states, actions and rewards, $\tau = \{s_0, a_0, r_1, .., s_T, a_T\}$. Note that for convenience in the following derivation we have defined the reward, r_{t+1}, to arrive at the next time step, after action a_t was taken.

$$\begin{aligned}
\mathbb{E}_\pi[J_\diamond] &= \sum_\tau P(\tau) J_\diamond(\tau) && (2)\\
&= \sum_\tau P(\tau) \Big(\sum_{t=0}^{T} \gamma^t \big(\boxdot(a_{\leq t}) - \boxdot(a_{<t})\big) R_{t+1} \Big) && (3)\\
&= \sum_{t=0}^{T} \gamma^t \Big(\sum_\tau P(\tau) \big(\boxdot(a_{\leq t}) - \boxdot(a_{<t})\big) R_{t+1} \Big) && (4)\\
&= \sum_{t=0}^{T} \gamma^t J_t && (5)\\
& && (6)
\end{aligned}$$

We note that for each time step the term, J_t is of the form:

$$J_t = \sum_\tau P(\tau) f(\tau_{\leq t}) g(\tau_{>t}), \tag{7}$$

where $f(a_{\leq t}, s_{\leq t}) = \big(\boxdot(a_{\leq t}) - \boxdot(a_{<t})\big)$ and $g(\tau_{>t}) = R_{t+1}$.

Next we use:

$$\begin{aligned}
P(\tau) &= P(\tau_{\leq t}) P(\tau_{>t} | \tau_{\leq t}) && (8)\\
&= P(\tau_{\leq t}) P(\tau_{>t} | s_t, a_t), && (9)
\end{aligned}$$

where in the last step we have used the Markov property. Substituting we obtain:

$$J_t = \sum_{\tau} P(\tau_{\leq t})P(\tau_{>t}|s_t, a_t)f(a_{\leq t}, s_{\leq t})g(\tau_{>t}) \tag{10}$$

$$= \sum_{\tau_{\leq t}} P(\tau_{\leq t})f(a_{\leq t}, s_{\leq t})\sum_{\tau_{>t}} P(\tau_{\geq t}|s_t, a_t)g(\tau_{>t}) \tag{11}$$

$$\tag{12}$$

If we substitute back for g and f we obtain:

$$J_t = \sum_{\tau_{\leq t}} P(\tau_{\leq t})\big(\boxdot(a_{\leq t}) - \boxdot(a_{<t})\big)\sum_{\tau_{>t}} P(\tau_{\geq t}|s_t, a_t)R_{t+1} \tag{13}$$

$$= \sum_{\tau_{\leq t}} P(\tau_{\leq t})\big(\boxdot(a_{\leq t}) - \boxdot(a_{<t})\big)\mathbb{E}[R_{t+1}|s_t, a_t] \tag{14}$$

$$= \sum_{\tau_{\leq t}} P(\tau_{\leq t})\big(\boxdot(a_{\leq t}) - \boxdot(a_{<t})\big)Q(s_t, a_t) \tag{15}$$

$$\tag{16}$$

Putting all together we obtain the final form:

$$\mathbb{E}_{\pi}[J_{\diamond}] = \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t \Big(\boxdot(a_{\leq t}) - \boxdot(a_{<t})\Big)Q(s_t, a_t)\right] \tag{17}$$

2 Experimental Details

2.1 Random MDPs

We use the `mdptoolbox.example.rand()` function from PyMDPToolbox to generate random MDP transition functions with five states and four actions per state.

The reward is a function only of state, and is sampled from $\mathcal{N}(5, 10)$. We use $\gamma = 0.95$. When sampling for the stochastic estimators, we use batches of 512 rollouts of length 50 steps unless the batch size is otherwise specified.

We only compute higher order derivatives of the derivative of the first parameter at each order, to save computation.

For the sweeps over λ and τ we use 200 batches for each value of λ or τ. To simulate function approximation error in our analysis of the impact of τ, we add a gaussian noise with standard deviation 10 to the true value function.

2.2 MAML experiments

We use the following hyperparameters for our MAML experiments:

Parameter	**Value**
γ	0.97
hidden layer size	100
number of layers	2
task batch size	20 trajectories
meta batch size	40 tasks
inner loop learning rate	0.1
outer loop optimiser	Adam
outer loop learning rate	0.0005
outer loop τ	1.0
reward noise	Uniform(-0.01, 0.01) at each timestep

We also normalise all advantages in each batch (per task).