CS 6101 Week #4 Notes: Actor-Critic Introduction, Value Functions and Q-Learning

Note taking: Alexandre Gravier, Joel Lee
LaTeX transcription: Alexandre Gravier <al.gravier@gmail.com>

Contents

1 Recap about policy gradients

We define $J(\theta) \doteq E_{\tau \sim p_\theta(\tau)} \left[\sum_t r\left(\mathbf{s}_t, \mathbf{a}_t\right)\right]$ so that the objective of RL can be defined as an optimization exercise consisting in finding an assignment of policy parameters $\theta^\star = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} J(\theta)$.

$J(\theta)$ is not usually optimizable as such due to f.e. dimensionality issues, so we use a sample-based unbiased estimate: $J(\theta) \approx \frac{1}{N} \sum_i \sum_t r\left(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}\right)$. Taking the gradient of $J(\theta)$ along θ allows maximizing the expected reward as per the policy.

1.1 Policy differentiation with a "convenient identity"

Let $r(\tau) \doteq \sum_{t=1}^{T} r\left(\mathbf{s}_t, \mathbf{a}_t\right)$ the total reward of a trajectory τ.

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta E_{\tau \sim p_\theta(\tau)}[r(\tau)] && && \text{(1a)}\\
&= \nabla_\theta \int \pi_\theta(\tau) r(\tau)\, d\tau && \text{by definition of expectation} && \text{(1b)}\\
&= \int \nabla_\theta \pi_\theta(\tau) r(\tau)\, d\tau && \text{by linearity} && \text{(1c)}
\end{aligned}$$

At this point, the expression $\int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau$ seems rather intractable. This is where the following "convenient identity" can be used to derive a tractable expression of $\nabla_\theta J(\theta)$.

A convenient identiy

$$\pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) = \pi_\theta(\tau) \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta \pi_\theta(\tau) \tag{2}$$

Furthermore, we can expand the definition of $\pi_\theta\left(\tau\right)$ and take its logarithm:

$$\begin{aligned}
\pi_\theta\left(\tau\right) &= p\left(\mathbf{s}_1\right) \prod_{t=1}^{T} \pi_\theta\left(\mathbf{a}_t \mid \mathbf{s}_t\right) p\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right) && \text{(3a)}\\
\Leftrightarrow \log \pi_\theta\left(\tau\right) &= \log p\left(\mathbf{s}_1\right) + \sum_{t=1}^{T} \log \pi_\theta\left(\mathbf{a}_t \mid \mathbf{s}_t\right) + \log p\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right) && \text{(3b)}
\end{aligned}$$

Exploration in Computer Science Research: Deep Reinforcement Learning (CS 6101, 2019), National University of Singapore.

Using (**??**) and (**??**) in (**??**), the gradient of the objective becomes:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) r(\tau) d\tau \qquad \text{using (??)} && \text{(4a)}\\
&= E_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(\tau) r(\tau) \right] \qquad \text{by definition of expectation} && \text{(4b)}\\
&= E_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \left[\log p\left(\mathbf{s}_1\right) + \sum_{t=1}^{T} \log \pi_\theta\left(\mathbf{a}_t \mid \mathbf{s}_t\right) + \log p\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right) \right] r(\tau) \right] && \text{(4c)}
\end{aligned}$$

We note that in the expression $\nabla_\theta \left[\log p\left(\mathbf{s}_1\right) + \sum_{t=1}^{T} \log \pi_\theta\left(\mathbf{a}_t \mid \mathbf{s}_t\right) + \log p\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right) \right]$ of the gradient w.r.t. θ in (**??**), the terms $\log p\left(\mathbf{s}_1\right)$ and $\log p\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right)$ are independent of θ, so we are left with:

$$\begin{aligned}
\nabla_\theta J(\theta) &= E_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \left[\sum_{t=1}^{T} \log \pi_\theta\left(\mathbf{a}_t \mid \mathbf{s}_t\right) \right] r(\tau) \right] && \text{(5a)}\\
&= E_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t \mid \mathbf{s}_t\right) \right) r(\tau) \right] \qquad \text{by linearity} && \text{(5b)}\\
&= E_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t \mid \mathbf{s}_t\right) \right) \sum_{t=1}^{T} r\left(\mathbf{s}_t, \mathbf{a}_t\right) \right] \qquad \text{by definition of } r(\tau) && \text{(5c)}
\end{aligned}$$

In Equation (**??**), the gradient of J is now a computable function of π_θ only.

We earlier mentioned the sample estimate of $J(\theta) \approx \frac{1}{N} \sum_i \sum_t r\left(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}\right)$; similarly $\nabla_\theta J(\theta)$ is approximated with samples, leading us to the algorithm:

$$\textbf{REINFORCE algorithm:} \begin{cases} \text{sample } \left\{\tau^i\right\} \text{ from } \pi_\theta\left(\mathbf{a}_t \mid \mathbf{s}_t\right) \\ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t \mid \mathbf{s}_t\right) \right) \sum_{t=1}^{T} r\left(\mathbf{s}_t, \mathbf{a}_t\right) \right) \\ \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \end{cases} \tag{6}$$

In (**??**), there is no use of the Markov property, so the algorithm can be used as such on POMDPs.

1.2 The bad news

We introduce some simplifying notation:

$$\begin{aligned}
\nabla_\theta J(\theta) &\approx \frac{1}{N} \sum_{i=1}^{N} \left(\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t \mid \mathbf{s}_t\right) \right) \sum_{t=1}^{T} r\left(\mathbf{s}_t, \mathbf{a}_t\right) \right) && \text{(7a)}\\
&\approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta\left(\tau\right) r\left(\tau\right) && \text{(7b)}
\end{aligned}$$

The big problem with vanilla REINFORCE lies in variance: if you were to repeatedly collect a small finite number N of samples, estimating each time the gradient based on these samples, you would observe that these estimates of the gradient vary a lot.

That means that the gradient descent step will likely take us away from the goal, and convergence will be very slow, or may not happen.

Additionally, given two sets of sample trajectories differing only by a constant factor, but differently centered around a total reward of 0, the basic policy gradient algorithm in (**??**) may change the parameters very differently.

There are two tricks that help with the large variance of the samples, both of which should always be used because they have no drawbacks.

1.3 Variance reduction with causality: the "rewards to go" trick

In (**??**), the gradient approximation can be improved by making use of causality: the policy at time t' cannot affect the reward at time t when $t < t'$.

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(\mathbf{a}_t \mid \mathbf{s}_t \right) \right) \sum_{t=1}^{T} r \left(\mathbf{s}_t, \mathbf{a}_t \right) \right) \tag{8a}$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \left(\underbrace{\nabla_\theta \log \pi_\theta \left(\mathbf{a}_t \mid \mathbf{s}_t \right)}_{\textit{gradient at t}} \underbrace{\sum_{t'=1}^{T} r \left(\mathbf{s}_{t'}, \mathbf{a}_{t'} \right)}_{\textit{this total reward should be computed from time t}} \right) \right) \quad \text{by distributivity} \tag{8b}$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \left(\nabla_\theta \log \pi_\theta \left(\mathbf{a}_t \mid \mathbf{s}_t \right) \underbrace{\sum_{t'=t}^{T} r \left(\mathbf{s}_{t'}, \mathbf{a}_{t'} \right)}_{\textit{reward to go}} \right) \right) \quad \text{by causality} \tag{8c}$$

Intuitively, the "**rewards to go**" trick comes from the observation that at time t, all past rewards cannot be affected by policy decisions.

We define the "reward to go" function $\hat{Q} : [\![1, N]\!] \times [\![1, T]\!] \mapsto \mathbb{R}$ as:

$$\hat{Q}_{i,t} \doteq \sum_{t'=t}^{T} r \left(\mathbf{s}_{t'}, \mathbf{a}_{t'} \right) \tag{9}$$

Therefore, the gradient of the objective function approximation with rewards to go is:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(\mathbf{a}_t \mid \mathbf{s}_t \right) \hat{Q}_{i,t} \tag{10}$$

We should note that most often, in policy gradient, we search the optimal policy within a class of policy functions that is time-invariant. However, to get the optimal policy in a finite horizon problem ($T < \infty$), we may need a time-dependent policy (e.g. in the last time step, there is no more need to care about energy preservation).

1.4 Variance reduction with baselines

Ideally, policy gradients should only depend on the relative values of the trajectory samples under the policy. But in REINFORCE, policy gradients dependent on the value of the total rewards of the policy samples: given the same set of samples shifted by a constant reward, the gradient direction and norm may change drastically. This is very undesirable.

The intuition behind the baseline trick is to subtract a constant "average reward" baseline from the trajectory rewards, so as to make trajectories that are less rewarding than average less likely, and those more rewarding more likely.

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta \left(\tau \right) \left(r \left(\tau \right) - b \right) \qquad \text{where } b = \frac{1}{N} \sum_{i=1}^{N} r(\tau) \tag{11}$$

Regardless of the baseline b used in computing the gradient, and the policy gradient defined in (**??**) remains an unbiased estimator of the gradient of the objective.

Proof that the gradient estimator in Eq. (**??**) *is unbiased for any* b. To prove that the expression in (**??**) is an unbiased estimator of $\nabla_\theta J(\theta)$, we need to prove that $E\left[\nabla_\theta \log \pi_\theta\left(\tau\right)\left(r\left(\tau\right)-b\right)\right] \stackrel{?}{=} E\left[\nabla_\theta \log \pi_\theta\left(\tau\right) r\left(\tau\right)\right]$, which — by linearity of expectation — is equivalent to:

$$E\left[\nabla_\theta \log \pi_\theta\left(\tau\right) b\right] \stackrel{?}{=} 0 \tag{12}$$

We expand the left-hand side:

$$\begin{aligned}
E\left[\nabla_\theta \log \pi_\theta\left(\tau\right) b\right] &= \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) b \, d\tau && \text{by definition of expectation} && \text{(13a)}\\
&= \int \nabla_\theta \pi_\theta(\tau) b \, d\tau && \text{by substitution with (??)} && \text{(13b)}\\
&= b \nabla_\theta \int \pi_\theta(\tau) \, d\tau && \text{by linearity} && \text{(13c)}\\
&= b \nabla_\theta \int 1 && \text{as } \pi_\theta \text{ is a probability distribution} && \text{(13d)}\\
&= 0 && && \text{(13e)}
\end{aligned}$$

Hence, any baseline b may be added to Eq. (**??**), and it will remain an unbiased estimator of the gradient of $J(\theta)$. □

Therefore, after subtracting an average reward baseline from the trajectories' reward, Eq. (**??**) remains an unbiased estimate of the gradient of the objective, but with a lower variance.

Choosing $b = \frac{1}{N}\sum_{i=1}^{N}$ works well, but it is not the baseline that results in the lowest variance estimator. To derive the best baseline, we can express that at its minimum, the derivative of the variance in function of the baseline is null.

The variance of the gradient of the objective is, by definition:

$$\begin{aligned}
\text{Var} &= E\left[\left(\nabla_\theta J(\theta)\right)^2\right] - E[\nabla_\theta J(\theta)]^2 && && \text{(14a)}\\
&= E_{\tau\sim\pi_\theta(\tau)}\left[\left(\nabla_\theta \log \pi_\theta(\tau)(r(\tau)-b)\right)^2\right] - \underbrace{E_{\tau\sim\pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau)(r(\tau)-b)\right]^2}_{\text{as shown in Eqs. (??), this term is independent of b}} && && \text{(14b)}
\end{aligned}$$

As we observe that only the first term depends on b,

$$\begin{aligned}
\frac{d\text{Var}}{db} &= \frac{d}{db} E_{\tau\sim\pi_\theta(\tau)}\left[\left(\nabla_\theta \log \pi_\theta(\tau)(r(\tau)-b)\right)^2\right] && && \text{(15a)}\\
&= \frac{d}{db} E\left[\left(\nabla_\theta \log \pi_\theta(\tau)\right)^2 \left(r(\tau)-b\right)^2\right] && && \text{(15b)}\\
&= \frac{d}{db} E\left[g(\tau)^2 (r(\tau)-b)^2\right] && \text{(we introduce } g(\tau) \doteq \nabla_\theta \log \pi_\theta(\tau)\text{)} && \text{(15c)}\\
&= \frac{d}{db}\left(E\left[g(\tau)^2 r(\tau)^2\right] - 2bE\left[g(\tau)^2 r(\tau)\right] + b^2 E\left[g(\tau)^2\right]\right) && && \text{(15d)}\\
&= \frac{d}{db}\left(-2bE\left[g(\tau)^2 r(\tau)\right] + b^2 E\left[g(\tau)^2\right]\right) && && \text{(15e)}\\
&= -2E\left[g(\tau)^2 r(\tau)\right] + 2bE\left[g(\tau)^2\right] && && \text{(15f)}
\end{aligned}$$

We want to find b s.t. $\frac{d\text{Var}}{db} = 0$:

$$\frac{d\text{Var}}{db} = 0 \tag{16a}$$

$$\implies -2E\left[g(\tau)^2 r(\tau)\right] + 2bE\left[g(\tau)^2\right] = 0 \tag{16b}$$

$$\implies b = \frac{E\left[g(\tau)^2 r(\tau)\right]}{E\left[g(\tau)^2\right]} \tag{16c}$$

We have derived the baseline that minimizes the variance of the objective function:

The optimal baseline

$$\hat{b} = \frac{E\left[\left(\nabla_\theta \log \pi_\theta(\tau)\right)^2 r(\tau)\right]}{E\left[\left(\nabla_\theta \log \pi_\theta(\tau)\right)^2\right]} \tag{17}$$

This the expected reward weighted by the square of the gradient magnitude, **element-wise**: the baseline $\hat{b}$ is also defined per element of the vector of parameters, and the $g(\tau)^2$ is an element-wise square of the gradient vector. Each scalar in the gradient baseline vector is weighted by its contribution to the gradient.

It is possible to implement $\hat{b}$, and it is guaranteed to be the optimal baseline w.r.t variance reduction. However, researchers often choose the simpler solution of the expected reward baseline, as it is easier to implement.

2 On-policy and off-policy algorithms