

Policy Gradient



Farnaz Adib Yaghmaie

Linköping University, Sweden
farnaz.adib.yaghmaie@liu.se

March 12, 2021

What is Policy Gradient?

The most ambitious way of solving an RL problem

- Policy: The agent's decision
- Value function: how good the agent does in a state
- Model: The agent's interpretation of the environment

Agent's goal: To learn the policy by directly optimizing the total reward.

$$J = \mathbf{E}_{\tau \sim \pi_{\theta}}[R(T)]$$

How?

Optimization by perturbation:

- Consider a stochastic parametric policy with parameter θ
- Observe the total reward as a result of perturbation
- Optimize the parameters of the policy by finding $\nabla_{\theta} J$

A simple math rule based on $\nabla_p \log p = \frac{1}{p}$

$$\nabla_{\theta} \log p = \nabla_p \log p \nabla_{\theta} p = \frac{1}{p} \nabla_{\theta} p.$$

$$\boxed{\nabla_{\theta} p = p \nabla_{\theta} \log p} \tag{1}$$

We will have a closer look at the following components in J

$$J = \mathbf{E}_{\tau \sim \pi_{\theta}}[\mathbf{R}(T)]$$

- π_{θ} : The parametric pdf of the policy
- τ : A sampled trajectory and the expectation is defined over the probability of the trajectory
- $\mathbf{R}(T)$: The total reward

Why to consider a stochastic policy instead of a deterministic one?

- To enable learning by deviating from the deterministic policy
- If a deterministic policy is considered, the agents remains in a local optimum forever.

How to define the pdf?

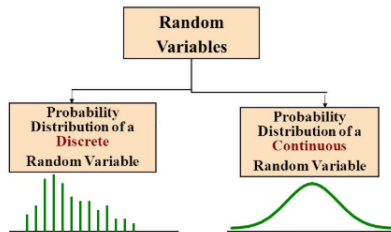
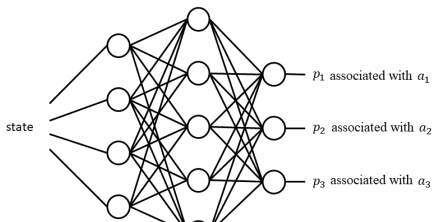


Photo Credit: @ <https://towardsdatascience.com/probability-distributions-discrete-and-continuous-7a94ede66dc0>

Generates the pdf $\pi_{\theta} = \text{network}(s)$



```
network = keras.Sequential([  
    keras.layers.Dense(30, input_dim=n_s, activation='relu'),  
    keras.layers.Dense(30, activation='relu'),  
    keras.layers.Dense(n_a, activation='softmax')])
```


Continuous action space

Select a Gaussian distribution as the pdf π_θ and generate the mean

$$\pi_\theta = \frac{1}{\sqrt{(2\pi\sigma^2)^{n_a}}} \exp\left[-\frac{1}{2\sigma^2}(a - \mu_\theta(s))^T(a - \mu_\theta(s))\right]$$

For example, for a linear policy

$$\mu_\theta(s) = \theta s$$

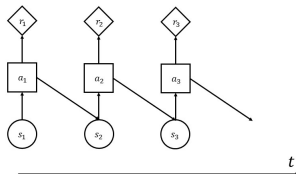
Discrete:

Parameterize the pdf

Continuous:

Parameterize the mean of a
Gaussian pdf

Sampling a trajectory



Select $a_t \sim \pi_\theta$, $t = 1, \dots, T$ and step the environment

$$\tau = (s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_{T+1})$$

The probability of the trajectory

$$P(\tau|\theta) = \prod_{t=1}^T p(s_{t+1}|s_t, a_t)p(a_t|\theta).$$

- $p(s_{t+1}|s_t, a_t)$: the model of the environment
- $p(a_t|\theta)$: The pdf π_θ evaluated at a_t .
 - Discrete action space: $\pi_\theta = \text{network}(s)$. So, $p(a_t|\theta)$ is obtained by indexing into the output vector $\text{network}(\text{state})$.
 - Continuous action space:

$$p(a_t|\theta) = \frac{1}{\sqrt{(2\pi\sigma^2)^{n_a}}} \exp\left[-\frac{1}{2\sigma^2}(a_t - \mu_\theta(s_t))^T(a_t - \mu_\theta(s_t))\right]$$

Love math? Dive in!

$$\begin{aligned}
 \nabla_{\theta} J &= \nabla_{\theta} \mathbf{E}[R(T)] \\
 &= \int_{\tau} \nabla_{\theta} P(\tau|\theta) R(T) \\
 &= \int_{\tau} P(\tau|\theta) \nabla_{\theta} \log P(\tau|\theta) R(T), \quad \text{using log-derivative trick} \\
 &= \mathbf{E}_{\tau \sim \pi_{\theta}} [\nabla_{\theta} \log P(\tau|\theta) R(T)] \\
 &= \mathbf{E}_{\tau \sim \pi_{\theta}} \left[\left(\nabla_{\theta} \sum_{t=1}^T \log \underbrace{p(s_{t+1}|s_t, a_t)}_{\text{Dynamics}} + \nabla_{\theta} \sum_{t=1}^T \log p(a_t|\theta) \right) R(T) \right] \\
 &= \mathbf{E}_{\tau \sim \pi_{\theta}} [\nabla_{\theta} \sum_{t=1}^T \log p(a_t|\theta) R(T)]
 \end{aligned} \tag{2}$$

But how to compute?

Use pre-built cost functions for the classification task in ML.

Policy Gradient:

$$J = \sum_{t=1}^T R(T) \mathbf{1} \log p(a_t | \theta)$$

- Number of actions n_a
- Trajectory length T
- Weight $R(T)$
- State s_t
- Target label for state s_t and action a
- $p(a_t | \theta)$: probability of a_t by the network at state s_t

Classification:

$$J_{\text{wcec}} = -\frac{1}{M} \sum_{m=1}^M \sum_{c=1}^C w_c y_m^c \log h_{\theta}(x_m, c)$$

- Number of classes C
- Data length M
- Weight w_c
- Image x_m
- Target label for x_m for class c ; y_m^c
- $h_{\theta}(x_m, c)$: The probability of x_m belongs to class c by the network

Summary of optimizing the parameters in the discrete action space

PG is similar to the classification task!

- The network should produce probability
- The cost to be optimized is a weighted cross entropy cost
- The weights are $R(T)$

We can compute $\mathbf{E}_{\tau \sim \pi_\theta} [\nabla_\theta \sum_{t=1}^T \log p(a_t | \theta) R(T)]$ easily!

$$\nabla_\theta J = \frac{1}{\sigma^2 |\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=1}^T (a_t - \mu_\theta(s_t)) \frac{d\mu_\theta(s_t)}{d\theta}^\dagger R(T).$$

If we consider a linear policy $\mu_\theta(s_t) = \theta s_t$

$$\nabla_\theta J = \frac{1}{\sigma^2 |\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=1}^T (a_t - \theta s_t) s_t^\dagger R(T).$$

Discrete:

Assign a cross entropy cost function and let the ML library optimize the parameter!

Continuous:

Use

$$\nabla_{\theta} J = \frac{1}{\sigma^2 |\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=1}^T (a_t - \theta s_t) s_t^{\dagger} R(T)$$

and optimize θ with a gradient algorithm, e.g.

$$\theta = \theta + \alpha \nabla_{\theta} J.$$

Putting all together

We build/consider a parametric pdf $\pi_{\theta}(s)$. Then, we iterate:

1 Collect data

- Observe s and sample $a \sim \pi_{\theta}(s)$.
- Apply a and observe r .
- Add s , a , r to the history.

2 Update the parameter θ

- We calculate the total reward.
- We optimize the policy by a gradient algorithm.

PG: The most ambitious way of solving an RL problem

- Directly optimizes the reward for MDP
- No model, no bellman equation
- Random search is a special case
- Can be extremely good or bad
- Take a look at implementation on my github

`Crash_course_on_RL/pg_notebook.ipynb`

Email your questions to

farnaz.adib.yaghmaie@liu.se