Chapter 11 – Policy Search

Author: Gianmarco Scarano

gianmarcoscarano@gmail.com

1. Introduction

Introducing this chapter, we can say a few words about the problems deriving from a tabular/parameterized state-action value function:

- If action space is large, very expensive computations
- States might appear identical in featurization
- Optimal policy through ϵ-greedy is not always enough

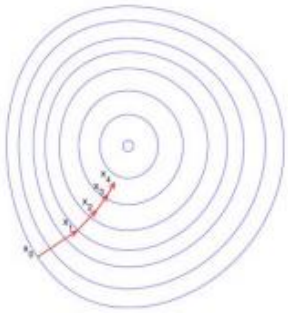
We can, then, directly represent and learn a parametrized policy:

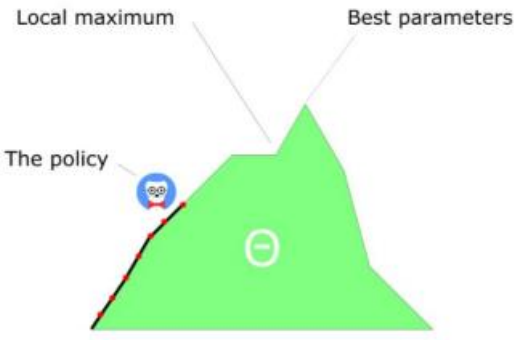
$$a \sim \pi_{\theta}(s)$$

In order to do so, we want to find the parameters θ such that π_θ is the best policy through gradient-free optimization such as Hill Climbing Simplex, Cross-Entropy method (CEM), Covariance Matrix Adaptation.

In this chapter, we'll discuss an efficient and widely used technique to find a parametrized policy: Policy Gradient.

2. Policy Gradient





This is guaranteed to converge to a local maximum or global minimum, but often converges only to a local maximum.

- **Policy:** $\pi_\theta(a|s)$
- **Performance function**: $J(\pi_\theta) = \mathbb{E}_{\pi_\theta}[\sum_{h=0}^{\infty} \gamma^h r_h]$
  - Considering the infinite setting
  - The expectation is done on our policy π_θ.
- **Parameters**: $\theta_{t+1} = \theta_t + \eta \cdot \nabla_\theta J(\pi_\theta)|_{\theta=\theta_t}$
  - Taking the gradient of the performance function J and take a step in the direction of the gradient.

The problem now is to compute the gradient of the performance function ($\nabla_\theta J(\pi_\theta)$), since it doesn't depend explicitly on θ, but on the state distribution that is generated by the policy ($\mathbb{E}_{\pi_\theta}$), hence depends on the policy π itself.

$$\frac{\partial V(s_0,\theta)}{\partial \theta_k} \approx \frac{V(s_0,\theta+\epsilon u_k) - V(s_0,\theta)}{\epsilon}$$

$$\begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow k\text{-th entry}$$

Uses n evaluations to compute policy gradient in n dimensions

Works for arbitrary policies, even if policy is not differentiable

The easiest way to overcome this problem is to use finite differences, where we have a variation ϵ on the θ vector.

In fact, it is given by the difference of the initial value, given $\theta + \epsilon \cdot u_k$ and the initial state given only θ.

The second way is to compute the gradient in an analytically way through differentiable policy classes.

2.1 Differentiable policy classes

In order to have an analytical gradient, we have to have a differentiable policy, such as the Softmax.

Given $\phi(s, a')$ as the feature vector and θ as the parameters, we can define 3 kinds of differentiable policy classes:

Softmax Linear Policy	Softmax Policy	Gaussian Policy
$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$	$\pi_\theta(a \mid s) = \frac{\exp(f_\theta(s,a))}{\sum_{a'} \exp(f_\theta(s,a'))}$	$\pi_\theta(a \mid s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(a - f_\theta(s))^2}{2\sigma^2}\right)$

Where f can be, for example, a Neural Network. Also, Softmax can be described as a classifier for discrete actions, while Gaussian policy can be described as a classifier for continuous action space.

2.2 Likelihood Ratio Policy Gradient

Now we know that, even if we can compute the gradient of the policy, the objective function J still doesn't depend on the policy.

For this reason, let's suppose we have a usual trajectory $\tau = \{s_0, a_0, s_1, a_1, \dots\}$ and its probability distribution, depending on θ, is:

$$p_\theta(\tau) = \mu(s_0) \cdot \pi_\theta(a_0|s_0) \cdot P(s_1|s_0, a_0) \cdot \pi_\theta(a_1, s_1) \dots$$

The problem here is that we included θ, but we don't have access to the transition function $P(s_{t+1}|s_t, a_t)$.

So, we can rewrite our objective (cost) function as a function of the trajectory, so:

$$J(\pi_\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \underbrace{\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \right]}_{R(\tau)} = \sum_\tau P(\tau;\theta) R(\tau)$$

probability of each trajectory from the distribution

We now want to maximize our J that now depends on the policy π, and it's equivalent to:

$$\arg\max_\theta J(\pi_\theta) = \arg\max_\theta \sum_\tau P(\tau;\theta) R(\tau)$$

The term on the right, now explicitly depends on the policy, so we can take directly the gradient.

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_\tau P(\tau;\theta) R(\tau) = \sum_\tau P(\tau;\theta) \nabla_\theta \log P(\tau;\theta) R(\tau)$$

Let's do an additional step, which is taking out the P (which is the expectation) and calculate it approximately by taking several trajectories (m), collecting the sum of the rewards, and dividing by the sum of the trajectories that I have:

$$\approx \quad \hat{g} = (1/m) \sum_{i=1}^{m} R(\tau^{(i)}) \nabla_\theta \log P(\tau^{(i)};\theta)$$

The last thing that we might want to compute is the logarithm of P. Actually, we know that that object includes a transition function inside, but do we actually need it? The answer is no.

By making P explicit, which we know is

$$p_\theta(\tau) = \mu(s_0) \cdot \pi_\theta(a_0|s_0) \cdot P(s_1|s_0, a_0) \cdot \pi_\theta(a_1, s_1) \dots$$

and by taking the logarithm version of it, then, we can infer that P and μ don't depend on θ, so we can cancel them out, remaining with:

$$= \sum_{t=0}^{T-1} \underbrace{\nabla_\theta \log \pi_\theta(a_t|s_t)}_{\text{no dynamics model required!}}$$

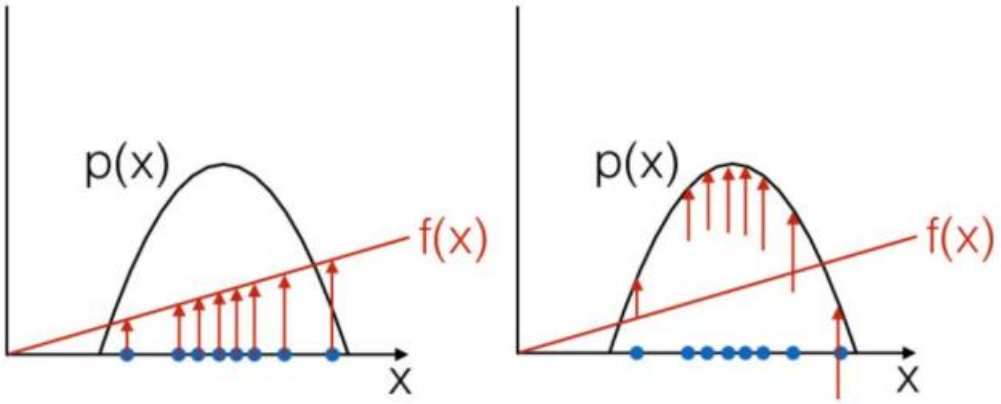
Called **score function**

$$= (1/m) \sum_{i=1}^{m} \boxed{R(\tau^{(i)})} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)})$$

Measure how good the sample trajectory is

That parameter $R(\tau^{(i)})$ will let me move in the direction of the gradient of the policy. Visualizing this:

$$\hat{g}_i = f(x_i) \nabla_\theta \log p(x_i|\theta)$$



This means that we are actually moving in the direction in such a way that we increase the log-probability of a certain sample in proportion to how good it is ($f(x_i)$). We are doing it in such a way that we are increasing the likelihood of sampling a certain trajectory which has a high reward.

$$\nabla_\theta J(\pi_\theta) = \frac{1}{1-\gamma} \mathbb{E}_{s,a \sim d_\mu^{\pi_\theta}} \left[\nabla_\theta \ln \pi_\theta(a|s) \cdot Q^{\pi_\theta}(s,a) \right]$$

$$\nabla_\theta J(\pi_\theta) = \sum_{h=0}^{H-1} \mathbb{E}_{s_h, a_h \sim \mathbb{P}_h^{\pi_\theta}} \left[\nabla \ln \pi_\theta(a_h|s_h) \cdot Q_h^{\pi_\theta}(s_h, a_h) \right]$$

This result can be generalized in the **Policy Gradient Theorem**, which is nothing else than replacing the sum of the rewards with $\frac{1}{1-\gamma}$. We replace $R(\tau^{(i)})$ with Q since $R(\tau^{(i)})$ it's a specific sample of the value function. If we have to do this in general, we must take Q.

The second row refers to the finite setting, where we simply do the summation up to the horizon H.