1 REINFORCE

Let's start from first principles and derive the REINFORCE update. Assume that our policy (in the bandit case) is parametrised by a parameter θ. The "exact" objective function is given by

$$\nu(\theta) = -\int \pi_\theta(a) q(a)\, da$$

Assuming that π_θ is continuously differentiable with respect to θ (for differentation to make sense and to justify pulling the gradient into the integral) and that $\pi(a) > 0$ for all a, we can calculate the gradient as

$$\begin{aligned}
\nabla\nu(\theta) &= -\int \nabla\pi_\theta(a) q(a)\, da \\
&= -\int \pi_\theta(a) \frac{\nabla\pi_\theta(a)}{\pi_\theta(a)} q(a)\, da \\
&= -\int \pi_\theta(a) \nabla \log \pi_\theta(a) q(a)\, da \\
&= -\mathbb{E}[\nabla \log \pi_\theta(A_t) q(A_t)] \\
&= -\mathbb{E}[\nabla \log \pi_\theta(A_t) R_t].
\end{aligned}$$

The SGD update is just a sample of the above:

$$-\nabla \log \pi_\theta(A_t) R_t.$$

We cannot easily perform this update in PyTorch. One approach is to define a surrogate objective function:

$$\ell_t = -\log \pi_\theta(A_t) R_t.$$

Note that ℓ_t is not an unbiased estimate of $\nu(\theta)$, but taking the gradient of ℓ_t does give an unbiased estimate of $\nu(\theta)$.

Exercise 1. *Assume the reward funtion is $R(a) = -(a-10)^2$. Derive the REINFORCE update for a policy parametrized as a normal distribution. Parametrise the mean and standard deviation separately. Run in PyTorch.*

2 U N I F I C A T I O N

Our main goal in this section is to unify some objective functions used in supervised learning and reinforcement learning. In regression, we saw that the mean-squared error objective comes from maximizing a log-likelihood. We also now know that the log-likelihood objective can be written as a cross-entropy objective. The maximum log-likelihood (MLL) objective is

$$-\frac{1}{t}\sum_{i=1}^{t} \log(p(Y_i \mid \theta)).$$

The cross-entropy (CE) objective, commonly used in classification, is

$$\begin{aligned}
\mathrm{CE}(p^*, p_\theta) &= -\int p^*(y) \log p(y \mid \theta)\, dy \\
&= \mathbb{E}[-\log p_\theta(Y \mid \theta)]
\end{aligned}$$

Therefore, we recover the MLL objective from the CE objective.

Let's try to write the policy gradient objective as a CE objective. Recall that the policy gradient objective is

$$\nu(\theta) = -\int \pi_\theta(a) q(a)\, da. \tag{1}$$

One way to frame what we are trying to accomplish is that we are trying to estimate the distribution of the optimal policy. Let π^* be an optimal policy (assuming existence, of course). Given that the CE objective is not symmetric, there are two possible objectives.

$$\text{CE}(\pi^*, \pi_\theta) = -\int \pi^*(a) \log \pi_\theta(a)\, da$$
$$\text{CE}(\pi_\theta, \pi^*) = -\int \pi_\theta(a) \log \pi^*(a)\, da$$

Which one should we choose? The first one is not a good candidate because we are not selecting actions according to the optimal policy at first. We do not have an oracle to give us π^*. Rather, data is generated based on the actions of an initially suboptimal agent. If we wish to form stochastic estimates of our objective, we better make sure that any integrals are over a policy we actually possess. This argument leaves us with the second objective.

Note that the second objective is somewhat similar to the policy gradient objective, where $\log \pi^*(a)$ is replaced with $q(a)$. How do we solidify this connection? One way is to to define "good" policy π^* via a Boltzmann distribution (softmax) over the (true) action values.

$$\pi^*(a) = Z^{-1} e^{q(a)\tau^{-1}}.$$

Z is a normalizing constant and τ is called the *temperature*, which controls the entropy of the distribution. Note that this policy is not necessarily the optimal policy (hence why we call it "good"), as there is still a non-zero chance of taking each action. Taking logs, we have

$$\log \pi^*(a) = \tau^{-1} q(a) - \log Z. \tag{2}$$

We can recover the policy gradient objective now. In particular, with this definition of π^*, we have

$$\begin{aligned} -\arg\min_\theta \int \pi_\theta(a) \log \pi^*(a)\, da &= -\arg\min_\theta \int \pi_\theta(a)(\tau^{-1} q(a) - \log Z)\, da \\ &= -\arg\min_\theta \int \pi_\theta(a) q(a)\, da. \end{aligned}$$

Let's now try to unify value-based and policy-based methods. We define the KL-divergence between two distributions:

$$\text{KL}(\pi_\theta, \pi^*) := \int \pi_\theta(a) \log \frac{\pi_\theta(a)}{\pi^*(a)}\, da. \tag{3}$$

Note that the KL-divergence is not symmetric. Taking gradients, we have

$$\begin{aligned} \nabla \text{KL}(\pi_\theta, \pi^*) &= \nabla \left(-\int \pi_\theta(a) \log \pi^*(a)\, da + \int \pi_\theta(a) \log \pi_\theta(a)\, da \right) \\ &= \nabla \text{CE}(\pi_\theta, \pi^*) - \nabla \text{Entropy}(\pi_\theta) \\ &= -\mathbb{E}[\nabla \log \pi_\theta(A_t)\, (R_t - \log \pi_\theta(A_t))]. \end{aligned} \tag{4}$$

As usual, we assume $\pi_\theta(a) > 0$ for all a in order to apply the REINFORCE trick. In particular, this assumption is satisfied if π_θ is parametrised as a Boltzmann distribution.

We have an additional $\log \pi_\theta(A_t)$ term in Equation (4), which is suggestive of entropy. If our policy parametrization is also Boltzmann, and we estimate action values with $\hat{q}_\theta$, we can substitute $\log \pi_\theta(A_t)$ (and an analagous equation for $\log \pi_\theta(A_t)$) from Equation (2) into Equation (4) to obtain

$$\nabla \text{KL}(\pi_\theta, \pi^*) = \mathbb{E}[(R_t - \tau^{-1} \hat{q}_\theta(A_t) + \log Z) \nabla \hat{q}_\theta(A_t)]. \tag{5}$$

This looks quite a bit like a Q-Learning update for bandits. Note that $-\tau^{-1} q(A_t)$ in Equation (5) is distinct from the baseline term in actor-critic, since the term that would be a baseline here is action-dependent.