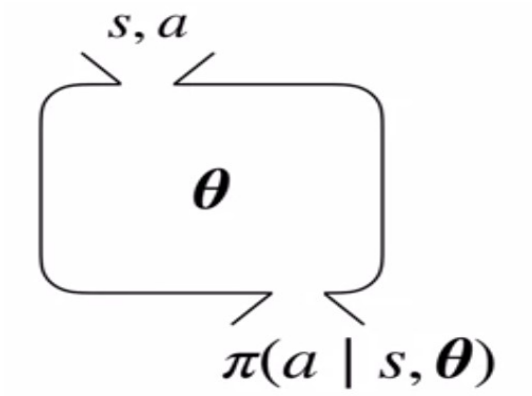
Policy Gradient Methods

So far, we have learned all the values of actions and selected the actions based on their estimated action values. Their policies were not considered for further action-value estimates. However, in the Policy Gradient Methods, we instead learn a parameterized policy that helps us in selecting the action without concerning the value function. Eventhough, the value functions may still be required to learn the policy parameters, but for the selection of the actions, a value function is not directly required.



Just like the function approximation, we use θ to represent the parameter vector.
The parameterized policy is given as:

$$\pi(a|s,\boldsymbol{\theta}) = \Pr\{A_t = a \mid S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}\}$$

It denotes the probability of choosing an action “a” at time “t”, given the environment is in state “s” at time “t” with parameter θ.

Constraints on the Policy Parameterization

1. Probability of Selecting an action must be >= 0.

$$\pi(a \mid s, \boldsymbol{\theta}) \geq 0 \qquad \text{for all } a \in \mathcal{A} \text{ and } s \in \mathcal{S}$$

2. For each states sum of probabilities over all action must be equal to 1.

$$\sum_{a \in \mathcal{A}} \pi(a \mid s, \boldsymbol{\theta}) = 1 \quad \text{for all } s \in \mathcal{S}$$

That is why we could not use a linear function directly as we did in value function approximation because there is no way to guarantee that linear function will sum to one.
Therefore, to satisy this condition we use a **softmax policy**:

$$\pi(a \mid s, \boldsymbol{\theta}) \doteq \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_{b \in \mathcal{A}} e^{h(s,b,\boldsymbol{\theta})}}$$

Numerator = Action Preference
Denominator = Sum of all the Actions

Consider an example of a grid with an agent at center. The agent is able to take four actions. Right, Left, Up and Down. In the following diagram the probability of taking the action UP is higher while other have respectively lower probabilities such that the sum of the probabilities will be equal to 1 and dont exceed it.

Stochastic Policy

The reasons behind using the stochastic policy class is because it smooths out the optimization problem.
In a deterministic policy, considering a grid world, if we make a change on an action, the change in the action makes a significant change in the outcome while in a stochastic policy, its much more smoother.
Since, the distribution is continuous and smoother, hence the computation of gradient descent becomes much more easier.
The output of a deterministic policy is an action to take while the output of the stochastic policy is the probability to choose an action at certain time steps.

Reasons for Policy over Value Estimation:

1. Policy Evaluation comparatively consists of lesser parameters. So, computational advantage.
2. Better Convergence rate.

Derivation for Policy Gradient Estimation

Our Objective Function:

$$\max_{\theta} U(\theta) = \max_{\theta} \sum_{\tau} P(\tau;\theta) R(\tau)$$

where,

$$U(\theta) = \mathrm{E}[\sum_{t=0}^{H} R(s_t, u_t); \pi_\theta] = \sum_{\tau} P(\tau;\theta) R(\tau)$$

$$U(\theta) = \sum_{\tau} P(\tau;\theta) R(\tau)$$

Taking the gradient w.r.t. θ gives

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \sum_{\tau} P(\tau;\theta) R(\tau) \\
&= \sum_{\tau} \nabla_\theta P(\tau;\theta) R(\tau) \\
&= \sum_{\tau} \frac{P(\tau;\theta)}{P(\tau;\theta)} \nabla_\theta P(\tau;\theta) R(\tau) \\
&= \sum_{\tau} P(\tau;\theta) \frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)} R(\tau) \\
&= \sum_{\tau} P(\tau;\theta) \nabla_\theta \log P(\tau;\theta) R(\tau)
\end{aligned}$$

Approximate with the empirical estimate for m sample paths under policy π_θ:

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

Here, T or the Tau stands for the Trajectory which is a sequence of states and actions observed by an agent.

$$\nabla_\theta \log P(\tau^{(i)};\theta) = \nabla_\theta \log \left[\prod_{t=0}^{H} \underbrace{P(s_{t+1}^{(i)}|s_t^{(i)},u_t^{(i)})}_{\text{dynamics model}} \cdot \underbrace{\pi_\theta(u_t^{(i)}|s_t^{(i)})}_{\text{policy}} \right]$$

$$= \nabla_\theta \left[\sum_{t=0}^{H} \log P(s_{t+1}^{(i)}|s_t^{(i)},u_t^{(i)}) + \sum_{t=0}^{H} \log \pi_\theta(u_t^{(i)}|s_t^{(i)}) \right]$$

$$= \nabla_\theta \sum_{t=0}^{H} \log \pi_\theta(u_t^{(i)}|s_t^{(i)})$$

$$= \sum_{t=0}^{H} \underbrace{\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)})}_{\text{no dynamics model required!!}}$$

The first equation disappears because of the absence of theta parameter.

Hence, we consider methods for learning the policy parameter based on the gradient of some performance measure of U(θ) with respect to the policy parameter. These methods seek to maximize the performance, so that the updates approximate gradient ascent in U:

$$\theta_{t+1} = \theta_t + \alpha U(\theta_t)'$$