Policy Gradient for Multi-Armed Bandits

June 11, 2025

Goal: Maximize Expected Reward

We want to find policy parameters θ such that:

$$J(\theta) = \mathbb{E}_{a\sim\pi_\theta}[R(a)]$$

This is the expected reward under the current policy $\pi_\theta(a)$, parameterized by action preferences $\theta = [H(0), H(1), \ldots, H(k-1)]$.

1. Policy Definition (Softmax over Preferences)

Theory:

$$\pi_\theta(a) = \frac{e^{H(a)}}{\sum_b e^{H(b)}}$$

Code:

```
def softmax(self):
    exp_prefs = np.exp(self.preferences - np.max(self.preferences))
    return exp_prefs / np.sum(exp_prefs)
```

Match:

- `self.preferences` corresponds to $H(a)$
- Computes $\pi_\theta(a)$
- Subtracting `np.max()` prevents overflow. The `np.max(self.preferences)` is to give numerical stability. Big $H(a)$ can cause $e^{H(a)}$ to blow up.

2. Action Sampling (Exploration)

Theory: $A_t \sim \pi_\theta(a)$
Code:

```
def select_arm(self):
    probs = self.softmax()
    return np.random.choice(self.k, p=probs)
```

Match:

- Computes $\pi_\theta(a)$ via softmax
- Samples action A_t based on these probabilities. `np.random.choice(self.k, p=probs)` will pick the options based on the value of probs.

3. Policy Update (REINFORCE)

Theory:

$$\theta \leftarrow \theta + \alpha R_t \nabla_\theta \log \pi_\theta(a_t)$$

$$\nabla_\theta \log \pi_\theta(a_t) = \begin{cases} 1 - \pi(a_t) & \text{if } j = a_t \\ -\pi(j) & \text{otherwise} \end{cases}$$

Take example case with 3 actions: 1, 2, 3:

$$\pi(1) = \frac{e^{H(1)}}{e^{H(1)} + e^{H(2)} + e^{H(3)}} \rightarrow \log \pi(1) = H(1) - \log(e^{H(1)} + e^{H(2)} + e^{H(3)})$$

$$\pi(2) = \frac{e^{H(2)}}{e^{H(1)} + e^{H(2)} + e^{H(3)}} \rightarrow \log \pi(2) = H(2) - \log(e^{H(1)} + e^{H(2)} + e^{H(3)})$$

$$\pi(3) = \frac{e^{H(3)}}{e^{H(1)} + e^{H(2)} + e^{H(3)}} \rightarrow \log \pi(3) = H(3) - \log(e^{H(1)} + e^{H(2)} + e^{H(3)})$$

When action 1 is taken:

$$\frac{\partial \log \pi(1)}{\partial H(1)} = 1 - \frac{\partial}{\partial H(1)}(\log(e^{H(1)} + e^{H(2)} + e^{H(3)})) = 1 - \frac{e^{H(1)}}{e^{H(1)} + e^{H(2)} + e^{H(3)}} = 1 - \pi(1)$$

$$\frac{\partial \log \pi(1)}{\partial H(2)} = 0 - \frac{\partial}{\partial H(2)}(\log(e^{H(1)} + e^{H(2)} + e^{H(3)})) = -\frac{e^{H(2)}}{e^{H(1)} + e^{H(2)} + e^{H(3)}} = -\pi(2)$$

$$\frac{\partial \log \pi(1)}{\partial H(3)} = 0 - \frac{\partial}{\partial H(3)}(\log(e^{H(1)} + e^{H(2)} + e^{H(3)})) = -\frac{e^{H(3)}}{e^{H(1)} + e^{H(2)} + e^{H(3)}} = -\pi(3)$$

Update Rule:

$$H(j) \leftarrow H(j) + \alpha R_t(\mathbb{I}_{j=a_t} - \pi(j))$$

Code:

```python
def update(self, arm, reward):
    probs = self.softmax()
    grad_log_pi = -probs
    grad_log_pi[arm] += 1
    self.preferences += self.alpha * reward * grad_log_pi
```

4. Intuition Recap

Step	Theory	Code Intuition
Policy	$\pi(a) = \frac{e^{H(a)}}{\sum_b e^{H(b)}}$	`softmax()` Action probabilities
Sampling	$A_t \sim \pi(a)$	`np.random.choice(...)` Stochastic action choice
Update	$\theta \leftarrow \theta + \alpha R_t(\mathbb{I}_{j=a} - \pi(j))$	`update()` Increase pref for good actions