Probability Concepts and its Application in POLICY GRADIENT

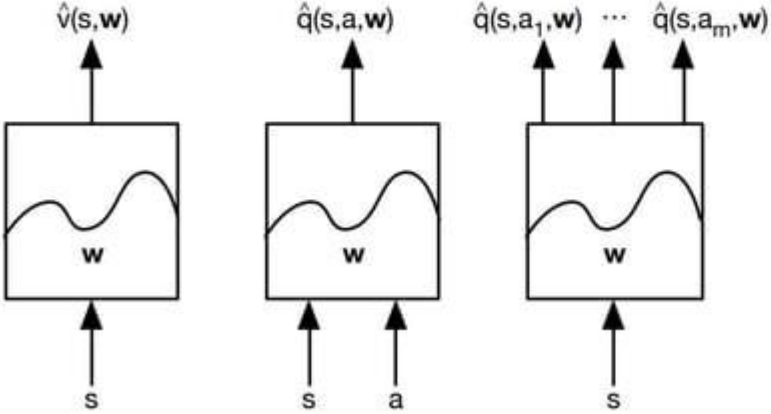
DR. SRIPARNA SAHA
CSE DEPT, IIT PATNA

Outline

- Pitfall of Value-based Reinforcement Learning
- Policy gradient
- Variance reduction
- Policy in policy gradient
- Off-policy policy gradient
- Reference

Value-based Reinforcement Learning

In previous lecture, we introduce how to use neural network to approximate value function and how to learn the optimal policy in discrete action space.



Value-based Reinforcement Learning

In Deep Q Network, we use neural network to approximate the action value function Q(s, a).

$$Q_\theta(s,a) \approx Q^\pi(s,a)$$

The greedy policy is

$$a = \underset{a}{argmax} Q_\theta(s,a)$$

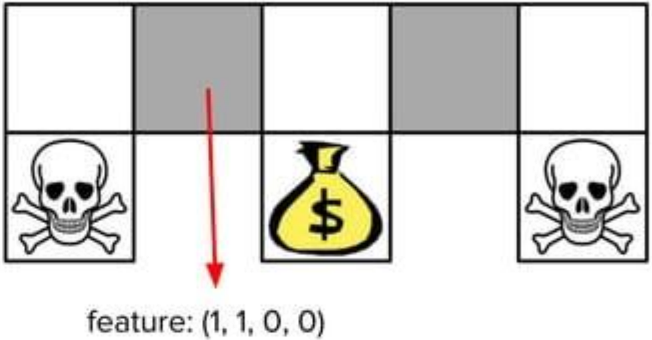
Value-based Reinforcement Learning

- The optimal policy learned from value-based method is **deterministic**
- It's hard to be applied in continuous action problem

$$\pi_*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_{a \in \mathcal{A}} q_*(s, a) \\ 0, & \text{otherwise.} \end{cases}$$

Pitfall of Value-based Reinforcement Learning

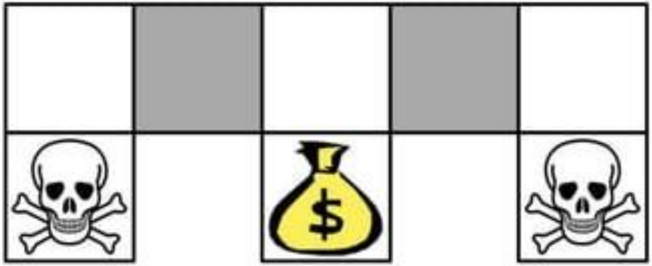
Consider the following simple maze, the features are constructed by 4 elements and each element means whether facing the wall in that direction (N, S, W, E).



Pitfall of Value-based Reinforcement Learning

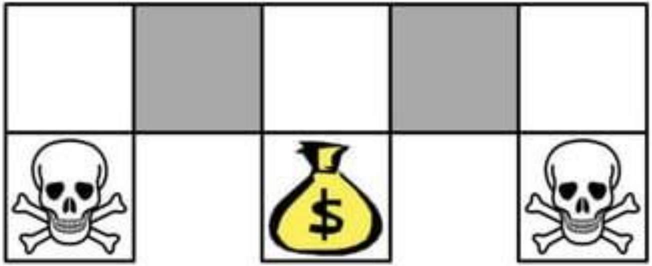
For deterministic policy:

- It will move east either or west in **both** grey states.
- It may get stuck and never reach the goal state.



Pitfall of Value-based Reinforcement Learning

Although well-defined observation could help the agent to distinguish the difference in different states, sometimes we prefer to use **stochastic** policy.



Pitfall of Value-based Reinforcement Learning

In robotics, the action(control) is often continuous. We need to decide the degree/torque in the robotic arm given observation.

It's hard to use **argmax** to demonstrate the optimal action of robotic arm, so we need other solutions in continuous control problem.

$$a = \underset{a}{argmax} Q_\theta(s, a)$$

Value-based and Policy-based RL

Value-Based

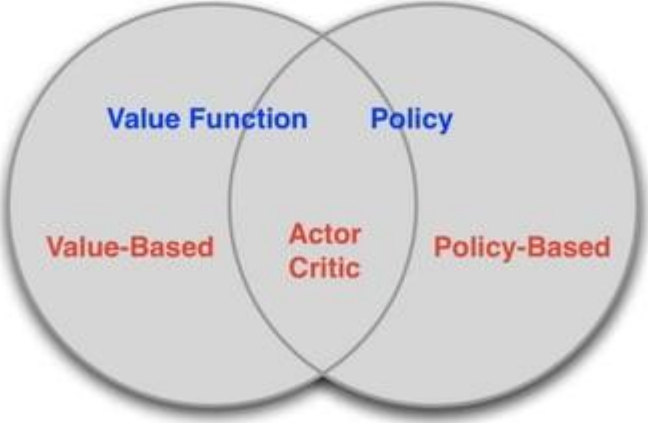
- Learnt Value Function
- Implicit policy

Policy-Based

- **No Value Function**
- **Learnt Policy**

Actor-Critic

- Learnt Value Function
- Learnt Policy



Policy Gradient

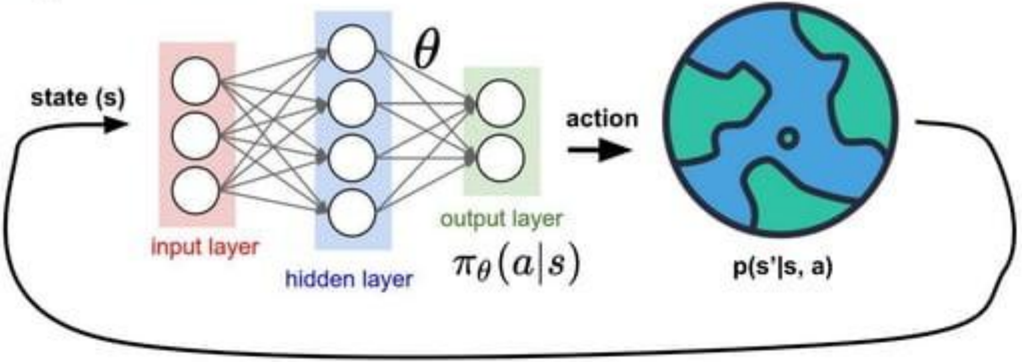
The objective of reinforcement learning is to maximize the expected episodic rewards (here, we take episodic task as our example)

$$J(\theta) = E[\sum_t^T r(s_t, a_t)]$$

We define $\sum_t^T r(s_t, a_t)$ as $r(\tau)$ and τ is represented for the episodic trajectory, the objective can be expressed as following:

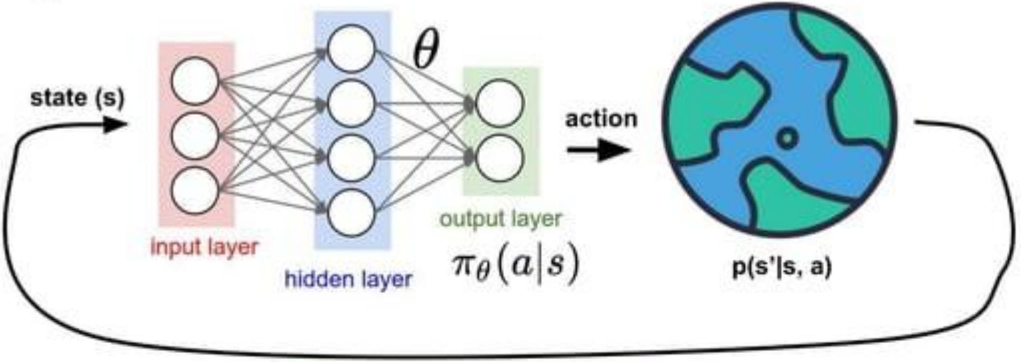
$$J(\theta) = E[r(\tau)]$$

Policy Gradient



$$\underbrace{p_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

Policy Gradient



$$\boxed{J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[r(\tau)]}$$

In this example, the sample is a episodic trajectory not the experience for each transition (s, a, r, s')

Policy Gradient

How to do we find the optimal parameters of neural network?

$$\theta^* = \underset{\theta}{argmax}\, E_{\tau \sim \pi_\theta(\tau)}[r(\tau)] \text{ ???}$$

Policy Gradient

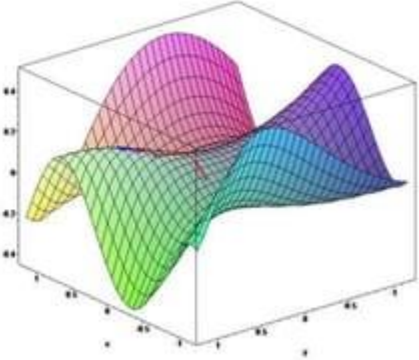
How to do we find the optimal parameters of neural network?

$$\xcancel{\theta^* = \underset{\theta}{argmax}\, E_{\tau\sim\pi_\theta(\tau)}[r(\tau)]}$$

Gradient Ascent ! (maximize objective)

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[r(\tau)]$$

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$



*** Math Caution ***

Policy Gradient

$$J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[r(\tau)] = \int \pi_\theta(\tau) r(\tau) d\tau$$

$$r(\tau) \rightarrow \sum_{t=1}^{T} r(s_t, a_t)$$

tips:

$$\underline{\nabla_\theta \pi_\theta(\tau)} = \pi_\theta \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \underline{\pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau)}$$

Policy Gradient

$$J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[r(\tau)] = \int \pi_\theta(\tau) r(\tau) d\tau$$

$$\sum_{t=1}^{T} r(s_t, a_t)$$

tips:

$$\underline{\nabla_\theta \pi_\theta(\tau)} = \pi_\theta \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \underline{\pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau)}$$

$$\nabla_\theta J(\theta) = \int \underline{\nabla_\theta \pi_\theta(\tau)} r(\tau) d\tau = \int \underline{\pi_\theta(\tau) \nabla_\theta log \pi_\theta(\tau)} r(\tau) d\tau$$

$$= E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

Policy Gradient

$$J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[r(\tau)] = \int \pi_\theta(\tau) r(\tau) d\tau$$

$$\sum_{t=1}^{T} r(s_t, a_t)$$

tips:

$$\underline{\nabla_\theta \pi_\theta(\tau)} = \pi_\theta \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \underline{\pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau)}$$

$$\nabla_\theta J(\theta) = \int \underline{\nabla_\theta \pi_\theta(\tau)} r(\tau) d\tau = \int \underline{\pi_\theta(\tau) \nabla_\theta log \pi_\theta(\tau)} r(\tau) d\tau$$

$$= E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

We just sample trajectories using current policy and adjust the likelihood of trajectories by episodic rewards.

Policy Gradient

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau = \int \pi_\theta(\tau) \nabla_\theta log \pi_\theta(\tau) r(\tau) d\tau$$

$$= E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \boxed{\pi_\theta(\tau)} r(\tau)]$$

$$\boxed{\underbrace{\pi_\theta(s_1, a_1, s_2, a_2, ..., s_T, a_T)}_{\pi_\theta(\tau)}} = p(s1) \prod_{t=1}^{T} \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t)$$

Policy Gradient

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau = \int \pi_\theta(\tau) \nabla_\theta log \pi_\theta(\tau) r(\tau) d\tau$$

$$= E_{\tau \sim \pi_\theta(\tau)}[\underline{\nabla_\theta \log \pi_\theta(\tau)} r(\tau)]$$

$$\log \pi_\theta(\tau) = \log p(s_1) + \sum_{t=1}^{T} (\log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t))$$

$$\underline{\nabla_\theta \log \pi_\theta(\tau)} = \nabla_\theta \left[\cancel{\log p(s_1)} + \sum_{t=1}^{T} (\log \pi_\theta(a_t|s_t) + \cancel{\log p(s_{t+1}|s_t, a_t)}) \right]$$

Policy Gradient

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau = \int \pi_\theta(\tau) \nabla_\theta log \pi_\theta(\tau) r(\tau) d\tau$$

$$= E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \underbrace{\log \pi_\theta(\tau)}_{\sum_{t=1}^{T} \log \pi_\theta(a_t|s_t)} \underbrace{r(\tau)}_{\sum_{t=1}^{T} r(s_t, a_t)}]$$

Policy Gradient

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau = \int \pi_\theta(\tau) \nabla_\theta log \pi_\theta(\tau) r(\tau) d\tau$$

$$= E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

$$= E_{\tau \sim \pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)\left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right]$$

Policy Gradient

The gradient of objective:

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \underline{\nabla_\theta \log \pi_\theta(a_t|s_t)}\right)\left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right]$$

Adjust the action probability taken in that trajectory.

According to the magnitude of episodic rewards

Policy Gradient

The gradient of objective:

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)\left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right]$$

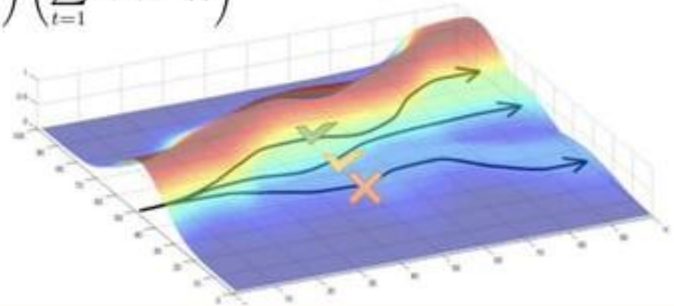
In practice, we replace expectation by sampling multiple trajectories.

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})\right)$$

Vanilla Policy Gradient - REINFORCE algorithm

REINFORCE algorithm:

1. sample trajectory τ^i from $\pi_\theta(a_t|s_t)$
2. $$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)$$
3. $$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$



Features of Policy-Based RL

Advantages

- Better convergence
- Effective in high-dimensional or continuous action space
- Stochastic policy

Disadvantages:

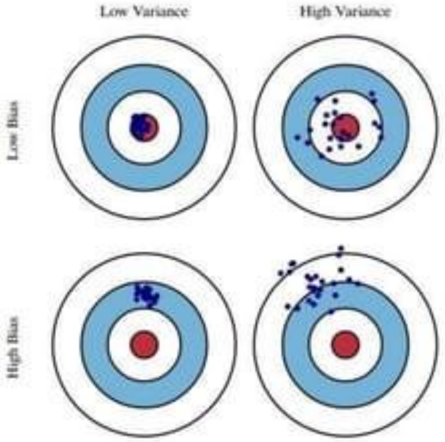
- Local optimal rather than global optimal
- Inefficient learning (learned by episodes)
- High variance

REINFORCE: bias and variance

The estimator is unbiased, because it use true rewards to evaluate policy.

The estimator of REINFORCE is known to have high variance because of huge difference in episodic rewards. High variance results in slow convergence.

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\boxed{\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)}$$



Variance reduction

There are two method to reduce variance

1. Causality
2. Baseline

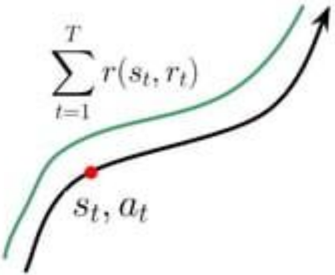
$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\boxed{\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)}$$

Variance reduction: causality

Original:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)$$

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\sum_{t'=1}^{T} r(s_{i,t'},a_{i,t'})\right]$$



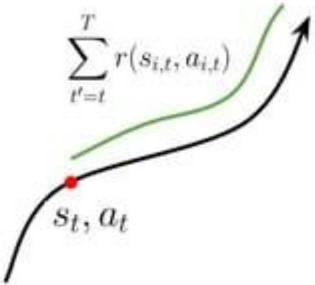
Variance reduction: causality

Original:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)$$

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\sum_{t'=1}^{T} r(s_{i,t'},a_{i,t'})\right]$$

Causality: policy at time t' cannot affect reward at time t when t < t'



Variance reduction: causality

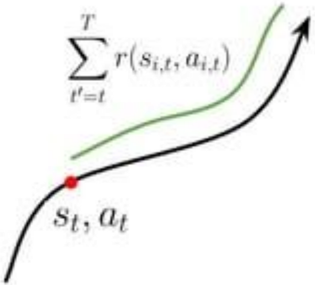
Original:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)$$

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\sum_{t'=1}^{T} r(s_{i,t'},a_{i,t'})\right]$$

Causality: policy at time t' cannot affect reward at time t when t < t'

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\widehat{Q}(s_{i,t},a_{i,t})\right]$$



Variance reduction: causality

Original:

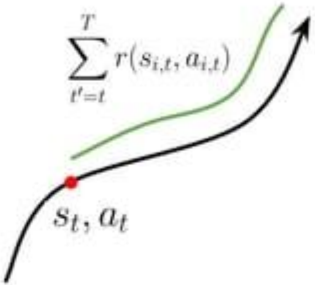
$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)$$

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\sum_{t'=1}^{T} r(s_{i,t'},a_{i,t'})\right]$$

Causality: policy at time t' cannot affect reward at time t when t < t'

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\widehat{Q}(s_{i,t},a_{i,t})\right]$$

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\sum_{t'=t}^{T} r(s_{i,t'},a_{i,t'})\right]$$



REINFORCE: reduce variance

There are two method to reduce variance

Original: $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$

1. Causality:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'}) \right]$$

2. Baseline

Variance reduction: baseline

baseline:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau)[r(\tau) - \boxed{b}]$$

Variance reduction: baseline

baseline:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau)[r(\tau) - \boxed{b}]$$

we can choose baseline like:

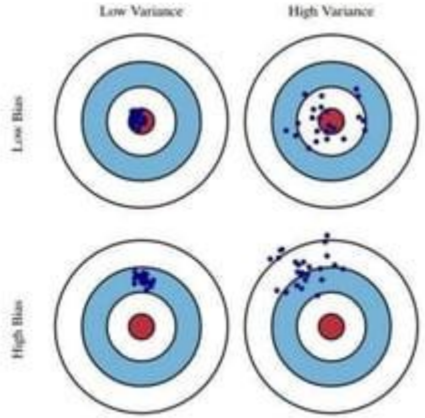
$$b = \frac{1}{N}\sum_{i=1}^{N} r(\tau)$$ Average sampled trajectories' rewards

Variance reduction: baseline

baseline:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau)[r(\tau) - \boxed{b}]$$

Do baselines introduce bias in expectation?



Variance reduction: baseline

baseline:

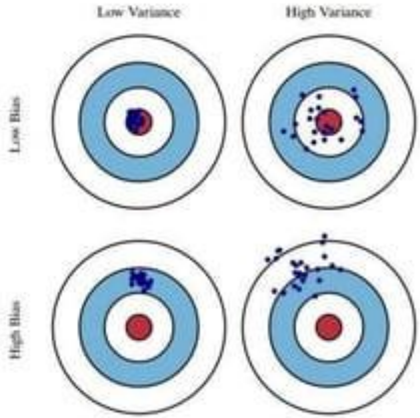
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau)[r(\tau) - \boxed{b}]$$

Do baselines introduce bias in expectation?

Analyze:

$$E_{\tau \sim \pi_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b) \right]$$

$$E_{\tau \sim \pi_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(\tau) b \right]$$



Variance reduction: baseline

$$
\begin{aligned}
E_{\tau\sim\pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau) b\right] &= \int \pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) b \, d\tau \\
&= \int \nabla_\theta \pi_\theta(\tau) b \, d\tau \\
&= b\nabla_\theta \int \pi_\theta(\tau) d\tau \\
&= b\nabla_\theta 1 \\
&= 0
\end{aligned}
$$

*baseline is independent of policy

Reduce variance with baseline won't make model biased, as long as the baseline is independent of the policy (not action-related)

Variance reduction: baseline

$$E_{\tau\sim\pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)\right]$$

- Subtracting a baseline is unbiased in expectation, It won't make the estimator biased.
- The baseline can be any function, random variable, as long as it does not vary with action.

Variance reduction: baseline

Variance:

$$Var[x] = E[x^2] - E[x]^2$$

$$Var = E_{\tau\sim\pi_\theta(\tau)}\left[\left(\nabla_\theta \log \pi_\theta(\tau)(r(\tau)-b)\right)^2\right] - E_{\tau\sim\pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau)(r(\tau)-b)\right]^2$$

Related Paper:
VARIANCE REDUCTION FOR POLICY GRADIENT WITH ACTION-DEPENDENT FACTORIZED BASELINES, Cathy Wu*, Aravind Rajeswaran*, Yan Duan, Vikash Kumar, Alexandre M Bayen, Sham Kakade, Igor Mordatch, Pieter Abbeel OpenAI (currently under review at ICLR 2018)

Variance reduction: causality + baseline

In previous, we introduce 2 method to reduce variance.

- Causality

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'})\right]$$

- Baseline

$$E_{\tau\sim\pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)\right]$$

Question: Can we combine two variance reduction method together?

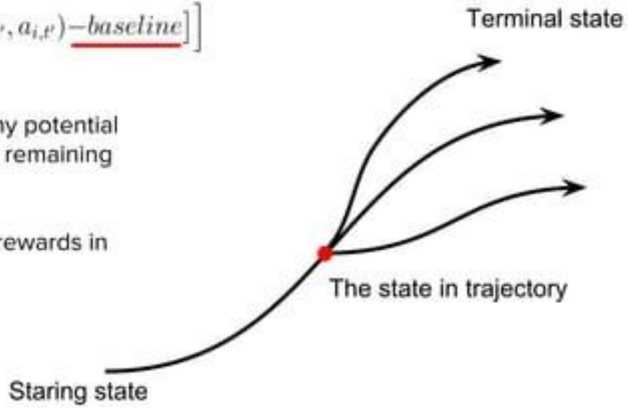
Variance reduction: causality + baseline

The ideal form:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left[\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'}) \underline{- baseline}\right]\right]$$

If you are in certain state of trajectory, there are many potential path to reach different terminal state. Of course, the remaining rewards would also be different.

The naive concept is to find the average remaining rewards in that state, in other words, value function.

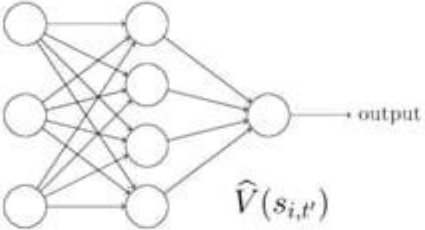


Variance reduction: causality + baseline

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left[\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'}) - \boxed{\widehat{V}(s_{i,t'})}\right]\right]$$

We can learn value function by the method mentioned before (tabular method or using function approximator)

In REINFORCE algorithm, the agent play with environment until reaching terminal state. We know the remaining rewards at each step so that we can use Monte Carlo method to evaluate policy, and the loss could be MSE.



Policy

Policy Gradient can be apply to discrete action space problem so as continuous action space problem.

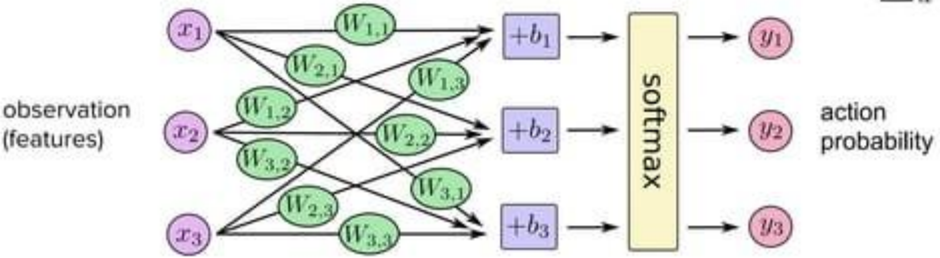
- Discrete action problem: Softmax Policy
- Continuous action problem: Gaussian Policy

Policy: Softmax Policy

Here, we suppose the function approximator is h(s, a) and θ is its parameters

we will sample the action according to its softmax probability.

$$\pi_\theta(s,a) = \frac{e^{h(s,a|\theta)}}{\sum_{a'} e^{h(s,a'|\theta)}}$$



Policy: Gaussian Policy

In continuous control, a Gaussian policy is common.

- Gaussian distribution:

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- Gaussian Policy: we use neural network to approximate **mean**, and sample the action according to Gaussian distribution. The **variance** can also be parameterized.

Policy: Gaussian Policy

Here, we use **fixed variance** and mean value is computed by **linear combination** of state features, where $\phi(s)$ is used to features transformation.

$$\mu(s) = \phi(s)^T \theta$$

The gradient of the log of the policy is

$$\nabla_\theta \log \pi_\theta(s, a) = \frac{(a - \mu(s))\phi(s)}{\sigma^2}$$

In neural network, you just need to backpropagate the sampled action probability.

Reason of Inefficient Learning

The main reasons for inefficient learning in REINFORCE are:

- The REINFORCE algorithm is on-policy
- We need to learning by Monte-Carlo method

Reason of Inefficient Learning

The main reasons for inefficient learning in REINFORCE are:

- The REINFORCE algorithm is on-policy

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)\right)\left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right]$$

The learning process of REINFORCE:

1. Sample multiple trajectories from $\tau \sim \pi_\theta(\tau)$
2. Fit model

If the sampled trajectories are from different distribution, the learning result would be wrong.

Reason of Inefficient Learning

The main reason for inefficient learning in REINFORCE is:

- The REINFORCE algorithm is on-policy

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)\left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right]$$

- We need to learning by Monte-Carlo method
  - In vanilla policy gradient, we sample multiple trajectories but just update model once. In contrast to TD learning, vanilla policy gradient learns much slower.

Reason of Inefficient Learning

The main reason for inefficient learning in REINFORCE is:

- The REINFORCE algorithm is on-policy

  Improve by **importance sampling**

- We need to learning by Monte-Carlo method

  Improve by **Actor-Critic**

Importance sampling

Importance sampling is a statistical technique that we could use to estimate the properties from different distribution here.

Suppose the objective is $E_{X\sim p(X)}\left[f(X)\right]$, but the data are sampled from q(X), we can do such transformation:

$$E_{X\sim p(X)}\left[f(X)\right] = \sum p(x)f(x)$$
$$= \sum q(x)\frac{p(x)}{q(x)}f(x) = E_{X\sim q(X)}\left[\frac{p(x)}{q(x)}f(x)\right]$$

Objective of on-policy policy gradient : $J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[r(\tau)\right]$

Objective of off-policy policy gradient: $J(\theta) = E_{\tau\sim\mu_\phi(\tau)}\left[\frac{\pi_\theta(\tau)}{\mu_\phi(\tau)}r(\tau)\right]$

Off-policy & importance sampling

- target policy (π_θ): the learning policy, which we are interested in.
- behavior policy (μ): the policy used to collect samples.

We sample the trajectories from μ the objective would be:

$$J(\theta) = E_{\tau\sim\mu(\tau)}\Big[\frac{\pi_\theta(\tau)}{\mu(\tau)}r(\tau)\Big]$$

$$\pi_\theta(\tau) = p(s_1)\prod_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)$$

$$\frac{\pi_\theta(\tau)}{\mu(\tau)} = \frac{p(s_1)\prod_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)}{p(s_1)\prod_{t=1}^{T}\mu(a_t|s_t)p(s_{t+1}|s_t,a_t)}$$

Off-policy & importance sampling

- target policy (π_θ): the learning policy, which we are interested in.
- behavior policy (μ): the policy used to collect samples.

We sample the trajectories from μ the objective would be:

$$J(\theta) = E_{\tau\sim\mu(\tau)}\Big[\frac{\pi_\theta(\tau)}{\mu(\tau)}r(\tau)\Big]$$

$$\pi_\theta(\tau) = p(s_1)\prod_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)$$

$$\frac{\pi_\theta(\tau)}{\mu(\tau)} = \frac{\cancel{p(s_1)}\prod_{t=1}^{T}\pi_\theta(a_t|s_t)\cancel{p(s_{t+1}|s_t,a_t)}}{\cancel{p(s_1)}\prod_{t=1}^{T}\mu(a_t|s_t)\cancel{p(s_{t+1}|s_t,a_t)}} = \boxed{\frac{\prod_{t=1}^{T}\pi_\theta(a_t|s_t)}{\prod_{t=1}^{T}\mu(a_t|s_t)}}$$

Off-policy & importance sampling

- target policy (π_θ): the learning policy, which we are interested in.
- behavior policy (μ): the policy used to interact with environment.

We sample the trajectories from μ the objective would be:

$$J(\theta) = E_{\tau\sim\mu(\tau)}\Big[\frac{\pi_\theta(\tau)}{\mu(\tau)}r(\tau)\Big]$$

$$\pi_\theta(\tau) = p(s_1)\prod_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)$$

$$\frac{\pi_\theta(\tau)}{\mu(\tau)} = \frac{\cancel{p(s_1)}\prod_{t=1}^{T}\pi_\theta(a_t|s_t)\cancel{p(s_{t+1}|s_t,a_t)}}{\cancel{p(s_1)}\prod_{t=1}^{T}\mu(a_t|s_t)\cancel{p(s_{t+1}|s_t,a_t)}} = \frac{\prod_{t=1}^{T}\pi_\theta(a_t|s_t)}{\prod_{t=1}^{T}\mu(a_t|s_t)}$$

Importance sampling ratio ends up depending only on the two policies and the sequence.

Off-policy & importance sampling

Suppose the off-policy objective function is:

$$J(\theta) = E_{\tau \sim \mu_\phi(\tau)}\left[\frac{\boxed{\pi_\theta(\tau)}}{\boxed{\mu_\phi(\tau)}} r(\tau)\right]$$

target policy (learner neural net)

behavior policy (expert/behavior neural net)

Off-policy & importance sampling

Suppose the off-policy objective function is:

$$J(\theta) = E_{\tau \sim \mu_\phi(\tau)}\left[\frac{\pi_\theta(\tau)}{\mu_\phi(\tau)} r(\tau)\right]$$

$$\nabla_\theta J(\theta) = \nabla_\theta E_{\tau \sim \mu_\phi(\tau)}\left[\frac{\pi_\theta(\tau)}{\mu_\phi(\tau)} r(\tau)\right] = E_{\tau \sim \mu_\phi(\tau)}\left[\frac{\pi_\theta(\tau)}{\mu_\phi(\tau)} \nabla_\theta \log \pi_\theta(\tau) r(\tau)\right]$$

Off-policy & importance sampling

Suppose the off-policy objective function is:

$$\frac{\prod_{t=1}^{T} \pi_\theta(a_t|s_t)}{\prod_{t=1}^{T} \mu(a_t|s_t)}$$

$$J(\theta) = E_{\tau\sim\mu_\phi(\tau)}\left[\frac{\pi_\theta(\tau)}{\mu_\phi(\tau)} r(\tau)\right]$$

$$\nabla_\theta J(\theta) = \nabla_\theta E_{\tau\sim\mu_\phi(\tau)}\left[\frac{\pi_\theta(\tau)}{\mu_\phi(\tau)} r(\tau)\right] = E_{\tau\sim\mu_\phi(\tau)}\left[\frac{\pi_\theta(\tau)}{\mu_\phi(\tau)} \nabla_\theta \log \pi_\theta(\tau) r(\tau)\right]$$

Off-policy & importance sampling

Suppose the off-policy objective function is:

$$J(\theta) = E_{\tau \sim \mu_\phi(\tau)}\left[\frac{\pi_\theta(\tau)}{\mu_\phi(\tau)} r(\tau)\right]$$

$$\nabla_\theta J(\theta) = \nabla_\theta E_{\tau \sim \mu_\phi(\tau)}\left[\frac{\pi_\theta(\tau)}{\mu_\phi(\tau)} r(\tau)\right] = E_{\tau \sim \mu_\phi(\tau)}\left[\frac{\pi_\theta(\tau)}{\mu_\phi(\tau)} \nabla_\theta \log \pi_\theta(\tau) r(\tau)\right]$$

$$= \quad E_{\tau \sim \mu_\phi(\tau)}\left[\left(\prod_{t=1}^{T} \frac{\pi_\theta(a_t|s_t)}{\mu_\phi(a_t|s_t)}\right)\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)\left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right]$$

How about causality?

Off-policy & importance sampling

Suppose the off-policy objective function is:

$$J(\theta) = E_{\tau \sim \mu_\phi(\tau)}\left[\frac{\pi_\theta(\tau)}{\mu_\phi(\tau)} r(\tau)\right]$$

$$\nabla_\theta J(\theta) = \nabla_\theta E_{\tau \sim \mu_\phi(\tau)}\left[\frac{\pi_\theta(\tau)}{\mu_\phi(\tau)} r(\tau)\right] = E_{\tau \sim \mu_\phi(\tau)}\left[\frac{\pi_\theta(\tau)}{\mu_\phi(\tau)} \nabla_\theta \log \pi_\theta(\tau) r(\tau)\right]$$

$$= \quad E_{\tau \sim \mu_\phi(\tau)}\left[\left(\prod_{t=1}^{T} \frac{\pi_\theta(a_t|s_t)}{\mu_\phi(a_t|s_t)}\right)\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)\left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right]$$

$$= \quad E_{\tau \sim \mu_\phi(\tau)}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\left(\prod_{t'=1}^{t} \frac{\pi_\theta(a_{t'}|s_{t'})}{\mu_\phi(a_{t'}|s_{t'})}\right)\left(\sum_{t'=t}^{T} r(s_{t'}, a_{t'})\right)\right]$$

Off-policy & importance sampling

The gradient of off-policy objective:

$$\nabla_\theta J(\theta) = \quad E_{\tau\sim\mu_\phi(\tau)}\Big[\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)\Big(\prod_{t'=1}^{t}\frac{\pi_\theta(a_{t'}|s_{t'})}{\mu_\phi(a_{t'}|s_{t'})}\Big)\Big(\sum_{t'=t}^{T} r(s_{t'}, a_{t'})\Big)\Big]$$

future action won't affect current weight

Off-policy & importance sampling

The gradient of off-policy objective:

$$\nabla_\theta J(\theta) = \quad E_{\tau \sim \mu_\phi(\tau)}\Big[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\Big(\prod_{t'=1}^{t} \frac{\pi_\theta(a_{t'}|s_{t'})}{\mu_\phi(a_{t'}|s_{t'})}\Big)\Big(\sum_{t'=t}^{T} r(s_{t'}, a_{t'})\Big)\Big]$$

1. This is the general form of off-policy policy gradient. if we use on-policy learning, the form is as same as vanilla policy gradient (importance sampling ratio is 1)

2. In practice, We store trajectories along with its action probability each step, and then update the neural network by adding importance sampling ratio.

Reason of Inefficient Learning

The main reason for inefficient learning in REINFORCE is:

- The REINFORCE algorithm is on-policy

  We've already discussed.

- We need to learning by Monte-Carlo method

  In the next section

Reference

- CS 294, Berkeley lecture 4: http://rll.berkeley.edu/deeprlcourse/
- David Silver RL course lecture7: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/pg.pdf
- Baseline Substraction from Shan-Hung Wu https://www.youtube.com/watch?v=XnXRzOBOPc8
- Andrej Karpathy's blog: http://karpathy.github.io/2016/05/31/rl/
- Policy Gradient in pytorch https://github.com/pytorch/examples/tree/master/reinforcement_learning

Outline

- Pitfall of Value-based Reinforcement Learning
  - Value-based policy is deterministic
  - Hard to handle continuous control
- Policy gradient
- Variance reduction
  - Causality
  - Baseline
- Policy in policy gradient
  - Softmax policy
  - Gaussian policy
- Off-policy policy gradient
  - Importance sampling