Reinforce

Our goal is to learn a policy π_θ with parameters θ which maximizes the reward within an episode:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} r_t\right]. \tag{1}$$

Here, we define τ as a single 'trajectory' $\tau = \{s_t, a_t, r_t\}_{t=0}^{T}$. The simplest way to optimize our objective is to use gradient descent with the gradient given by

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]. \tag{2}$$

However, this requires differentiating through the environment which is in general intractable. Instead, we use the linearity of expectations and the identity $\nabla_\theta \log f(\theta) = f(\theta)^{-1}\nabla_\theta f(\theta)$ to write

$$\begin{aligned}
\nabla_\theta J(\theta) &= \int \nabla_\theta p_\theta(\tau) R_\tau d\tau && (3)\\
&= \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) R_\tau d\tau && (4)\\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[R_\tau \nabla_\theta \log p_\theta(\tau)\right], && (5)
\end{aligned}$$

where we define $R_\tau := \sum_{t=0}^{T} r_t|\tau$. Since the environment does not depend on θ, we can simplify the calculation of $\nabla_\theta \log p_\theta(\tau)$:

$$\begin{aligned}
\nabla_\theta \log p_\theta(\tau) &= \nabla_\theta\left[\log p(s_0) + \sum_{t=0}^{T} \log p(s_{t+1}|s_t, a_t) \log \pi_\theta(a_t|s_t)\right] && (6)\\
&= \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t). && (7)
\end{aligned}$$

Inserting this in the expression for $\nabla_\theta J(\theta)$ and taking a Monte Carlo estimate of the expectation gives rise to the REINFORCE algorithm (Williams, 1992):

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta}\left[R_\tau \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\right] && (8)\\
&\approx \frac{1}{N}\sum_{\tau\sim\pi_\theta}\left(\sum_{t=0}^{T} r_t\right)\left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\right). && (9)
\end{aligned}$$

Variance reduction

While the REINFORCE algorithm is unbiased, it also has high variance, which can make learning slow and unstable. It is therefore common to introduce modifications which reduce the variance. The first of these comes from noting that an action taken at time t cannot affect the reward received at times $< t$. A corollary of this is that

$$\mathbb{E}_{\tau\sim\pi_\theta}\left[\left(\sum_{t'=0}^{t-1} r_{t'}\right)\nabla_\theta \log \pi_\theta(a_t|s_t)\right] = 0. \tag{10}$$

This allows us to define $R_t := \sum_{t'=t}^{T} r_{t'}$ and rewrite our REINFORCE update as

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{\tau \sim \pi_\theta} \sum_{t=0}^{T} R_t \nabla_\theta \log \pi_\theta(a_t|s_t). \tag{11}$$

It is straightforward to show that for any constant B:

$$\mathbb{E}_{\tau\sim\pi_\theta}\left[B\nabla_\theta \log \pi_\theta(a_t|s_t)\right] = B \int p_\theta(\tau)\nabla_\theta \log \pi_\theta(a_t|s_t)d\tau = B \int p_\theta(a_t, s_t)\nabla_\theta \log \pi_\theta(a_t|s_t)ds_t da_t \tag{12}$$

$$= B \int \nabla_\theta p_\theta(a_t, s_t)ds_t da_t = B\nabla_\theta \int p_\theta(a_t, s_t)ds_t da_t = B\nabla_\theta 1 = 0. \tag{13}$$

We can thus subtract a *baseline* from the reward-to-go R_t to get

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{\tau \sim \pi_\theta} \sum_{t=0}^{T} (R_t - B) \nabla_\theta \log \pi_\theta(a_t|s_t). \tag{14}$$

If we do a bit of maths we can derive the value of B which minimizes the variance of the estimator, ignoring the causality stuff above and defining $g(\tau) := \nabla_\theta \log p_\theta(\tau)$:

$$\sigma^2 = \mathbb{E}\left[g(\tau)^2(r(\tau) - B)^2\right] - \mathbb{E}\left[g(\tau)(r(\tau) - B)\right]^2 \tag{15}$$

$$= \mathbb{E}\left[g(\tau)^2(r(\tau) - B)^2\right] - \mathbb{E}\left[g(\tau)r(\tau)\right]^2 \tag{16}$$

$$\frac{d\sigma^2}{dB} = \mathbb{E}\left[g(\tau)^2(B - r(\tau))B\right] \tag{17}$$

$$\tag{18}$$

This is minimized when

$$\mathbb{E}\left[g(\tau)^2(B - r(\tau))\right] = 0 \tag{19}$$

$$B\mathbb{E}\left[g(\tau)^2\right] = \mathbb{E}\left[g(\tau)^2 r(\tau)\right] \tag{20}$$

$$B = \frac{\mathbb{E}\left[g(\tau)^2 r(\tau)\right]}{\mathbb{E}\left[g(\tau)^2\right]}. \tag{21}$$

That is, using the probability-weighted average reward as a baseline minimizes the variance of our estimator among constant baselines. However, the baseline can in fact be a function of the state without biasing our gradient estimator. If B depends on the state s_t we get

$$\mathbb{E}_{\tau\sim\pi_\theta}\left[B(s_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\right] = \int_{s_t} B(s_t)p(s_t)\left[\nabla_\theta \int_{a_t} \pi_\theta(a_t|s_t)da_t\right] ds_t \tag{22}$$

$$= \int_{s_t} B(s_t)p(s_t)\left[\nabla_\theta 1\right] ds_t = 0 \tag{23}$$

A common choice here is the expected future reward

$$V(s_t) = \mathbb{E}[\sum_{t'=t}^{T} r_t|s_t] \tag{24}$$

This gives rise to a so-called actor-critic algorithm

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{\tau \sim \pi_\theta} \sum_{t=0}^{T} (R_t - V(s_t)) \nabla_\theta \log \pi_\theta(a_t|s_t). \tag{25}$$

The actor-critic update is unbiased but can still have high variance due to the R_t term which sums rewards over a stochastic trajectory. One way to reduce this variance is to instead use a 'bootstrapped' estimate, noting that

$$\mathbb{E}\left[R_t\right] = \mathbb{E}[r_t + V(s_{t+1})], \tag{26}$$

with the right-hand side having lower variance than the left-hand side. We can thus use $r_t + V(s_{t+1})$ as an estimate of R_t to define the 'advantage function' $A_t = r_t + V(s_{t+1}) - V(s_t)$, which gives rise to the 'advantage actor critic' (A2C) algorithm:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{\tau \sim \pi_\theta} \sum_{t=0}^{T} (r_t + V(s_{t+1}) - V(s_t)) \nabla_\theta \log \pi_\theta(a_t|s_t). \tag{27}$$

While we have considered two extreme cases of a full Monte-Carlo sample of R_t and a 'one-step' bootstrap, the sum in R_t can be truncated to any order with $R_{t'}$ replaced by $V(s_{t'})$. In theory this estimator remains unbiased – however, in practice our estimate of $V(s_t)$ will be inexact, which introduces a bias to our updates. The bootstrapping procedure outlined above thus leads to a tradeoff between the bias and variance of our update step.

Implementation

Automatic differentiation

For the A2C algorithm outlined above, it is common to parameterize both the policy $\pi_\theta(a_t|s_t)$ and the value function $V_\phi(s_t)$ with neural networks – often with a subset of shared parameters between θ and ϕ. This leaves the problem of updating the parameters. In order to use out-of-the-box automatic differentiation software, we do this by defining an auxiliary utility (i.e. negative loss)

$$\tilde{J}(\theta) = \frac{1}{N} \sum_{\tau \sim \pi_\theta} \sum_{t=0}^{T} (r_t + V(s_{t+1}) - V(s_t)) \log \pi_\theta(a_t|s_t), \tag{28}$$

where r_t and $V(s_t)$ are constant w.r.t. θ. While $\tilde{J}(\theta)$ has no intrinsic interpretation, it is chosen such that

$$\nabla_\theta \tilde{J}(\theta) = \nabla_\theta J(\theta). \tag{29}$$

Importantly, we no longer have to differentiate through the environment – only the policy.

The gradient w.r.t. the value function loss is given by

$$\nabla_\phi \mathcal{L}_V = \nabla_\phi \sum_t \frac{1}{2} (V_\phi(s_t) - R_t)^2 = \sum_t (V_\phi(s_t) - R_t) \nabla_\phi (V_\phi(s_t). \tag{30}$$

When π_θ and V_ϕ share parameters, we also need to balance these the gradients w.r.t. θ and ϕ in our update step. We do this by defining $\delta_t = R_t - V(s_t)$, where R_t can optionally be a bootstrapped estimate, introducing a hyperparameter β_V, and maximizing the utility

$$\mathcal{L} = \frac{1}{N} \sum_{\tau \sim \pi_\theta} \sum_{t=0}^{T} (\log \pi_\theta(a_t|s_t) + \beta_V V(s_t)) \delta_t. \tag{31}$$

Importantly, we do not propagate gradients w.r.t. ϕ through the computation of δ_t.

References

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.