# Reinforcement Learning - Policy Gradient Methods

Amir Globerson

Reinforcement learning describes what is arguably the most general machine learning setup: an agent acting interacting with its environment. We shall focus on the setting of a Markov Decision Process which makes several assumptions about the probabilistic properties of the environment.

Below are the components of such a model:

- A set of world states $\mathcal{S}$. The set of world states over time is a sequence of random variables $S_0, \ldots, S_t$.
- A set of actions the agent can take $\mathcal{A}$. The set of actions the agent takes are the random variables $A_0, \ldots, A_t$.
- State transition distribution. Describes the probability that the state will change to $s'$ given that it was $s$ and the agent took action $a$. Namely:
$$p(s'|s,a) = \mathbb{P}\left[S_{t+1} = s'|S_t = s, A_t = a\right] \tag{1}$$
We shall assume that $S_{t+1}$ is independent of previous actions and states given $S_t, _t$. Namely:
$$\mathbb{P}\left[S_{t+1}|S_0, \ldots, S_t, A_0, \ldots, A_t\right] = \mathbb{P}\left[S_{t+1}|S_t, A_t\right] \tag{2}$$
- A reward function specifying the reward an agent gets for taking action $a$ in state $s$. Denote by $r(s,a)$.
- A policy $\pi : \mathcal{A} \to \mathcal{S}$ defined which action the agent takes given its state. In some cases we may consider a stochastic policy where the agent samples an action given a state. In that case we will have $\pi : \mathcal{A}, \mathcal{S} \to \mathbb{R}^+$ so that:
$$\pi(a|s) = \mathbb{P}\left[A_{t+1} = a|S_{t+1} = s\right] \tag{3}$$

An instantiation of this process is a set of random variables for states $S_0, \ldots, S_t$ actions $A_0, \ldots, A_t$ and rewards $R_0, \ldots, R_t$. The goal of the learning process is to find a good policy. There are various ways of measuring the quality of a hypothesis. Since a reward is given at each time $t$ we want to take all of them into account but weight them so the sum does not diverge. The common way of doing this is via a discount parameter $\gamma > 0$. The cumulative reward of a policy is then:

$$\sum_{t=0}^{\infty} \gamma^t R_t \tag{4}$$

## 1 Model Free Policy Learning

Our eventual gual is to learn a good policy. Namely, a policy which maximizes our expected reward. One way to do this is to learn all the parameters specifying the MDP (e.g., transition probabilities, reward function etc) and learn the best policy for those. This would be called a model based approach. On the other hand, we can take a more "discriminative" approach, and try to directly learn a policy without worrying about learning the model. This will be called a model free approach, and is our focus in these notes.

## 2 Policy Gradient

When the state is a continuous vector, we cannot store the policy $\pi(s)$ as a table of values. In such cases, it makes sense to parameterize the policy as a function of the state. Here we focus on stochastic policies $\pi(a|s)$. Namely, we assume a conditional distribution defined via parameters $\boldsymbol{\theta}$.

$$\pi_\theta(a|s) = f(a, s, \boldsymbol{\theta}) \tag{5}$$

Here $f$ can be a complex neural net. For example it can be a deep net taking $s$ as input with a softmax layer for the output $a$ (assuming $a$ is discrete).

We focus on the finite horizon case with cumulative reward:

$$G(S_{0:T}, A_{0:T}) = \sum_{t=0}^{T} r(S_t, A_t) \tag{6}$$

Where we use $S_{0:T}, A_{0:T}$ to denote the sequence of actions and states.

The expected cumulative reward is then:

$$L(\boldsymbol{\theta}) = \mathbb{E}\left[G(S_{0:T}, A_{0:T})\right] = \mathbb{E}\left[\sum_{t=0}^{T} r(S_t, A_t)\right] \tag{7}$$

And we would like to calculate its gradient so we can do gradient descent on $L(\boldsymbol{\theta})$. There's a nice trick for doing this, as we show below. The source of randomness is $S, A$. So we can write (denoting $v = (s_{0:T}, a_{0:T})$ for a complete sequence of action and state:)

$$L(\boldsymbol{\theta}) = \sum_{s_{0:T}, a_{0:T}} p_{\boldsymbol{\theta}}(s_{0:T}, a_{0:T}) G(s_{0:T}, a_{0:T}) = \sum_{v} p_{\boldsymbol{\theta}}(v) G(v) \tag{8}$$

where we used $p_{\boldsymbol{\theta}}$ to denote the fact that $p$ depends on the parameters.

### 2.1 The Likelihood Ratio Approach

We next describe an elegant trick for estimating the gradient of the reward with respect to the policy parameters. The resulting methods is known as *"likelihood ratio policy gradient"*.

Let's write the gradient of $L(\boldsymbol{\theta})$ and express it a bit differently:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) &= \sum_v G(v) \nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(v) \\ &= \sum_v p_{\boldsymbol{\theta}}(v) G(v) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(v) \\ &= \mathbb{E}\left[G(V) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(V)\right]\end{aligned}$$

Now note that due to Markovity:

$$p_\theta(v) = p(S_{0:T}, a_{0:T}) = p(s_0) \prod_{t=1}^{T} p(s_t | s_{t-1}, a_{t-1}) \prod_{t=0}^{T} \pi_{\boldsymbol{\theta}}(a_t | s_t) \tag{9}$$

Take the log and denote all elements not dependent on $\theta$ by $c$. Then:

$$\log p_\theta(v) = c + \sum_{t=0}^{T} \log \pi_{\boldsymbol{\theta}}(a_t | s_t) \tag{10}$$

So we can rewrite Equation 9 as:

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \mathbb{E}\left[G(S_{0:T}, A_{0:T}) \sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t | S_t)\right] \tag{11}$$

We now have an expression of the gradient as an expected value over samples of $S_{0:T}, A_{0:T}$. We can therefore estimate it using empirical average, and use it within gradient descent as follows:

- Use current parameter $\boldsymbol{\theta}_t$ to sample $n$ sequences of state and action. Denote these by $s_0^i, \ldots, s_T^i, a_0^i, \ldots, a_T^i$ for $i = 1, \ldots, n$.
- Estimate the gradient wrt $\boldsymbol{\theta}_t$ using an empirical average to approximate the expectation in Equation 11. Namely:

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \approx \boldsymbol{\delta}_t = \frac{1}{n} \sum_{i=1}^{n} G(s_{0:T}^i, a_{0:T}^i) \sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t^i | s_t^i) \tag{12}$$

- Perform gradient ascent step: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta \boldsymbol{\delta}_t$. Of course you can use fancier gradient descent methods like AdaGrad, Adam etc.

As $n$ grows we will of course get a more stable estimate of the gradient, and generally there will be variance around the true value of the gradient. We next describe a method for reducing this variance.

## 2.2 The REINFORCE Algorithm

In 1992 Williams [2] proposed the class of REINFORCE algorithms. These are based on the likelihood ratio trick, but with additional important variance reducing components. To develop these we first notice that for any parameter $\theta_i$ and any $b_i \in \mathbb{R}$, the identity in Equation 9 can be extended as follows:

$$\nabla_{\theta_i} L(\boldsymbol{\theta}) = \mathbb{E}\left[(G(V) - b_i)\,\nabla_{\theta_i}\log p_{\boldsymbol{\theta}}(V)\right] \tag{13}$$

Namely, we are setting a "baseline" value for the reward, and the baseline differs per parameter. To see why Equation 13 is true, we just need to show that the added term due to $b_i$ is zero. This is true since:

$$\mathbb{E}\left[b_i \nabla_{\theta_i}\log p_{\boldsymbol{\theta}}(V)\right] = b_i \sum_v \nabla_{\theta_i} p_{\boldsymbol{\theta}}(v) = b_i \nabla_{\theta_i} \sum_v p_{\theta}(v) = b_i \nabla_{\theta_i} 1 = 0 \tag{14}$$

where we used the fact that $p_{\boldsymbol{\theta}}(v)$ is a distribution and therefore normalizes to 1.

We can now use Equation 13 to estimate the gradient as before. But the current expression is more general, as we can always set $b_i = 0$ to arrive at the previous one. We would like to set $b$ such that the variance of the estimator is minimized. More precisely, define the random variable:

$$Z = (G(V) - b_i)\,\nabla_{\theta_i}\log p_{\boldsymbol{\theta}}(V) \tag{15}$$

Then $\nabla_{\theta_i} L(\boldsymbol{\theta}) = \mathbb{E}[Z]$ and we will estimate it by sampling $Z_1, \ldots, Z_n$ and calculating $\frac{1}{n}\sum_i Z_i$. The variance of the estimate is therefore $\frac{\sigma_Z^2}{n}$ where $\sigma_Z^2$ is the variance of $Z$. We therefore would like to set $b_i$ to minimize this variance.

$$\sigma_Z^2 = \mathbb{E}\left[Z^2\right] - \mathbb{E}^2[Z] \tag{16}$$

As we saw above, the expected value of $Z$ does not depend on $b_i$, and therefore only the first term above depends on $b_i$. We therefore have (dropping the index $i$ for now)

$$0 = \nabla_{b_i}\sigma_Z^2 = \nabla_{b_i}\mathbb{E}\left[Z^2\right] = \mathbb{E}\left[(G(V) - b_i)\left(\nabla_{\theta_i}\log p_{\boldsymbol{\theta}}(V)\right)^2\right] \tag{17}$$

So that the optimal $b_i$ is:

$$b_i = \frac{\mathbb{E}\left[G(V)\left(\nabla_{\theta_i}\log p_{\boldsymbol{\theta}}(V)\right)^2\right]}{\mathbb{E}\left[\left(\nabla_{\theta_i}\log p_{\boldsymbol{\theta}}(V)\right)^2\right]} \tag{18}$$

Luckily this depends only on $\nabla_{\theta_i}\log p_{\boldsymbol{\theta}}(V)$ which can be calculated from samples of the policy as in Equation 10. The resulting estimate is based on $n$ state-action sequences as before (namely $s_0^i, \ldots, s_T^i, a_0^i, \ldots, a_T^i$), and given by:

$$b_i \approx \frac{\sum_{i=1}^n G(s_{0:T}^i, a_{0:T}^i)\left(\sum_{t=0}^T \nabla_{\theta_i}\log \pi_{\boldsymbol{\theta}}(a_t^i|s_t^i)\right)^2}{\sum_{i=1}^n \left(\sum_{t=0}^T \nabla_{\theta_i}\log \pi_{\boldsymbol{\theta}}(a_t^i|s_t^i)\right)^2} \tag{19}$$

## 2.3 Further Variance Reduction - GOMDP and Policy Gradient Theorem

Variance can be reduced further by noticing that our expected gradient has some redundant elements. To see this write Equation 11 as:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) &= \mathbb{E}\left[\left(\sum_{k=0}^{T} r(S_k, A_k)\right)\sum_{t=0}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(A_t|S_t)\right] \\ &= \mathbb{E}\left[(\sum_{k,t=0}^{T} r(S_k, A_k)\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(A_t|S_t)\right]\end{aligned}$$

We now note that not all these $k, t$ terms are needed as some are in fact zero in expectation. To see this consider the case $k < t$ (i.e., we are looking at a reward given before time $t$). Then:

$$\mathbb{E}\left[r(S_k, A_k)\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(A_t|S_t)\right] = \sum_{s_k,a_k,s_t,a_t} p_{\boldsymbol{\theta}}(s_k)\pi_{\boldsymbol{\theta}}(a_k|s_k)p_{\boldsymbol{\theta}}(s_t|s_k)\pi_{\boldsymbol{\theta}}(a_t|s_t)\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t|s_t)r(s_k, a_k) \tag{20}$$

where we have taken care to denote whatever probabilities are dependent on $\boldsymbol{\theta}$. Now using the derivative of log we get that this equals:

$$\begin{aligned}&\sum_{s_k,a_k,s_t,a_t} p_{\boldsymbol{\theta}}(s_k)\pi_{\boldsymbol{\theta}}(a_k|s_k)p_{\boldsymbol{\theta}}(s_t|s_k)\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(a_t|s_t)r(s_k, a_k) \\ = &\sum_{s_k,a_k,s_t} p_{\boldsymbol{\theta}}(s_k)\pi_{\boldsymbol{\theta}}(a_k|s_k)p_{\boldsymbol{\theta}}(s_t|s_k)\nabla_{\boldsymbol{\theta}}\sum_{a_t}\pi_{\boldsymbol{\theta}}(a_t|s_t) = 0\end{aligned}$$

We conclude that we can drop all these terms. So our expression simplifies to:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) &= \mathbb{E}\left[\left(\sum_{k=0}^{T} r(S_k, A_k)\right)\sum_{t=0}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(A_t|S_t)\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(A_t|S_t)\sum_{k=t}^{T} r(S_k, A_k)\right]\end{aligned}$$

The sum over $k$ may be interpreated as the future reward given that we start at state $S_k$ and take action $A_k$. Thus, it intuitively can be replaced with $Q(S_k, A_k)$. This is indeed true and is known as the Policy Gradient Theorem [1] . The nice implication of this is that we can use various methods for estimating $Q$ and then plug these into a policy gradient method.

## References

[1] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063, 1999.

[2] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.