DDA4230 Reinforcement learning Variance reduction

Lecture 19

Lecturer: Baoxiang Wang *Scribe: Baoxiang Wang*

1 Goal of this lecture

In this lecture we discuss variance reduction methods for Monte-Carlo methods.
Suggested reading: Chapter 13 of *Reinforcement learning: An introduction*; Search publications of your interest.

2 Recap: Policy gradient

The policy gradient theorem states that

$$\nabla_\theta J(\theta) = \sum_{s\in\mathcal{S}} \rho^{\pi_\theta}(s \mid s_0) \sum_{a\in\mathcal{A}} Q^{\pi_\theta}(s,a) \nabla_\theta \pi_\theta(a \mid s)\,.$$

2.1 REINFORCE

We can sample N trajectories following the policy π and use the empirical mean to estimate the gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi[Q^\pi(s,a) \nabla_\theta \log \pi_\theta(a \mid s)]\,.$$

For $Q^\pi(s,a)$, we can use return $G_t = \sum \gamma^t r_t$ to estimate. For $\nabla_\theta \log \pi_\theta(a \mid s)$, it depends on the form of the policy, but can be directly calculated. The discount γ^t in the last step is usually omitted in practice.

Algorithm 1: REINFORCE (Monte-Carlo method)

Initialize the policy parameter θ
for *each episode* **do**
  Sample one trajectory on policy π_θ: $s_0, a_0, r_0, s_1, a_1, \ldots, s_T$
  for *each* $t = 0, 1, \ldots, T$ **do**
    $G_t \leftarrow \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$
    $\theta \leftarrow \theta + \alpha \gamma^t G_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)$

2.2 REINFORCE with baselines

One problem of policy gradient method is high variance. A natural solution is to subtract a baseline $b(s)$ from Q^π, i.e.,

$$\nabla_\theta J(\theta) \propto \sum_{s\in\mathcal{S}} \rho^{\pi_\theta}(s \mid s_0) \sum_{a\in\mathcal{A}} (Q^\pi(s,a) - b(s)) \nabla \pi_\theta(a \mid s)\,.$$

The baseline can be any function, even a random variable, as long as it does not depend on the action a. The update rule that we end up with is a new version of REINFORCE that includes a general baseline

$$\theta \leftarrow \theta + \alpha\gamma^t(G_t - b(s_t))\nabla_\theta \log \pi_\theta(a_t \mid s_t)\,.$$

One natural choice for the baseline is an estimate of the state value function $\hat{V}(s, \boldsymbol{w})$, where $\boldsymbol{w} \in R^d$ is a weight vector to be learned. We can use the same method as we adopted in learning $\boldsymbol{\theta}$ to learn $\boldsymbol{w}$.

Algorithm 2: REINFORCE with baseline

Initialize the policy parameter $\boldsymbol{\theta}$ and $\boldsymbol{w}$ at random.
for *each episode* **do**
  Sample one trajectory under policy π_θ: $s_0, a_0, r_0, s_1, a_1, r_1 \ldots, s_T$
  for *each* $t = 1, 2, \ldots, T$ **do**
    $G_t \leftarrow \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$
    $\delta \leftarrow G_t - \hat{V}(s_t, \boldsymbol{w})$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha_{\boldsymbol{w}} \delta \nabla_{\boldsymbol{w}} \hat{V}(s_t, \boldsymbol{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \gamma^t \delta \nabla_{\boldsymbol{\theta}} \log \pi_\theta(a_t \mid s_t)$

3 Control variates

The control variates method is a variance reduction technique used in Monte Carlo methods. It exploits information about the errors in estimates of known quantities to reduce the error of an estimate of an unknown quantity.

Let the unknown parameter of interest be μ and assume that we have a statistic X such that the expected value of X is μ. That is, $\mathbb{E}[X] = \mu$, where X is an unbiased estimator of μ. Suppose we calculate another statistic Y such that $\mathbb{E}[Y] = \eta$ is a known value. Then,

$$X' = X + c(Y - \eta)$$

is also an unbiased estimator of μ, for any $c \in \mathbb{R}$. The variance of this new estimator X' is

$$\begin{aligned}
\mathbb{V}[X'] &= \mathbb{V}[X + c(Y - \eta)] \\
&= \mathbb{V}[X] + c^2\mathbb{V}[Y - \eta] + 2c\mathrm{Cov}(X, Y - \eta) \\
&= \mathbb{V}[Y - \eta]c^2 + 2\mathrm{Cov}(X, Y)c + \mathbb{V}[X]\,.
\end{aligned}$$

As c is arbitrary, minimizing this quadratic variance yields

$$c = -\frac{\mathrm{Cov}(X, Y)}{\mathbb{V}[Y]}\,,$$

which is the best parameter for variance reduction. This optimal choice of c guarantees variance reduction, as

$$\mathbb{V}[X'] = (1 - \mathrm{Corr(X,\ Y)})\mathbb{V}[X]\,.$$

Recall that policy gradient estimates $\nabla_\theta J(\theta) = \mathbb{E}_\pi[Q^\pi(s,a)\nabla_\theta \log \pi_\theta(a \mid s)]$, which indicates the vanilla estimator as

$$X = Q^\pi(s,a)\nabla_{\theta_i} \log \pi_{\theta_i}(a \mid s)\,,$$

for some $\theta_i \in \theta$. Let $Y = \nabla_{\theta_i} \log \pi_{\theta_i}(a \mid s)$, where $\mathbb{E}[Y] = 0$, the estimator

$$X' = X + cY = (Q^\pi(s,a) - c)\nabla_{\theta_i} \log \pi_{\theta_i}(a \mid s)$$

is unbiased. We choose

$$\begin{aligned} c &= -\frac{\mathrm{Cov}(X,Y)}{\mathbb{V}[Y]} \\ &= -\frac{\mathbb{E}[(X - \mathbb{E}[X])\mathbb{E}[Y - \mathbb{E}[Y]]}{\mathbb{E}[(Y - \mathbb{E}[Y])^2]} \\ &= -\frac{\mathbb{E}[XY]}{\mathbb{E}[Y^2]} \\ &= -\frac{\mathbb{E}[Q^\pi(s,a)(\nabla_{\theta_i} \log \pi_{\theta_i}(a \mid s))^2]}{\mathbb{E}[(\nabla_{\theta_i} \log \pi_{\theta_i}(a \mid s))^2]}\,. \end{aligned}$$

As both terms are unknown, we cannot proceed further analytically. One approach is to run Monte-Carlo estimation of both the dividend and the divisor and use them to calculate c. Note that this does not introduce any bias by the arbitrarity of c. Though, the variance reduction is no longer guaranteed with the Monte-Carlo estimation of c.

Monte-Carlo estimation also have respective c for each parameter $\theta_i \in \theta$, which introduces heavy computational overhead in practice. We commonly simply "cancel" $(\nabla_{\theta_i} \log \pi_{\theta_i}(a \mid s))^2$ and use

$$c = \mathbb{E}[Q^\pi(s,a)] = \mathbb{E}[V(s)]\,,$$

which results in the REINFORCE with baseline algorithm.

4 Actor–critic methods

In practice, most of the variance is from the Monte-Carlo estimation G_t of $Q(s_t, a_t)$. An estimation with much lower variance can be obtained by estimating a parametrized $Q(s,a)$ and bootstrap the estimation into the policy gradient. This results in a biased estimator but with much lower variance. One way to estimate the value function is the temporal-difference method, which has been discussed in previous lectures. With this bootstrap, the methods is called actor-critic.

Actor-critic methods consist of two models.

- The critic updates the value function parameters $\boldsymbol{w}$.
- The actor updates the policy parameters $\boldsymbol{\theta}$ in the direction suggested by the critic.

Note that although the REINFORCE with baseline method learns both a policy and a state value function, we do not consider it to be an actor–critic method because its state value function is used only as a baseline instead of a critic.

One-step actor–critic methods replace the full return of REINFORCE with the one-step return and use a learned state value function as the baseline, as

$$\begin{aligned}\boldsymbol{\theta_{t+1}} &= \boldsymbol{\theta_t} + \alpha_{\boldsymbol{\theta}}(G_t - \widehat{V(s_t, \boldsymbol{w})}\ )\nabla \log \pi_\theta(a_t \mid s_t) \\ &= \boldsymbol{\theta_t} + \alpha_{\boldsymbol{\theta}}(r_t + \gamma \widehat{V(s_{t+1}, \boldsymbol{w})} - \widehat{V(s_t, \boldsymbol{w})}\ )\nabla \log \pi_\theta(a_t \mid s_t)\,.\end{aligned}$$

This algorithm then takes two inputs: a differentiable policy parametrized by $\pi_\theta(a \mid s)$ and a differentiable state value function parametrized by $\hat{V}(s, \boldsymbol{w})$.

Algorithm 3: One-step actor–critic (episodic)

Initialize the policy parameter $\boldsymbol{\theta}$ and $\boldsymbol{w}$ at random. **for** *each episode* **do**
  Initialize s_0, the first state of each episode
  for *each* $t = 0, 1, \ldots, T-1$ **do**
    sample $a \sim \pi(a \mid s_t, \boldsymbol{\theta})$
    take action a and observe s', r
    $\delta \leftarrow r + \gamma \widehat{V(s', \boldsymbol{w})} - \widehat{V(s, \boldsymbol{w})}$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha_{\boldsymbol{w}} \delta \nabla_{\boldsymbol{w}} \widehat{V(s, \boldsymbol{w})}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \delta \nabla_{\boldsymbol{\theta}} \log \pi(a \mid s, \boldsymbol{\theta})$
    $s' \leftarrow s$

5 Other variance reduction methods

Reward to go This variance reduction method exploits causality, as it is based on the fact that the future does not effect the past. The policy at time t cannot effect the rewards we get until time t and therefore when MC sampling the action value function $Q(s, a)$ we can use the return G_t which is based only on future rewards relative to t (instead of G_0). This has been already integrated into our algorithm charts.

The variance is usually reduced in this way in practice because we are summing up less random variables. The variance reduction does not hold rigorously, but some intuition can be give for a specific case of independent rewards. If we assume that the rewards in a trajectory are independent of each other, then the variance of the sum of N random variables is equal to the sum of the individual variances

$$\mathbb{V}[\sum_{i=1}^{T} r_i] = \sum_{i=1}^{T} \mathbb{V}[r_i]\,.$$

Because the variance is a non-negative function, this sum monotonically increases as N increases.

Discount factor The use of a discount factor also reduces the variance because it puts less emphasis on rewards far into the future. Rewards in the far future are more likely to have high variance because in many cases the far future is more uncertain than the near future. We may choose to go in different directions and therefore arrive at various states

that are very different from our current state. This is also already integrated into the algorithm charts.

Batch learning If we are to use the online version described in our algorithms, we will be updating the gradient based on a single environment interaction and that will be a very noisy signal (high variance). In practice, we prefer to use batches, and this can be achieved with the asynchronous advantage actor-critic (A3C) algorithm, which is the actor-critic algorithm (A2C) we described above plus the asynchronization of generalizing samples. In a nutshell, it runs multiple agents in parallel and synchronized their gradient signals which reduce the variance.

Advantage normalization In a batch, we collect the advantage data $A(s,a) = Q(s,a) - V(s)$ on multiple experiences to for the data $\{A(s,a)\}_N$. Advantage normalization is to normalize the data to zero mean and optionally unit variance within each batch. This introduces bias in general, but have some effect of variance reduction in practice.

Diagonal variance For continuous case, the policy can be defined as the normal probability density over a real-valued scalar action, with mean and standard deviation given by parametric function approximators

$$\pi(a \mid s, \boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^k \det(\Sigma)}} \exp(-\frac{1}{2}(a-\mu)^T \Sigma^{-1}(a-\mu)) \,.$$

This general setting generates actions within which the dimensions can be dependent with each other. By removing this dependency, i.e. by setting $\Sigma = \mathrm{diag}(\sigma, \ldots, \sigma)$, with either fixed or learnable σ, the sample complexity of the system can be reduced in general in practice.

Eligibility traces Eligibility traces, also known as the generalized advantage estimator, is an interpolation between TD(0) and TD(1) and an analogous of using n-step Monte-Carlo sampling with bootstrapping on later steps, where n correspond to a geometrically decreasing factor λ.

$$G_{t,\lambda} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n} \,.$$

This balances the variance introduced by MC sampling and the unstable estimation process of TD(0).

Reward clip and gradient clip A simple approach is to clip either the gradient estimator or the reward. The former brings $Q(s,a)\nabla \log \pi(a \mid s)$ to a bounded range and the latter brings $Q(s,a)$ to a bounded range. Both of which will be effectively reducing the variance and can prevent estimation outlier to be included into the calculation. However these brute-force tricks can cause large bias and can trim useful information that should have been included in the computation (for example, say, in episodic task where completing the task gives a lump sum reward of $+100$ which each step cost -1, and the reward is trimmed to $[-1, +1]$).

Advanced statistical variance reduction tools We refer the reader to checkout advanced tools in statistical variance reduction and checkout their presence in reinforcement learning.

References

[1] Mnih, Volodymyr, et al. "Asynchronous methods for deep reinforcement learning." International conference on machine learning. 2016.

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction; 2nd Edition.* 2018.

[3] John Schulman, et al. "High-dimensional continuous control using generalized advantage estimation." ICLR 2016.

[4] David Silver, et al. "Deterministic policy gradient algorithms." ICML. 2014.

[5] Adrien, E. Intuitive Explanation of Policy Gradient. 2018. Availabel here.

[6] Lilian, W. Policy Gradient Algorithms. 2018. Available here.

Acknowledgement

This lecture notes partially use material from *Reinforcement learning: An introduction.* Actor-critic is prepared by Ke Li.