Reinforcement Learning

Chapter 5: RL via Policy Gradient

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

University of Toronto

Fall 2025

Policy Network

$$\mathbf{x}(s, a) \longrightarrow \boxed{\pi_{\boldsymbol{\theta}}(\cdot)} \longrightarrow \pi(a|s)$$

Policy networks are used in two sets of deep RL approaches

- *Policy gradient approaches*
  - ↳ *We do not use a value network and directly approximate optimal policy*
- *Actor-critic approaches*
  - ↳ *We keep the value network to examine the approximated optimal policy*
  - ↳ *This is the most practically-robust approach we can use*

Policy Network

Policy Network

Policy network is an approximation model that maps state-action features to a conditional probability distribution

$$\mathbf{x}\left(s,a\right) \longrightarrow \boxed{\pi_{\boldsymbol{\theta}}\left(\cdot\right)} \longrightarrow \pi\left(a|s\right)$$

+ *How can we realize such a network? It is not any network! It should return probabilities!*
– Yes! That's right! Let's see a few examples

Recall: *Feature*

Feature Representation of State-Actions

Feature representation maps each state-action pair into a vector of features that correspond to that state and action, i.e.,

$$\mathbf{x}\left(\cdot\right) : \mathbb{S} \times \mathbb{A} \mapsto \mathbb{R}^J$$

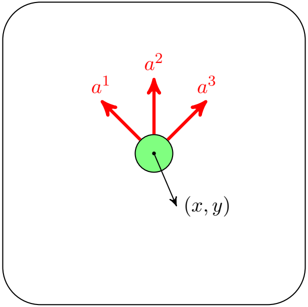
for some integer J that is the feature dimension

Attention

Note that are now in the most general case: states and actions can be either discrete or continuous

Example: *Moving Particle*

We are controlling a moving particle that could move in the 2D space

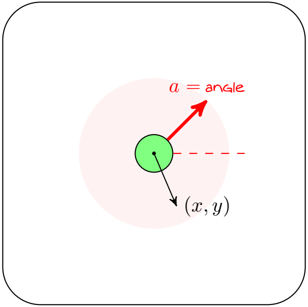


We can set the feature vector

$$\mathbf{x}(s, a) = \begin{bmatrix} x \\ y \\ a \end{bmatrix}$$

Example: *Moving Particle*

We have the same moving particle that could move in any direction



We can set the feature vector

$$\mathbf{x}\left(s, a\right) = \begin{bmatrix} x \\ y \\ a \end{bmatrix}$$

New Notation

+ *Shall we see now an example of a policy network?*
– Sure! Just last point to mention before

New Notation

As we think of a generic action and state space, we use a simple notation

$$\int_a f(a) = \begin{cases} \displaystyle\sum_{a \in \mathbb{A}} f(a) & \textit{discrete } a \\ \displaystyle\int_{\mathbb{A}} f(a)\,\mathrm{d}a & \textit{continuous } a \end{cases}$$

Example: *Softmax*

The most basic example is to assume a linear mapping

$$\pi_{\boldsymbol{\theta}}(a|s) = \boldsymbol{\theta}^{\mathsf{T}} \mathbf{x}(s, a)$$

+ *But how can we guarantee that it returns a probability?! Shall we assume*

$$\int_a \pi_{\boldsymbol{\theta}}(a|s) = \int_a \boldsymbol{\theta}^{\mathsf{T}} \mathbf{x}(s, a) = 1$$

– *Well! We can do that, but there is a better way to convert a linear function into a probability distribution*

Example: *Softmax*

Softmax

Softmax is a vector-activated neuron that maps input $\mathbf{x}(s, a)$ *into*

$$\mathit{Soft}_{\max}^{\boldsymbol{\theta}}(\mathbf{x}(s, a)) = \frac{\exp\left\{\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}(s, a)\right\}}{\displaystyle\int_{a} \exp\left\{\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}(s, a)\right\}}$$

We can now simply set

$$\pi_{\boldsymbol{\theta}}(a|s) = \mathit{Soft}_{\max}^{\boldsymbol{\theta}}(\mathbf{x}(s, a))$$

As we are going to have

$$\int_{a} \pi_{\boldsymbol{\theta}}(a|s) = \int_{a} \mathit{Soft}_{\max}^{\boldsymbol{\theta}}(\mathbf{x}(s, a)) = 1$$

Example: *Gaussian*

Another approach is to use a Gaussian policy that is controllable with some parameters: say at state s we only look at the state representation $\mathbf{x}(s)$

$$\begin{aligned}\pi_{\boldsymbol{\theta}}(a|s) &\equiv \mathcal{N}\left(\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}(s), \sigma^2\right) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{\left(a - \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}(s)\right)^2}{2\sigma^2}\right\}\end{aligned}$$

We may train this network by

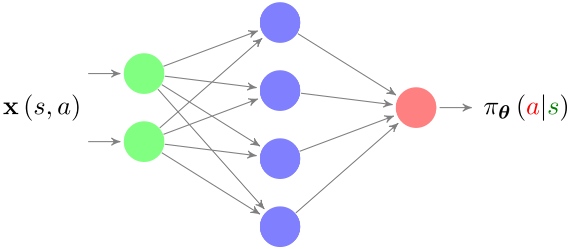
- *either only learning $\boldsymbol{\theta}$*
- *or learning both $\boldsymbol{\theta}$ and σ^2*

Example: *DPN*

In practice, we are more interested to train

Deep Policy Network $\equiv$ *DPN*

as it can learn a richer class of policies



And we very well know how to make it return a probability distribution!

Training Policy Network

Let's now train the policy network: *assume a general network as*

$$\mathbf{x}(s,a) \longrightarrow \boxed{\pi_{\boldsymbol{\theta}}(\cdot)} \longrightarrow \pi(a|s)$$

+ *How can we train it? What should be the loss?*
– *Well! We know what we want?*

> *We want ro have a policy that maximizes value at all states, i.e.,*
>
> $$\boldsymbol{\theta}^\star = \operatorname*{argmax}_{\boldsymbol{\theta}} v_{\pi_{\boldsymbol{\theta}}}(s)$$
>
> *for all states* $s \in \mathbb{S}$

Since we are more happy with minimization we can alternatively say ☺

$$\boldsymbol{\theta}^\star = \operatorname*{argmin}_{\boldsymbol{\theta}} -v_{\pi_{\boldsymbol{\theta}}}(s)$$

Training Policy Network

+ *But that is weird! We have so many states! For which one we should do it?!*
- That's right! We should find a way around it

This naive training reduces to a multi-objective optimization

$$\boldsymbol{\theta}^\star = \operatorname*{argmin}_{\boldsymbol{\theta}} -v_{\pi_{\boldsymbol{\theta}}}\left(s\right)$$

with the number of objectives being as much as the number of states!

Say we have N states: we need to have simultaneously

$$\boldsymbol{\theta}^\star = \operatorname*{argmin}_{\boldsymbol{\theta}} -v_{\pi_{\boldsymbol{\theta}}}\left(s^1\right) \qquad \dots \qquad \boldsymbol{\theta}^\star = \operatorname*{argmin}_{\boldsymbol{\theta}} -v_{\pi_{\boldsymbol{\theta}}}\left(s^N\right)$$

which is not necessarily possible!

Finding Loss

A classical remedy to such multi-objective optimization is to scalarize

$$\boldsymbol{\theta}^\star = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} -p\left(s^1\right) v_{\pi_{\boldsymbol{\theta}}}\left(s^1\right) - \ldots - p\left(s^N\right) v_{\pi_{\boldsymbol{\theta}}}\left(s^N\right)$$

Or better to say: to minimize the average return over all states, i.e.,

$$\begin{aligned}\mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) &= \int_s v_{\pi_{\boldsymbol{\theta}}}\left(s\right) p\left(s\right) \\ &= \mathbb{E}_{S\sim p}\left\{v_{\pi_{\boldsymbol{\theta}}}\left(S\right)\right\}\end{aligned}$$

+ *OK! But what is $p\left(s\right)$?! Do we have it? Or shall we assume it?*
– *Neither and both* ☺ *Let's try a simple setting first*

Finding Loss

Let's consider a simple case: *we have an episodic environment whose a sample trajectory looks like*

$$\tau : S_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T$$

We denote the whole trajectory by τ to keep our notation simple

---

Assume we have no discount; then, we could say that a sample return is

$$G_0 = \sum_{t=0}^{T-1} R_{t+1}$$

and that the value for sample state S_0

$$v_{\pi_{\boldsymbol{\theta}}}(S_0) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \{G_0 | S_0\}$$

Finding Loss

Say we fix our starting state to $S_0 = s_0$: *we get a sample trajectory as*

$$\tau\left(s_0\right): s_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T$$

The value of the starting state s_0 is then given by

$$\begin{aligned}
v_{\pi_{\boldsymbol{\theta}}}\left(s_0\right) &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left\{G_0 \middle| s_0\right\} \\
&= \int\limits_{r_1,\ldots,r_T} \int\limits_{s_1,\ldots,s_T} \int\limits_{a_0,\ldots,a_{T-1}} \left(\sum_{t=0}^{T-1} r_{t+1}\right) \pi_{\boldsymbol{\theta}}\left(a_0 \middle| s_0\right) p\left(s_1, r_1 \middle| s_0, a_0\right) \\
&\qquad \ldots \left(a_{T-1} \middle| s_{T-1}\right) p\left(s_T, r_T \middle| s_{T-1}, a_{T-1}\right) \\
&= \int\limits_{\tau(s_0)} \left(\sum_{t=0}^{T-1} r_{t+1}\right) \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}\left(a_t \middle| s_t\right) p\left(s_{t+1}, r_{t+1} \middle| s_t, a_t\right)
\end{aligned}$$

Finding Loss

Say we fix our starting state to $S_0 = s_0$: *we get a sample trajectory as*

$$\tau(s_0): s_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T$$

Let's define the return of this trajectory as

$$\mathrm{g}(\tau(s_0)) = \sum_{t=0}^{T-1} r_{t+1}$$

This an outcome of random variable

$$G(\tau(s_0)) = \sum_{t=0}^{T-1} R_{t+1}$$

We can now write

$$v_{\pi_{\boldsymbol{\theta}}}(s_0) = \int_{\tau(s_0)} \mathrm{g}(\tau(s_0)) \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}(a_t | s_t) \, p(s_{t+1}, r_{t+1} | s_t, a_t)$$

Finding Loss

Say we fix our starting state to $S_0 = s_0$: *we get a sample trajectory as*

$$\tau\left(s_0\right): s_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T$$

Note that we could look at this term as an expectation

$$\begin{aligned} v_{\pi_{\boldsymbol{\theta}}}\left(s_0\right) &= \int\limits_{\tau(s_0)} \mathrm{g}\left(\tau\left(s_0\right)\right) \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}\left(a_t | s_t\right) p\left(s_{t+1}, r_{t+1} | s_t, a_t\right) \\ &= \mathbb{E}_{\tau(s_0) \sim \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}(a_t|s_t) p(s_{t+1}, r_{t+1}|s_t, a_t)} \left\{G\left(\tau\left(s_0\right)\right)\right\} \end{aligned}$$

Initial Conclusion

Distribution of $\tau\left(s_0\right)$ for a given s_0 which includes all next states is specified by policy and environment

Finding Loss

Now, let's assume a randomly chosen starting state S_0: *then, we have*

$$\tau : S_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T$$

We choose it with some distribution $p(s_0)$; *thus, we have*

$$\begin{aligned}
\mathcal{J}(\pi_{\boldsymbol{\theta}}) &= \mathbb{E}_{S_0 \sim p}\left\{ v_{\pi_{\boldsymbol{\theta}}}(S_0) \right\} \\
&= \int_{s_0} \int_{\tau(s_0)} \mathrm{g}(\tau(s_0))\, p(s_0) \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}(a_t | s_t)\, p(s_{t+1}, r_{t+1} | s_t, a_t) \\
&= \int_{\underset{\tau}{\uparrow}} \mathrm{g}(\tau)\, p(s_0) \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}(a_t | s_t)\, p(s_{t+1}, r_{t+1} | s_t, a_t)
\end{aligned}$$

average over all possible trajectories

Finding Loss: *Estimating Form*

Now, let's define the overall distribution of trajectory τ as

$$p_{\pi_{\boldsymbol{\theta}}}\left(\tau\right) = p\left(s_0\right) \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}\left(a_t | s_t\right) p\left(s_{t+1}, r_{t+1} | s_t, a_t\right)$$

Then we could compute the average return of the environment as

$$\mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) = \int_{\tau} \mathrm{g}\left(\tau\right) p_{\pi_{\boldsymbol{\theta}}}\left(\tau\right)$$
$$= \underbrace{\mathbb{E}_{\tau \sim p_{\pi_{\boldsymbol{\theta}}}}}_{\text{distribution of trajectory}} \left\{ \overbrace{G\left(\tau\right)}^{\text{return of trajectory}} \right\}$$

Final Conclusion

Part of distribution of τ is assumed and remaining by policy and environment

Training Policy Network: *Gradient Descent*

We have the loss ready: *let's start training the policy network*

- + *What do you mean by training?*
- – *Simply, we want to find the network parameters that minimize the loss*

$$\boldsymbol{\theta}^\star = \operatorname*{argmin}_{\boldsymbol{\theta}} -\mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right)$$

We can use gradient descent: we consider learning rate α and update as

$$\begin{aligned}\boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} - \alpha\nabla\left\{-\mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right)\right\}\\ &\leftarrow \boldsymbol{\theta} + \alpha\nabla\mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right)\end{aligned}$$

So, we need to compute $\nabla\mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right)$ with respect to $\boldsymbol{\theta}$

Training Policy Network: *Gradient Descent*

We are using gradient descent (ascent)

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right)$$

and we need the gradient: so, we write

$$\begin{aligned}\nabla \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) &= \nabla \int_{\tau} \mathrm{g}\left(\tau\right) p_{\pi_{\boldsymbol{\theta}}}\left(\tau\right) = \int_{\tau} \mathrm{g}\left(\tau\right) \nabla p_{\pi_{\boldsymbol{\theta}}}\left(\tau\right) \\ &= \int_{\tau} \mathrm{g}\left(\tau\right) \nabla \left\{ \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}\left(a_t | s_t\right) p\left(s_{t+1}, r_{t+1} | s_t, a_t\right) p\left(s_0\right) \right\}\end{aligned}$$

+ *It looks challenging!*
- *Let's take a deeper look*

Training Policy Network: *Gradient Descent*

There is a trick that might help us in this respect

the so-called log-derivative trick

Log-Derivative Trick

For any positive function $f(\cdot) : \mathbb{R}^J \mapsto \mathbb{R}_+$ *we have by definition*

$$\nabla f(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) \nabla \log f(\boldsymbol{\theta})$$

Let's apply the log-derivative trick to our problem

Training Policy Network: *Gradient Descent*

Applying the log-derivative trick to our problem, we have

$$\nabla \mathcal{J}(\pi_{\boldsymbol{\theta}}) = \int_{\tau} \mathrm{g}(\tau) \nabla p_{\pi_{\boldsymbol{\theta}}}(\tau)$$

$$= \int_{\tau} \mathrm{g}(\tau) p_{\pi_{\boldsymbol{\theta}}}(\tau) \nabla \log p_{\boldsymbol{\theta}}(\tau)$$

$$= \int_{\tau} \left[\mathrm{g}(\tau) \nabla \log p_{\pi_{\boldsymbol{\theta}}}(\tau)\right] p_{\pi_{\boldsymbol{\theta}}}(\tau) = \mathbb{E}_{\tau \sim p_{\pi_{\boldsymbol{\theta}}}} \left\{ G(\tau) \nabla \log p_{\pi_{\boldsymbol{\theta}}}(\tau) \right\}$$

+ *Why should that be helpful?!*
– *Let's see how* $\log p_{\pi_{\boldsymbol{\theta}}}(\tau)$ *looks*

Training Policy Network: *Gradient Descent*

Consider one instant trajectory: *we have a particular outcome*

$$\tau : s_0, a_0 \xrightarrow{r_1} s_1, a_1 \xrightarrow{r_2} \cdots \xrightarrow{r_{T-1}} s_{T-1}, a_{T-1} \xrightarrow{r_T} s_T$$

Using the definition of $p_{\boldsymbol{\theta}}(\tau)$, we can write

$$\log p_{\pi_{\boldsymbol{\theta}}}(\tau) = \log \left\{ p(s_0) \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}(a_t | s_t)\, p(s_{t+1}, r_{t+1} | s_t, a_t) \right\}$$

$$= \underbrace{\log p(s_0)}_{\text{does not depend in } \boldsymbol{\theta}} + \sum_{t=0}^{T-1} \log \pi_{\boldsymbol{\theta}}(a_t | s_t) + \underbrace{\sum_{t=0}^{T-1} \log p(s_{t+1}, r_{t+1} | s_t, a_t)}_{\text{does not depend in } \boldsymbol{\theta}}$$

Training Policy Network: *Gradient Descent*

Consider one instant trajectory: *we have a particular outcome*

$$\tau : s_0, a_0 \xrightarrow{r_1} s_1, a_1 \xrightarrow{r_2} \cdots \xrightarrow{r_{T-1}} s_{T-1}, a_{T-1} \xrightarrow{r_T} s_T$$

The gradient of $\log p_{\boldsymbol{\theta}}(\tau)$ *is hence given by*

$$\nabla \log p_{\pi_{\boldsymbol{\theta}}}(\tau) = \nabla \sum_{t=0}^{T-1} \log \pi_{\boldsymbol{\theta}}(a_t | s_t) = \sum_{t=0}^{T-1} \nabla \log \pi_{\boldsymbol{\theta}}(a_t | s_t)$$

If we have a random sample trajectory

$$\tau : S_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T$$

we can similarly write

$$\nabla \log p_{\pi_{\boldsymbol{\theta}}}(\tau) = \sum_{t=0}^{T-1} \nabla \log \pi_{\boldsymbol{\theta}}(A_t | S_t)$$

Training Policy Network: *Gradient Descent*

Back to our main problem: *we have a random trajectory*

$$\tau : S_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T$$

and want to find the gradient of loss; so, we can write

$$\begin{aligned}
\nabla \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) &= \mathbb{E}_{\tau \sim p_{\pi_{\boldsymbol{\theta}}}} \left\{ G\left(\tau\right) \nabla \log p_{\pi_{\boldsymbol{\theta}}}\left(\tau\right) \right\} \\
&= \mathbb{E}_{\tau \sim p_{\pi_{\boldsymbol{\theta}}}} \left\{ G\left(\tau\right) \sum_{t=0}^{T-1} \nabla \log \pi_{\boldsymbol{\theta}}\left(A_t | S_t\right) \right\} \\
&= \mathbb{E}_{\tau \sim p_{\pi_{\boldsymbol{\theta}}}} \left\{ \left(\sum_{t=0}^{T-1} R_{t+1} \right) \sum_{t=0}^{T-1} \nabla \log \pi_{\boldsymbol{\theta}}\left(A_t | S_t\right) \right\}
\end{aligned}$$

We can estimate it via Monte-Carlo!

Training Policy Network: *SGD*

Say we set the weights of policy network to $\boldsymbol{\theta}$: *we sample K trajectories*

$$\tau : S_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T$$

from the environment using policy $\pi_{\boldsymbol{\theta}}$, and then estimate the gradient as

$$\hat{\nabla} \mathcal{J}(\pi_{\boldsymbol{\theta}}) = \frac{1}{K} \sum_{k=1}^{K} G(\tau_k) \sum_{t=0}^{T-1} \nabla \log \pi_{\boldsymbol{\theta}} (A_t [k] | S_t [k])$$

We can then use gradient descent to update $\boldsymbol{\theta}$ as

$$\begin{aligned} \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \alpha \hat{\nabla} \mathcal{J}(\pi_{\boldsymbol{\theta}}) \\ &\leftarrow \boldsymbol{\theta} + \frac{\alpha}{K} \sum_{k=1}^{K} G(\tau_k) \sum_{t=0}^{T-1} \nabla \log \pi_{\boldsymbol{\theta}} (A_t [k] | S_t [k]) \end{aligned}$$

Training Policy Network: *SGD*

+ *Isn't that again too slow?! We should wait for a single update!*
- Sure! We can go for SGD

Using SGD, we could take a single sample gradient

$$\hat{\nabla}\mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) = G\left(\tau\right)\sum_{t=0}^{T-1}\nabla\log\pi_{\boldsymbol{\theta}}\left(A_t|S_t\right)$$

and then update the policy network as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha G\left(\tau\right)\sum_{t=0}^{T-1}\nabla\log\pi_{\boldsymbol{\theta}}\left(A_t|S_t\right)$$

First Policy Gradient Algorithm

```
PG_v1():
```

1: *Initiate with* $\boldsymbol{\theta}$ *and learning rate* α
2: ***for*** *episode* $= 1 : K$ ***do***
3: *Sample a trajectory with policy* $\pi_{\boldsymbol{\theta}}$

$$\tau : S_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T$$

4: *Compute return* $G(\tau)$
5: ***for*** $t = 0 : T-1$ ***do***
6: *Update policy network* $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha G(\tau) \nabla \log \pi_{\boldsymbol{\theta}}(A_t | S_t)$
7: ***end for***
8: ***end for***

+ *Is it a kind of known algorithm?*
– With a bit of modification *it reduces to REINFORCE algorithm proposed by Ronald J. Williams in 1992*

REINFORCE: *First Official Algorithm*

```
REINFORCE():
```

1: *Initiate with* $\boldsymbol{\theta}$ *and learning rate* α
2: ***for*** *episode* $= 1 : K$ ***do***
3: *Sample a trajectory with policy* $\pi_{\boldsymbol{\theta}}$

$$\tau : S_0, A_0 \xrightarrow{R_1} S_1, A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{T-1}} S_{T-1}, A_{T-1} \xrightarrow{R_T} S_T$$

4: ***for*** $t = 0 : T-1$ ***do***
5: *Update policy network* $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha G_t \nabla \log \pi_{\boldsymbol{\theta}} (A_t | S_t)$
6: ***end for***
7: ***end for***

+ *But we are now computing a different gradient? Why should it work?!*
- This is because of the *Policy Gradient Theorem which says that we should update proportional to* $\nabla \log \pi_{\boldsymbol{\theta}} (A_t | S_t)$

SGD: *General Setting*

Let's have a more generic analysis: *assume we start at a random state S_0 that is chosen according to*

$$S_0 \sim p\left(s_0\right)$$

We start acting via the policy $\pi_{\boldsymbol{\theta}}$ and transit to a new state

$$S_0, A_0 \xrightarrow{R_1} S_1$$

We could then say that the average value of the policy is

$$\mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) = \mathbb{E}_{S_0 \sim p}\left\{v_{\pi_{\boldsymbol{\theta}}}\left(S_0\right)\right\}$$

We need the gradient of this value against $\boldsymbol{\theta}$ to train the policy network

SGD: *General Setting*

We can open up the loss expression

$$\mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) = \int_{s_0} v_{\pi_{\boldsymbol{\theta}}}\left(s_0\right) p\left(s_0\right)$$

and write the gradient as

$$\begin{aligned}\nabla \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) &= \nabla \int_{s} v_{\pi_{\boldsymbol{\theta}}}\left(s_0\right) p\left(s_0\right) \\ &= \int_{s} \nabla v_{\pi_{\boldsymbol{\theta}}}\left(s_0\right) p\left(s_0\right)\end{aligned}$$

Let's compute $\nabla v_{\pi_{\boldsymbol{\theta}}}\left(s_0\right)$

SGD: *General Setting*

We can use the marginalization rule to expand $v_{\pi_{\boldsymbol{\theta}}}(s_0)$

$$v_{\pi_{\boldsymbol{\theta}}}(s_0) = \int_{a_0} q_{\pi_{\boldsymbol{\theta}}}(s_0, a_0)\, \pi_{\boldsymbol{\theta}}(a_0|s_0)$$

So the gradient $\nabla v_{\pi_{\boldsymbol{\theta}}}(s_0)$ *is computed using chain rule as*

$$\begin{aligned}
\nabla v_{\pi_{\boldsymbol{\theta}}}(s_0) &= \nabla \int_{a_0} q_{\pi_{\boldsymbol{\theta}}}(s_0, a_0)\, \pi_{\boldsymbol{\theta}}(a_0|s_0) \\
&= \int_{a_0} \nabla q_{\pi_{\boldsymbol{\theta}}}(s_0, a_0)\, \pi_{\boldsymbol{\theta}}(a_0|s_0) + \int_{a_0} q_{\pi_{\boldsymbol{\theta}}}(s_0, a_0)\, \nabla \pi_{\boldsymbol{\theta}}(a_0|s_0)
\end{aligned}$$

Let's compute $\nabla q_{\pi_{\boldsymbol{\theta}}}(s_0, a_0)$ *next*

SGD: *General Setting*

We can use Bellman equation to expand $q_{\pi_{\boldsymbol{\theta}}}(s_0, a_0)$ as

$$q_{\pi_{\boldsymbol{\theta}}}(s_0, a_0) = \mathcal{R}(s_0, a_0) + \gamma \int_{s_1} v_{\pi_{\boldsymbol{\theta}}}(s_1)\, p(s_1|s_0, a_0)$$

So the gradient reads

$$\begin{aligned}
\nabla q_{\pi_{\boldsymbol{\theta}}}(s_0, a_0) &= \nabla \left\{ \mathcal{R}(s_0, a_0) + \gamma \int_{s_1} v_{\pi_{\boldsymbol{\theta}}}(s_1)\, p(s_1|s_0, a_0) \right\} \\
&= \underbrace{\nabla \mathcal{R}(s_0, a_0)}_{0} + \gamma \int_{s_1} \nabla v_{\pi_{\boldsymbol{\theta}}}(s_1)\, p(s_1|s_0, a_0) \\
&= \gamma \int_{s_1} \nabla v_{\pi_{\boldsymbol{\theta}}}(s_1)\, p(s_1|s_0, a_0)
\end{aligned}$$

SGD: *General Setting*

Now, let's put back all gradients gradually towards beginning of computation

$$\nabla \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) = \int_{s_0} \nabla v_{\pi_{\boldsymbol{\theta}}}\left(s_0\right) p\left(s_0\right)$$

$$= \int_{s_1} \nabla v_{\pi_{\boldsymbol{\theta}}}\left(s_1\right) p_{\pi_{\boldsymbol{\theta}}}\left(s_1\right) + \int_{s_0}\int_{a_0} q_{\pi_{\boldsymbol{\theta}}}\left(s_0, a_0\right) \nabla \pi_{\boldsymbol{\theta}}\left(a_0 | s_0\right) p\left(s_0\right)$$

where we define the marginal distribution of s_1 as

$$p_{\pi_{\boldsymbol{\theta}}}\left(s_1\right) = \int_{s_0}\int_{a_0} p\left(s_1 | s_0, a_0\right) \pi_{\boldsymbol{\theta}}\left(a_0 | s_0\right) p\left(s_0\right)$$

SGD: *General Setting*

Since $p_{\pi_{\boldsymbol{\theta}}}(s_1)$ *and* $p(s_0)$ *are distributions, we have*

$$\int_{s_1} p_{\pi_{\boldsymbol{\theta}}}(s_1) = \int_{s_0} p(s_0) = 1$$

So, we could modify our final expression as

$$\begin{aligned}
\nabla \mathcal{J}(\pi_{\boldsymbol{\theta}}) &= \int_{s_1} \nabla v_{\pi_{\boldsymbol{\theta}}}(s_1)\, p_{\pi_{\boldsymbol{\theta}}}(s_1) \int_{s_0} p(s_0) \\
&\quad + \int_{s_0}\int_{a_0} q_{\pi_{\boldsymbol{\theta}}}(s_0, a_0)\, \nabla \pi_{\boldsymbol{\theta}}(a_0|s_0)\, p(s_0) \int_{s_1} p_{\pi_{\boldsymbol{\theta}}}(s_1) \\
&= \int_{s_1}\int_{s_0} p_{\pi_{\boldsymbol{\theta}}}(s_1)\, p(s_0) \left[\nabla v_{\pi_{\boldsymbol{\theta}}}(s_1) + \int_{a_0} q_{\pi_{\boldsymbol{\theta}}}(s_0, a_0)\, \nabla \pi_{\boldsymbol{\theta}}(a_0|s_0) \right]
\end{aligned}$$

SGD: *General Setting*

If we keep on progressing in the trajectory as $t \to \infty$, we will see

$$\nabla \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) = \int_s d_{\pi_{\boldsymbol{\theta}}}\left(s\right) \int_a q_{\pi_{\boldsymbol{\theta}}}\left(s,a\right) \nabla \pi_{\boldsymbol{\theta}}\left(a|s\right)$$

for some distribution $d_{\pi_{\boldsymbol{\theta}}}\left(s\right)$ that is the average marginal distribution of states under policy $\pi_{\boldsymbol{\theta}}$, i.e.,

$$\int_s d_{\pi_{\boldsymbol{\theta}}}\left(s\right) = 1$$

Finally, using the log-derivative trick we have

$$\nabla \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) = \int_s d_{\pi_{\boldsymbol{\theta}}}\left(s\right) \int_a q_{\pi_{\boldsymbol{\theta}}}\left(s,a\right) \pi_{\boldsymbol{\theta}}\left(a|s\right) \nabla \log \pi_{\boldsymbol{\theta}}\left(a|s\right)$$

Policy Gradient Theorem

This can be equivalently written as

$$\nabla \mathcal{J}(\pi_{\boldsymbol{\theta}}) = \int_s \int_a d_{\pi_{\boldsymbol{\theta}}}(s)\, \pi_{\boldsymbol{\theta}}(a|s) q_{\pi_{\boldsymbol{\theta}}}(s,a) \nabla \log \pi_{\boldsymbol{\theta}}(a|s)$$
$$= \mathbb{E}_{S\sim d_{\pi_{\boldsymbol{\theta}}}, A|S\sim \pi_{\boldsymbol{\theta}}} \{ q_{\pi_{\boldsymbol{\theta}}}(S,A) \nabla \log \pi_{\boldsymbol{\theta}}(A|S) \}$$

which concludes the policy gradient theorem proved by Sutton et al. in 1992

Policy Gradient Theorem

For a policy network with non-zero probabilities, the gradient of the average trajectory return is always given by

$$\nabla \mathcal{J}(\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{S\sim d_{\pi_{\boldsymbol{\theta}}}, A|S\sim \pi_{\boldsymbol{\theta}}} \{ q_{\pi_{\boldsymbol{\theta}}}(S,A) \nabla \log \pi_{\boldsymbol{\theta}}(A|S) \}$$

Policy Gradient Theorem: *Implication*

Policy Gradient Theorem

For a policy network with non-zero probabilities, the gradient of the average trajectory return is always given by

$$\nabla \mathcal{J}\left(\pi_{\boldsymbol{\theta}}\right) = \mathbb{E}_{S \sim d_{\pi_{\boldsymbol{\theta}}}, A|S \sim \pi_{\boldsymbol{\theta}}} \left\{ q_{\pi_{\boldsymbol{\theta}}}\left(S, A\right) \nabla \log \pi_{\boldsymbol{\theta}}\left(A|S\right) \right\}$$

+ *OK! That sounds nice! But what is special about it?!*
– It says *to train a policy network, you only need gradient of log likelihood*
+ *Then what?!*
– *Well! We could have much more complicated terms! We will talk about it more in the next sections*