Reinforcement Learning and Optimal Control

IFT6760C, Fall 2021

Pierre-Luc Bacon

October 27, 2021

Policy gradient methods

- You are given a class of parameterized (typically stationary) policies within Π^{MR}: ie. $d_\theta(a|s)$ where θ are parameters to learn
- Important: d_θ needs to be a differentiable function of θ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

Pros:

- Can leverage "structure" in policy space
  - Can provide prior knowledge about the kind of policies to consider
- Applies to continuous state and action spaces

Cons:

- Typically high variance
- Doesn't leverage structure in value space/DP results

Objective

Our goal is to:

$$\text{maximize} \;\; J(\theta) = \mathbb{E}_{\tau \sim p_\theta} \left[G(\tau) \right] \quad ,$$

where $G(\tau) = \sum_{t=1}^{T} r(S_t, A_t)$ and p_θ the distribution over trajectories induced by d_θ interacting with the MDP.

- We are facing a stochastic optimization problem with a distributional dependency and no structural component.

LR for RL

Applying a change of measure (the LR approach), we get:

$$\text{maximize} \quad J(\theta) = \mathbb{E}_{\tau \sim q}\left[G(\tau)\rho(\tau, \theta, q)\right] \quad ,$$

where $\rho(\tau, \theta, q) = p_\theta(\tau)/q(\tau)$ is the likelihood ratio.

- Consequence: we no longer have distributional dependency on θ: we pushed θ inside the expectation as structural parameters.

The gradient of J with respect to θ is:

$$DJ(\theta) = \mathbb{E}_{\tau \sim q}\left[G(\tau) D_2 \rho(\tau, \theta, q)\right] \quad .$$

The likelihood ratio is a Martingale

Let $\tau = (s_1, a_1, \ldots, s_T, a_T)$, the likelihood of a trajectory τ under d_θ is:

$$p(\tau;\theta) = p(s_1)\left(\prod_{t=1}^{T-1} d_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t)\right) d_\theta(a_T|s_T)$$

Therefore:

$$\frac{p(\tau;\theta)}{q(\tau)} = \prod_{t=1}^{T} \frac{d_\theta(a_t|s_t)}{d(a_t|s_t)} \quad .$$

where d is a given stationary policy in MR. The likelihood ratio is a Martingale, ie:

$$\mathbb{E}\left[\rho_{1:t+1} \mid \tau_{1:t}\right] = \mathbb{E}\left[\frac{d_\theta(A_{t+1}|S_{t+1})}{d(A_{t+1}|S_{t+1})} \,\middle|\, \tau_{1:t}\right] \rho_{1:t} = \rho_{1:t} \quad .$$

Using the Extended Conditional Monte Carlo Method

Using the law of total expectation, we can show that:

$$J(\theta) = \mathbb{E}_{\tau \sim q}\left[G(\tau)\rho(\tau, \theta, q)\right] = \mathbb{E}_{\tau \sim q}\left[\sum_{t=1}^{T} r(S_t, A_t)\mathbb{E}\left[\rho_{1:T} \mid \tau_{1:t}\right]\right]$$

And using the fact that the LR is a Martingale:

$$J(\theta) = \mathbb{E}_{\tau \sim d}\left[\sum_{t=1}^{T} r(S_t, A_t)\prod_{k=1}^{t}\frac{d_\theta(A_k|S_k)}{d(A_k|S_k)}\right] \quad .$$

LR + CMC + Martingale

We then get:

$$J(\theta) = \mathbb{E}_{\tau\sim d}\left[\sum_{t=1}^{T} r(S_t, A_t) \prod_{k=1}^{t} \frac{d_\theta(A_k|S_k)}{d(A_k|S_k)}\right] \quad .$$

If we pick the specific case $d_\theta = d$ as a sampling policy, we obtain the **score function** expression:

$$DJ(\theta) = \mathbb{E}_{\tau\sim d}\left[\sum_{t=1}^{T} r(S_t, A_t) \sum_{k=1}^{t} D_\theta \log d_\theta(A_k|S_k)\right] \quad .$$

SF + CMC + Martingale

The resulting estimator, call it $\hat{D}^{\triangledown}$ (mnemonic: lower triangular), taken over N trajectories is then:

$$\hat{D}^{\triangledown} J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{T} r_{i,j} \sum_{k=1}^{j} D_\theta \log d_\theta(a_{i,k} | s_{i,k})$$

where $r_{i,j}$ denotes the jth reward from the ith trajectory (same for $a_{i,k}$ and $s_{i,k}$). The inner most term can also be computed recursively:

$$\hat{D}^{\triangledown} J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{T} r_{i,j} z_{i,j}$$

$$z_{i,j} = D_\theta \log d_\theta(a_{i,j} | s_{i,j}) + z_{i,j-1} \quad .$$

and $z_{i,\bullet}$ is the eligibility trace for the ith trajectory.

SF + CMC + Martingale + change of bounds

We have:

$$DJ(\theta) = \mathbb{E}_{\tau\sim d}\left[\sum_{t=1}^{T} r(S_t, A_t) \sum_{k=1}^{t} D_\theta \log d_\theta(A_k|S_k)\right] .$$

The indices in the above expression are such that $1 \leq k \leq t \leq T$. Instead of taking $1 \leq t \leq T$ and $k \leq t \leq T$, we can use instead $1 \leq k \leq T$ and $k \leq t \leq T$. This gives us:

$$DJ(\theta) = \mathbb{E}_{\tau\sim d}\left[\sum_{t=1}^{T} D_\theta \log d_\theta(A_t|S_t) \sum_{k=t}^{T} r(S_k, A_k)\right] .$$

Estimator

The resulting (offline) estimator, call it $\hat{D}^{\triangle}$ (mnemonic: upper triangular) is then:

$$\hat{D}^{\triangle} J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{T} D_\theta \log d_\theta(a_{i,j}|s_{i,j}) \sum_{k=j}^{T} r(s_{i,k}, a_{i,k}) \quad .$$

This estimator is the most frequently encountered in modern deep RL and is typically implemented using the SAA perspective, that is, by defining a surrogate objective:

$$\hat{J}^{\triangle}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{T} \log d_\theta(a_{i,j}|s_{i,j}) \sum_{k=j}^{T} r(s_{i,k}, a_{i,k}) \quad ,$$

and the gradient $D\hat{J}^{\triangle}$ is the computed using automatic differentiation.