Reinforcement Learning

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 4: Policy Gradient I

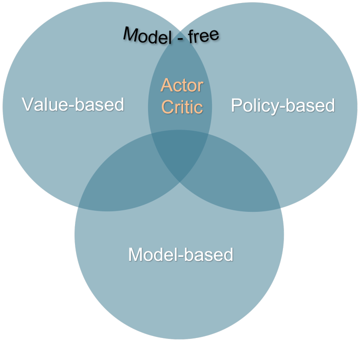
Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)

EE-568 (Spring 2025)

License Information for Reinforcement Learning (EE-568)

- This work is released under a Creative Commons License with the following terms:
- **Attribution**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees must give the original authors credit.
- **Non-Commercial**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees may not use the work for commercial purposes – unless they get the licensor's permission.
- **Share Alike**
  - The licensor permits others to distribute derivative works only under a license identical to the one that governs the licensor's work.
- Full Text of the License

Overview of reinforcement learning approaches



- ○ Value-based RL
  - Learn the optimal value functions $V^\star, Q^\star$ (or the best approximation $V_{\theta^\star}, Q_{\theta^\star}$)
  - Generate the optimal policy

$$\pi^\star(a|s) = \arg\max_{a \in \mathcal{A}} Q^\star(s,a)$$

  - Algorithms: Monte Carlo, SARSA, Q-learning, etc.

- ○ Policy-based RL
  - Learn the optimal policy $\pi^\star$

- ○ Model-based RL
  - Learn the model P and r and then do planning

Value-based methods

- Advantages
  - Easy to generate policy from the learned value function [19], e.g., via greedy selections.
  - Controlling the bias-variance tradeoff (e.g., via MC or TD(n)) is well-studied [29, 28, 7].
  - We have good theory for tabular and linear function approximation settings [26, 23].

- Disadvantages:
  - Do not scale to high-dimensional or continuous action spaces [15].
  - Instability with off-policy learning under function approximation [2, 5, 4].
    - Combining TD-learning, function approximation, and off-policy learning is sometimes called *the deadly triad* [24].
  - A small value error may lead to a large policy error [19].

Optimization formulation for policy-based methods

○ Idea: Parameterize the policy as $\pi_\theta(a|s)$ and then find the best parameter θ maximizing the cumulative reward

Policy optimization

We write an optimization formulation that will be used by the policy-based methods in the sequel as follows:

$$\max_\theta J(\pi_\theta) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 \sim \mu, \pi_\theta\right] = \mathbb{E}_{s\sim\mu}[V^{\pi_\theta}(s)].$$

Observations: ○ Here μ is the initial state distribution.

○ *Alternatively, one may consider the average reward objective:

$$J_{\text{avg}}(\pi_\theta) = \liminf_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} r(s_t, a_t) | s_0 \sim \mu, \pi_\theta\right],$$

which we do not cover in this lecture.

○ Stochastic policies: $\pi_\theta(a|s) = \mathsf{P}(a|s, \theta)$ is a distribution over the action space.

How to parameterize policies for discrete actions?

- In general, we choose a parameterization that gives us an advantage
  - Direct parameterization

$$\pi_\theta(a|s) = \theta_{s,a}, \quad \text{where } \theta_{s,a} \geq 0 \text{ and } \sum_{a\in\mathcal{A}} \theta_{s,a} = 1.$$

  - Softmax parameterization

$$\pi_\theta(a|s) = \frac{\exp(\theta_{s,a})}{\sum_{a'\in\mathcal{A}} \exp(\theta_{s,a'})}, \quad \text{where } \theta \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{S}|}.$$

  - Log-linear parameterization

$$\pi_\theta(a|s) = \frac{\exp\left(\theta \cdot \phi(s,a)\right)}{\sum_{a'\in\mathcal{A}} \exp\left(\theta \cdot \phi(s,a')\right)}, \quad \text{where } \phi(s,a) \in \mathbb{R}^d \text{ and } \theta \in \mathbb{R}^d.$$

  - Neural softmax parameterization

$$\pi_\theta(a|s) = \frac{\exp\left(h_\theta(s,a)\right)}{\sum_{a'\in\mathcal{A}} \exp\left(h_\theta\left(s,a'\right)\right)}, \quad \text{where } h_\theta(s,a) \text{ represents a neural network.}$$

How to parameterize policies for continuous actions?

- Continuous probability distributions: Gaussian, Beta, Dirichlet, etc.

Example

For example, we can use the Gaussian parameterization as follows

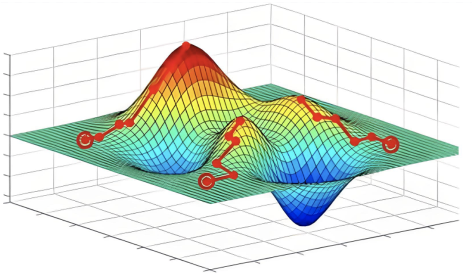
$$\pi_\theta(a|s) = \frac{1}{\sqrt{2\pi}\sigma_\theta(s)} \exp\left(-\frac{(a-\mu_\theta(s))^2}{2\sigma_\theta(s)^2}\right)$$

where $\mu_\theta(s), \sigma_\theta(s)$ are two differentiable function approximators.

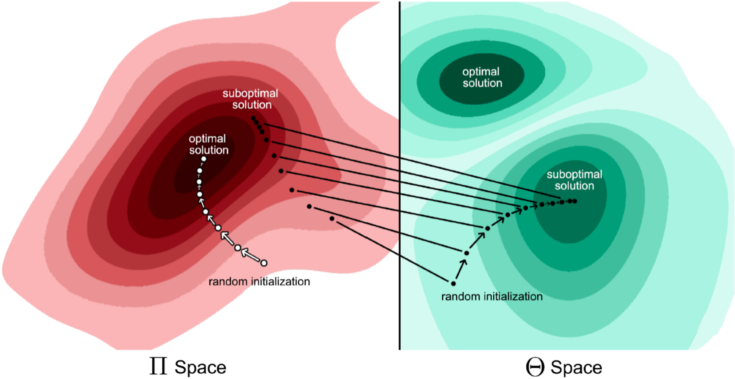
How to optimize over the given policy parameterization?

- **Gradient-free methods**
  - Hill climbing [20]
  - Simulated annealing [13]
  - Evolutionary strategies [16, 21]
  - ....

- **Gradient-based methods (our focus)**
  - Policy gradient method [25]
  - Natural policy gradient method [10]
  - ....



How to optimize over the given policy parameterization?



Policy space Π vs parameter space Θ (figure from [27])

Policy gradient method

- In general, we cannot exactly compute the gradient $\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}})$ of the objective.
- A natural idea is to consider stochastic gradients:

$$\theta_{t+1} \longleftarrow \theta_t + \alpha_t \widehat{\nabla}_\theta J(\pi_{\theta_t}),$$

where $\widehat{\nabla}_\theta J(\pi_{\theta_t})$ is a stochastic estimate of the gradient $\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}})$ at θ_t.

Q1: How do we construct a *good* estimate of $\nabla_\theta J(\pi_\theta)$?

Q2: Where does it converge to and how fast?

Monte Carlo estimation

- Consider the following objective: $F(\theta) = \mathbb{E}_{\xi \sim \mathsf{p}(\xi)}[f(\theta, \xi)]$.
- Applying the Leibniz integral rule, the gradient of the objective can be written as

$$\nabla_\theta F(\theta) = \nabla_\theta \int f(\theta, \xi)\mathsf{p}(\xi) d\xi = \int \nabla_\theta f(\theta, \xi)\mathsf{p}(\xi) d\xi = \mathbb{E}_{\xi \sim \mathsf{p}(\xi)}[\nabla_\theta f(\theta, \xi)].$$

- Here are some unbiased gradient estimators (single-sample and batch):

$$\widehat{\nabla}_\theta F(\theta) = \nabla_\theta f(\theta, \xi), \text{ where } \xi \sim \mathsf{p}(\xi).$$

$$\widehat{\nabla}_\theta F(\theta) = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta f(\theta, \xi_i), \text{ where } \xi_1, \ldots, \xi_n \sim \mathsf{p}(\xi).$$

Monte Carlo estimation with score functions

- Now, consider the following parameterization: $F(\theta) = \mathbb{E}_{\xi \sim \mathsf{p}_\theta(\xi)}[f(\xi)]$.
- The gradient of the parameterization can be written as

$$\nabla_\theta F(\theta) = \int f(\xi) \nabla_\theta \mathsf{p}_\theta(\xi) d\xi = \int \mathsf{p}_\theta(\xi) f(\xi) \nabla_\theta \log \mathsf{p}_\theta(\xi) d\xi = \mathbb{E}_{\xi \sim \mathsf{p}_\theta(\xi)}[f(\xi) \nabla_\theta \log \mathsf{p}_\theta(\xi)].$$

- Here are some unbiased gradient estimators (single-sample and batch):

$$\widehat{\nabla}_\theta F(\theta) = f(\xi) \nabla_\theta \log \mathsf{p}_\theta(\xi), \text{ where } \xi \sim \mathsf{p}_\theta(\xi).$$

$$\widehat{\nabla}_\theta F(\theta) = \frac{1}{n} \sum_{i=1}^{n} f(\xi_i) \nabla_\theta \log \mathsf{p}_\theta(\xi_i), \text{ where } \xi_1, \ldots, \xi_n \sim \mathsf{p}_\theta(\xi).$$

Remarks:

- The term $\nabla_\theta \log \mathsf{p}_\theta(\xi)$ is called the *score function*.
- For further explanation on score function, please see these EE-556 slides [6].

Parametric policy optimization

- We would like to express J as a function of trajectories drawn from the parameterized distribution.
- Recall the discounted cumulative reward objective:

$$\max_{\theta} J(\pi_\theta) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 \sim \mu, \pi_\theta\right] = \mathbb{E}_{\tau \sim \mathsf{p}_\theta}[R(\tau)],$$

where $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$ is the total reward over the random trajectory.

Observations:

- $\tau = (s_0, a_0, s_1, \dots)$ is a random trajectory with probability distribution p_θ.
- This distribution p_θ corresponds to the unique Markov process defined via

$$\begin{aligned} s_0 &\sim \mu(\cdot), & \\ a_t &\sim \pi_\theta(\cdot|s_t), & (t = 0, 1, \dots); \\ s_{t+1} &\sim \mathsf{P}(\cdot|s_t, a_t), & (t = 0, 1, \dots). \end{aligned}$$

- We have $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \mathsf{p}_\theta}[R(\tau) \cdot \nabla_\theta \log \mathsf{p}_\theta(\tau)]$.
- Note that $\nabla_\theta \log \mathsf{p}_\theta(\tau) = \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t)$.

Derivation–I

- Step 1: Define the objective to maximize expected cumulative reward:

$$J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] = \sum_\tau \mathsf{p}(\tau|\theta)R(\tau),$$

where the trajectory probability is given by:

$$\mathsf{p}_\theta(\tau) := \mathsf{p}(\tau|\theta) = \mathsf{p}(s_0)\prod_{t=0}^{\infty}\pi_\theta(a_t|s_t)\mathsf{P}(s_{t+1}|s_t,a_t).$$

- Step 2: Compute the gradient with respect to the policy parameters θ:

$$\nabla_\theta J(\pi_\theta) = \nabla_\theta\left[\sum_\tau \mathsf{p}(\tau|\theta)R(\tau)\right] = \sum_\tau R(\tau)\nabla_\theta \mathsf{p}(\tau|\theta),$$

where we apply the log-derivative trick ($\nabla_\theta f(\theta) = f(\theta)\nabla_\theta \log f(\theta)$) to obtain the following:

$$\nabla_\theta J(\pi_\theta) = \sum_\tau \mathsf{p}(\tau|\theta)R(\tau)\nabla_\theta \log \mathsf{p}(\tau|\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)\nabla_\theta \log \mathsf{p}(\tau|\theta)]$$

Derivation–II

∘ Step 3: Simplify the gradient by substituting the log of the trajectory probability:

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

and note that since $\log p(s_0)$ and $\log p(s_{t+1}|s_t, a_t)$ do not depend on θ, their gradients vanish:

$$\nabla_\theta \log \mathsf{p}(\tau|\theta) = \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t),$$

resulting in the expression we will use in the sequel:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau) \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t)\right].$$

Policy gradient theorem (version 1): REINFORCE expression

Policy gradient theorem (REINFORCE) [30]

In the paper by Williams in 1992, the acronym REINFORCE stands for "REward Increment = Nonnegative Factor times Offset Reinforcement times Characteristic Eligibility." It encapsulates the essence of the method that computes the gradient:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim \mathrm{p}_\theta}\left[R(\tau)\left(\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)\right]. \tag{1}$$

Remarks:

- ○ The term $\nabla_\theta \log \pi_\theta(a|s) = \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)}$ is called the *score function*.
- ○ For differentiable policies, the score function can often be easily computed.
- ○ For example, for log-linear parameterization $\pi_\theta(a|s) = \frac{\exp(\theta\cdot\phi(s,a))}{\sum_{a'\in\mathcal{A}}\exp(\theta\cdot\phi(s,a'))}$, we have

$$\nabla_\theta \log \pi_\theta(a|s) = \phi(s,a) - \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[\phi(s,a)].$$

- ○ Note that $\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[\nabla_\theta \log \pi_\theta(a|s)] = 0$. (Why?)

Policy gradient estimator

REINFORCE estimator

Given that we expressed J as a function of trajectories drawn from the parameterized distribution, we can use the following recipe:

- Generate an episode $\tau = (s_0, a_0, r_0, s_1, \ldots)$ from policy π_θ;
- Construct $\widehat{\nabla}_\theta J(\pi_\theta) = \left(\sum_{t=0}^{\infty} \gamma^t r_t\right) \cdot \left(\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)$.

Remarks:

- ○ A single trajectory under π_θ is enough to obtain an unbiased policy gradient estimator.
- ○ It is achieved without the knowledge of the transition probabilities.
- ○ REINFORCE has a high variance due to the environment's randomness.
- ○ Notice that $\pi_\theta(a_{t_2}|s_{t_2})$ does not affect $\sum_{t=0}^{t_1} r(s_t, a_t)$ if $t_2 > t_1$.

Policy gradient estimator

REINFORCE estimator

Given that we expressed J as a function of trajectories drawn from the parameterized distribution, we can use the following recipe:

- Generate an episode $\tau = (s_0, a_0, r_0, s_1, \ldots)$ from policy π_θ;
- Construct $\widehat{\nabla}_\theta J(\pi_\theta) = \left(\sum_{t=0}^{\infty} \gamma^t r_t\right) \cdot \left(\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)$.

Remarks:

- ○ A single trajectory under π_θ is enough to obtain an unbiased policy gradient estimator.
- ○ It is achieved without the knowledge of the transition probabilities.
- ○ REINFORCE has a high variance due to the environment's randomness.
- ○ Notice that $\pi_\theta(a_{t_2}|s_{t_2})$ does not affect $\sum_{t=0}^{t_1} r(s_t, a_t)$ if $t_2 > t_1$.
  - We will use this observation to come up with a second version of the policy gradient theorem.

Policy gradient theorem (version 2): Action-value expression

Policy gradient theorem (Action-value function)

We can express the gradient of the objective in terms of the action-value as well as the score functions as follows:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \mathsf{p}_\theta}\left[\sum_{t=0}^{\infty} \gamma^t Q^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)\right]. \tag{2}$$

Remarks:

- Using the fact that $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]]$, we can rewrite the action-value expression as follows:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \mathsf{p}_\theta}\left[\sum_{t=0}^{\infty}\Big(\sum_{t'=t}^{\infty} \gamma^{t'} r(s_{t'}, a_{t'})\Big) \nabla_\theta \log \pi_\theta(a_t|s_t)\right]. \tag{\textsc{Reward2Go}}$$

- Recall the REINFORCE expression, which is given by

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \mathsf{p}_\theta}\left[\sum_{t=0}^{\infty}\Big(\sum_{t'=0}^{\infty} \gamma^{t'} r(s_{t'}, a_{t'})\Big) \nabla_\theta \log \pi_\theta(a_t|s_t)\right]. \tag{REINFORCE}$$

- If the policy π_θ can not be applied to the environment, we can estimate Q^{π_θ} via OPE.

Policy gradient estimator using reward-to-go

Gradient estimator using reward-to-go

Given that we expressed J as a function of trajectories drawn from the parameterized distribution, we can use the following recipe:

- Generate an episode $\tau = (s_0, a_0, r_0, s_1, \ldots)$ from policy π_θ;
- Construct $\widehat{\nabla}_\theta J(\pi_\theta) = \sum_{t=0}^{\infty} \gamma^t G_t \cdot \nabla_\theta \log \pi_\theta(a_t|s_t)$, where $G_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$.

Remarks: ○ The expression above is an unbiased estimator of the policy gradient.

○ This estimator is not necessarily cheaper than REINFORCE.

○ Reward-to-go ensures that each action is reinforced based on only the future rewards it contributes.

- Early actions have little effect on the later rewards.
- Actions are updated based only on their individual, relevant impact.
- Policy optimization is more directed and is more effective.

○ Unfortunately, this estimator might still have a high variance.

Policy gradient estimator using reward-to-go

Gradient estimator using reward-to-go

Given that we expressed J as a function of trajectories drawn from the parameterized distribution, we can use the following recipe:

- Generate an episode $\tau = (s_0, a_0, r_0, s_1, \dots)$ from policy π_θ;
- Construct $\widehat{\nabla}_\theta J(\pi_\theta) = \sum_{t=0}^{\infty} \gamma^t G_t \cdot \nabla_\theta \log \pi_\theta(a_t|s_t)$, where $G_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$.

Remarks:

○ The expression above is an unbiased estimator of the policy gradient.

○ This estimator is not necessarily cheaper than REINFORCE.

○ Reward-to-go ensures that each action is reinforced based on only the future rewards it contributes.

- Early actions have little effect on the later rewards.
- Actions are updated based only on their individual, relevant impact.
- Policy optimization is more directed and is more effective.

○ Unfortunately, this estimator might still have a high variance.

○ Next, we will introduce a way to reduce variance.

Policy gradient theorem (version 3): Baseline expression

Policy gradient theorem (Baseline)

For any function $b\colon \mathcal{S} \to \mathbb{R}$, we have

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \mathsf{p}_\theta} \left[\sum_{t=0}^{\infty} \gamma^t [Q^{\pi_\theta}(s_t, a_t) - b(s_t)] \nabla_\theta \log \pi_\theta(a_t | s_t) \right]. \tag{3}$$

Recall version 2 of the policy gradient theorem where we have the above expression but with $b(s_t) = 0$.

Remarks:

- ○ For any baseline $b(s)$ that does not depend on the actions:

$$\mathbb{E}_{a \sim \pi_\theta}[b(s) \nabla_\theta \log \pi_\theta(a|s)] = 0.$$

- ○ A natural choice of baseline is the value function: $b(s) = V^{\pi_\theta}(s)$.
- ○ We call $A^{\pi_\theta}(s, a) := Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$ the advantage function.
- ○ Mainly employed as a variance reduction mechanism.

Proof of baseline expression

Derivation:

- Notice that $\sum_a \pi_\theta(a|s) = 1$ for any $s \in \mathcal{S}$.
- For any $b(s)$ that is independent of actions, we have the following:

$$
\begin{aligned}
\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[b(s)\nabla_\theta \log \pi_\theta(a|s)\right] &= b(s)\sum_a \pi_\theta(a|s)\frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} \\
&= b(s)\sum_a \nabla_\theta \pi_\theta(a|s) \\
&= b(s)\nabla_\theta \sum_a \pi_\theta(a|s) \\
&= b(s)\nabla_\theta 1 \\
&= 0.
\end{aligned}
$$

Summary: Policy gradient theorem (versions 1–3)

- REINFORCE expression:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \mathsf{p}_\theta}\left[R(\tau)\left(\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)\right].$$

- Action-value expression:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \mathsf{p}_\theta}\left[\sum_{t=0}^{\infty} \gamma^t Q^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)\right].$$

- Baseline expression:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \mathsf{p}_\theta}\left[\sum_{t=0}^{\infty} \gamma^t [Q^{\pi_\theta}(s_t, a_t) - b(s_t)] \nabla_\theta \log \pi_\theta(a_t|s_t)\right].$$

Example: Policy gradient learns cartpole

- The OpenAI Gym library provides a collection of environments for RL research.
- One of them is called the *cartpole*.
  - The goal is to balance a stick on a cartpole as long as possible, by moving it to the left or right.

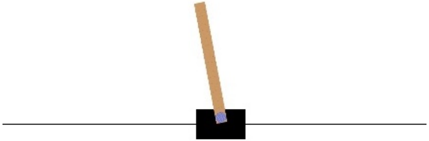


Figure: Cartpole environment. Actions are left (0) and right (1), the state is the position and velocity of the cart, and the angle and angular velocity of the pole (all encoded as floats). The episode ends when the pole is more than $\pm 12^{\circ}$ from vertical, or the cart moves more than 2.4 units from the center. Reward $+1$ for every step taken without terminating.

Example: Policy gradient learns cartpole (cont'd)

- We train an agent with neural softmax parameterization to solve this task.
  - Each time, we use a different version 1, 2 or 3 of the policy gradient theorem to build a gradient estimate.

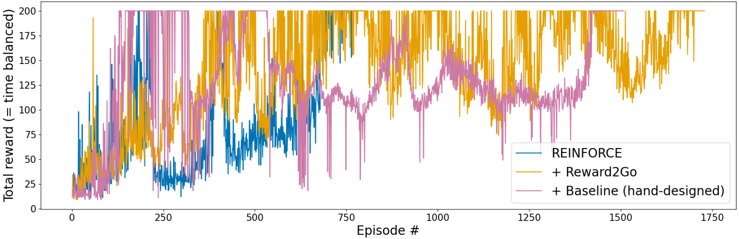


Figure: Learning curves for the cartpole environment

Example: Policy gradient learns cartpole (cont'd)

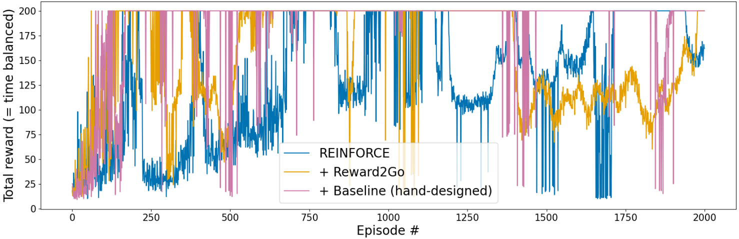


Figure: Learning curves for a different run and without stopping when the policy seems to have converged.

Remarks:

- ◦ The learning curves are only across one training run, explaining the large variance.
- ◦ Practical RL depends a lot on hyper-parameters, initialization and engineering tricks.
- ◦ The baseline we used is handcrafted not trained (negatively correlated with the pole angle).
- ◦ This is a very simple problem and we used a small feedforward NN.

Example: Policy gradient learns cartpole (cont'd)

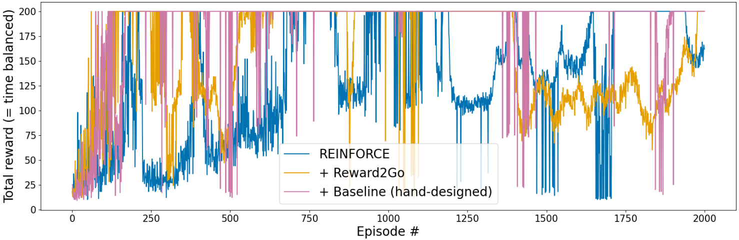


Figure: Learning curves for a different run and without stopping when the policy seems to have converged.

○ *Self-exercise:* Use the notebook we provide and check the learning curves and cartpole videos they generate.

- See moodle for the corresonding notebook. We recommend using Google Colab.
- Play around with the hyper-parameters (learning rate, discount factor, etc.).
- Try different baseline functions, and then try to use a learned one given by a NN (not handcrafted).

Policy gradient theorem (versions 4–5)

∘ Recall that the discounted state visitation distribution under a given policy π is given as below:

$$\lambda_\mu^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \mathbb{P}[s_t = s | s_0 \sim \mu, \pi].$$

Policy gradient theorem via occupancy measures

- Action-value expression:

$$\nabla_\theta J(\pi_\theta) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim\lambda_\mu^{\pi_\theta}}\left[\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)\right]\right]. \tag{4}$$

- Advantage expression:

$$\nabla_\theta J(\pi_\theta) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim\lambda_\mu^{\pi_\theta}}\left[\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[A^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)\right]\right]. \tag{5}$$

Remark: ∘ The proof follows immediately based on the definition of $\lambda_\mu^\pi(s)$.

Policy gradient theorem (versions 4–5): Remarks

- ∘ Constructing unbiased stochastic policy gradients requires sampling from $\lambda_\mu^\pi(s)$ (versions 4–5).
- ∘ This can be achieved by generating (s_T, a_T) with a random horizon $T \sim$ Geometric$(1-\gamma)$.
  - ▸ Indeed, it holds that

$$\mathbb{P}[s_T = s|s_0 \sim \mu, \pi] = \sum_{t=0}^{\infty} \gamma^t (1-\gamma)\mathbb{P}[s_t = s|s_0 \sim \mu, \pi] = (1-\gamma)\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}\{s_t = s\}|s_0 \sim \mu, \pi\right] = \lambda_\mu^\pi(s).$$

- ∘ Unbiased estimator of $A^{\pi_{\boldsymbol{\theta}}}(s,a)$ requires two random rollouts to estimate $Q^{\pi_{\boldsymbol{\theta}}}(s,a)$ and $V^{\pi_{\boldsymbol{\theta}}}(s)$, separately.

Exercise: Policy gradient under tabular parameterization

- Compute policy gradient under the direct and softmax parameterization in the tabular setting.

Direct parameterization

In direct parameterization, the policy is given by

$$\pi_\theta(a|s) = \theta_{s,a},$$

where $\theta_{s,a} \geq 0$ and $\sum_{a\in\mathcal{A}} \theta_{s,a} = 1$.

Softmax parameterization

In softmax parameterization, the policy is given by

$$\pi_\theta(a|s) = \frac{\exp(\theta_{s,a})}{\sum_{a'\in\mathcal{A}} \exp(\theta_{s,a'})},$$

where $\theta \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{S}|}$.

Exercise:

- Derive $\frac{\partial J(\pi_\theta)}{\partial \theta_{s,a}}$ via the chain-rule (the solutions are on the slide 34).

Monte Carlo policy gradient method

REINFORCE: Monte-Carlo policy-gradient method

```
Initialize the policy parameter $\theta_1 \in \mathbb{R}^d$, the step-size $\alpha > 0$, the baseline function $b(\cdot)$. Choose a total number of episodes $E$, i.e., the number of gradient updates.
for $e = 1, \ldots, E$ do
    Generate an episode $s_0, a_0, r_0, ..., s_T, a_T, r_T$ following $\pi_\theta$ using the chosen parameterization. Initialize the gradient estimate $\widehat{\nabla} J(\theta_e) = 0$.
    for each step $t = 0, 1, ..., T$ of the episode do
        Compute return $G_t \leftarrow \sum_{i=t}^{T} \gamma^{i-t} r_i$                    à la Reward2Go
        Compute advantage estimate $A_t \leftarrow G_t - b(s_t)$
        $\widehat{\nabla} J(\theta_e) \leftarrow \widehat{\nabla} J(\theta_e) + \gamma^t A_t \cdot \nabla_\theta \log \pi_\theta(a_t|s_t)$     à la Reward2Go
    end for
    $\theta_{e+1} \leftarrow \theta_e + \alpha \widehat{\nabla} J(\theta_e)$                note that the step-size might depend on $e$
end for
```

Remarks:

- ◦ The policy is updated only after generating a whole trajectory, which may not be efficient.
- ◦ We can use the idea of temporal difference learning to build policy gradient estimators.

Policy gradient method with value function estimation

Online actor-critic algorithm

```
Initialize $\theta_0$, $w_0$, state $s_0 \sim \mu$,$a_0 \sim \pi_{\theta_0}(\cdot|s_0)$
for each step $t = 0, 1, ..., T$ of the episode do
    Obtain $(r_t, s_{t+1}, a_{t+1})$ from $\pi_{\theta_t}$
    Compute temporal difference: $\delta_t = r_t + \gamma Q_{w_t}(s_{t+1}, a_{t+1}) - Q_{w_t}(s_t, a_t)$
    Compute policy gradient estimator:
        $\widehat{\nabla}_\theta J(\pi_{\theta_t}) = Q_{w_t}(s_t, a_t) \nabla_\theta \log \pi_{\theta_t}(a_t \mid s_t)$
    Update $\theta$: $\theta_{t+1} = \theta_t + \alpha \widehat{\nabla}_\theta J(\pi_{\theta_t})$
    Update $\boldsymbol{w}$: $w_{t+1} = w_t - \beta \delta_t \nabla_w Q_{w_t}(s_t, a_t)$
end for
```

Remarks:

◦ Approximating the value function in policy gradient introduces extra bias.

◦ There are various ways to estimate the advantage function [22].

Summary: Policy gradient methods

- Advantages
  - Directly optimize policy parameters (but still need to evaluate value functions)
  - Can deal with high-dimensional and continuous action spaces
  - Can learn stochastic policies

- Optimization Challenges:
  - Nonconcave landscape (in general, only converge to stationary points)
  - Sensitive to stepsize choice
  - High variance/bias of the policy gradient estimators

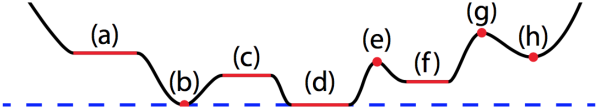


Figure: A non-convex function. (a) and (c) are plateaus, (b) and (d) are global minima, (f) and (h) are local minima, (e) and (g) are local maxima. [9]

Recap: Policy-based methods

Policy optimization

In policy optimization, we seek to obtain numerical solutions to the following objective function:

$$\max_{\theta} J(\pi_\theta) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 \sim \mu, \pi_\theta\right] = \mathbb{E}_{s\sim\mu}[V^{\pi_\theta}(s)].$$

Tabular parameterization

- Direct :

$$\pi_\theta(a|s) = \theta_{s,a}, \text{ with } \theta_{s,a} \geq 0, \sum\nolimits_a \theta_{s,a} = 1.$$

- Softmax:

$$\pi_\theta(a|s) = \frac{\exp(\theta_{s,a})}{\sum_{a'\in\mathcal{A}} \exp(\theta_{s,a'})}.$$

Non-tabular parameterization

- Softmax:

$$\pi_\theta(a|s) = \frac{\exp(f_\theta(s,a))}{\sum_{a'\in\mathcal{A}} \exp(f_\theta(s,a'))}.$$

- Gaussian:

$$\pi_\theta(a|s) \sim \mathcal{N}\left(\mu_\theta(s), \sigma_\theta^2(s)\right).$$

Recap: Policy gradient theorems

○ Recall that $\mathrm{p}_\theta(\tau)$ is the trajectory distribution and $\lambda_\mu^\pi(s)$ is the discounted state visitation distribution.

Policy gradient theorems

- REINFORCE expression is given by

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim \mathrm{p}_\theta}\left[R(\tau)\left(\sum_{t=0}^{\infty}\nabla_\theta \log \pi_\theta(a_t|s_t)\right)\right].$$

- Action-value expression is given by

$$\begin{aligned}\nabla_\theta J(\pi_\theta) &= \mathbb{E}_{\tau\sim \mathrm{p}_\theta}\left[\sum_{t=0}^{\infty}\gamma^t Q^{\pi_\theta}(s_t,a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\right]\\ &= \frac{1}{1-\gamma}\mathbb{E}_{s\sim\lambda_\mu^{\pi_\theta}, a\sim\pi_\theta(\cdot|s)}\left[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)\right].\end{aligned}$$

Policy gradient in tabular setting (solution)

- Direct parameterization: $\pi_\theta(a|s) = \theta_{s,a}$,

$$\frac{\partial J(\pi_\theta)}{\partial \theta_{s,a}} = \frac{1}{1-\gamma} \lambda_\mu^{\pi_\theta}(s) Q^{\pi_\theta}(s,a).$$

- Softmax parameterization: $\pi_\theta(a|s) \propto \exp(\theta_{s,a})$,

$$\frac{\partial J(\pi_\theta)}{\partial \theta_{s,a}} = \frac{1}{1-\gamma} \lambda_\mu^{\pi_\theta}(s) \pi_\theta(a|s) A^{\pi_\theta}(s,a).$$

Proofs:

- Recall that $\nabla_\theta J(\pi_\theta) = \frac{1}{1-\gamma} \sum_s \lambda_\mu^{\pi_\theta}(s) \sum_a Q^{\pi_\theta}(s,a) \nabla_\theta \pi_\theta(a|s)$.

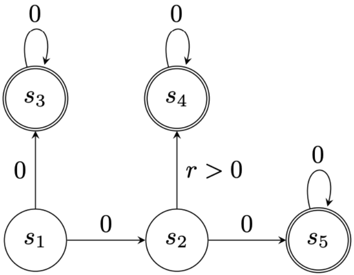
- Direct case: $\frac{\partial \pi_\theta(a|s)}{\partial \theta_{s',a'}} = \mathbb{1}\{s = s', a = a'\}$.

- Softmax case: $\frac{\partial \pi_\theta(a|s)}{\partial \theta_{s',a'}} = \pi_\theta(a|s) \mathbb{1}\{s = s', a = a'\} - \pi_\theta(a|s) \pi_\theta(a'|s) \mathbb{1}\{s = s'\}$.

Optimization challenge I: Nonconcavity

∘ In general, the objective $J(\pi_\theta)$ is nonconcave.

∘ This holds even for the tabular setting with direct or softmax parameterization.



a_1: move up, a_2: move right

Example (direct parameterization)

$$V^\pi(s_1) = \pi(a_2|s_1)\pi(a_1|s_2)r.$$

- Consider $\pi_{\text{mid}} = \frac{\pi_1+\pi_2}{2}$, where

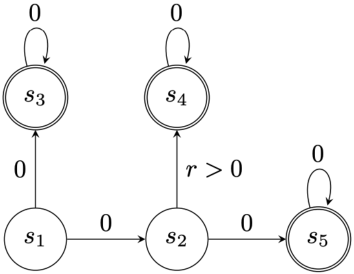
$$\begin{aligned} \pi_1(a_2|s_1) &= 3/4, & \pi_1(a_1|s_2) &= 3/4; \\ \pi_2(a_2|s_1) &= 1/4, & \pi_2(a_1|s_2) &= 1/4; \\ \pi_{\text{mid}}(a_2|s_1) &= 1/2, & \pi_{\text{mid}}(a_1|s_2) &= 1/2. \end{aligned}$$

- $V^{\pi_1}(s_1) = \frac{9}{16}r, V^{\pi_2}(s_1) = \frac{1}{16}r.$
- $V^{\pi_{\text{mid}}}(s_1) = \frac{1}{4}r < \frac{1}{2}(V^{\pi_1}(s_1) + V^{\pi_2}(s_1)).$

Optimization challenge I: Nonconcavity

- In general, the objective $J(\pi_\theta)$ is nonconcave.

- This holds even for tabular setting with direct or softmax parameterization.



a_1: move up, a_2: move right

Example (softmax parameterization)

$$\theta = (\theta_{a_1,s_1}, \theta_{a_2,s_1}, \theta_{a_1,s_2}, \theta_{a_2,s_2}),$$

$$V^{\pi_\theta}(s_1) = \frac{e^{\theta_{a_2,s_1}}}{e^{\theta_{a_1,s_1}} + e^{\theta_{a_2,s_1}}} \frac{e^{\theta_{a_1,s_2}}}{e^{\theta_{a_1,s_2}} + e^{\theta_{a_2,s_2}}} r.$$

- Consider

$$\begin{aligned} \theta_1 &= (\log 1, \log 3, \log 3, \log 1), \\ \theta_2 &= (-\log 1, -\log 3, -\log 3, -\log 1), \\ \theta_{\text{mid}} &= (\theta_1 + \theta_2)/2 = (0,0,0,0). \end{aligned}$$

- $V^{\pi_{\theta_1}}(s_1) = \frac{9}{16} r, V^{\pi_{\theta_2}}(s_1) = \frac{1}{16} r.$
- $V^{\pi_{\theta_{\text{mid}}}}(s_1) = \frac{1}{4} r < \frac{1}{2}(V^{\pi_{\theta_1}}(s_1) + V^{\pi_{\theta_2}}(s_1)).$

Convergence to stationary points (see Lecture 1)

Convergence of exact policy gradient method: $\theta_{t+1} = \theta_t + \alpha_t \nabla_\theta J(\pi_{\theta_t})$ (Nesterov, 2004 [18])

If the objective $J(\pi_\theta)$ is L-smooth and set $\alpha_t = \frac{1}{L}$, then we have the following guarantee:

$$\min_{t=0,\ldots,T-1} \|\nabla_\theta J(\pi_{\theta_t})\|_2^2 \leq \frac{2L(J(\pi_{\theta^\star}) - J(\pi_{\theta_0}))}{T}.$$

Convergence of stochastic policy gradient method: $\theta_{t+1} = \theta_t + \alpha_t \widehat{\nabla}_\theta J(\pi_{\theta_t})$ (Ghadimi and Lan, 2013 [8])

If the objective $J(\pi_\theta)$ is L-smooth and $\widehat{\nabla}_\theta J(\pi_\theta)$ is unbiased and has bounded variance by σ^2, then with a proper choice of the step-size, we have the following guarantee:

$$\min_{t=0,\ldots,T-1} \mathbb{E}\left[\|\nabla_\theta J(\pi_{\theta_t})\|_2^2\right] = O\left(\sqrt{\frac{L(J(\pi_{\theta^\star}) - J(\pi_{\theta_0}))\sigma^2}{T}}\right).$$

Questions: Can these rates be further improved? Do stationary points imply good performance?

Optimization challenge II: Vanishing gradient and saddle points

- In general, there are no guarantees on the quality of stationary points.
- Vanishing gradients can happen when using softmax parameterization.
- Vanishing gradients can happen when lacking sufficient exploration [1].

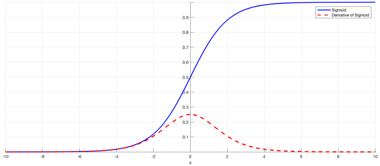


Figure: Softmax function: $\frac{e^\theta}{1+e^\theta} = \frac{1}{1+e^{-\theta}}$.

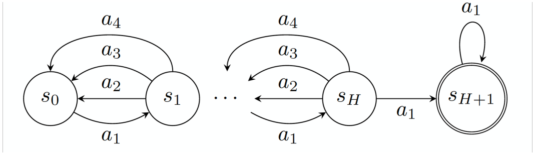


Figure: Example with $H + 2$ states and $\gamma = \frac{H}{H+1}$: rewards are everywhere 0 except at s_{H+1}. For small order p and θ such that $\theta_{s,a_1} < \frac{1}{4}$ for all s [1]: $\|\nabla^p V^{\pi_\theta}(s_0)\| \leq \left(\frac{1}{3}\right)^{H/4}$.

A simple example

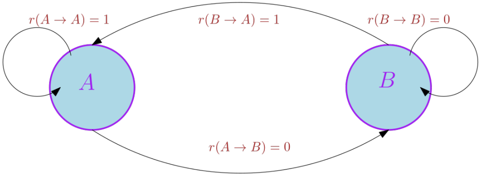


Figure: MDP with 2 states and 2 actions

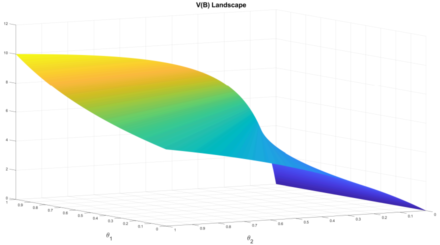
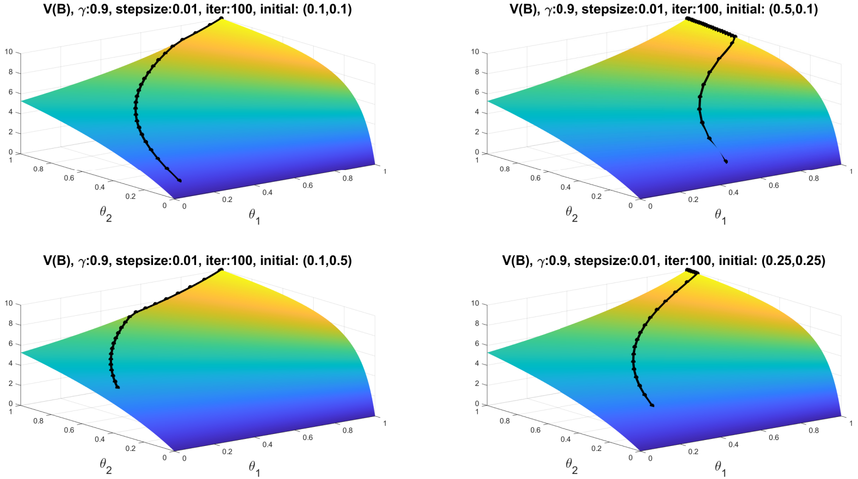


Figure: $V^{\pi}(B)$ under direct parameterization

A simple example (cont'd)



V(B), γ:0.9, stepsize:0.01, iter:100, initial: (0.1,0.1)

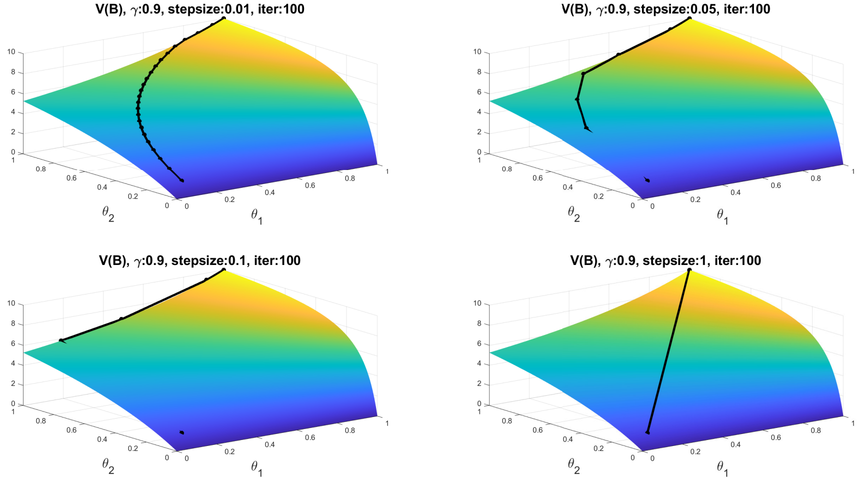
V(B), γ:0.9, stepsize:0.01, iter:100, initial: (0.5,0.1)

V(B), γ:0.9, stepsize:0.01, iter:100, initial: (0.1,0.5)

V(B), γ:0.9, stepsize:0.01, iter:100, initial: (0.25,0.25)

Figure: PG with different initial points

A simple example (cont'd)



V(B), γ:0.9, stepsize:0.01, iter:100

V(B), γ:0.9, stepsize:0.05, iter:100

V(B), γ:0.9, stepsize:0.1, iter:100

V(B), γ:0.9, stepsize:1, iter:100

Figure: PG with different stepsizes

Fundamental questions

Question 1

When do policy gradient methods converge to an optimal solution? If so, how fast?

Remarks:

- Optimization wisdom: GD/SGD could converge to the global optima for "convex-like" functions:

$$J(\pi^\star) - J(\pi) = O(\|\nabla J(\pi)\|).$$

- We focus on the tabular setting with exact gradients.

Question 2

How can we avoid vanishing gradients and improve the convergence?

Remarks:

- Optimization wisdom: Use divergence with good curvature information.
- Switch to natural policy gradient by exploiting geometry.

Performance difference lemma (PDL)

Performance difference lemma (Kakade and Langford, 2002 [11])

For any two policy π, π', the following holds

$$J(\pi) - J(\pi') = \frac{1}{1-\gamma}\mathbb{E}_{s\sim\lambda_\mu^\pi,\ a\sim\pi(\cdot|s)}\left[A^{\pi'}(s,a)\right].$$

The difference in expected returns between two policies π and π' depends on how the new policy π samples actions compared to the advantage function of the old policy π', weighted by the state distribution $\lambda_\mu^\pi(s)$:

- If $A^{\pi'}(s,a) > 0$ for actions that π selects more often, then $J(\pi) > J(\pi')$, i.e., π improves upon π'.
- If $A^{\pi'}(s,a) < 0$ for actions that π selects more often, then π performs worse than π'.
- If π is a small update from π', this lemma helps approximate the expected improvement.

Remarks:

◦ Here $\lambda_\mu^\pi(s) = (1-\gamma)\mathbb{E}[\sum_{t=0}^\infty \gamma^t \mathbb{1}_{\{s_t=s\}}|s_0 \sim \mu, \pi]$ is the state visitation distribution.

◦ Here $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$ is the advantage function.

◦ *Can be used to show policy improvement theorem for policy iteration (self-exercise).

◦ *Can also be used to show policy gradient theorem (self-exercise).

◦ Proof follows from the definition of value functions (see supplementary material, slide 11).

Key insight: Policy optimization is convex-like in the full policy space

- Performance difference lemma:

$$J(\pi^\star) - J(\pi) = \frac{1}{1-\gamma} \sum_s \lambda_\mu^{\pi^\star}(s) \sum_a \pi^\star(a|s) A^\pi(s,a).$$

- Policy gradient theorem (tabular setting):

$$\frac{\partial J(\pi)}{\partial \pi(a|s)} = \frac{1}{1-\gamma} \lambda_\mu^\pi(s) Q^\pi(s,a) \qquad \text{(direct parameterization).}$$

$$\frac{\partial J(\pi)}{\partial \pi(a|s)} = \frac{1}{1-\gamma} \lambda_\mu^\pi(s) \pi(a|s) A^\pi(s,a) \qquad \text{(softmax parameterization).}$$

- This seems to imply gradient dominance type properties:

$$J(\pi^\star) - J(\pi) = O(\max_{\bar\pi \in \Delta} \langle \bar\pi - \pi, \nabla J(\pi) \rangle),$$

which is crucial to ensure global optimality.

- *Closely related to the Łojasiewicz inequality, which is more nuanced.

*Policy optimization (direct parameterization)

◦ We first consider the direct parameterization in the tabular setting.

Policy optimization under direct parameterization

$$\max_{\pi \in \Delta(\mathcal{A})^{|\mathcal{S}|}} J(\pi) := \mathbb{E}_{s\sim\mu}[V^\pi(s)],$$

where $\Delta(\mathcal{A})^{|\mathcal{S}|} = \{\pi : \pi(a|s) \geq 0, \sum_{a\in\mathcal{A}} \pi(a|s) = 1, \forall s\}$. For brevity, we denote this set as Δ.

Remarks:

- If $\pi \in \Delta$ is optimal, then it satisfies the first-order optimality condition:

  $$\langle \bar{\pi} - \pi, \nabla J(\pi)\rangle \leq 0, \forall\, \bar{\pi} \in \Delta,$$

  or equivalently, $\max_{\bar{\pi}\in\Delta} \langle \bar{\pi} - \pi, \nabla J(\pi)\rangle = 0$.
- Does the reverse statement hold?

*Gradient dominance property

Gradient mapping domination

For any policy π, we have

$$J(\pi^\star) - J(\pi) \leq \left\| \frac{\lambda_\mu^{\pi^\star}}{\lambda_\mu^\pi} \right\|_\infty \cdot \max_{\bar{\pi} \in \Delta} \langle \bar{\pi} - \pi, \nabla J(\pi) \rangle .$$

Remarks:

- This shows that $J(\pi)$ satisfies the Polyak-Łojasiewicz (PL) condition [12].
- Any first-order stationary point is thus globally optimal ($\max_{\bar{\pi} \in \boldsymbol{\Delta}} \langle \bar{\pi} - \pi, \nabla \mathbf{J}(\pi) \rangle = \mathbf{0} \Rightarrow \mathbf{J}(\pi) = \mathbf{J}(\pi^\star)$).
- The term $\left\| \frac{\lambda_\mu^{\pi^\star}}{\lambda_\mu^\pi} \right\|_\infty$ is called the distribution mismatch coefficient.
  - This coefficient captures the hardness of the exploration problem.
  - Note that in the vanishing gradient example, this coefficient can be exponentially large.
  - Note that $\max_\pi \left\| \frac{\lambda_\mu^{\pi^\star}}{\lambda_\mu^\pi} \right\|_\infty \leq \frac{1}{1-\gamma} \left\| \frac{\lambda_\mu^{\pi^\star}}{\mu} \right\|_\infty$, since $\forall \pi, \lambda_\mu^\pi(s) \geq (1-\gamma)\mu(s)$.
- Proof via performance difference lemma (see supplementary material, slide 12).

Projected policy gradient method

Projected policy gradient method

By projected policy gradient method, we mean the iteration variant below

$$\pi_{t+1} = \Pi_\Delta(\pi_t + \eta \nabla J(\pi_t)),$$

where the projection is given by $\Pi_\Delta(\pi) = \arg\min_{\pi' \in \Delta} \|\pi - \pi'\|_2^2$.

Remarks:

- ○ Take a gradient ascent step and project onto the simplex set (can be computed efficiently).
- ○ *Generalized gradient mapping*: $G(\pi_t) = \frac{1}{\eta}(\pi_{t+1} - \pi_t)$, or equivalently, $\pi_{t+1} = \pi_t + \eta G(\pi_t)$.
- ○ If π is optimal, then $G(\pi) = 0$.
- ○ Convergence of the gradient mapping [17]: If $J(\pi)$ is L-smooth, then we have

$$\min_{t \le T} \|G(\pi_t)\|_2^2 \le \frac{2L(J(\pi^\star) - J(\pi_0))}{T}.$$

*Convergence of projected policy gradient method

Theorem (Agarwal et al., 2020 [1])

Assume access to exact gradients. Let $\eta = \frac{(1-\gamma)^3}{2\gamma|A|}$. Then, the projected policy gradient method achieves the following guarantee:

$$\min_{t<T} J(\pi^\star) - J(\pi_t) \leq \frac{8\sqrt{\gamma|\mathcal{S}||\mathcal{A}|}}{(1-\gamma)^3\sqrt{T}} \left\| \frac{\lambda_\mu^{\pi^\star}}{\mu} \right\|_\infty .$$

Remarks:

- See supplementary material, slide 15 for the proof.
- We have large constants $\frac{\sqrt{|\mathcal{S}||\mathcal{A}|}}{(1-\gamma)^3} \left\| \frac{\lambda_\mu^{\pi^\star}}{\mu} \right\|_\infty$ in the bound and a slow rate $\frac{1}{\sqrt{T}}$ in T.
- In the tabular setting, VI or PI converges linearly, which is much faster.
- However: Linear convergence of PG can be shown with larger step-sizes through line-search [3].

Policy optimization (softmax parameterization)

∘ We now consider the softmax parameterization in the tabular setting.

Policy optimization under softmax parameterization

Consider the objective

$$\max_{\theta} J(\pi_\theta) := \mathbb{E}_{s\sim\mu}[V^{\pi_\theta}(s)], \qquad \text{where} \qquad \pi_\theta(a|s) = \frac{\exp(\theta_{s,a})}{\sum_{a'} \exp(\theta_{s,a'})}.$$

Softmax policy gradient method

By softmax policy gradient method, we mean the iteration variant below

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta J(\pi_{\theta_t}), \quad \text{where} \quad \frac{\partial J(\theta)}{\partial \theta_{s,a}} = \frac{1}{1-\gamma} \lambda_\mu^{\pi_\theta}(s) \pi_\theta(a|s) A^{\pi_\theta}(s,a).$$

*Convergence of softmax policy gradient method

Convergence of softmax policy gradient (Mei et al., 2020 [14])

Assume access to exact gradients, let $\eta \leq \frac{(1-\gamma)^3}{8}$. Then, the iterates of the softmax policy gradient method satisfy the following guarantee:

$$J(\pi^\star) - J(\pi_{\theta_T}) \leq \frac{16|\mathcal{S}|}{c^2(1-\gamma)^5 T} \left\| \frac{\lambda_\mu^{\pi^\star}}{\mu} \right\|_\infty^2,$$

where $c = [\min_{s,t} \pi_{\theta_t}(a^\star(s)|s)]^{-1} > 0$.

Remark:

- ◦ See supplementary material, slide 17 for the derivation of the algorithm.
- ◦ The proof is similar to the one in the projected setting.
- ◦ Once more, we have a slow rate and large constants in the bound.

Natural policy gradient method (NPG)

○ Natural policy gradient (NPG) allows us to overcome the large constant in the convergence bound.

Natural policy gradient (Kakade, 2002 [10])

By natural policy gradient (NPG), we mean the following iteration variant below:

$$\theta_{t+1} = \theta_t + \eta (F_{\theta_t})^\dagger \nabla J(\pi_{\theta_t}),$$

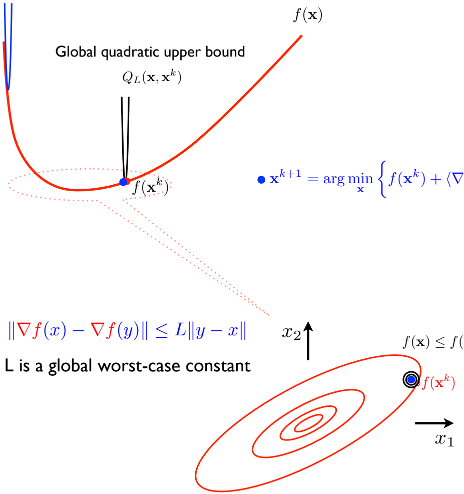
where

- F_θ is the Fisher information matrix:

$$F_\theta = \mathbb{E}_{s \sim \lambda_\mu^{\pi_\theta}, a \sim \pi_\theta(\cdot|s)} \left[\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^\top \right].$$

- $C^\dagger$ is the pseudoinverse of a matrix C.

How can we better adapt to the local geometry?



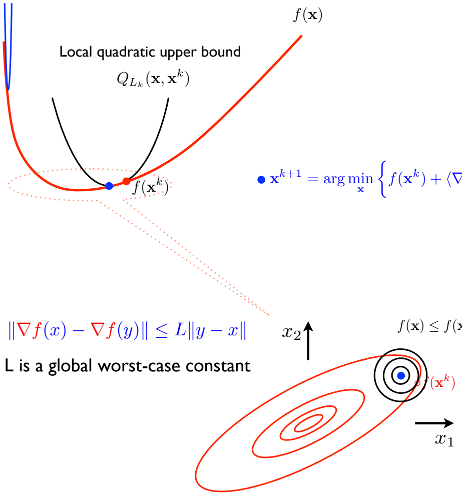
● $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L}{2} \|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$

$\|\nabla f(x) - \nabla f(y)\| \leq L\|y - x\|$

L is a global worst-case constant

$f(\mathbf{x}) \leq f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{L}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2$

How can we better adapt to the local geometry?


$f(\mathbf{x})$
Local quadratic upper bound
$Q_{L_k}(\mathbf{x}, \mathbf{x}^k)$
$f(\mathbf{x}^k)$
x_2
$f(\mathbf{x}^k)$
x_1

- $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L_k}{2} \|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$

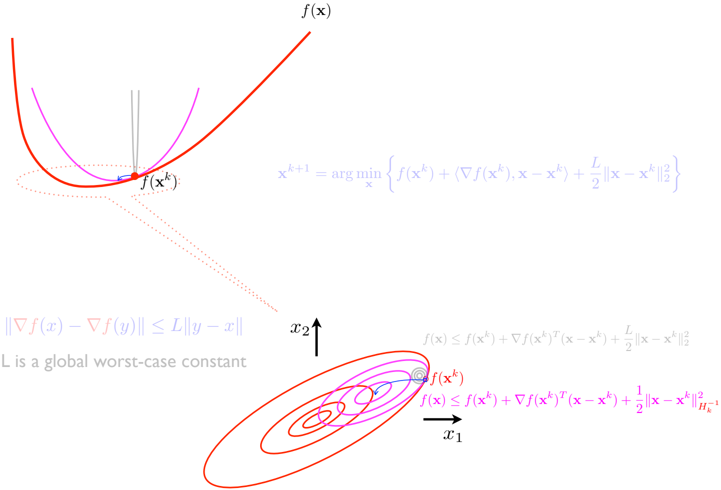
$$\|\nabla f(x) - \nabla f(y)\| \leq L\|y - x\|$$

L is a global worst-case constant

$$f(\mathbf{x}) \leq f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{L_k}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2$$

applies only locally

How can we better adapt to the local geometry?



$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$$

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|y - x\|$$

L is a global worst-case constant

$$f(\mathbf{x}) \leq f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{L}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2$$

$$f(\mathbf{x}) \leq f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{1}{2}\|\mathbf{x} - \mathbf{x}^k\|_{H_k^{-1}}^2$$

NPG under softmax parameterization

- Under softmax parameterization, NPG allows closed-form updates that can be computed efficiently.
- Consider softmax parameterization $\pi_\theta(a|s) = \frac{\exp(\theta_{s,a})}{\sum_{a'} \exp(\theta_{s,a'})}$ and denote $\pi_t = \pi_{\theta_t}$.
- The policy update below shows that NPG then correponds to a policy mirror ascent update.

NPG update

Under softmax parameterization, NPG corresponds to the following parameter update involving the advantage function $A^{\pi_{\theta_t}}(s,a)$,

$$\theta_{t+1} = \theta_t + \frac{\eta}{1-\gamma} A^{\pi_{\theta_t}},$$

which is equivalent to the following policy update:

$$\pi_{t+1}(a|s) = \pi_t(a|s) \frac{\exp(\eta A^{\pi_t}(s,a)/(1-\gamma))}{\sum_{a'} \pi_t(a'|s) \exp(\eta A^{\pi_t}(s,a')/(1-\gamma))}.$$

Convergence of NPG under softmax parameterization

Convergence of NPG with softmax parameterization [1]

Assume access to the advantage function A^{π_θ}. For any $\eta \geq (1-\gamma)^2 \log|\mathcal{A}|$ and $T > 0$, NPG with softmax parameterization satisfies the following guarantee:

$$J(\pi^\star) - J(\pi_{\theta_T}) \leq \frac{2}{(1-\gamma)^2 T}.$$

Remarks:

- ○ We have dimension-free convergence, i.e. no dependence on $|\mathcal{A}|, |\mathcal{S}|$.
- ○ We have no dependence on distribution mismatch coefficient.

Questions: Why? What about the function approximation setting? Can we further improve the convergence?

Next week!

- Recap on policy gradient methods
- A deeper look at the natural policy gradient method

References I

[1] Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan.
Optimality and approximation with policy gradient methods in markov decision processes.
In *Conference on Learning Theory*, pages 64–66. PMLR, 2020.
41, 51, 59, 75, 76

[2] Leemon Baird.
Residual algorithms: Reinforcement learning with function approximation.
In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.
4

[3] Jalaj Bhandari and Daniel Russo.
On the linear convergence of policy gradient methods for finite mdps.
In *International Conference on Artificial Intelligence and Statistics*, pages 2386–2394. PMLR, 2021.
51, 76

[4] Justin A. Boyan and Andrew W. Moore.
Generalization in reinforcement learning: Safely approximating the value function.
In *Advances in Neural Information Processing Systems 7*, pages 369–376. MIT Press, 1995.
4

[5] Steven Bradtke.
Reinforcement learning applied to linear quadratic regulation.
In S. Hanson, J. Cowan, and C. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann, 1992.
4

References II

[6] Volkan Cevher.
Lecture 4: The role of computation.
https://www.epfl.ch/labs/lions/wp-content/uploads/2025/02/Lecture_4_2024.pdf, 2024.
Accessed: 2025-02-21.
12

[7] P. Cichosz.
Truncating temporal differences: On the efficient implementation of td(lambda) for reinforcement learning, 1995.
4

[8] Saeed Ghadimi and Guanghui Lan.
Stochastic first- and zeroth-order methods for nonconvex stochastic programming.
SIAM Journal on Optimization, 23(4):2341–2368, 2013.
40

[9] Benjamin D Haeffele and René Vidal.
Global optimality in tensor factorization, deep learning, and beyond.
arXiv preprint arXiv:1506.07540, 2015.
34

[10] S. Kakade.
A natural policy gradient.
In *Advances in Neural Information Processing Systems (NeurIPS)*, 2001.
8, 54

References III

[11] Sham Kakade and John Langford.

Approximately optimal approximate reinforcement learning.

In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

46

[12] Hamed Karimi, Julie Nutini, and Mark Schmidt.

Linear convergence of gradient and proximal-gradient methods under the polyak-undefinedojasiewicz condition.

In *European Conference on Machine Learning and Knowledge Discovery in Databases - Volume 9851*, ECML PKDD 2016, pages 795–811, Berlin, Heidelberg, 2016. Springer-Verlag.

49

[13] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi.

Optimization by simulated annealing.

science, 220(4598):671–680, 1983.

8

[14] Jincheng Mei, Chenjun Xiao, Csaba Szepesvari, and Dale Schuurmans.

On the global convergence rates of softmax policy gradient methods.

In *International Conference on Machine Learning*, pages 6820–6829. PMLR, 2020.

53, 80

References IV

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis.

Human-level control through deep reinforcement learning.

Nature, 518(7540):529–533, February 2015.

4

[16] David E Moriarty, Alan C Schultz, and John J Grefenstette.

Evolutionary algorithms for reinforcement learning.

Journal of Artificial Intelligence Research, 11:241–276, 1999.

8

[17] Yu Nesterov.

Gradient methods for minimizing composite functions.

Mathematical Programming, 140(1):125–161, 2013.

50, 74, 75

[18] Yurii Nesterov.

Introductory Lectures on Convex Optimization.

Kluwer, Boston, MA, 2004.

40

References V

[19] Satinder Singh Richard and Richard C. Yee.
An upper bound on the loss from approximate optimal-value functions.
In *Machine Learning*, pages 227–233, 1994.
4

[20] Stuart J Russell and Peter Norvig.
Artificial intelligence a modern approach.
2010.
8

[21] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever.
Evolution strategies as a scalable alternative to reinforcement learning.
arXiv preprint arXiv:1703.03864, 2017.
8

[22] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel.
High-dimensional continuous control using generalized advantage estimation.
arXiv preprint arXiv:1506.02438, 2015.
33

[23] Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesvári.
Convergence results for single-step on-policy reinforcement-learning algorithms.
Machine learning, 38(3):287–308, 2000.
4

References VI

[24] Richard S Sutton, Andrew G Barto, et al.
Introduction to reinforcement learning, volume 135.
MIT press Cambridge, 1998.
4

[25] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al.
Policy gradient methods for reinforcement learning with function approximation.
In *Conference on Neural Information Processing Systems*, pages 1057–1063, 1999.
8

[26] V.B. Tadic.
On the almost sure rate of convergence of linear stochastic approximation algorithms.
IEEE Transactions on Information Theory, 50(2):401–409, 2004.
4

[27] Chen Tessler, Guy Tennenholtz, and Shie Mannor.
Distributional policy optimization: An alternative approach for continuous control.
Advances in Neural Information Processing Systems, 32:1352–1362, 2019.
9

[28] Harm van Seijen, Ashique Rupam Mahmood, Patrick M. Pilarski, Marlos C. Machado, and Richard S. Sutton.
True online temporal-difference learning.
CoRR, abs/1512.04087, 2015.
4

References VII

[29] Christopher John Cornish Hellaby Watkins.
Learning from Delayed Rewards.
PhD thesis, King's College, Cambridge, UK, May 1989.
4

[30] Ronald J Williams.
Simple statistical gradient-following algorithms for connectionist reinforcement learning.
Machine learning, 8(3-4):229–256, 1992.
16

Supplementary Material

Deferred proofs

Proof of action-value expression

Proof

For any state s_0, we have

$$\begin{aligned}
\nabla V^{\pi_\theta}(s_0) &= \nabla \sum_{a_0} \pi_\theta(a_0|s_0) Q^{\pi_\theta}(s_0, a_0) && \text{(by definition of } Q^{\pi_{\boldsymbol{\theta}}}\text{)} \\
&= \sum_{a_0} \nabla \pi_\theta(a_0|s_0) Q^{\pi_\theta}(s_0, a_0) + \sum_{a_0} \pi_\theta(a_0|s_0) \nabla Q^{\pi_\theta}(s_0, a_0) && \text{(by chain rule)} \\
&= \sum_{a_0} \nabla \pi_\theta(a_0|s_0) Q^{\pi_\theta}(s_0, a_0) + \sum_{a_0} \pi_\theta(a_0|s_0) \nabla \left(r(s_0, a_0) + \gamma \sum_{s_1} \mathsf{P}(s_1|s_0, a_0) V^{\pi_\theta}(s_1) \right) \\
&= \sum_{a_0} \pi_\theta(a_0|s_0) \nabla \log \pi_\theta(a_0|s_0) Q^{\pi_\theta}(s_0, a_0) + \gamma \sum_{a_0, s_1} \pi_\theta(a_0|s_0) \mathsf{P}(s_1|s_0, a_0) \nabla V^{\pi_\theta}(s_1).
\end{aligned}$$

Proof of action-value expression (cont'd)

Continued.

By induction, we have

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \sum_{s_0} \mu(s_0) \nabla V^{\pi_\theta}(s_0) \\
&= \mathbb{E}_{\tau \sim \mathsf{p}_\theta} \left[Q^{\pi_\theta} \left(s_0, a_0\right) \nabla \log \pi_\theta \left(a_0 | s_0\right) \right] + \gamma \mathbb{E}_{\tau \sim \mathsf{p}_\theta} \left[\nabla V^{\pi_\theta} \left(s_1\right) \right] \\
&= \mathbb{E}_{\tau \sim \mathsf{p}_\theta} \left[Q^{\pi_\theta} \left(s_0, a_0\right) \nabla \log \pi_\theta \left(a_0 | s_0\right) \right] + \gamma \mathbb{E}_{\tau \sim \mathsf{p}_\theta} \left[Q^{\pi_\theta} \left(s_1, a_1\right) \nabla \log \pi_\theta \left(a_1 | s_1\right) \right] + \dots \\
&= \mathbb{E}_{\tau \sim \mathsf{p}_\theta} \left[\sum_{t=0}^{\infty} \gamma^t Q^{\pi_\theta}(s_t, a_t) \nabla \log \pi_\theta(a_t | s_t) \right].
\end{aligned}$$

□

Proof of performance difference lemma

Derivation:

$$
\begin{aligned}
V^{\pi}(s) - V^{\pi'}(s) &= \mathbb{E}_{\tau \sim \mathsf{p}_\pi(\tau)}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s\Big] - V^{\pi'}(s) \\
&= \mathbb{E}_{\tau \sim \mathsf{p}_\pi(\tau)}\Big[\sum_{t=0}^{\infty} \gamma^t \Big(r(s_t, a_t) + V^{\pi'}(s_t) - V^{\pi'}(s_t)\Big) | s_0 = s\Big] - V^{\pi'}(s) \\
&= \mathbb{E}_{\tau \sim \mathsf{p}_\pi(\tau)}\Big[\sum_{t=0}^{\infty} \gamma^t \Big(r(s_t, a_t) + \gamma V^{\pi'}(s_{t+1}) - V^{\pi'}(s_t)\Big) | s_0 = s\Big] \\
&= \mathbb{E}_{\tau \sim \mathsf{p}_\pi(\tau)}\Big[\sum_{t=0}^{\infty} \gamma^t \Big(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathsf{P}(\cdot | s_t, a_t)}[V^{\pi'}(s_{t+1})] - V^{\pi'}(s_t)\Big) | s_0 = s\Big] \\
&= \mathbb{E}_{\tau \sim \mathsf{p}_\pi(\tau)}\Big[\sum_{t=0}^{\infty} \gamma^t \Big(Q^{\pi'}(s_t, a_t) - V^{\pi'}(s_t)\Big) | s_0 = s\Big] \\
&= \mathbb{E}_{\tau \sim \mathsf{p}_\pi(\tau)}\Big[\sum_{t=0}^{\infty} \gamma^t A^{\pi'}(s_t, a_t) | s_0 = s\Big]
\end{aligned}
$$

Remark: ○ We use a telescoping trick to go from line 2 to line 3!

Proof of gradient dominance

Derivation:

$$
\begin{aligned}
J(\pi^\star) - J(\pi) &= \frac{1}{1-\gamma} \sum_s \lambda_\mu^{\pi^\star}(s) \sum_a \pi^\star(a|s) A^\pi(s,a) \\
&= \frac{1}{1-\gamma} \sum_s \frac{\lambda_\mu^{\pi^\star}(s)}{\lambda_\mu^\pi(s)} \lambda_\mu^\pi(s) \sum_a \pi^\star(a|s) A^\pi(s,a) \\
&\leq \frac{1}{1-\gamma} \left\| \frac{\lambda_\mu^{\pi^\star}}{\lambda_\mu^\pi} \right\|_\infty \times \max_{\bar{\pi}\in\Delta} \sum_{s,a} \lambda_\mu^\pi(s) \bar{\pi}(a|s) A^\pi(s,a) \\
&= \frac{1}{1-\gamma} \left\| \frac{\lambda_\mu^{\pi^\star}}{\lambda_\mu^\pi} \right\|_\infty \times \max_{\bar{\pi}\in\Delta} \sum_{s,a} \lambda_\mu^\pi(s) (\bar{\pi}(a|s) - \pi(a|s)) A^\pi(s,a) \\
&= \frac{1}{1-\gamma} \left\| \frac{\lambda_\mu^{\pi^\star}}{\lambda_\mu^\pi} \right\|_\infty \times \max_{\bar{\pi}\in\Delta} \sum_{s,a} \lambda_\mu^\pi(s) (\bar{\pi}(a|s) - \pi(a|s)) Q^\pi(s,a) \\
&= \left\| \frac{\lambda_\mu^{\pi^\star}}{\lambda_\mu^\pi} \right\|_\infty \times \max_{\bar{\pi}\in\Delta} \langle \bar{\pi} - \pi, \nabla J(\pi) \rangle.
\end{aligned}
$$

Supplementary Material

Projected policy pradient and softmax policy gradient

Projected policy gradient method

Projected policy gradient method

By projected policy gradient method, we mean the iteration variant below

$$\pi_{t+1} = \Pi_\Delta(\pi_t + \eta \nabla J(\pi_t)),$$

where the projection is given by $\Pi_\Delta(\pi) = \arg\min_{\pi' \in \Delta} \|\pi - \pi'\|_2^2$.

Remarks:

- ○ Take a gradient ascent step and project onto the simplex set (can be computed efficiently).
- ○ *Generalized gradient mapping*: $G(\pi_t) = \frac{1}{\eta}(\pi_{t+1} - \pi_t)$, or equivalently, $\pi_{t+1} = \pi_t + \eta G(\pi_t)$.
- ○ If π is optimal, then $G(\pi) = 0$.
- ○ Convergence of the gradient mapping [17]: If $J(\pi)$ is L-smooth, then we have

$$\min_{t \le T} \|G(\pi_t)\|_2^2 \le \frac{2L(J(\pi^\star) - J(\pi_0))}{T}.$$

Convergence of projected policy gradient method

Theorem (Agarwal et al., 2020 [1])

Assume access to exact gradients. Let $\eta = \frac{(1-\gamma)^3}{2\gamma|\mathcal{A}|}$. Then, the following holds

$$\min_{t<T} J(\pi^\star) - J(\pi_t) \leq \frac{8\sqrt{\gamma|\mathcal{S}||\mathcal{A}|}}{(1-\gamma)^3\sqrt{T}} \left\| \frac{\lambda_\mu^{\pi^\star}}{\mu} \right\|_\infty .$$

Proof sketch:

- ○ Show that the objective $J(\pi)$ is L-smooth with $L = \frac{2\gamma|\mathcal{A}|}{(1-\gamma)^3}$ and $J(\pi) \leq \frac{1}{1-\gamma}$.
- ○ Invoke convergence on gradient mapping: $\min_{t\leq T} \|G(\pi_t)\|_2^2 \leq \frac{2L(J(\pi^\star)-J(\pi_0))}{T}$.
- ○ Invoke the relationship between gradient mapping and approximation of stationary point [17]:

$$\max_{\bar{\pi}\in\Delta} \langle \bar{\pi} - \pi_{t+1}, \nabla J(\pi_{t+1}) \rangle \leq (1 + L\eta) \cdot \|G(\pi_t)\|_2 \cdot \|\pi_{t+1} - \pi_t\|_2 .$$

- ○ Use the gradient dominance for global convergence.

A closer look at the convergence

Theorem (Agarwal et al., 2020 [1])

Assume access to exact gradients. Let $\eta = \frac{(1-\gamma)^3}{2\gamma|A|}$. Then, the following holds

$$\min_{t<T} J(\pi^\star) - J(\pi_t) \leq \frac{8\sqrt{\gamma|\mathcal{S}||\mathcal{A}|}}{(1-\gamma)^3\sqrt{T}} \left\| \frac{\lambda_\mu^{\pi^\star}}{\mu} \right\|_\infty .$$

Remarks:

- We have large constants in the bound and a slow rate in T.
- In the tabular setting, VI or PI converges linearly, which is much faster.
- However: Linear convergence of PG can be shown with larger step-sizes through line-search [3].

A closer look at the PG method

- The projected PG update can also be viewed as

$$\begin{aligned}\pi_{t+1} &:= \Pi_\Delta(\pi_t + \eta \nabla J(\pi_t)) \\ &= \arg\max_{\pi\in\Delta} \left\{ \langle \nabla J(\pi_t), \pi \rangle - \frac{1}{2\eta}\|\pi - \pi_t\|_2^2 \right\}.\end{aligned}$$

- As $\eta \to \infty$, this reduces to the policy iteration update:

$$\pi_{t+1}(\cdot|s) = \arg\max_{\pi(\cdot|s)\in\Delta(\mathcal{A})} \sum_a \pi(s|a) Q^{\pi_t}(s,a).$$

- In other words, the policy gradient method can be viewed as an approximation of policy iteration

$$\arg\max_{\pi\in\Delta} \left\{ \langle \nabla J(\pi_t), \pi \rangle - \frac{1}{2\eta}\|\pi - \pi_t\|_2^2 \right\} = \arg\max_{\pi\in\Delta} \left\{ \langle Q^{\pi_t}, \pi \rangle_{\lambda_\mu^{\pi_t}} - \frac{1}{2\eta'}\|\pi - \pi_t\|_2^2 \right\}, \tag{6}$$

where $\frac{\partial J(\pi)}{\partial \pi(a|s)} = \frac{1}{1-\gamma}\lambda_\mu^\pi(s) Q^\pi(s,a)$ and $\langle \cdot, \cdot \rangle_{\lambda_\mu^\pi}$ is the reweighted inner product by λ_μ^π.

From gradient descent to mirror descent: Exploiting the non-Euclidean geometry

- We can adapt PG in the simplex with mirror descent updates:

$$\pi_{t+1} := \arg\max_{\pi\in\Delta} \left\{ \langle \nabla J(\pi_t), \pi \rangle - \frac{1}{\eta} \sum_s \lambda_\mu^{\pi_t}(s) \mathsf{KL}\left(\pi(\cdot|s) || \pi_t(\cdot|s)\right) \right\},$$

  where $\mathsf{KL}\left(p||q\right) = \sum_i p_i \log\left(\frac{p_i}{q_i}\right)$ is the Kullback-Leibler divergence.

- The policy mirror descent update can be further simplified as

$$\pi_{t+1}(a|s) = \pi_t(a|s) \frac{\exp(\eta Q^t(s,a)/(1-\gamma))}{\sum_{a'} \pi_t(a'|s) \exp(\eta Q^t(s,a')/(1-\gamma))}.$$

- We will see that this is the so-called *natural policy gradient* method under softmax parameterization.

- As $\eta \to \infty$, this also reduces to the policy iteration update.

Policy optimization (softmax parameterization)

○ We now consider the softmax parameterization in the tabular setting.

Policy optimization under softmax parameterization

$$\max_{\theta} J(\pi_\theta) := \mathbb{E}_{s\sim\mu}[V^{\pi_\theta}(s)], \quad \text{where } \pi_\theta(a|s) = \frac{\exp(\theta_{s,a})}{\sum_{a'} \exp(\theta_{s,a'})}.$$

Softmax policy gradient method

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta J(\pi_{\theta_t}), \quad \text{where} \quad \frac{\partial J(\theta)}{\partial \theta_{s,a}} = \frac{1}{1-\gamma} \lambda_\mu^{\pi_\theta}(s) \pi_\theta(a|s) A^{\pi_\theta}(s,a).$$

Gradient dominance and global convergence

Gradient dominance (Mei et al., 2020 [14])

$$J(\pi^\star) - J(\pi_\theta) \leq [\min_s \pi_\theta(a^\star(s)|s)]^{-1} \sqrt{S} \cdot \left\| \frac{\lambda_\mu^{\pi^\star}}{\lambda_\mu^{\pi_\theta}} \right\|_\infty \cdot \|\nabla_\theta J(\pi_\theta)\|_2.$$

Convergence of softmax policy gradient (Mei et al., 2020 [14])

Assume access to exact gradients, let $\eta \leq \frac{(1-\gamma)^3}{8}$. Then, the following holds

$$J(\pi^\star) - J(\pi_{\theta_T}) \leq \frac{16|\mathcal{S}|}{c^2(1-\gamma)^5 T} \left\| \frac{\lambda_\mu^{\pi^\star}}{\mu} \right\|_\infty^2,$$

where $c = [\min_{s,t} \pi_{\theta_t}(a^\star(s)|s)]^{-1} > 0$.

Remark: ○ Proof follows similarly as the tabular setting with slow rate and large constants in the bound.