

Online and Reinforcement Learning (2025)

Home Assignment 4

Davide Marchi 777881

Contents

1	Policy Gradient Methods	2
1.1	Baseline	2
1.2	Lunar	3
2	Improved Parametrization of UCB1	5
3	Introduction of New Products	6
4	Empirical comparison of FTL and Hedge	6

1 Policy Gradient Methods

1.1 Baseline

We are given that the policy gradient theorem can be generalized to include an arbitrary baseline $b(s)$:

$$\nabla_{\theta} J(\pi) = \sum_{s \in S} \mu_{\pi}(s) \sum_{a \in A} \nabla_{\theta} \pi(s, a) (Q_{\pi}(s, a) - b(s)),$$

where:

- S is the state space.
- A is the action space.
- $\pi(s, a)$ is the probability of choosing action a in state s .
- $\mu_{\pi}(s)$ is the stationary state distribution under policy π .
- $Q_{\pi}(s, a)$ is the state-action value function.

The term

$$\sum_{a \in A} \nabla_{\theta} \pi(s, a) b(s)$$

acts as a control variate, and we must show that its expectation is zero, i.e.,

$$\mathbb{E} \left[\sum_{a \in A} \nabla_{\theta} \pi(s, a) b(s) \right] = 0.$$

Proof

For any state $s \in S$, note that $\pi(s, \cdot)$ is a probability distribution over A . Therefore, by definition:

$$\sum_{a \in A} \pi(s, a) = 1.$$

Differentiating both sides of the equation with respect to θ , we obtain:

$$\sum_{a \in A} \nabla_{\theta} \pi(s, a) = \nabla_{\theta} \left(\sum_{a \in A} \pi(s, a) \right) = \nabla_{\theta} (1) = 0.$$

Since $b(s)$ does not depend on the action a , it can be factored out of the summation:

$$\sum_{a \in A} \nabla_{\theta} \pi(s, a) b(s) = b(s) \sum_{a \in A} \nabla_{\theta} \pi(s, a) = b(s) \cdot 0 = 0.$$

Taking the expectation with respect to the stationary distribution $\mu_\pi(s)$, we have:

$$\mathbb{E}_{s \sim \mu_\pi} \left[\sum_{a \in A} \nabla_{\theta} \pi(s, a) b(s) \right] = \sum_{s \in S} \mu_\pi(s) \cdot 0 = 0.$$

Thus, we conclude that

$$\mathbb{E} \left[\sum_{a \in A} \nabla_{\theta} \pi(s, a) b(s) \right] = 0.$$

1.2 Lunar

1. Derivation of the Analytical Expression for the Score Function

I consider a softmax policy defined by

$$\pi(s, a) = \frac{\exp(\theta_a^\top s)}{\sum_{b \in A} \exp(\theta_b^\top s)},$$

where θ_a is the parameter vector corresponding to action a and $s \in \mathbb{R}^d$ is the state feature vector.

Taking the logarithm of the policy, I have:

$$\log \pi(s, a) = \theta_a^\top s - \log \left(\sum_{b \in A} \exp(\theta_b^\top s) \right).$$

I now differentiate this expression with respect to the parameters θ_i , for any action i . There are two cases:

Case 1: $i = a$ Differentiate $\log \pi(s, a)$ with respect to θ_a :

$$\nabla_{\theta_a} \log \pi(s, a) = \nabla_{\theta_a} [\theta_a^\top s] - \nabla_{\theta_a} \log \left(\sum_{b \in A} \exp(\theta_b^\top s) \right).$$

The first term is simply:

$$\nabla_{\theta_a} (\theta_a^\top s) = s.$$

For the second term, using the chain rule,

$$\nabla_{\theta_a} \log \left(\sum_{b \in A} \exp(\theta_b^\top s) \right) = \frac{1}{\sum_b \exp(\theta_b^\top s)} \cdot \nabla_{\theta_a} \left(\sum_b \exp(\theta_b^\top s) \right).$$

Since only the term with $b = a$ depends on θ_a , it follows that

$$\nabla_{\theta_a} \left(\sum_b \exp(\theta_b^\top s) \right) = \exp(\theta_a^\top s) s.$$

Thus,

$$\nabla_{\theta_a} \log \left(\sum_b \exp(\theta_b^\top s) \right) = \frac{\exp(\theta_a^\top s)}{\sum_b \exp(\theta_b^\top s)} s = \pi(s, a) s.$$

Therefore, for $i = a$,

$$\nabla_{\theta_a} \log \pi(s, a) = s - \pi(s, a) s = (1 - \pi(s, a)) s.$$

Case 2: $i \neq a$ For $i \neq a$, the first term is zero (since θ_i does not appear in $\theta_a^\top s$), and only the normalization term contributes:

$$\nabla_{\theta_i} \log \pi(s, a) = -\nabla_{\theta_i} \log \left(\sum_b \exp(\theta_b^\top s) \right).$$

Again, only the term with $b = i$ depends on θ_i , so

$$\nabla_{\theta_i} \left(\sum_b \exp(\theta_b^\top s) \right) = \exp(\theta_i^\top s) s,$$

and hence,

$$\nabla_{\theta_i} \log \pi(s, a) = -\frac{\exp(\theta_i^\top s)}{\sum_b \exp(\theta_b^\top s)} s = -\pi(s, i) s.$$

Combined Expression Thus, for every action i , the gradient is given by

$$\nabla_{\theta_i} \log \pi(s, a) = \begin{cases} (1 - \pi(s, a)) s, & \text{if } i = a, \\ -\pi(s, i) s, & \text{if } i \neq a. \end{cases}$$

In vector form (where the policy parameters are arranged in rows corresponding to actions), this can be compactly written as:

$$\nabla_{\theta} \log \pi(s, a) = (e_a - \pi(s)) s^\top,$$

with e_a denoting the one-hot vector for the action a .

2. Implementation of the Gradient Function

In my implementation, I only added the parts required to compute the analytical gradient for the softmax policy. The modified function `gradient_log_pi` in my `Softmax_policy` class is as follows:

```
def gradient_log_pi(self, s, a):
    # Compute the probability vector for state s
    prob = self.pi(s)
    # Compute the gradient for each action (outer product of prob and s)
```

```

grad = - np.outer(prob, s)
# For the taken action a, add s to obtain (1 - pi(s,a))*s
grad[a] += s
return grad

```

Listing 1: Modified `gradient_log_pi` function

This code implements exactly the formula derived above.

3. Verification of the Gradient Implementation

To verify my implementation, I used the numerical approximation of the gradient in the function `gradient_log_pi_test`. I run the notebook cell to compare my analytical gradient with the numerical gradient for a range of random perturbations on the policy parameters. In all cases the analytical and numerical gradients agreed within the requested tolerance. This confirms that the derivation and implementation of `gradient_log_pi` are correct.

Also, here we present the graph showing the accumulated reward increasing over the number of episodes, which indicates that the policy is actually improving over time.

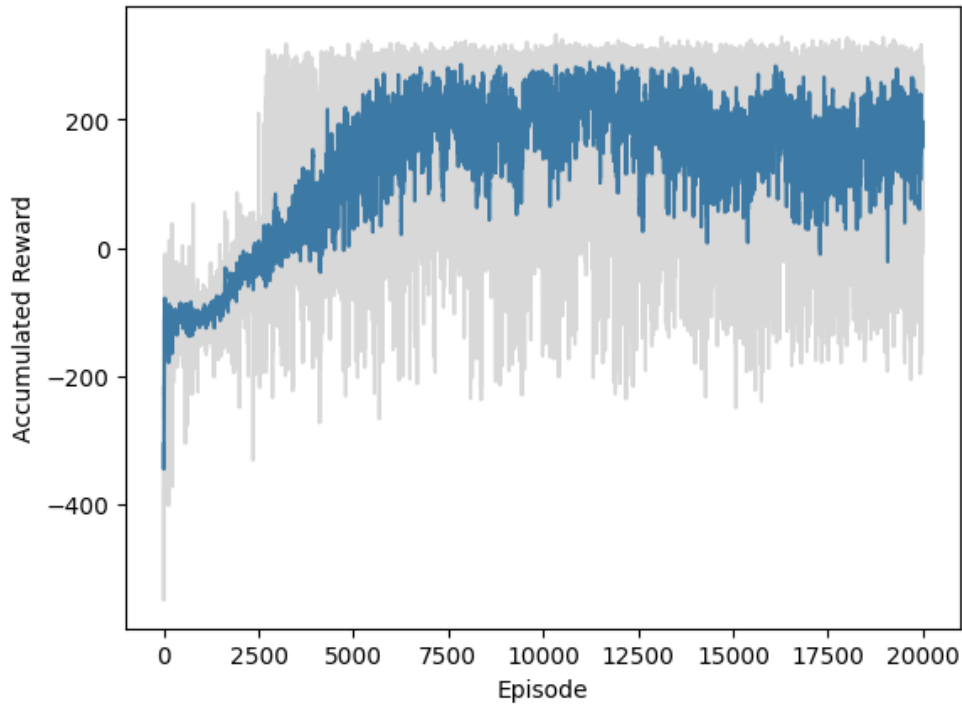


Figure 1: Accumulated reward over episodes.

2 Improved Parametrization of UCB1

(Optional, but highly recommended)

3 Introduction of New Products

4 Empirical comparison of FTL and Hedge