Reinforcement Learning **Lecture–7**

Policy Gradient Methods

Anubhav Gupta

All the methods covered till now are action-value methods, which learn/approximate the values of actions and then use some ϵ-greedy policy to select the best action. Let us now consider methods to learn the policy and select actions without consulting the value function every time. Value function might still be used to learn the policy, but once the policy is learned value function is not required to select the actions.

Reinforcement learning methods can be categorized as follows depending on how the policy is learned:

- **Value based methods:** Value based methods learn the value function and then use techniques such as ϵ-greedy to get the policy.
- **Policy based methods:** These methods directly learn the policy without computing the value functions.
- **Actor-Critic methods:** Actor-Critic methods learn the policy by also learning the value functions explicitly.

Policy based methods can learn stochastic policies and they usually have better convergence properties than value-based methods. These methods are also useful in high-dimensional or continuous action spaces. These methods, however, typically converge to a local rather than global optimum. Evaluating a policy explicitly is often inefficient and exhibit high variance.

1 Policy Objective Functions

Let's denote by $\theta \in \mathbb{R}^{d'}$ parameter vector for the policy. Thus the probability to take action a at time t given that we are in state s is given by

$$\pi_\theta(s, a) = \pi(a \mid s, \theta) = P(A_t = a \mid S_t = s, \theta_t = \theta)$$

Several loss functions can be used to measure the quality of a policy π_θ.

- In episodic environments we can use the **value of start state**

$$J_1(\theta) = V^\pi\theta(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

- In continuing environments we can use **average value**

$$J_{avV}(\theta) = \sum_s d^\pi_\theta(s) V^\pi_\theta(s)$$

  as well as **average reward** per time-step

$$J_{avR} = \sum_s d^\pi_\theta(s) \sum_a \pi_\theta(s, a) \mathcal{R}^a_s$$

  Here d^{π_θ} is the stationary distribution of Markov chain for π_θ.

Now that we have loss function $J(\theta)$, the objective of policy based learning is to find parameter vector θ that maximizes $J(\theta)$. For this purpose, methods such as hill climbing, simplex, which don't use gradient can be used. However, better performance is achieved by gradient based methods for optimizing $J(\theta)$ such as gradient descent, quasi-newton methods, etc.

2 Policy Gradient

Policy gradient methods seek to find the optimal parameters θ by searching for a local maximum by ascending in the direction of gradient of $J(\theta)$ w.r.t. parameter vector θ. i.e.

$$\delta\theta = \alpha\nabla_\theta J(\theta)$$

where $\nabla_\theta J(\theta)$ is the policy gradient

$$\nabla_\theta J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \frac{\partial J(\theta)}{\partial \theta_2} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta'_d} \end{bmatrix}$$

and α is step size parameter.

2.1 Score Function

Let's assume that the policy π_θ is differentiable whenever it is non-zero. Then we can use the trick of *likelihood ratios* to get following identity:

$$\begin{aligned} \nabla_\theta \pi_\theta(s,a) &= \pi_\theta(s,a)\frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} \\ &= \pi_\theta(s,a)\nabla_\theta \log\, \pi_\theta(s,a) \end{aligned}$$

the quantity $\nabla_\theta \log\, \pi_\theta(s,a)$ is known as the *score function*. Let's now compute score functions for some of the simple policies.

2.1.1 Softmax Policy

Let $\phi(s,a)$ denote the features for state-action pair (s,a) and actions are weighted using linear combination of features $\phi(s,a)^T\theta$. In softmax policy, probability of an action is proportional to exponent of it's weight. i.e.

$$\pi_\theta(s,a) = \frac{e^{\phi(s,a)^T\theta}}{\sum_{a'} e^{\phi(s,a')^T\theta}}$$

The score function is now computed as follows:

$$\begin{aligned} \nabla_\theta \log\, \pi_\theta(s,a) &= \nabla_\theta \phi(s,a)^T\theta - \nabla_\theta \log \sum_{a'} e^{\phi(s,a')^T\theta} \\ &= \phi(s,a) - \frac{\sum_{a'} e^{\phi(s,a')^T\theta}\phi(s,a')}{\sum_{a'} e^{\phi(s,a')^T\theta}} \\ &= \phi(s,a) - \sum_{a'} \pi_\theta(s,a')\phi(s,a') \\ &= \phi(s,a) - \mathbb{E}_{\pi_\theta}\phi(s,\cdot) \end{aligned}$$

2.1.2 Gaussian Policy

Gaussian policy is a natural policy for continuous action spaces. Mean action for a state s is linear combination of state features $\mu(s) = \phi(s)^T\theta$ with fixed variance. Actions in state s are then selected based on this probability distribution $a \sim \mathcal{N}(\mu(s), \sigma^2)$. Score function of Gaussian policy is given by

$$\begin{aligned} \nabla_\theta \log\, \pi_\theta(s,a) &= \nabla_\theta \log\, P(a \mid s,\theta) \\ &= \nabla_\theta \log\, \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(a-\mu(s))^2}{2\sigma^2}} \\ &= \nabla_\theta - \frac{1}{2\sigma^2}(a-\mu(s))^2 \\ &= \frac{1}{\sigma^2}(a-\mu(s))\nabla_\theta \phi(s)^T\theta \\ &= \frac{(a-\mu(s))\phi(s)}{\sigma^2} \end{aligned}$$

2.2 Policy Gradient Theorem

Let's consider the simplified episodic environment of one-step MDP where we always start in state $s \sim d(s)$ and the MDP terminates after one time-step with reward $r = \mathcal{R}_{s,a}$. Then loss function for the policy parameter vector is

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\pi_\theta}[r] \\ &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(s,a) \mathcal{R}_{s,a} \end{aligned}$$

Then we can use the likelihood ratios to formulate the policy gradient as follows:

$$\begin{aligned} \nabla_\theta J(\theta) &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(s,a) \nabla_\theta \log \pi_\theta(s,a) \mathcal{R}_{s,a} \\ &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) r\right] \end{aligned}$$

Theorem. *The policy gradient theorem states that for any differential policy $\pi_\theta(s,a)$ and for any policy objective objective $J = J_1, J_{avR},$ or $\frac{1}{1-\gamma} J_{avV}$, the policy gradient is given by*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta log\ \pi_\theta(s,a) Q^{\pi_\theta}(s,a)\right]$$

Policy gradient theorem generalizes the likelihood ratio approach to multi-step MDPs. it applies to all the objective functions mentioned earlier - start state objective, average reward objective and average value objective.

3 Monte-Carlo Policy Gradient (REINFORCE)

REINFORCE is a policy gradient algorithm which updates parameters by stochastic gradient ascent. The gradient of loss function $J(\theta)$ is given by the policy gradient theorem. REINFORCE is a Monte-Carlo algorithm which uses return G_t as an unbiased estimate of $Q^{\pi_\theta}(s_t, a_t)$. The classical REINFORCE algorithm is stated in algorithm-1.

Algorithm 1: REINFORCE algorithm (episodic control) for estimating $\pi_\theta \approx \pi_*$

Input: A differential policy parameterization $\pi(a \mid s, \theta)$
Algorithm Parameters: Step size $\alpha \in (0, 1]$
Initialization: Initialize policy parameter vector $\theta \in \mathbb{R}^{d'}$ arbitrarily

```
for each episode until convergence do
    Generate an episode S_0, A_0, R_1, S_1, ..., S_{T-1}, A_{T-1}, R_T following policy π(· | ·, θ)
    for each time-step t = 0, 1, ..., T − 1 of episode do
        G ← Σ_{k=t+1}^{T} γ^{k−t−1} R_k
        θ ← θ + α γ^t G ∇_θ log π(A_t | S_t, θ)
    end
end
```

3.1 REINFORCE with Baseline

Monte-Carlo policy gradient methods, such as REINFORCE exhibit high variance. One common way to reduce variance of policy gradient methods is to use a baseline function. The idea is to subtract some baseline function from policy gradient such that it reduces the variance without changing the expectation.

$$\begin{aligned} \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) B(s)\right] &= \sum_{s \in \mathcal{S}} d^{\pi_\theta}(s) \sum_a \nabla_\theta \pi_\theta(s,a) B(s) \\ &= \sum_{s \in \mathcal{S}} d^{\pi_\theta}(s) B(s) \nabla_\theta \sum_a \pi_\theta(s,a) \\ &= 0 \end{aligned}$$

So, any function that doesn't depend on actions can be chosen as a baseline function $B(s)$. A good baseline function is the state value function $B(s) = V^{\pi_\theta}(s)$. For this baseline function, the policy gradient theorem can be rewritten using the **advantage function** $A^{\pi_\theta}(s,a)$.

$$A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$
$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log\, \pi_\theta(s,a) A^{\pi_\theta}(s,a)\right]$$

The advantage function can be interpreted as telling how much better than usual it is to take action a from state s then being in state s in general. Let us approximate the state-value function $V^{\pi_\theta}(s)$ by parameter vector $\mathbf{w}$. By using this advantage function, REINFORCE algorithm with baseline is given in algorithm-2.

Algorithm 2: REINFORCE algorithm with Baseline (episodic control) for estimating $\pi_\theta \approx \pi_*$

Input: A differential policy parameterization $\pi(a \mid s, \theta)$
A differential state-value function parameterization $\hat{v}(s, \mathbf{w})$
Algorithm Parameters: Step sizes $\alpha^\theta \in (0,1]$, $\alpha^{\mathbf{w}} \in (0,1]$
Initialization: Initialize policy parameter vector $\theta \in \mathbb{R}^{d'}$, state-value function parameter vector $\mathbf{w} \in \mathbb{R}^d$ arbitrarily

for *each episode until convergence* **do**
  Generate an episode $S_0, A_0, R_1, S_1, \cdots, S_{T-1}, A_{T-1}, R_T$ following policy $\pi(\cdot \mid \cdot, \theta)$
  for *each time-step* $t = 0, 1, \cdots, T-1$ *of episode* **do**
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
    $\delta \leftarrow G - \hat{v}(S_t, \mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla_{\mathbf{w}} \hat{v}(S_t, \mathbf{w})$
    $\theta \leftarrow \theta + \alpha^\theta \gamma^t \delta \nabla_\theta \log\, \pi(A_t \mid S_t, \theta)$
  end
end

4 Actor-Critic Methods

Actor-Critic methods learn two sets of parameters. *Critic* updates action-value function parameters, whereas parameters learned by *actor* are used for policy approximation and depend on parameters suggested by critic. Even though REINFORCE-with-baseline algorithm discussed above learns both a policy and a state-value function, this algorithm is not considered to be actor-critic. This is because the state value function is not used for bootstrapping, but only as a baseline for the state whose estimate is being updated. The outline of actor-critic algorithms is follows:

- Critic estimates the action-value function

$$Q_{\mathbf{w}}(s,a) \approx Q^{\pi_\theta}(s,a)$$

- Actor updates the policy estimate by using this approximate policy gradient

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log\, \pi_\theta(s,a) Q_{\mathbf{w}}(s,a)\right]$$
$$\Delta\theta = \alpha \nabla_\theta \log\, \pi_\theta(s,a) Q_{\mathbf{w}}(s,a)$$

The problem that critic solves is identical to policy evaluation. Critic measures how good is policy π_θ for current parameters θ. For this purpose, we can again use all the methods previously discussed such as MC, TD(0), TD(λ), least-squares methods, etc.

Similar to REINFORCE-with-baseline algorithm, the advantage function instead of the action-value function can significantly reduce variance of policy gradient. So, the critic should really estimate the advantage function. This can be achieved by estimating both $V^{\pi_\theta}(s)$ and $Q^{\pi_\theta}(s,a)$. This can be achieved by using two approximators and hence two parameter vectors $\mathbf{w_1}$ and $\mathbf{w_2}$, then updating both parameter vectors by e.g. TD learning.

4.1 One-step Actor-Critic Methods

The main advantage of one-step methods is that they are fully online and incremental. For true value function $V^{\pi_\theta}(s)$, the TD error

$$\delta^{\pi_\theta} = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$$

is an unbiased estimate of the advantage function. i.e.

$$\begin{aligned}\mathbb{E}_{\pi_\theta}\left[\delta^{\pi_\theta} \mid s, a\right] &= \mathbb{E}_{\pi_\theta}\left[r + \gamma V^{\pi_\theta}(s') \mid s, a\right] - V^{\pi_\theta}(s) \\ &= Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s) \\ &= A^{\pi_\theta}(s, a)\end{aligned}$$

So, TD error is used to compute the policy gradient. i.e.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log\, \pi_\theta(s, a)\delta^{\pi_\theta}\right]$$

One-step actor-critic method uses the approximate TD error

$$\delta_{\mathbf{w}} = r + \gamma\hat{v}(s', \mathbf{w}) - \hat{v}(s, \mathbf{w})$$

This only requires one set of critic parameters $\mathbf{w}$. The pseudocode for one-step actor-critic methods is given in algorithm-3.

Algorithm 3: One-step Actor-Critic (episodic control) for estimating $\pi_\theta \approx \pi_*$

Input: A differential policy parameterization $\pi(a \mid s, \theta)$
A differential state-value function parameterization $\hat{v}(s, \mathbf{w})$
Algorithm Parameters: Step sizes $\alpha^\theta \in (0, 1]$, $\alpha^{\mathbf{w}} \in (0, 1]$
Initialization: Initialize policy parameter vector $\theta \in \mathbb{R}^{d'}$, state-value function parameter vector $\mathbf{w} \in \mathbb{R}^d$ arbitrarily

for *each episode until convergence* **do**
  Initialize S (first state of the episode)
  $I \leftarrow 1$
  for *each step of episode* **do**
    $A \sim \pi(\cdot \mid S, \theta)$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma\hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\nabla_{\mathbf{w}}\hat{v}(S, \mathbf{w})$
    $\theta \leftarrow \theta + \alpha^\theta I\delta\nabla_\theta \log\, \pi(A \mid S, \theta)$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$
  end
end

4.2 Actors and Critics at Different Time-Scales

Critic can estimate value function $V_\theta(s)$ from many targets at different time-scales.

- For Monte-Carlo, the target is the return G_t (also used by REINFORCE-with-baseline)

$$\Delta\mathbf{w} = \alpha(G_t - V_\theta(s))\nabla_{\mathbf{w}}\hat{v}(s, \mathbf{w})$$

- For TD(0), the target is the TD target

$$\Delta\mathbf{w} = \alpha(r + \gamma\hat{v}(s', \mathbf{w}) - \hat{v}(s, \mathbf{w})\nabla_{\mathbf{w}}\hat{v}(s, \mathbf{w})$$

- For TD(λ), the target is λ-return G_t^λ

$$\Delta\mathbf{w} = \alpha(G_t^\lambda - V_\theta(s))\nabla_{\mathbf{w}}\hat{v}(s, \mathbf{w})$$

4.3 Actor-Critic with Eligibility Traces

The forward-views of n-step and TD(λ) actor-critic methods can be generalized using eligibility traces. The idea is to keep separate eligibility traces for the actor and critic. The complete pseudocode for actor-critic method with eligibility traces is given in algorithm-4.

Algorithm 4: Actor-Critic with eligibility traces (episodic control) for estimating $\pi_\theta \approx \pi_*$

Input: A differential policy parameterization $\pi(a \mid s, \theta)$
A differential state-value function parameterization $\hat{v}(s, \mathbf{w})$
Algorithm Parameters: Trace-decay rates $\lambda^\theta \in [0,1]$, $\lambda^{\mathbf{w}} \in [0,1]$, Step sizes $\alpha^\theta \in (0,1]$, $\alpha^{\mathbf{w}} \in (0,1]$
Initialization: Initialize policy parameter vector $\theta \in \mathbb{R}^{d'}$, state-value function parameter vector $\mathbf{w} \in \mathbb{R}^d$ arbitrarily

for *each episode until convergence* **do**
  Initialize S (first state of the episode)
  $\mathbf{z}^\theta \leftarrow \mathbf{0}$ (d'-component ET vector)
  $\mathbf{z}^w \leftarrow \mathbf{0}$ (d-component ET vector)
  $I \leftarrow 1$
  for *each step of episode* **do**
    $A \sim \pi(\cdot \mid S, \theta)$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$
    $\mathbf{z}^{\mathbf{w}} \leftarrow \gamma \lambda^{\mathbf{w}} \mathbf{z}^{\mathbf{w}} + \nabla_{\mathbf{w}} \hat{v}(S, \mathbf{w})$
    $\mathbf{z}^\theta \leftarrow \gamma \lambda^\theta \mathbf{z}^\theta + I \nabla_\theta \log \pi(A \mid S, \theta)$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \mathbf{z}^{\mathbf{w}}$
    $\theta \leftarrow \theta + \alpha^\theta \delta \mathbf{z}^\theta$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$
  end
end

Approximating the policy gradient using actor-critic method introduces bias and a biased policy gradient may not find the best solution. However if the value function approximation is chosen carefully, then we can avoid introducing any bias and we can still follow the exact policy gradient.

Theorem. *If the following two conditions are satisfied:*

1. *Value function approximator is compatible to the policy*

$$\nabla_{\mathbf{w}} Q_{\mathbf{w}}(s,a) = \nabla_\theta log\ \pi_\theta(s,a)$$

2. *Value function parameters* $\mathbf{w}$ *minimize the mean squared error*

$$\epsilon = \mathbb{E}_{\pi_\theta}\left[(Q^{\pi_\theta}(s,a) - Q_{\mathbf{w}}(s,a))^2\right]$$

Then the policy gradient is exact, i.e.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta log\ \pi_\theta(s,a) Q_{\mathbf{w}}(s,a)\right]$$

This theorem is called ***compatibility function approximation theorem****.*

References

[1] Reinforcement Learning: An Introduction (2nd Edition), by Richard S. Sutton, Andrew G. Barto.

[2] David Silver's course "Introduction to Reinforcement Learning" available at `https://www.davidsilver.uk/teaching/`