Week 7: Policy Gradient

Ardhito Nurhadyansah[1], Mohamad Arvin Fadriansyah[2], and Ian Suryadi Timothy H[3]

[1]2106750206
[2]2006596996
[3]2106750875

1 Policy Parameterization for Discrete Action Spaces

In reinforcement learning with discrete action spaces, we often represent the policy $\pi(a \mid s; \boldsymbol{\theta})$ using a parameterized categorical distribution. One commonly used approach is the **softmax (Gibbs/Boltzmann) parameterization**, where the probability of selecting an action a in state s is defined as:

$$\pi(a \mid s; \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\theta}^\top \phi(s, a))}{\sum_{a' \in A} \exp(\boldsymbol{\theta}^\top \phi(s, a'))},$$

where:

- $\phi(s, a) \in \mathbb{R}^d$ is a feature vector for state-action pair (s, a),
- $\boldsymbol{\theta} \in \mathbb{R}^d$ is the parameter vector,
- $\mathcal{A}$ is the set of all possible discrete actions.

This softmax function ensures that:

- Each action probability is positive: $\pi(a \mid s; \boldsymbol{\theta}) > 0$,
- The probabilities sum to one: $\sum_{a \in A} \pi(a \mid s; \boldsymbol{\theta}) = 1$.

The distribution $\pi(\cdot \mid s; \boldsymbol{\theta})$ is also known as a **categorical distribution** over actions, and this formulation aligns with the Gibbs distribution in statistical physics, which favors higher-scoring actions (higher $\boldsymbol{\theta}^\top \phi(s, a)$).

Alternative Interpretation: Preference-Based Parameterization. Sutton (2018) also presents a preference-based view of softmax policy parameterization, where each action a is assigned a scalar preference $H_t(a) \in \mathbb{R}$. These preferences determine the policy via:

$$\pi_t(a) = \frac{\exp(H_t(a))}{\sum_{b \in A} \exp(H_t(b))}.$$

Only the relative differences between preferences matter. For example, adding a constant to all $H_t(a)$ values has no effect on the resulting action probabilities. This highlights the invariance property of the softmax formulation. Initially, all preferences are often set equally (e.g., $H_1(a) = 0$) to induce uniform exploration.

Connection to Logistic Function. In the case of two actions a_1 and a_2, this softmax reduces to a logistic (sigmoid) function:

$$\pi(a_1) = \frac{1}{1 + \exp(-(H(a_1) - H(a_2)))},$$

which is widely used in statistics and neural networks. This emphasizes the connection between policy gradients and logistic regression models.

Gradient of the Softmax Policy. When applying policy gradient methods, we need the gradient of the log-policy:

$$\nabla_{\boldsymbol{\theta}} \log \pi(a \mid s; \boldsymbol{\theta}) = \phi(s, a) - \sum_{a' \in A} \pi(a' \mid s; \boldsymbol{\theta}) \phi(s, a').$$

This expression follows from the quotient rule and is essential for computing the policy gradient in REINFORCE and actor-critic methods.

Interpretation. This softmax parameterization encourages exploration: even suboptimal actions have non-zero probability of being selected. The scale of $\boldsymbol{\theta}^\top \phi(s, a)$ can influence how deterministic or stochastic the policy behaves.

2 Discounted Reward and the Policy Gradient Theorem

In reinforcement learning, when working with continuing tasks, we often aim to maximize the expected *discounted return*. This leads to defining the performance objective as:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \right] = \mathbf{P}_\pi^* \mathbf{r}_\pi,$$

where:

- $\boldsymbol{\theta} \in \mathbb{R}^d$ is the policy parameter,
- $\gamma \in [0, 1)$ is the discount factor,
- $\pi_{\boldsymbol{\theta}}$ is the parameterized policy,
- $\mathbf{P}_\pi^* = (\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}$ is the discounted resolvent (inverse Bellman operator),
- $\mathbf{r}_\pi \in \mathbb{R}^{|\mathcal{S}|}$ is the reward vector under policy π.

However, the distribution over states under this discounted formulation is no longer a proper probability distribution. Instead, it is a weighted occupancy distribution that reflects how often states are visited, discounted over time. Formally, this state distribution is:

$$\mu_\gamma(s) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \Pr(S_t = s),$$

which corresponds to a *geometric distribution* over state visitation (trial until first success). This reflects that recent states are weighted more heavily than distant ones.

2.1 How to Sample a State from the Discounted Distribution

Let $\widetilde{P}_\pi^\gamma \propto (1 - \gamma) P_\pi^\gamma$ denote the normalized discounted state distribution. To sample a single state $S \sim \widetilde{P}_\pi^\gamma$, follow this procedure:

1. Sample a length $L \sim \mathrm{Geo}(1 - \gamma)$, representing the number of steps to run the episode. This reflects the discounted weighting of time steps.
2. Run an episode (trajectory) from $t = 0$ to $t = L - 1$ using the current policy $\pi_{\boldsymbol{\theta}}$.
3. Let the sampled state be:

$$S_{L-1}.$$

   All previous states can be discarded. However, in practice, people often use *all* states $\{S_0, \ldots, S_{L-1}\}$ to construct an unbiased gradient estimate more efficiently.

This method allows sampling from the discounted state distribution using a simple rejection-free procedure based on the geometric distribution.

2.2 Gradient of the Discounted Performance

When differentiating the performance objective with respect to parameters $\boldsymbol{\theta}$, a challenge arises:

- The derivative of the state distribution $\mu_\gamma(s)$ with respect to $\boldsymbol{\theta}$ is unknown.
- This term is not sampling friendly, which makes gradient estimation difficult.

To illustrate the problem, consider the gradient of the value function from a start state s_0:

$$\nabla_{\boldsymbol{\theta}} v_\gamma^\pi(s_0) = \sum_{s \in \mathcal{S}} (\mathbf{P}_\pi^\gamma)(s \mid s_0) \sum_{a \in \mathcal{A}} q_\gamma^\pi(s, a) \nabla_{\boldsymbol{\theta}} \log \pi(a \mid s; \boldsymbol{\theta}).$$

This formulation shows two key difficulties:

- The discounted transition matrix $\mathbf{P}_\pi^\gamma$ is not a proper probability distribution over states because its rows do not sum to 1.
- The dependency on future state visitation makes $\nabla_{\boldsymbol{\theta}} v_\gamma^\pi(s_0)$ nontrivial to estimate via sampling, since it relies on **off-policy-like expectations** over future trajectories.

Despite this, we can still derive a useful result. The **Policy Gradient Theorem** states:

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu_\gamma(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|s; \boldsymbol{\theta}) q^\pi(s, a),$$

where the gradient does *not* involve the derivative of the state distribution $\mu_\gamma(s)$, and the proportionality constant depends on γ.

This result allows us to construct stochastic estimates of the policy gradient using sample trajectories.

2.3 Sampling-Friendly Estimation

Using the identity:

$$\nabla_{\boldsymbol{\theta}} \pi(a \mid s; \boldsymbol{\theta}) = \pi(a \mid s; \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log \pi(a \mid s; \boldsymbol{\theta}),$$

we can rewrite the gradient as:

$$\nabla J(\boldsymbol{\theta}) \propto \mathbb{E}_\pi[q^\pi(s, a) \nabla_{\boldsymbol{\theta}} \log \pi(a \mid s; \boldsymbol{\theta})],$$

which forms the basis of REINFORCE and other policy gradient algorithms.

2.4 REINFORCE with Discounted Return

In practice, for episodic tasks, we define the update rule as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \gamma^t G_t \nabla_{\boldsymbol{\theta}} \log \pi(A_t \mid S_t; \boldsymbol{\theta}_t),$$

where $G_t = \sum_{k=t}^T \gamma^{k-t} R_{k+1}$ is the sampled discounted return. This form emphasizes the time-weighted contribution of each reward.

2.5 Geometric Distribution Intuition

The discounted state distribution $\mu_\gamma(s)$ resembles a geometric distribution with success probability $1 - \gamma$, as it defines the likelihood of visiting a state within discounted trials. This interpretation helps justify the form of the weighting in the gradient expression.

2.6 Conclusion

Although the discounted reward setting does not induce a proper probability distribution over states, the policy gradient theorem provides a powerful result that avoids this difficulty by leveraging the structure of the expected return and enabling sampling-based learning algorithms like REINFORCE.

3 Citations and References