Policy Gradient Methods

Tabular methods and Function Approximators are value-based which means the policy looks at the value of the state-actions Q and choose the action with optimal estimated return. In policy gradient methods, we use the function approxmiators to not predict the Q values but to estimate teh probabilities of taking each action:

$$\pi(a|s,\theta) \in [0,1] \tag{1}$$

and thus the function approximator **is** the policy that gives out the distribution of probabilities over the actions rather than the values for you to read:

$$\pi(s,\theta) = [p(a_1), p(a_2), \cdots, p(a_n)] \tag{2}$$

and your action to take, a, is sampled from this distribution:

$$a \sim \pi(a|s,\theta) \tag{3}$$

and thus your policy and actions change more smoothly during learning,whereas in value-based methods, you manually choose the action using greedy or e-greedy methods:

$$a = \arg\max_a \hat{q}(s,a). \tag{4}$$

Representing Policies using Neural Networks

Neural networks can learn the weights and propagate the extracted features of the state vector to approximate the values of the state:

$$\hat{y} = \phi_2(\phi_1(x \cdot w_1) \cdot w_2). \tag{5}$$

To train the neural network to output the vector of probability, we compress the output values to values between 0 and 1 such that:

$$\hat{y} = [0,1]^n \tag{6}$$

where n is the number of possible actions to take.

Softmax Function

In order to achieve the compression, we apply softmax to normalize the values so that not only they compress to $[0,1]$, they also all sum up to 1:

$$\sigma(x_i) = \frac{e^{x_i}}{\sum e^{x_i}} \tag{7}$$

where the values of the individual outputs are raised to powers of e to get a vector of probability distribution over the possible actions:

$$\pi(a|s,\theta) = [\sigma(a_1), \sigma(a_2), \cdots, \sigma(a_n)]. \tag{8}$$

Policy Evaluation

We define performance measure of the policy/neural network as $J(\theta)$, so that we can compare them such that:

$$J_{\pi_1}(\theta) > J_{\pi_2}(\theta) \Rightarrow \text{We consider } J_{\pi_1}(\theta) \text{ is better than } J_{\pi_2}(\theta). \tag{9}$$

As such the optimal policy has the optimal paramaters θ that give us the maximal performance:

$$\pi_*(a \mid s, \theta) = \arg\max_{\theta} J(\theta). \tag{10}$$

We obtain the performance estimate $\hat{J}(\theta)$ from experience samples during the simulations, and we perform the Stochastic Gradient Ascent to improve the performance with regards to its parameters:

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta) \tag{11}$$

where

$$\nabla J(\theta) = [\frac{\partial \hat{J}(\theta)}{\partial \theta_1}, \frac{\partial \hat{J}(\theta)}{\partial \theta_2}, \cdots, \frac{\partial \hat{J}(\theta)}{\partial \theta_n}]. \tag{12}$$

Policy Gradient Theorem