Policy-Gradient Methods

Ayça Özçelikkale

Dept. of Electrical Engineering, Uppsala University

Course Map

Overview of the Course

- Planning by Dynamic Programing
- Model-free prediction and control with tabular methods
- Function Approximation: We can use function approximation
  - to approximate value functions (next lecture in Actor-Critic)
  - to parametrize policies (this lecture and the next lecture)
  - to model the environment†
- Model-based RL
  - Tabular Model-based RL Methods: Dyna-Q and Dyna-Q+ (last lecture)
  - General Model-based RL Methods:†
- Policy-Gradient Methods (this lecture and the next lecture)
  - REINFORCE
  - REINFORCE with baseline
  - One-step Actor Critic

† 5hp course: You're not responsible to know the details of algorithms/methods but you should be aware of the general ideas and advantages/disadvantages

MAIN IDEA: The policy determines which actions to take. Let's learn a policy directly instead of relying on value functions!

What are Policy-Gradient Methods?

Short Review of Previous Lectures: Up to now, we have used the value or the action-value function to find a good policy. Then, a policy is generated directly from the value functions.

- **Example:** Q-learning
  - The learned action-value function Q approximates the optimal action-value function $q_*(s, a)$
  - The policy is implicit, for instance we use ϵ-greedy.

Now: We will parametrize the policy with $\boldsymbol{\theta} \in \mathbb{R}^{d'}$:

$$\pi_{\boldsymbol{\theta}}(a|s) = P[A_t = a|S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}]$$

$\pi_{\boldsymbol{\theta}}(a|s)$: Probability action a is taken at time t given that the environment is in state s and the policy is parametrized by the parameter $\boldsymbol{\theta}$.

We will now directly search for a good policy π by searching over the parameters $\boldsymbol{\theta}$.

Example - Policy Parametrization

Let the action space be discrete. We can use soft-max for policy parametrization:

- $h_{\boldsymbol{\theta}}(s,a)$ denotes preferences: $h_{\boldsymbol{\theta}}(s,a)$ is high when action a is preferred over other actions)
- Action probabilities are proportional to exponentiated preferences:

$$\pi_{\boldsymbol{\theta}}(a|s) \propto e^{h_{\boldsymbol{\theta}}(s,a)}$$

  In particular, set

$$\pi_{\boldsymbol{\theta}}(a|s) \triangleq \frac{e^{h_{\boldsymbol{\theta}}(s,a)}}{\sum_b e^{h_{\boldsymbol{\theta}}(s,b)}}$$

The above scheme is called soft-max in action preferences.

How to parametrize $h_{\boldsymbol{\theta}}(s,a)$?

- Example: Linear in the features $\boldsymbol{x}(s,a)$

$$h_{\boldsymbol{\theta}}(s,a) = \boldsymbol{\theta}^T \boldsymbol{x}(s,a)$$

- Example: $\text{ANN}_\theta(s,a)$

Why are these methods called Policy Gradient Methods?

Let $J(\boldsymbol{\theta})$ be our scalar performance measure. Finding the best policy is now equivalent to maximizing $J(\boldsymbol{\theta})$ over $\boldsymbol{\theta}$:

$$\max_{\boldsymbol{\theta} \in \mathbb{R}^{d'}} J(\boldsymbol{\theta})$$

Policy gradient methods seek for $\boldsymbol{\theta}$ using (approximate) gradient ascent. Hence, the updates for policy gradient methods approximate the **gradient ascent** updates

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha^{\boldsymbol{\theta}} \nabla J(\boldsymbol{\theta}_t)$$

where $\alpha^{\boldsymbol{\theta}} > 0$ is the step size parameter.

Classification of RL Approaches

Value-Based

- Learn Value Function
- Policy is Implicit

Actor-Critic

- Learn Value Function (Critic)
- Learn Policy (Actor)

Policy-Based

- No Value Function
- Learn Policy

Motivation

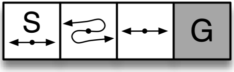
Policy-gradient methods are promising from the following perspectives:

- Deal with high-dimensional or continuous action spaces directly
- Directly learn stochastic policies
  - In some environments, we absolutely need stochastic policies
    - Example: Poker
- Better convergence properties
  - Action probabilities change smoothly as a function of the policy parameters $\rightarrow$ It is possible to directly approximate gradient ascent on the performance
- Incorporate a prior information about the problem

Note that policy-gradient methods also have some disadvantages:

- Typically converge to a local rather than global optimum
- Evaluating a policy is typically inefficient and high variance

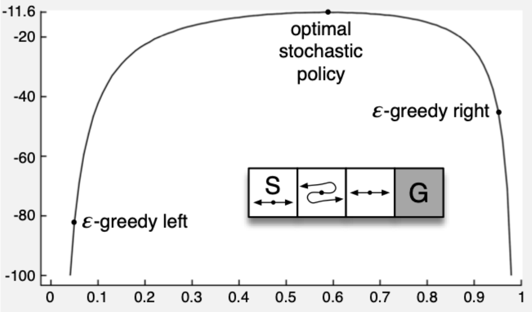
Example: Short corridor with switched actions



This is an example where the optimal policy is stochastic.

- A corridor of three rooms and one terminating state at the right.
  - We start at the left-most room S.
- Reward: -1 per step
- Actions: Left, Right
  - Middle room reverses the actions!
- Features: $x(s, right) = [1, 0]^T$ and $x(s, left) = [0, 1]^T$, $\forall s$
  - All states appear identical under function approximation!
- An action-value method with ϵ-greedy can only have two forms:
  - Right with probability $1 - \epsilon/2, \ \forall s$
  - Left with probability $1 - \epsilon/2, \ \forall s$

Optimal policy is stochastic



Performance $J(\boldsymbol{\theta}) = v_{\pi_{\boldsymbol{\theta}}}(S)$ versus the probability of choosing action "right"

For ϵ-greedy left/right, $\epsilon = 0.1$

Exercise: Action Values vs Action Preferences

Consider the policy parametrization with soft-max in action preferences:

$$\pi_{\boldsymbol{\theta}}(a|s) \triangleq \frac{e^{h_{\boldsymbol{\theta}}(s,a)}}{\sum_b e^{h_{\boldsymbol{\theta}}(s,b)}}$$

An idea is to use to first estimate the action values $q(s,a)$ with a value-based method; and then use $\hat{q}(s,a)$ instead of $h_{\boldsymbol{\theta}}(s,a)$ in the above equation.

Do you think that this idea will work?

Reminder: Gradient Ascent -A general idea for minimizing functions

Notation- Gradient: Let $J(\boldsymbol{\theta}) : \mathbb{R}^{d'} \rightarrow \mathbb{R}$ be a scalar valued function with a vector input. Gradient of $J(\boldsymbol{\theta})$ is given by

$$\nabla J \triangleq \frac{\partial J}{\partial \boldsymbol{\theta}} \triangleq \begin{bmatrix} \frac{\partial J}{\partial \boldsymbol{\theta}_1} \\ \frac{\partial J}{\partial \boldsymbol{\theta}_2} \\ \vdots \\ \frac{\partial J}{\partial \boldsymbol{\theta}_{d'}} \end{bmatrix} \in R^{d' \times 1},$$

The gradient evaluated at a particular value $\bar{\boldsymbol{\theta}}$ is denoted by the following:

$$\nabla J(\bar{\boldsymbol{\theta}}) = \left. \frac{\partial J}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}}$$

.

Gradient Ascent Idea: If you want to minimize a function $J(\boldsymbol{\theta})$ iteratively, start from an initial point $\boldsymbol{\theta}_i$ (here i is the iteration index!), and move in the direction of maximum "ascent":

$$\begin{aligned} \boldsymbol{\theta}_{i+1} &= \boldsymbol{\theta}_i + \Delta \boldsymbol{\theta}_i \\ &= \boldsymbol{\theta}_i + \alpha^{\boldsymbol{\theta}} \nabla J(\boldsymbol{\theta}_i) \end{aligned}$$

where $\alpha^{\boldsymbol{\theta}} > 0$ is the step size.

Objective Function $J(\boldsymbol{\theta})$

- We focus on the episodic case
- We assume that every episode starts at some particular non-random state s_0
- The performance measure is defined as the value function evaluated at the starting state s_0 if we follow the policy π_θ

$$J(\boldsymbol{\theta}) \triangleq v_{\pi_{\boldsymbol{\theta}}}(s_0)$$

- For the sake of clarity, analytical development assumes no discounting, i.e. $\gamma = 1$

How to find the policy gradient?

Challenge: Policy parameter $\boldsymbol{\theta}$ affects both of the following

- Action selections
  - Given a state, the effect of $\boldsymbol{\theta}$ on the actions and the reward can be calculated easily
- State distribution
  - The effect of $\boldsymbol{\theta}$ on the state distribution is a function of the unknown environment

Policy Gradient Theorem gives a neat answer to this challenge:

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_a q_{\pi_{\boldsymbol{\theta}}}(S_t, a)\nabla\pi_{\boldsymbol{\theta}}(a|S_t)\right]$$

- The expectation is over the state distribution under $\pi_{\boldsymbol{\theta}}$ (We know how to approximate this type of expressions!)
- RHS does not involve the derivative of the state distribution!

We look at $\nabla J(\boldsymbol{\theta})$ more carefully:

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_a q_{\pi_{\boldsymbol{\theta}}}(S_t, a)\nabla\pi_{\boldsymbol{\theta}}(a|S_t)\right] \\
&= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_a \pi_{\boldsymbol{\theta}}(a|S_t) q_{\pi_{\boldsymbol{\theta}}}(S_t, a)\frac{\nabla\pi_{\boldsymbol{\theta}}(a|S_t)}{\pi_{\boldsymbol{\theta}}(a|S_t)}\right] \\
&= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[q_{\pi_{\boldsymbol{\theta}}}(S_t, A_t)\frac{\nabla\pi_{\boldsymbol{\theta}}(A_t|S_t)}{\pi_{\boldsymbol{\theta}}(A_t|S_t)}\right] \\
&= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[G_t\frac{\nabla\pi_{\boldsymbol{\theta}}(A_t|S_t)}{\pi_{\boldsymbol{\theta}}(A_t|S_t)}\right] \\
&= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[G_t \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)\right]
\end{aligned}
$$

where in the last two lines we have used $\mathbb{E}_{\pi}[G_t|S_t, A_t] = q_{\pi}(S_t, A_t)$ and $\nabla \ln f(\boldsymbol{\theta}) = \frac{\nabla f(\boldsymbol{\theta})}{f(\boldsymbol{\theta})}$

We use $\nabla J(\boldsymbol{\theta})$ for stochastic gradient ascent updates

Gradient Ascent: To maximize $J(\boldsymbol{\theta})$, start from an estimate $\boldsymbol{\theta}_t$ and update using

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha^{\boldsymbol{\theta}} \nabla J(\boldsymbol{\theta}_t)$$

From the previous slide: $\nabla J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[G_t \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t) \right]$

Hence, we have

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha^{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[G_t \nabla \ln \pi_{\boldsymbol{\theta}_t}(A_t|S_t) \right]$$

where $\nabla \ln \pi_{\boldsymbol{\theta}_t}(\cdot)$ is the gradient of $\ln \pi_{\boldsymbol{\theta}}(\cdot)$ with respect to $\boldsymbol{\theta}$ evaluated at the particular value $\boldsymbol{\theta}_t$.

Stochastic Gradient Ascent: Use a sample instead of an expectation

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha^{\boldsymbol{\theta}} G_t \nabla \ln \pi_{\boldsymbol{\theta}_t}(A_t|S_t)$$

This is the REINFORCE update.

We look at REINFORCE updates more closely

REINFORCE update from the previous slide:

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \Delta\boldsymbol{\theta}_t \\ &= \boldsymbol{\theta}_t + \alpha^{\boldsymbol{\theta}} G_t \nabla \ln \pi_{\boldsymbol{\theta}_t}(A_t|S_t) \\ &= \boldsymbol{\theta}_t + \alpha^{\boldsymbol{\theta}} G_t \frac{\nabla \pi_{\boldsymbol{\theta}_t}(A_t|S_t)}{\pi_{\boldsymbol{\theta}_t}(A_t|S_t)}\end{aligned}$$

Note that $\nabla \pi_{\boldsymbol{\theta}_t}(A_t|S_t)$ is the direction in $\boldsymbol{\theta}$ space that most increases the probability of taking action A_t when at state S_t

Each update $\Delta\boldsymbol{\theta}_t$ is in the direction of $\nabla \pi_{\boldsymbol{\theta}_t}(A_t|S_t)$ with a scaling so that

- Proportional to the return G_t
- Inversely proportional to the action probability

Reinforce: Monte-Carlo Policy Gradient Control for finding π_*

Input:Differentiable policy parametrization $\pi_{\boldsymbol{\theta}}(a|s)$.

Initialization: Set step size $\alpha^{\boldsymbol{\theta}} > 0$. Initialize the policy parameters $\boldsymbol{\theta} \in \mathbb{R}^{d'}$.

repeat (for each episode)

Generate an episode using $\pi_{\boldsymbol{\theta}}$: $S_0, A_0, R_1, S_1, A_1, \ldots, R_T$.

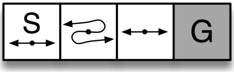
repeat : (for each step of the episode $t = 0, 1, \ldots T-1$)

$$G \leftarrow \sum_{k=t+1^T} \gamma^{k-t-1} R_k$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t G_t \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)$$

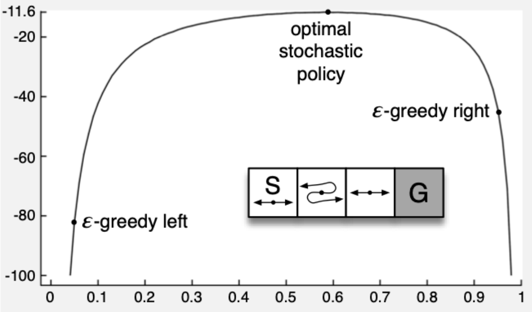
Desired Output $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Recall the environment “Short corridor with switched actions”



- A corridor of three rooms and one terminating state at the right.
  - We start at the left-most room S.
- Reward: -1 per step
- Actions: Left, Right
  - Middle room reverses the actions!
- Features: $x(s, right) = [1, 0]^T$ and $x(s, left) = [0, 1]^T$, $\forall s$
  - All states appear identical under function approximation!
- An action-value method with ϵ-greedy can only have two forms:
  - Right with probability $1 - \epsilon/2$, $\forall s$
  - Left with probability $1 - \epsilon/2$, $\forall s$

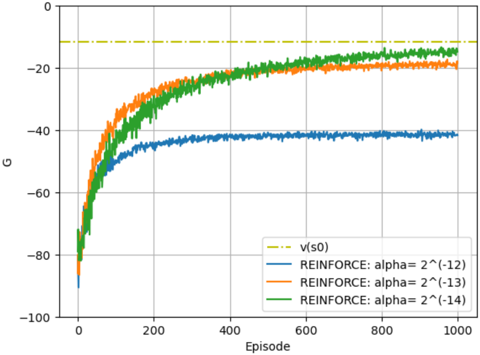
Reminder: Optimal policy is stochastic



Performance $J(\boldsymbol{\theta}) = v_{\pi_{\boldsymbol{\theta}}}(S)$ versus the probability of choosing action "right"

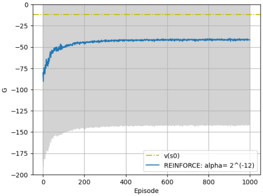
For ϵ-greedy left/right, $\epsilon = 0.1$

Convergence of REINFORCE

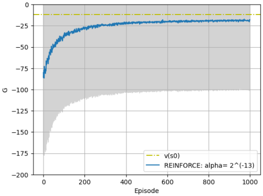


Total reward on episode (Average over 300 runs)

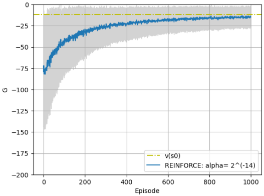
REINFORCE is of high variance



$\alpha^{\boldsymbol{\theta}} = 2^{-12}$



$\alpha^{\boldsymbol{\theta}} = 2^{-13}$

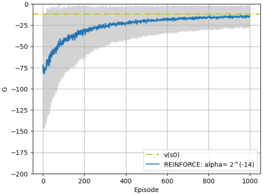


$\alpha^{\boldsymbol{\theta}} = 2^{-14}$

- Blue curves: mean of the total reward over runs ,i.e. same with the corresponding curve in the former figure
- We color the area between “mean-standard deviation” and “mean+standard deviation” with light gray
- To make the plots easier to read, we have done the following: standard deviation is smoothed along episode axis and capped with 100

REINFORCE with baseline has lower variance

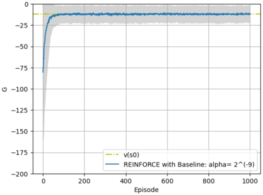
REINFORCE



$\alpha^{\boldsymbol{\theta}} = 2^{-14}$

V.S.

REINFORCE with baseline



$\alpha^{\boldsymbol{\theta}} = 2^{-9}$

Reminders: Classification of RL Approaches

Value-Based

- Learn Value Function
- Policy is Implicit

Actor-Critic

- Learn Value Function (Critic)
- Learn Policy (Actor)

Policy-Based

- No Value Function
- Learn Policy

Reminders:

Policy $\pi_{\boldsymbol{\theta}}(a|s)$ is parametrized by the policy parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{d'}$

Performance measure is value function evaluated at the starting state s_0 if we follow the policy π_θ:

$$J(\boldsymbol{\theta}) \triangleq v_{\pi_{\boldsymbol{\theta}}}(s_0)$$

Aim: Find the best policy by maximizing $J(\boldsymbol{\theta})$ over $\boldsymbol{\theta}$: $\max_{\boldsymbol{\theta} \in \mathbb{R}^{d'}} J(\boldsymbol{\theta})$

Main Idea for Policy Gradient Methods:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha^{\boldsymbol{\theta}} \nabla J(\boldsymbol{\theta}_t)$$

Policy Gradient Theorem:

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\sum_a q_{\pi_{\boldsymbol{\theta}}}(S_t, a) \nabla \pi_{\boldsymbol{\theta}}(a|S_t) \right] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[G_t \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t) \right]$$

REINFORCE Update:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} G_t \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)$$

Preliminaries: Baseline[†]

We can add a baseline $b(s)$ to obtain a generalization of the policy-gradient theorem

$$\begin{aligned}\nabla J(\boldsymbol{\theta}) &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_a q_{\pi_{\boldsymbol{\theta}}}(S_t, a)\nabla\pi_{\boldsymbol{\theta}}(a|S_t)\right] \\ &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_a (q_{\pi_{\boldsymbol{\theta}}}(S_t, a) - b(S_t))\nabla\pi_{\boldsymbol{\theta}}(a|S_t)\right]\end{aligned}$$

since we have

$$\begin{aligned}\sum_a b(s)\nabla\pi_{\boldsymbol{\theta}}(a|s) = b(s)\sum_a \nabla\pi_{\boldsymbol{\theta}}(a|s) &= b(s)\nabla\sum_a \pi_{\boldsymbol{\theta}}(a|s) \\ &= b(s)\nabla 1 \\ &= 0\end{aligned}$$

- This equation holds as long as the baseline does not depend on the actions.
- Note that the baseline can depend on the state s

† : Definition of baseline from Merriam-Webster dictionary: a line serving as a basis, especially one of known measure or position used (as in surveying or navigation) to calculate or locate something

REINFORCE with baseline

Update for REINFORCE with baseline

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} (G_t - b(S_t)) \nabla \ln \pi_{\boldsymbol{\theta}}(A_t | S_t)$$

- Baseline does not change the expected value of the update
- For $b(s) = 0$, we obtain the plain REINFORCE algorithm.
- The baseline $b(s)$ should vary with the state.
  - Example: If all actions from a given state has high values, it is better to have a high baseline so that we can differentiate between the actions.
- Idea: Use $b(s) = \hat{v}(S_t, \boldsymbol{w})$ where $\hat{v}(\cdot)$ is an estimate parametrized by $\boldsymbol{w} \in \mathbb{R}^d$.

Reinforce with Baseline for finding π_*

Input: Differentiable policy parametrization $\pi_{\boldsymbol{\theta}}(a|s)$; differentiable value function parametrization $\hat{v}(s, \boldsymbol{w})$

Initialization: Set step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\boldsymbol{w}} > 0$. Initialize the policy parameters $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and value function approximation parameters $\boldsymbol{w} \in \mathbb{R}^{d}$.

repeat (for each episode)

Generate an episode using $\pi_{\boldsymbol{\theta}}$: $S_0, A_0, R_1, S_1, A_1, \ldots, R_T$.

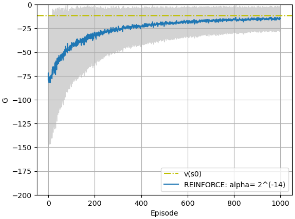
repeat : (for each step of the episode $t = 0, 1, \ldots T-1$)

$$G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$
$$\delta = G - \hat{v}(S_t, \boldsymbol{w})$$
$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha^{\boldsymbol{w}} \delta \nabla \hat{v}(S_t, \boldsymbol{w})$$
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)$$

Desired Output $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Example Revisited: Short Corridor with switched actions

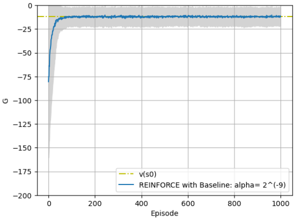
REINFORCE



$\alpha^{\boldsymbol{\theta}} = 2^{-14}$

V.S.

REINFORCE with baseline



$\alpha^{\boldsymbol{\theta}} = 2^{-9}$, $\alpha^{\boldsymbol{w}} = 2^{-6}$

- REINFORCE with baseline has lower variance.
- REINFORCE with baseline learns faster.
- How to set step sizes $\alpha^{\boldsymbol{w}}$ and $\alpha^{\boldsymbol{\theta}}$?
  - $\alpha^{\boldsymbol{w}}$: We have a rule of thumb

$$\alpha^{\boldsymbol{w}} = \frac{0.1}{\mathbb{E}[\|\nabla \hat{v}(S_t, \boldsymbol{w})\|^2]} = \frac{0.1}{\mathbb{E}[\|\boldsymbol{x}(S_t, \boldsymbol{w})\|^2]}$$

  - $\alpha^{\boldsymbol{\theta}}$: More difficult to choose.

Actor-Critic Methods

Actor-critic methods is a large family of methods which uses the following **main idea:**
Use the value function not only as a baseline but also as a "critic"

- critic uses the value function for bootstrapping, updating the value estimate for a state from the estimated values of subsequent states

Note: Although the conceptual distinction is important, algorithms with a state-dependent baseline is sometimes classified also as actor-critic algorithms.

Example: One-step Actor-Critic

REINFORCE with baseline

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} (G_t - \hat{v}(S_t, \mathbf{w})) \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)$$

One-step Actor-Critic

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} (R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})) \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)$$

One-step Actor-Critic for finding π_*

Input: Differentiable policy parametrization $\pi_{\boldsymbol{\theta}}(a|s)$; differentiable value function parametrization $\hat{v}(s, \boldsymbol{w})$

Initialization: Set step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\boldsymbol{w}} > 0$. Initialize the policy parameters $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and value function approximation parameters $\boldsymbol{w} \in \mathbb{R}^{d}$.

repeat (for each episode)

Initialize first state S of the episode.

$I \leftarrow 1$

repeat : (for each time step, while S is not terminal)

Choose action A using $\pi_{\boldsymbol{\theta}}(\cdot|S)$

Take action A, observe S', R

$\delta = R + \gamma \hat{v}(S', \boldsymbol{w}) - \hat{v}(S, \boldsymbol{w})$ (If S' terminal, set $\hat{v}(S', \boldsymbol{w})$)

$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha^{\boldsymbol{w}} \delta \nabla \hat{v}(S, \boldsymbol{w})$

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi_{\boldsymbol{\theta}}(A|S)$

$I \leftarrow \gamma I$

$S \leftarrow S'$

Desired Output $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Comparison

REINFORCE

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} G_t \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)$$

- high variance, updates unbiased, inconvenient for online/continuing problems

REINFORCE with a state-dependent baseline $\hat{v}(S_t, \mathbf{w})$ found using G_t

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} (G_t - \hat{v}(S_t, \mathbf{w})) \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)$$

- lower variance, updates unbiased, inconvenient for online/continuing problems

One-step Actor-Critic

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} (R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})) \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)$$

- lower variance, updates biased, convenient for online/continuing problems

Interpreting One-step Actor-Critic in terms of Advantage Function

Advantage Function:

$$A_{\pi_{\boldsymbol{\theta}}}(s,a) = q_{\pi_{\boldsymbol{\theta}}}(s,a) - v_{\pi_{\boldsymbol{\theta}}}(s)$$

We can write policy gradient in terms of the advantage function

$$\begin{aligned}\nabla J(\boldsymbol{\theta}) &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[q_{\pi_{\boldsymbol{\theta}}}(S_t, A_t)\nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)\right] \\ &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[A_{\pi_{\boldsymbol{\theta}}}(S_t, A_t)\nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)\right]\end{aligned}$$

REINFORCE: Approximate policy gradients by estimating $q_{\pi_{\boldsymbol{\theta}}}(S_t, A_t)$

Actor-Critic: Approximate policy gradients by estimating $A_{\pi_{\boldsymbol{\theta}}}(S_t, A_t)$

- Example: Use function approximation for both $q_{\pi_{\boldsymbol{\theta}}}(s,a)$ and $v_{\pi_{\boldsymbol{\theta}}}(s)$
- Example (One-Step Actor-Critic): Use function approximation for $v_{\pi_{\boldsymbol{\theta}}}(s)$ and use TD idea to find estimates for $q_{\pi_{\boldsymbol{\theta}}}(s,a)$

Preliminaries: Normal Distribution

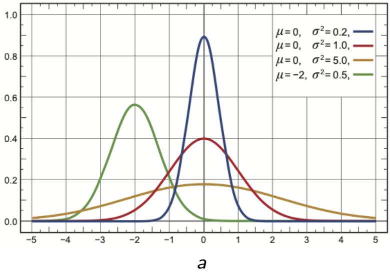
Probability density function (pdf) of a scalar random variable x is defined by the following relationship:

$$\text{Prob}[z_L \leq a \leq z_H] = \int_{z_L}^{z_H} p_a(\bar{a})d\bar{a}$$

The pdf for normal (i.e. Gaussian) distribution is given by[1]

$$p_a(a) = \mathcal{N}(a;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(\frac{(a-\mu)^2}{2\sigma^2})$$

where μ is the mean and σ is the standard deviation.



[1] π does not denote the policy in this formula; it is just the number $\pi \approx 3.1415$

Policy Parametrization for Continuous Actions

Popular Idea: Define the policy π as the normal probability density over a real-valued scalar action

$$\pi(a|s,\boldsymbol{\theta}) = \mathcal{N}(a; \mu(s,\boldsymbol{\theta}), \sigma(s,\boldsymbol{\theta}))$$

where $\mu(s,\boldsymbol{\theta})$ and $\sigma(s,\boldsymbol{\theta})$ are function approximators parametrized by $\boldsymbol{\theta}$.

Example:

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_\mu \\ \boldsymbol{\theta}_\sigma \end{bmatrix}$$

$$\mu(s,\boldsymbol{\theta}) = \boldsymbol{\theta}_\mu^T \boldsymbol{x}_\mu(s)$$

$$\sigma(s,\boldsymbol{\theta}) = \exp(\boldsymbol{\theta}_\sigma^T \boldsymbol{x}_\sigma(s))$$

where $\boldsymbol{x}_\mu(s)$ and $\boldsymbol{x}_\sigma(s)$ are feature vectors.

Example: Use an ANN whose output denotes μ and σ

A Revisit to Policy Gradient Theorem

Discrete Action Space

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_a q_{\pi_{\boldsymbol{\theta}}}(S_t, a)\nabla \pi_{\boldsymbol{\theta}}(a|S_t)\right]$$
$$= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[G_t \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)\right]$$

Continuous Action Space

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\int_a q_{\pi_{\boldsymbol{\theta}}}(S_t, a)\nabla \pi_{\boldsymbol{\theta}}(a|S_t)da\right]$$
$$= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[G_t \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)\right]$$

We need $\nabla \pi_{\boldsymbol{\theta}}(a|S_t)$!

With the above parametrization with normal pdf, we can easily calculate $\nabla \ln \pi_{\boldsymbol{\theta}}(a|s)$, i.e. $\nabla_{\boldsymbol{\theta}_u} \ln \pi_{\boldsymbol{\theta}}(a|s)$ and $\nabla_{\boldsymbol{\theta}_\sigma} \ln \pi_{\boldsymbol{\theta}}(a|s)$.

For instance,

$$\nabla_{\boldsymbol{\theta}_u} \ln \pi_{\boldsymbol{\theta}}(a|s) = \nabla_{\boldsymbol{\theta}_u}\left(-\frac{(a-\mu(s,\boldsymbol{\theta}))^2}{2\sigma(s,\boldsymbol{\theta})^2}\right)$$
$$= \frac{1}{\sigma(s,\boldsymbol{\theta})^2}(a-\mu(s,\boldsymbol{\theta}))\boldsymbol{x}_\mu(s)$$

Self-study Exercise: Find $\nabla_{\boldsymbol{\theta}_\sigma} \ln \pi_{\boldsymbol{\theta}}(a|s)$.

Recap: Pros and Cons of Policy Gradient Methods

Pros:

- Deal with high-dimensional or continuous action spaces directly
- Directly learn stochastic policies
- Better convergence properties
  - directly approximate gradient ascent on the performance
- Incorporate a prior information about the problem

Cons:

- Typically converge to a local rather than global optimum
- Typically sample inefficient and high variance

Discussions: Different Policy Gradient Ideas from the Literature -A2C

Advantage Actor Critic (A2C):

Similar to the before:

- estimate advantage function (but with n-step returns)
- a shared ANN to parametrize $\pi_{\boldsymbol{\theta}}$ and $\hat{v}_{\boldsymbol{\theta}}$
- multiple-thread processing
- add an exploration term to the objective to prevent early convergence to deterministic policies

Discussions: Different Policy Gradient Ideas from the Literature - TRPO

Notation:
$r_t(\boldsymbol{\theta}) = \frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_{old}}(A_t|S_t)}$
Advantage function: $A_{\theta_{old}} = A_{\theta_{old}}(S_t, A_t)$

Trust Region Policy Optimization (TRPO):

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\theta_{old}}}[r_t(\boldsymbol{\theta}) A_{\theta_{old}}]$$

st.

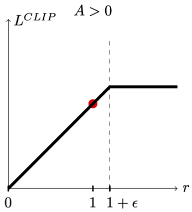
$$\mathbb{E}_{\pi_{\theta_{old}}}[D_{KL}\left(\pi_{\theta_{old}}(.|S_t)|\pi_\theta(.|S_t)\right)] \leq \epsilon_{TRPO}$$

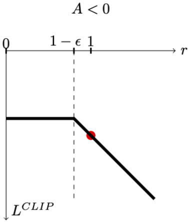
Discussions: Different Policy Gradient Ideas from the Literature -PPO

Notation:
$r_t(\boldsymbol{\theta}) = \frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_{old}}(A_t|S_t)}$
Advantage function: $A_{\theta_{old}} = A_{\theta_{old}}(S_t, A_t)$

Proximal Policy Optimization (PPO):

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\theta_{old}}}[\min\left(r_t(\boldsymbol{\theta})A_{\theta_{old}}, \text{clip}(r_t(\boldsymbol{\theta}), 1+\epsilon, 1-\epsilon)A_{\theta_{old}}\right)]$$





IDEA: obtain a lower bound on the objective by ignoring the change in r_t when it would make the objective improve too much, and include it when it makes the objective worse

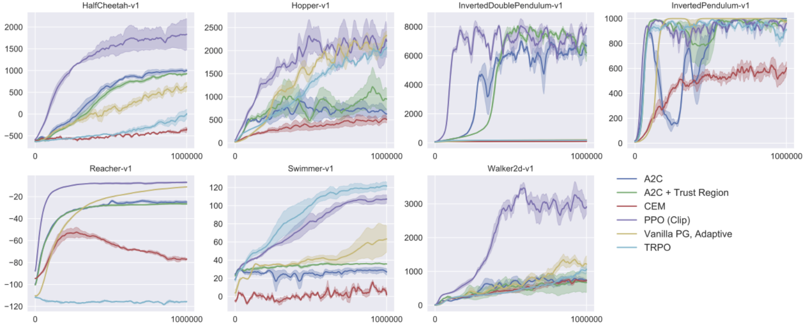
Mujoco: Continuous control tasks running in a physics simulator

Inverted Pendulum

Reacher

Swimmer

Comparison of Different Policy Gradient Algorithms



Summary - Key Concepts

Parametrization of the policy: soft-max policies for discrete action spaces, Gaussian parametrization for continuous actions

Aim: Find the best policy by maximizing $J(\boldsymbol{\theta}) = v_{\pi_{\boldsymbol{\theta}}}(s_0)$ over $\boldsymbol{\theta}$:

$$\max_{\boldsymbol{\theta} \in \mathbb{R}^{d'}} J(\boldsymbol{\theta})$$

Main Idea for Policy Gradient Methods:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha^{\boldsymbol{\theta}} \nabla J(\boldsymbol{\theta}_t)$$

Advantage Function:

$$A_{\pi_{\boldsymbol{\theta}}}(s, a) = q_{\pi_{\boldsymbol{\theta}}}(s, a) - v_{\pi_{\boldsymbol{\theta}}}(s)$$

Policy Gradient:

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[q_{\pi_{\boldsymbol{\theta}}}(S_t, A_t) \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)\right] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[A_{\pi_{\boldsymbol{\theta}}}(S_t, A_t) \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)\right]$$

Concluding Remarks -Comparison

REINFORCE

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} G_t \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)$$

- high variance, updates unbiased, inconvenient for online/continuing problems

REINFORCE with a state-dependent baseline $\hat{v}(S_t, \mathbf{w})$ found using G_t

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} (G_t - \hat{v}(S_t, \mathbf{w})) \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)$$

- lower variance, updates unbiased, inconvenient for online/continuing problems

One-step Actor-Critic

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} (R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})) \nabla \ln \pi_{\boldsymbol{\theta}}(A_t|S_t)$$

- lower variance, updates biased, convenient for online/continuing problems

References

- Sutton, Barto, Reinforcement Learning: An Introduction
- V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, andK. Kavukcuoglu. "Asynchronous methods for deep reinforcement learning". PMLR, 2016
- J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. "Trust region policy optimization", PMLR, 2015
- J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms", 2017