Machine Learning Lecture Notes 11/18

November 2025

Contents

1 Tackling Unknown MDPs with Learning

1.1 Monte Carlo Evaluation

The Bellman Equation will be less effective when the MDP becomes unknown. Idea: Learning with samples (s, a, r, s').

1. Value-based methods: Avoid learning p or R directly. Learn v_π or q_π through samples.

2. Model-based methods: Learn p and R directly.

We usually adopt the first method. In fact, we can define Monte Carlo estimation to estimate the expectation with empirical average.

Definition 1 *We define the MC estimation of* $v_\pi(s) = E_\pi[G_t \mid s_t = s]$ *as*

$$V(s)\frac{\sum_{i=1}^{n} G^{(i)}}{n},$$

where $G^{(i)}$ *represents the yield value of the* i*-th episode.*

Notice that we don't need to store the value of every $G^{(i)}$ to do that estimation, in fact, we only need the result of last episode, as this estimation can be transformed into an incremental update

$$V(s) \leftarrow V(s) + \frac{1}{n}(G^{(n)} - V(s)).$$

We can define the MC estimation of $q_\pi(s, a)$ in the similar way.

Definition 2 *We define the MC estimation of* $q_\pi(s, a) = E_\pi[G_t \mid s_t = s, a_t = a]$ *as*

$$Q(s, a)\frac{\sum_{i=1}^{n} G^{(i)}}{n},$$

where $G^{(i)}$ *represents the yield value of the* i*-th episode.*

It can be transformed into an incremental update

$$Q(s,a) \leftarrow Q(s,a) + \frac{1}{n}(G^{(n)} - Q(s,a)).$$

The estimation pipeline given a policy π is described as follows.
Goal: Input π, output V(an estimate of V_π). Initialize Q arbitrarily (0 for terminal states), then loop until convergence:

1. Collect an episode with π
2. For $t = 0, 1, \cdots$:
   (a) Compute return G_t
   (b) Increase counter: $N(s_t) \leftarrow N(s_t) + 1$
   (c) Increase total value: $V(s_t) \leftarrow V(s_t) + \frac{1}{N(s_t)}(G_t - V(s_t))$

Goal: Input π, output Q(an estimate of q_π). Initialize Q arbitrarily (0 for terminal states), then loop until convergence:

1. Collect an episode with π
2. For $t = 0, 1, \cdots$:
   (a) Compute return G_t
   (b) Increase counter: $N(s_t, a_t) \leftarrow N(s_t, a_t) + 1$
   (c) Increase total value: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \frac{1}{N(s_t, a_t)}(G_t - Q(s_t, a_t))$

Usually, we need to change the policy according to our estimation. The implicit greedy policy requires q_π. However, as

$$q_\pi(s_t, a_t) = R(s_t, a_t) + \gamma E_{s_{t+1} \sim p(\cdot|s_t, a_t)}[v_\pi(s_{t+1})]$$

we cannot get q_π explicitly by knowing v_π. Therefore, we usually estimate q_π. article graphicx amsmath amsfonts amsthm

2 Off-policy Correction with Importance Sampling

Goal: guarantee sufficient samples for every state-action pair.
A common policy is ϵ-greedy:

$$\pi(a|s) = \{\, 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}|} a = argmax_{a \in \mathcal{A}}\ q(s,a) \frac{\epsilon}{|\mathcal{A}|} otherwise$$

But sometimes π is not ϵ-soft. Now let us consider how Monte Carlo methods can estimate a non-exploratory policy.
There are two roles of policy:

1. Target policy π: the policy to learn.

2. Behavior policy π_b: the policy used to collect samples. Based on π but could be more exploratory.

An RL method is on-policy(off-policy) when $\pi_b = \pi$(else).
Off-policy methods allow for sample reuse after policy update. It is more general, but often converges slower.

Lemma 1

$$E_{x\sim p(\cdot)}[f(x)] = E_{x\sim q(\cdot)}[\frac{p(x)}{q(x)}f(x)]$$

Proof:

$$E_{x\sim p(\cdot)}[f(x)] = \int p(x)f(x)dx = \int q(x)\frac{p(x)}{q(x)}f(x)dx = E_{x\sim q(\cdot)}[\frac{p(x)}{q(x)}f(x)]$$

■

Theorem 1 *For any function f of τ,*

$$E_{\pi}[f(\tau)] = E_{\pi_b}[(\prod_{t=0}^{\infty}\frac{\pi(a_t|s_t)}{\pi_b(a_t|s_t)})f(\tau)]$$

Proof:

$$p_{\pi}(\tau) = \rho_0(S_0)\prod_{t=0}^{\infty}\pi(a_t|s_t)p(s_{t+1}|s_t,a_t)$$

$$\frac{p_{\pi}(\tau)}{p_{\pi_b}(\tau)} = \prod_{t=0}^{\infty}\frac{\pi(a_t|s_t)}{\pi_b(a_t|s_t)}$$

$$E_{\pi}[f(\tau)] = E_{\pi_b}[\frac{p_{\pi}(\tau)}{p_{\pi_b}(\tau)}f(\tau)] = E_{\pi_b}[(\prod_{t=0}^{\infty}\frac{\pi(a_t|s_t)}{\pi_b(a_t|s_t)})f(\tau)]$$

■

Remarks:

1. With IS, samples collected by π_b can be used to estimate E_{π}.

2. π_b must have greater empirical support ($\pi(a|s) \geq \epsilon \Rightarrow \pi_b(a|s) \geq \epsilon for small \epsilon$). This holds since π_b is more exploratory.

3. IS is empirically unstable when π_b is very different from π. It can only correct the distribution of "slightly" off-policy samples.

article amsmath amssymb algorithm algpseudocode booktabs multirow

3 Temporal Difference Learning

3.1 Core Idea

Combines the advantages of Monte Carlo (sampling) and Dynamic Programming (bootstrapping):

- **MC update**: $V(s_t) \leftarrow V(s_t) + \frac{1}{N(s_t)}(G_t - V(s_t))$
- **DP update**: $V \leftarrow B_\pi V$
- **TD update**: $V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$

3.2 n-step Return

Define n-step return:

$$G_t^{(n)} := r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n})$$

Special cases: n = 1 (TD): $\mathrm{G}_t^{(1)} = r_{t+1} + \gamma V(s_{t+1})$
$n = 2 : G_t^{(2)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 V(s_{t+2})$
$n = \infty \quad (MC) : G_t^{(\infty)} = \sum_{k=0}^{\infty} \gamma^k r_{t+1+k}$
Properties:

- As $n \to \infty$: bias decreases, variance increases

3.3 λ-return

Weighted combination of all n-step returns:

$$G_t^\lambda := (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

Characteristics:

- $\lambda = 0$: TD return (high bias, low variance)
- $\lambda = 1$: MC return (no bias, high variance)
- $\lambda \in (0, 1)$: trade-off between bias and variance

4 Temporal Difference Control

4.1 TD Target Construction

Control tasks estimate q_π, TD target:

$$r_{t+1} + \gamma Q(s_{t+1}, a')$$

4.2 On-policy Method: Sarsa

Sarsa: On-policy TD Control [1] Initialize Q arbitrarily (0 for terminal states) $s_0 \sim \rho_0(\cdot)$ $a_0 \sim \pi_b(\cdot|s_0)$ $t = 0, 1, \ldots$ until terminal Take a_t, observe r_{t+1}, s_{t+1} $a_{t+1} \sim \pi_b(\cdot|s_{t+1})$ $\hat{q}_t \leftarrow r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$ $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(\hat{q}_t - Q(s_t, a_t))$ convergence

4.3 Off-policy Method: Q-learning

Q-learning: Off-policy TD Control [1] Initialize Q arbitrarily (0 for terminal states) $s_0 \sim \rho_0(\cdot)$ $t = 0, 1, \ldots$ until terminal $a_t \sim \pi_b(\cdot|s_t)$ Take a_t, observe r_{t+1}, s_{t+1} $\hat{q}_t \leftarrow r_{t+1} + \gamma \max_{a' \in A} Q(s_{t+1}, a')$ $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(\hat{q}_t - Q(s_t, a_t))$ convergence

4.4 Key Differences

Method	Policy Type	TD Target
Sarsa	On-policy	$r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$
Q-learning	Off-policy	$r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a')$

[11pt]article
amsmath, amssymb geometry enumitem hyperref
margin=1in

5 Deep RL

5.1 From Tabular MDPs to Deep RL

- Tabular RL assumes small, finite state and action spaces with known transition and reward functions.
- Real-world domains (e.g., Go, robotics) involve vast or continuous state spaces, making tabular representation infeasible.
- Deep neural networks replace lookup tables to model value functions or policies, providing generalization to unseen states.

5.2 Deep RL: Representation and Optimization

- **Representation:** Replace tabular $Q(s, a)$ with parametric networks Q_ϕ.
- **Optimization:** Move from single-entry updates to mini-batch stochastic gradient descent.
- Training objective:

$$L(\phi) = \sum_i \left(\hat{q}^{(i)} - Q_\phi(s^{(i)}, a^{(i)})\right)^2.$$

5.3 Naive Deep Q-Learning

– Uses bootstrapped targets:

$$\hat{q}_t = r_{t+1} + \gamma \max_{a'} Q_\phi(s_{t+1}, a').$$

– Key issues:

[label=(3)]

1. **Correlated samples** due to sequential trajectories.
2. **Non-stationary targets** because the same network provides predictions and targets.

5.4 Deep Q-Networks (DQN)

– **Experience Replay:** Store transitions in a replay buffer; sample randomly to reduce correlations.

article graphicx amssymb

6 Issues of Naive Deep Q-learning

- Correlated samples ...
- Non-stationary targets $\rightarrow$ target network

Maintain another network $Q_{\phi'}$ to provide stationary TD targets spnanning multiple iterations. Synchronize $\phi^{'}$ with ϕ periodically.

6.1 DQN Algorithm

1. Initialize Q network Q_ϕ, target network $Q_{\phi'}$ and experience replay $\mathcal{D} = \phi$, then loop:

2. Collect data: Execute π_b to collect a dataset $\{(s_i, a_i, r_i, s_i^{'})\}$, append to $\mathcal{D}$.

3. Update network:

3.1 Sample a batch of data randomly from $\mathcal{D}$:

$$\{(s_i, a_i, r_i, s_i^{'})\} \sim \mathcal{D}$$

3.2 Update on the batch:

$$\hat{q}_t \leftarrow r_{t+1} + \gamma \max_{a^{'} \in \mathcal{A}} Q_{\phi'}(s_{t+1}, a^{'})$$

$$\mathcal{L}_\phi \leftarrow \sum_i (\hat{q}_i - Q_\phi(s_i, a_i))^2$$

$$\phi \leftarrow \phi - \alpha \nabla_{\phi} \mathcal{L}_{\phi}$$

4. Every several iterations, update the target network:

$$\phi^{'} \leftarrow \phi$$

6.2 Policy-based Methods

Drawbacks of value-based methods:

- Not appliable to large or continuous $\mathcal{A}$

  Policy extraction requires enumerating over $\mathcal{A}$.

- Difficult to utilize policy priors

  There are often policy priors too valuable to waste.

- Instable improvement of π over efforts

  $\pi^{'} \geq \pi$ don't hold under function approximation. More learning don't guarantee a better policy.

- Limited representation for stochastic policies

  Greedy policies is suboptimal for exploration or modeling diverse behaviors.

Policy-based methods: Parametrize a stochastic policy π_{θ}, directly optimize it through gradient ascend on the RL objective:

$$J(\pi_{\theta}) = E_{\pi_{\theta}}[\sum_{t=0}^{\infty} \gamma^{t} r_{t+1}]$$

$$\theta \leftarrow \theta + \alpha \hat{\nabla}_{\theta} J(\pi_{\theta})$$

article amsmath,amssymb geometry a4paper, margin=1in

7 Policy Gradient Theorem

7.1 Theorem Statement

The gradient of the policy performance function $J(\pi_{\theta})$ is given by:

$$\nabla_{\theta} J\left(\pi_{\theta}\right) = E_{\pi_{\theta}}\left[\left(\sum_{t=0}^{\infty} \nabla_{\theta} \log \pi_{\theta}\left(a_{t} | s_{t}\right)\right)\left(\sum_{t=0}^{\infty} \gamma^{t} r_{t+1}\right)\right]$$

7.2 Key Lemma

For a trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots)$, the gradient of the log-probability of the trajectory under policy π_θ satisfies:

$$\nabla_\theta \log p_{\pi_\theta}(\tau) = \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta \left(a_t | s_t\right)$$

7.2.1 Proof of the Lemma

The probability of trajectory τ is:

$$p_{\pi_\theta}(\tau) = \rho_0 \left(s_0\right) \prod_{t=0}^{\infty} \pi_\theta \left(a_t | s_t\right) p \left(s_{t+1} | s_t, a_t\right)$$

Taking the logarithm of both sides:

$$\log p_{\pi_\theta}(\tau) = \log \rho_0 \left(s_0\right) + \sum_{t=0}^{\infty} \log \pi_\theta \left(a_t | s_t\right) + \sum_{t=0}^{\infty} \log p \left(s_{t+1} | s_t, a_t\right)$$

Since $\rho_0(s_0)$ (initial state distribution) and $p(s_{t+1}|s_t, a_t)$ (state transition probability) are independent of θ, their gradients vanish. Thus:

$$\nabla_\theta \log p_{\pi_\theta}(\tau) = \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta \left(a_t | s_t\right)$$

7.3 Derivation of the Theorem

Starting from the definition of the performance function (expected discounted cumulative reward):

$$J(\pi_\theta) = E_{\pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \right]$$

Convert the expectation to an integral over trajectories:

$$J(\pi_\theta) = \int p_{\pi_\theta}(\tau) \left(\sum_{t=0}^{\infty} \gamma^t r_{t+1} \right) d\tau$$

Take the gradient with respect to θ and interchange gradient and integral:

$$\nabla_\theta J(\pi_\theta) = \int \nabla_\theta p_{\pi_\theta}(\tau) \left(\sum_{t=0}^{\infty} \gamma^t r_{t+1} \right) d\tau$$

Use the identity $\nabla_\theta p(\tau) = p(\tau) \nabla_\theta \log p(\tau)$:

$$\nabla_\theta J(\pi_\theta) = \int p_{\pi_\theta}(\tau) \nabla_\theta \log p_{\pi_\theta}(\tau) \left(\sum_{t=0}^{\infty} \gamma^t r_{t+1} \right) d\tau$$

Convert back to expectation over trajectories under π_θ:

$$\nabla_\theta J(\pi_\theta) = E_{\pi_\theta}\left[\nabla_\theta \log p_{\pi_\theta}(\tau)\left(\sum_{t=0}^{\infty}\gamma^t r_{t+1}\right)\right]$$

Substitute the key lemma to obtain the final theorem:

$$\nabla_\theta J(\pi_\theta) = E_{\pi_\theta}\left[\left(\sum_{t=0}^{\infty}\nabla_\theta \log \pi_\theta\left(a_t|s_t\right)\right)\left(\sum_{t=0}^{\infty}\gamma^t r_{t+1}\right)\right]$$

8 REINFORCE Algorithm (Reward-to-go Policy Gradient)

REINFORCE is a model-free policy gradient algorithm based on the above theorem. The steps are as follows:

8.1 Initialization

Initialize the policy parameter θ (e.g., random initialization).

8.2 Data Collection

Execute the current policy π_θ to collect a dataset of trajectories:

$$\{\tau_i\}_{i=1}^{n} \sim \pi_\theta, \quad \tau_i = (s_{i,0}, a_{i,0}, r_{i,1}, s_{i,1}, a_{i,1}, r_{i,2}, \ldots)$$

8.3 Compute Reward-to-go

For each trajectory τ_i and each time step t, compute the discounted cumulative reward (reward-to-go):

$$G_{i,t} = \sum_{k=0}^{\infty}\gamma^k r_{i,t+1+k}$$

8.4 Policy Update

1. Construct the surrogate objective function:

$$\mathcal{L}_\theta = \frac{1}{n}\sum_{i=1}^{n}\sum_{t=0}^{\infty}\log \pi_\theta\left(a_{i,t}|s_{i,t}\right) G_{i,t}$$

2. Update the policy parameter via gradient ascent:

$$\theta \leftarrow \theta + \alpha\nabla_\theta\mathcal{L}_\theta$$

where α is the learning rate.

9 Issues and Improvements of Policy Gradient Methods

9.1 Key Issue

High estimation variance: The outer expectation in the policy gradient theorem is estimated using a finite number of trajectory samples, leading to large variance in gradient estimates.

9.2 Improvement Direction

Reduce variance through modifications such as: - Baseline subtraction (e.g., using value function estimates) - Advantage estimation (e.g., A2C, A3C algorithms) - Trust region constraints (e.g., TRPO, PPO algorithms) article amsmath amssymb algorithm algpseudocode booktabs multirow

10 Policy Gradient Basics

The standard gradient for policy optimization is defined as:

$$\nabla_\theta J(\pi_\theta) = E_{\pi_\theta}\left[\left(\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)\left(\sum_{t=0}^{\infty} \gamma^t r_{t+1}\right)\right]$$

10.1 Issues of Policy Gradient Methods

Traditional policy gradient methods face several challenges:

- **High estimation variance**: Leading to unstable learning.
- **Low sample efficiency**: Requires many samples to estimate the gradient accurately.
- **No monotonic improvement guarantee**: There is no guarantee that each learning step improves the policy.

Improvement: Techniques like Conservative Update are often used to address these issues.

11 Reducing Estimation Variance

To reduce variance, we can rewrite the policy gradient with an explicit "weight" A_t at timestep t:

$$\nabla_\theta J(\pi_\theta) = E_{\pi_\theta}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t) A_t\right]$$

Modifying the form of A_t can lead to reduced estimation variance. Common variations include:

- **Vanilla**: $A_t = G_0$
- **Reward-to-go**: $A_t = G_t$
- **Baseline**: $A_t = G_t - V(s_t)$
- **GAE (λ-return)**: $A_t = GAE(\lambda, V, t)$

12 Reward-to-go

12.1 Definition and Intuition

Applying the reward-to-go modification changes the summation target:

$$A_t : \sum_{k=0}^{\infty} \gamma^k r_{k+1} \to \sum_{k=0}^{\infty} \gamma^k r_{t+1+k}$$

Intuition: Since π is Markov, the rewards $(r_1, r_2, \ldots, r_t)$ are received *before* reaching state s_t and thus should be independent of the action a_t.

12.2 Proof of Validity

We need to prove that removing past rewards does not bias the gradient. For simplicity, let $\gamma = 1$. We aim to show:

$$E_{\pi_\theta}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{k=1}^{t} r_k\right)\right] = 0$$

12.2.1 Lemma

First, we establish the following property:

$$E_{x\sim p_\theta(\cdot)}[\nabla_\theta \log p_\theta(x)] = 0$$

Proof of Lemma: $\mathrm{E}_{x\sim p_\theta(\cdot)}[\nabla_\theta \log p_\theta(x)] = \int p_\theta(x) \nabla_\theta \log p_\theta(x) dx$
$= \int \nabla_\theta p_\theta(x) dx$
$= \nabla_\theta \int p_\theta(x) dx$
$= \nabla_\theta 1$
$= 0$

12.2.2 Main Proof

Using the lemma, we can decompose the expectation: $\mathrm{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{k=1}^{t} r_k\right)\right] =$
$\sum_{t=0}^{\infty} E_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{k=1}^{t} r_k\right)\right]$
$= \sum_{t=0}^{\infty} E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)] E_{\pi_\theta}\left[\sum_{k=1}^{t} r_k\right]$
$= 0$ Since the expectation of the score function $\nabla_\theta \log \pi_\theta(a_t|s_t)$ is 0, the term involving past rewards vanishes. article amsmath amssymb amsfonts mathtools amsthm

13 Improving Policy Gradient

13.1 Issues of Policy Gradient Methods

Policy gradient methods suffer from several limitations:

1. **High estimation variance**: The outer expectation is estimated through only a constant amount of samples.
2. **Low sample efficiency**: Purely on-policy. Collected data are discarded after a single gradient update.
3. **No monotonic improvement guarantee**: No guarantee that each learning step improves the policy.

13.2 Improvement Strategies

Corresponding improvements:

- Variance reduction: Apply several variance reduction modifications
- Sample efficiency: Importance sampling
- Monotonic improvement: Conservative update

14 Reducing Estimation Variance

Rewrite policy gradient with explicit weight A_t at timestep t:

$$\nabla_\theta J(\pi_\theta) = E_{\pi_\theta}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) A_t\right]$$

Modify A_t to reduce estimation variance: $\mathrm{G}_0 \quad vanilla$
$\Rightarrow G_t \quad + reward - to - go$
$\Rightarrow G_t - V(s_t) \quad + baseline$
$\Rightarrow GAE(\tau, V, t) \quad + \lambda$-return

15 Reward-to-go

Apply reward-to-go modification:

$$A_t: \quad \sum_{k=0}^{\infty} \gamma^k r_{k+1} \to \sum_{k=0}^{\infty} \gamma^k r_{t+1+k}$$

Intuition: π is Markov. $(r_1, r_2, \cdots, r_t)$ are received before reaching s_t and thus should be independent of a_t.

Need to prove (let $\gamma = 1$ for simplicity):

$$E_{\pi_\theta}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \left(\sum_{k=1}^{t} r_k\right)\right] = 0$$

15.1 Fundamental Lemma

Lemma 2

$$E_{x\sim p_\theta(\cdot)}[\nabla_\theta \log p_\theta(x)] = 0$$

Proof: $\mathrm{E}_{x\sim p_\theta(\cdot)}[\nabla_\theta \log p_\theta(x)] = \int p_\theta(x)\nabla_\theta \log p_\theta(x)dx$
$= \int \nabla_\theta p_\theta(x)dx$
$= \nabla_\theta \int p_\theta(x)dx$
$= \nabla_\theta 1 = 0$

16 Baseline

Apply baseline modification:

$$A_t : \quad G_t \to G_t - V(s_t)$$

where V can be any function that depends only on s_t.

Intuition: Subtracting a baseline estimates the advantage function:

$$a_\pi(s_t, a_t) := q_\pi(s_t, a_t) - v_\pi(s_t)$$

a_π reflects the relative advantage of taking a_t at s_t, over the expected outcome of following π.

Need to prove:

$$E_{\pi_\theta}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t)V(s_t)\right] = 0$$

17 Generalized Advantage Estimation

Define n-step return:

$$G_t^{(n)} := r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n})$$

Define λ-return:

$$G_t^\lambda := (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

Apply λ-return at advantage estimation:

$$A_t^\lambda := G_t^\lambda - V(s_t)$$

18 Understanding Policy Gradient

$$\nabla_\theta J(\pi_\theta) = E_{\pi_\theta}\left[\sum_{t=0}^{\infty} \underbrace{\nabla_\theta \log \pi_\theta(a_t \mid s_t)}_{likelihooddirection} \underbrace{A_t}_{weight}\right]$$

Policy gradient modifies observed action likelihood based on its advantage function:

- If $A_t > 0$, a_t is above average, gradient ascent increases $\pi_\theta(a_t \mid s_t)$
- If $A_t < 0$, a_t is below average, likelihood is decreased

Major differences with supervised learning (imitation learning):

- Online data: Learn consequences of executing the learning policy itself
- Negative gradient: Push down likelihood on unwanted behaviors

19 Off-policy Policy Gradient with Importance Sampling

Goal: Split the collected dataset into minibatches for multiple gradient updates.

$\nabla_\theta J(\pi_\theta) = E_{\pi_\theta}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) A_t\right]$

$= E_{\pi_{old}}\left[\frac{p_{\pi_\theta}(\tau)}{p_{\pi_{old}}(\tau)} \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) A_t\right]$

$= E_{\pi_{old}}\left[\left(\prod_{t=0}^{\infty} \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}\right) \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) A_t\right]$

$\approx E_{\pi_{old}}\left[\sum_{t=0}^{\infty} \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)} \nabla_\theta \log \pi_\theta(a_t \mid s_t) A_t\right]$

$= E_{\pi_{old}}\left[\sum_{t=0}^{\infty} \nabla_\theta \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)} A_t\right]$

20 Monotonic Improvement

Lemma: Let $\gamma = 1$ for simplicity:

$$J(\pi) = J(\pi_{old}) + E_\pi\left[\sum_{t=0}^{\infty} a_{\pi_{old}}(s, a)\right]$$

Proof: $\mathrm{E}_\pi\left[\sum_{t=0}^{\infty} a_{\pi_{old}}(s_t, a_t)\right] = E_\pi\left[\sum_{t=0}^{\infty} r_t + v_{\pi_{old}}(s'_t) - v_{\pi_{old}}(s_t)\right]$

$= E_\pi\left[-v_{\pi_{old}}(s_0) + \sum_{t=0}^{\infty} r_t\right]$

$= -J(\pi_{old}) + J(\pi)$

If π and π_{old} are close enough, apply importance sampling: $\mathrm{J}(\pi) - J(\pi_{old}) =$

$E_{\pi_{old}}\left[\left(\prod_{t=0}^{\infty} \frac{\pi(a_t|s_t)}{\pi_{old}(a_t|s_t)}\right)\left(\sum_{t=0}^{\infty} a_{\pi_{old}}(s_t, a_t)\right)\right]$

$\approx E_{\pi_{old}}\left[\sum_{t=0}^{\infty} \frac{\pi(a_t|s_t)}{\pi_{old}(a_t|s_t)} A_t\right]$

Policy gradient involves two estimations:

1. The surrogate objective estimates this objective delta
2. Gradient ascent maximizes the surrogate objective in a local, first-order sense

Apply sufficiently small update in policy space. article amsmath amssymb amsfonts mathtools

Gradient ascent applies a small update in parameter space:

$$\theta \leftarrow \theta_{old} + \alpha \hat{\nabla}_\theta J(\pi_\theta)$$

However,

$$small \|\theta - \theta_{old}\| \;\not\Rightarrow\; small \|\pi_\theta - \pi_{old}\|.$$

20.1 How to measure $\|\pi_\theta - \pi_{old}\|$?

20.2 How to bound the update properly?

1. Trust Region Policy Optimization (TRPO)
2. Proximal Policy Optimization (PPO)

21 Trust Region Policy Optimization

Theorem:

$$J(\pi_\theta) - J(\pi_{old}) \;\geq\; L_\theta \;-\; C\, D_{\mathrm{KL}}^{\max}(\theta_{old}, \theta)$$

where L_θ is the surrogate objective, C is some constant, and

$$D_{\mathrm{KL}}^{\max}(\theta_{old}, \theta) := \max_{s \in \mathcal{S}} D_{\mathrm{KL}}(\pi_{old}(\cdot \mid s) \,\|\, \pi_\theta(\cdot \mid s))$$

is a policy-space distance metric based on KL divergence.

21.1 Practice

A complex implementation that solves

$$\max_\theta \; L_\theta \quad s.t. \quad D_{\mathrm{KL}}^{\pi_{old}}(\theta_{old}, \theta) \leq \delta.$$

22 Proximal Policy Optimization

PPO maximizes a clipped surrogate objective:

$$L_\theta := E_{\pi_{old}} \left[\sum_{t=0}^{\infty} \min(r_t(\theta) A_t,\; clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\, A_t) \right].$$

The gradient is:

$$\nabla_\theta L_\theta = E_{\pi_{old}}\left[\sum_{t=0}^{\infty} \nabla_\theta r_t(\theta)\, A_t \cdot I_t(\theta)\right],$$

where

$$I_t(\theta) = \{\, I\left(r_t(\theta) < 1+\epsilon\right), A_t > 0, I(r_t(\theta) > 1-\epsilon), A_t < 0.$$

Thus, $\nabla_\theta L_\theta$ at step t is nonzero only when it lies in a "trust region," where the importance-sampling (IS) ratio stays similar on the observed action.

23 Actor–Critic Style PPO

Initialize policy π_θ and value function V_ϕ, then loop:

23.1 1. Collect data

1. Execute π_θ to collect a dataset: $\{\tau_i\}_i \sim \pi_\theta$.
2. Critic inference: $V_{i,t} \leftarrow V_\phi(s_{i,t})$.
3. Compute λ-return:
$$G^\lambda_{i,t} \leftarrow GAE(\tau_i, V_i, t).$$
4. Actor inference:
$$\pi_{old}(a_{i,t} \mid s_{i,t}) \leftarrow \pi_\theta(a_{i,t} \mid s_{i,t}).$$

23.2 2. Update critic

Gradient descent on MSE loss:

$$L_\phi = \frac{1}{n}\sum_{i=1}^{n}\sum_{t=0}^{\infty}\left(G^\lambda_{i,t} - V_\phi(s_{i,t})\right)^2.$$

23.3 3. Update actor

Define:

$$r_{i,t}(\theta) = \frac{\pi_\theta(a_{i,t} \mid s_{i,t})}{\pi_{old}(a_{i,t} \mid s_{i,t})}, \qquad A_{i,t} = G^\lambda_{i,t} - V_{i,t}.$$

Surrogate objective:

$$L_\theta = \frac{1}{n}\sum_{i=1}^{n}\sum_{t=0}^{\infty} \min(r_{i,t}(\theta)A_{i,t},\ clip(r_{i,t}(\theta), 1-\epsilon,\, 1+\epsilon)\, A_{i,t}).$$

24 Summary

So far, we covered:

1. Tackling unknown MDPs with learning
2. Value-based methods
3. Policy-based methods
   (a) Policy gradient theorem
   (b) Improving policy gradient
   (c) Proximal Policy Optimization

25 Resources

- **Textbook:** *Reinforcement Learning: An Introduction* by Richard Sutton and Andrew Barto, MIT Press.
- **Online course:** CS 285 by Sergey Levine, UC Berkeley.
- **Online material:** OpenAI Spinning Up.