
DEEP REINFORCEMENT LEARNING

HOMEWORK 2 | POLICY-BASED METHODS

Professor Mohammad Hossein Rohban

Sharif University of Technology
Spring 2026



Contents

1	Policy Gradients & Variance Reduction	1
2	REINFORCE	2
3	Policy-Induced Distributions and Gradients	3
3.1	Occupancy Measures	3
3.2	Discounted Visitation and the Gradient of v_π	4
3.3	Undiscounted Gradients	4
4	Natural Policy Gradient (NPG)	6
4.1	Average-Reward Policy Optimization	6
4.2	The Differential Action-Value Function	6
4.3	The Ordinary Policy Gradient	7
4.4	The Fisher Geometry of Policies	8
4.5	Compatible Function Approximation	9
4.6	Why the Compatible Approximation is Advantage-Like	10
4.7	Natural Gradient and Greedy Policy Improvement	10
4.8	Local Policy Improvement for General Policies	11
4.9	Why This is Not Newton's Method	11
4.10	Natural Gradient as a Small Policy-Distribution Step	11
4.11	MiniTetris	12

Important Notes

- **Grading Policy:** This homework is considered complete once you reach a total of **100 points**. Up to *5 additional bonus points* may be earned; however, any score above 100 will still be recorded as 100. Bonus points do *not* transfer or overflow to other assignments.
- **Homework Deadline:** The assignment is due on the posted deadline. Please make sure to submit your work on time.
- **Delay Usage Policy:** You may use *at most 3 days* of your allowed late-day budget for this homework. Late days will be automatically applied if you submit after the deadline.
- **Solution Release:** Official solutions will be released exactly **3 days after the deadline**. No submissions will be accepted once the solutions are published.
- **Cheating and Collaboration Policy:** You must write all solutions in your own words and produce all code independently. Discussing general ideas is allowed, but sharing written solutions, code, or using someone else's work (including from online sources, previous semesters, or AI tools without proper authorization) is strictly prohibited. Violations will result in academic misconduct procedures.

Advice

Please avoid asking LLMs for the answer. Sure, doing so might give you an immediate reward of 5 points, but in the long run, it won't help you become the next Rockstar of DRL. Remember the key lesson of this course: chasing immediate rewards is rarely the best strategy. Instead, struggle with the problems, read the papers, and truly understand the algorithms. The satisfaction of solving a challenging problem yourself is far greater than any grade. Good luck! [1]

Grading Breakdown

Task / Criteria	Points
Policy Gradients & Variance Reduction	10 pts
The Score Function Estimator	3 pts
The Model-Free Property	2 pts
Zero-Bias Baseline	2 pts
The Optimal Baseline	3 pts
REINFORCE	28 pts
REINFORCE Implementation Variants	25 pts
Notebook Questions	3 pts
Policy-Induced Distributions and Gradients	25 pts
Occupancy Measures	9 pts
Discounted Visitation and the Gradient of v_π	5 pts
Undiscounted Gradients	11 pts
Natural Policy Gradient	47 pts
The Differential Action-Value Function	6 pts
Fisher Geometry of Policies	4 pts
Compatible Function Approximation	5 pts
Policy Improvement	12 pts
KL Interpretation and Conceptual Comparisons	4 pts
MiniTetris Implementation	16 pts
Bonus	+5 pts
Clarity and Quality of Code	+2 pts
Clarity and Quality of Report	+3 pts
Total	110+5 pts

1 Policy Gradients & Variance Reduction

In value-based methods, we learned a value function and derived a deterministic policy by taking the argmax over actions. In policy-based methods, we directly parameterize the policy $\pi_\theta(a|s)$ and optimize the parameters θ to maximize the expected return $J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$. Let $\tau = (s_0, a_0, s_1, a_1, \dots, s_T, a_T)$ denote a trajectory. Assume the probability of a trajectory is given by

$$P(\tau|\theta) = P(s_0) \prod_{t=0}^T \pi_\theta(a_t|s_t) P(s_{t+1}|s_t, a_t)$$

Question 1: The Score Function Estimator

[3 Points]

To optimize $J(\theta)$, we need its gradient $\nabla_\theta J(\theta)$. However, we cannot simply push the gradient inside the expectation because the expectation itself depends on the parameters θ . Show that the gradient of the objective function can be rewritten as an expectation over trajectories:

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

Question 2: The Model-Free Property

[2 Points]

In most reinforcement learning scenarios, the environment's transition dynamics $P(s_{t+1}|s_t, a_t)$ are completely unknown to the agent. Prove why we can still compute the policy gradient $\nabla_\theta \log P(\tau|\theta)$ from sample trajectories without needing to model or know the environment's transition dynamics. Show exactly what happens to the $P(s_{t+1}|s_t, a_t)$ term during the derivation.

Question 3: Zero-Bias Baseline

[2 Points]

Policy gradients suffer from high variance. To reduce this variance, we subtract a state-dependent function $b(s_t)$, known as a *baseline* [4], from our return. Prove that subtracting an arbitrary baseline $b(s_t)$ that depends only on the current state does not introduce any bias into the policy gradient estimate.

While any valid baseline $b(s_t)$ leaves the gradient unbiased, some baselines reduce variance better than others. Often, the value function $V^\pi(s_t)$ is used. However, it is not strictly the variance-minimizing baseline.

Question 4: The Optimal Baseline

[3 Points]

Derive the exact, optimal baseline $b^*(s_t)$ that strictly minimizes the variance of the policy gradient estimate.

2 REINFORCE

In this section, we study the Monte Carlo policy-gradient method known as **REINFORCE**.

The goal is to train a policy network that maximizes the expected discounted return. The core difficulty is that Monte Carlo policy-gradient estimates are unbiased but often have high variance. Therefore, this assignment compares four related estimators:

Implementation 1: REINFORCE Implementation

[25 Points]

1. REINFORCE without the causality trick, using the same full-episode return for every action.
2. REINFORCE with the causality trick, using reward-to-go.
3. REINFORCE with reward-to-go and a constant baseline.
4. REINFORCE with reward-to-go and a learned value-function baseline.

Question 1:

[3 Points]

Answer the questions inside the Jupyter notebook based on your experiments.

Algorithm 1: Vanilla REINFORCE

- 1: **Input:** policy π_θ , learning rate α , discount factor γ , number of episodes N
- 2: **for** episode = 1, ..., N **do**
- 3: Roll out one trajectory using $a_t \sim \pi_\theta(\cdot \mid s_t)$
- 4: Store rewards R_t and log-probabilities $\log \pi_\theta(a_t \mid s_t)$
- 5: Compute the full discounted return $G_0 = \sum_{k=0}^T \gamma^k R_k$
- 6: Form the loss $L(\theta) = - \sum_{t=0}^T \log \pi_\theta(a_t \mid s_t) G_0$
- 7: Update θ by gradient descent on $L(\theta)$
- 8: **end for**

3 Policy-Induced Distributions and Gradients

A central theme in policy-based reinforcement learning is that a policy can be studied through the distributions it induces over states and actions. In finite-horizon problems, this leads to occupancy measures and flow-conservation constraints. In discounted infinite-horizon problems, similar visitation quantities appear in the gradient of the value function. Finally, in the undiscounted average-reward setting, the usual value function must be replaced by the differential value, which satisfies the Poisson equation. Together, these problems show how policy-induced visitation structure connects occupancy measures, value gradients, and average-reward analysis.

3.1 Occupancy Measures

Consider a finite-horizon Markov Decision Process (MDP) defined by the tuple (S, A, P, R, γ) with horizon H . The state space S is partitioned into H layers: $S = S_1 \cup S_2 \cup \dots \cup S_H$, where $S_1 = \{s_{\text{init}}\}$ is the initial state. For a stochastic policy $\pi : S \rightarrow \Delta(A)$ (where $\Delta(A)$ is the probability simplex over actions), the *occupancy measure* $q_\pi : S \times A \rightarrow [0, 1]$ is defined as the probability that the agent visits state-action pair (s, a) when starting at s_{init} and following π for H steps.

Question 1: Markov Chain as a Special Case

[2 Points]

Suppose $|A| = 1$ (only one action).

- Explain how the MDP reduces to a Markov Chain (MC).
- In this MC, what does the occupancy measure q_π represent?
- If the MC were ergodic and we considered an infinite horizon ($H \rightarrow \infty$), how would the occupancy measure relate to the stationary distribution of the MC?

Question 2: Properties of the Occupancy Measure

[2 Points]

Show that for any policy π , the induced occupancy measure q_π satisfies the following linear constraints:

$$\sum_{a \in A} q_\pi(s_{\text{init}}, a) = 1, \quad (1)$$

for each $h = 1, \dots, H - 1$ and each $s' \in S_{h+1}$:

$$\sum_{s \in S_h} \sum_{a \in A} q_\pi(s, a) P(s' | s, a) = \sum_{a \in A} q_\pi(s', a). \quad (2)$$

(Interpretation: Equation (2) is a *flow conservation* condition: the probability of entering state s' equals the probability of being in s' .)

Question 3: From Constraints to Occupancy Measure

[4 Points]

Now suppose we have a function $q : S \times A \rightarrow [0, 1]$ that satisfies constraints (1) and (2). Define a policy π by:

$$\pi(a | s) = \frac{q(s, a)}{\sum_{a' \in A} q(s, a')} \quad \text{if } \sum_{a'} q(s, a') > 0,$$

and let $\pi(\cdot | s)$ be an arbitrary distribution over A otherwise. Prove that q is the occupancy measure induced by π , i.e., $q = q_\pi$.

Question 4: Utility of the Occupancy Measure Formulation

[1 Points]

- Express the expected total loss $v_\pi(s_{\text{init}})$ in terms of q_π and γ .
- Discuss one advantage and one disadvantage of optimizing over the space of occupancy measures (subject to constraints (1) and (2)) compared to optimizing directly over the policy space.

Hint for advantage: Consider the structure of the objective and constraints.

Hint for disadvantage: Think about the dimensionality and the relationship between q and π .

3.2 Discounted Visitation and the Gradient of v_π

In a discounted MDP, it holds for any $s \in \mathcal{S}$ that

$$\nabla_\theta v_\pi(s) = \sum_{s' \in \mathcal{S}} \Pr_\pi(s' | s) \sum_{a \in \mathcal{A}} \nabla_\theta \pi(a | s', \theta) q_\pi(s', a), \quad (3)$$

where

$$\Pr_\pi(s' | s) := [(I_n - \gamma P_\pi)^{-1}]_{ss'} \quad (4)$$

is the discounted total probability of transitioning from s to s' under policy π . Here, $[\cdot]_{ss'}$ denotes the entry in the s -th row and s' -th column.

Question 5: The Gradient of v_π

[5 Points]

Prove Equation 3 holds for all $s \in \mathcal{S}$ in a discounted MDP.

3.3 Undiscounted Gradients

In the undiscounted case, it is necessary to redefine state and action values. Since the undiscounted sum of the rewards, $\mathbb{E}[R_{t+1} + R_{t+2} + \dots | S_t = s]$, may diverge, the state and action values are defined in a special way:

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}[(R_{t+1} - \bar{r}_\pi) + (R_{t+2} - \bar{r}_\pi) + \dots | S_t = s], \\ q_\pi(s, a) &\doteq \mathbb{E}[(R_{t+1} - \bar{r}_\pi) + (R_{t+2} - \bar{r}_\pi) + \dots | S_t = s, A_t = a], \end{aligned}$$

where $\bar{r}_\pi$ is the average reward, which is determined when π is given. There are different names for $v_\pi(s)$ in the literature such as the differential reward or bias. Note that in this part $q_\pi(s, a)$ denotes the action-value function, following standard reinforcement learning notation. It can be verified that the state value defined above satisfies the following Bellman-like equation:

$$v_\pi(s) = \sum_a \pi(a | s, \theta) \left[\sum_r p(r | s, a) (r - \bar{r}_\pi) + \sum_{s'} p(s' | s, a) v_\pi(s') \right]. \quad (5)$$

Since $v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a | s, \theta) q_\pi(s, a)$, it holds that $q_\pi(s, a) = \sum_r p(r | s, a)(r - \bar{r}_\pi) + \sum_{s'} p(s' | s, a)v_\pi(s')$. The matrix-vector form of Equation 5 is

$$v_\pi = r_\pi - \bar{r}_\pi \mathbf{1}_n + P_\pi v_\pi, \quad (6)$$

where $\mathbf{1}_n = [1, \dots, 1]^\top \in \mathbb{R}^n$. Equation 6 is similar to the Bellman equation and it has a specific name called the **Poisson equation**.

Question 6: Solution of the Poisson Equation

[5 Points]

Show that

$$v_\pi^* = (I_n - P_\pi + \mathbf{1}_n d_\pi^\top)^{-1} r_\pi. \quad (7)$$

is a solution of the Poisson equation. Here, the distribution d_π is selected such that it is stationary distribution under π , satisfying

$$d_\pi^\top P_\pi = d_\pi^\top,$$

where P_π is the state transition probability matrix.

Question 7: Solution of the Poisson Equation

[6 Points]

Show that any solution of the Poisson equation has the following form:

$$v^\pi = v_\pi^* + c \mathbf{1}_n, \quad (8)$$

where $c \in \mathbb{R}$.

4 Natural Policy Gradient (NPG)

The usual policy gradient tells us how to improve the objective by changing the policy parameters. But the parameters themselves are not the real object of interest. The real object is the policy distribution. A policy parameter vector is only a coordinate system for a family of probability distributions.

This distinction matters. If we change coordinates, the same policy family may look different in parameter space, and an ordinary Euclidean gradient step may behave differently. The natural policy gradient corrects this by measuring the size of a policy update using the geometry of probability distributions.

By the end of the section, you should see why the natural policy gradient can be interpreted as a direction that moves the policy toward the action choices suggested by a greedy policy-improvement step.

4.1 Average-Reward Policy Optimization

We consider a finite Markov decision process with state space S , action space A , reward function R , and transition dynamics P .

The agent uses a stochastic policy $\pi_\theta(a|s)$, where $\theta \in \mathbb{R}^m$ parameterizes the policy. Unlike the discounted setting, here we consider a continuing task. The performance objective is not

$$\mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right].$$

Instead, the objective is the long-run average reward per time step. Intuitively, if a policy is run for a very long time, we ask: “How much reward does it collect per step on average?”

Under an ergodicity assumption (stable long-run distribution), each policy π induces a stationary state distribution $\rho^\pi(s)$. The average-reward objective is

$$\eta(\pi) = \sum_{s \in S} \rho^\pi(s) \sum_{a \in A} \pi(a|s) R(s, a).$$

Equivalently, one may view it as the infinite-horizon time average

$$\eta(\pi) = \lim_{T \rightarrow \infty} \frac{1}{T} \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} R(s_t, a_t) \right].$$

The optimization problem is therefore

$$\max_{\theta \in \mathbb{R}^m} \eta(\theta).$$

4.2 The Differential Action-Value Function

In the discounted setting, the action-value function measures the expected discounted future reward after taking action a in state s .

In the average-reward setting, we instead measure future reward relative to the policy’s long-run average reward. This gives the differential action-value function:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} (R(s_t, a_t) - \eta(\pi)) \mid s_0 = s, a_0 = a \right].$$

This quantity tells if we start from (s, a) , how much better or worse the future is compared to the average reward rate.

For the next task, assume that S and A are finite, rewards are bounded, and under the fixed policy π , the induced Markov chain is ergodic.

Question 1: Finiteness of the Differential Action-Value Function

[4 Points]

Prove that, under these assumptions,

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} (R(s_t, a_t) - \eta(\pi)) \mid s_0 = s, a_0 = a \right]$$

is finite for every state-action pair (s, a) .

Question 2: Advantage-Like Interpretation

[2 Points]

The corresponding state-value function is

$$J^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} [Q^\pi(s, a)] .$$

Show that replacing $Q^\pi(s, a)$ by

$$A^\pi(s, a) = Q^\pi(s, a) - J^\pi(s)$$

does not change the policy gradient.

4.3 The Ordinary Policy Gradient

The average-reward policy-gradient theorem gives

$$\nabla \eta(\theta) = \sum_{s,a} \rho^\pi(s) \nabla_\theta \pi_\theta(a|s) Q^\pi(s, a) .$$

Using the identity

$$\nabla_\theta \pi_\theta(a|s) = \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) ,$$

we can also write this as

$$\nabla \eta(\theta) = \mathbb{E}_{s \sim \rho^\pi, a \sim \pi_\theta(\cdot|s)} [\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s, a)] .$$

A standard policy-gradient update is

$$\theta \leftarrow \theta + \alpha \nabla \eta(\theta) .$$

This update treats θ -space as Euclidean. In other words, it implicitly measures the size of a parameter step $\Delta\theta$ using

$$\|\Delta\theta\|^2 = \Delta\theta^\top \Delta\theta .$$

The problem is that θ is only a coordinate system. The policy itself is a distribution. Two different parameterizations can represent the same family of policies, but ordinary gradient ascent may behave differently under these two parameterizations.

So the question is:

“What is the steepest direction if we measure distance between policies rather than raw parameters?”

4.4 The Fisher Geometry of Policies

A parameter vector θ is not important by itself. What matters is the policy distribution $\pi_\theta(\cdot|s)$. If we perturb the parameters from θ to $\theta + \Delta\theta$, the meaningful question is not simply How large $\Delta\theta$ is, but rather How much the action distribution change.

The Fisher matrix answers this local question. It tells us how sensitive the policy distribution is to small changes in the parameters.

Some directions in parameter space may look large in Euclidean norm but barely change the policy. Other directions may look small but drastically change the action probabilities. The Fisher matrix rescales directions according to their actual effect on the distribution.

For a small perturbation $\Delta\theta$, the quantity

$$\Delta\theta^\top F_s(\theta) \Delta\theta$$

can be interpreted as the local squared distance, at state s , between the old policy distribution $\pi_\theta(\cdot|s)$ and the perturbed policy distribution $\pi_{\theta+\Delta\theta}(\cdot|s)$.

So the Fisher matrix is not measuring distance between parameter vectors directly. It is measuring the local change in the probability distribution represented by those parameters.

The Fisher information matrix for $\pi_\theta(\cdot|s)$ is

$$F_s(\theta) = \mathbb{E}_{a \sim \pi_\theta(\cdot|s)} [\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^\top].$$

Since the policy is used across states, the natural metric for the policy class averages this statewise Fisher information over the stationary distribution:

$$F(\theta) = \mathbb{E}_{s \sim \rho^\pi} [F_s(\theta)].$$

The averaging over $\rho^\pi(s)$ means that changes to the policy are weighted according to how often states are visited under the current policy.

Equivalently,

$$F(\theta) = \sum_{s,a} \rho^\pi(s) \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^\top.$$

This matrix measures how much a small parameter perturbation changes the policy distribution.

The steepest ascent direction under this metric is

$$\tilde{\nabla} \eta(\theta) = F(\theta)^{-1} \nabla \eta(\theta).$$

This is the natural policy gradient.

This optimization problem is the local mathematical version of asking for the steepest ascent direction.

For a small parameter update $\Delta\theta$, we can approximate the change in the objective by the first-order Taylor expansion

$$\eta(\theta + \Delta\theta) \approx \eta(\theta) + \nabla \eta(\theta)^\top \Delta\theta.$$

Since $\eta(\theta)$ is fixed at the current parameter value, maximizing local improvement is equivalent to maximizing

$$\nabla \eta(\theta)^\top \Delta \theta.$$

However, without any constraint, this linear expression can be made arbitrarily large just by scaling $\Delta \theta$. Therefore, to compare directions fairly, we fix the size of the update.

If we were using the ordinary Euclidean geometry of parameter space, we would constrain

$$\Delta \theta^\top \Delta \theta = \epsilon.$$

But the natural-gradient viewpoint says that the size of an update should be measured by how much it changes the policy distribution. Locally, this size is measured by the Fisher metric:

$$\Delta \theta^\top F(\theta) \Delta \theta.$$

Question 3: Natural Gradient as Steepest Ascent

[4 Points]

Derive the natural-gradient direction by solving the local constrained optimization problem

$$\max_{\Delta \theta} \quad \nabla \eta(\theta)^\top \Delta \theta$$

subject to

$$\Delta \theta^\top F(\theta) \Delta \theta = \epsilon.$$

4.5 Compatible Function Approximation

Now we connect the natural gradient to a special linear approximation of Q^π .

Define the score feature

$$\psi^\pi(s, a) = \nabla_\theta \log \pi_\theta(a|s).$$

A compatible function approximator has the form

$$f^\pi(s, a; w) = w^\top \psi^\pi(s, a).$$

It is called compatible because it uses exactly the same score features that appear in the policy-gradient expression.

We fit w by minimizing the squared error

$$\epsilon(w, \pi) = \sum_{s,a} \rho^\pi(s) \pi_\theta(a|s) (w^\top \psi^\pi(s, a) - Q^\pi(s, a))^2.$$

The important fact is that the best compatible function approximator does not merely approximate Q^π . Its weight vector is exactly the natural-gradient direction.

Question 4: Compatible Function Approximation and Natural Gradient**[3 Points]**

Assume w minimizes

$$\epsilon(w, \pi) = \sum_{s,a} \rho^\pi(s) \pi_\theta(a|s) (w^\top \psi^\pi(s, a) - Q^\pi(s, a))^2.$$

Show that

$$w = F(\theta)^{-1} \nabla \eta(\theta).$$

4.6 Why the Compatible Approximation is Advantage-Like

The compatible function approximator has a special property: it cannot represent arbitrary state-dependent offsets. It represents the action-dependent part of the value function. In this sense, it is naturally advantage-like. The arguments below support this fact.

Question 5: Zero-Mean Property**[2 Points]**

Prove that

$$\mathbb{E}_{a \sim \pi_\theta(\cdot|s)} [f^\pi(s, a; w)] = 0.$$

4.7 Natural Gradient and Greedy Policy Improvement

A greedy policy-improvement step using the compatible approximation would choose actions according to

$$a \in \arg \max_{a'} f^\pi(s, a'; w).$$

The surprising connection is that the natural gradient points toward this greedy improvement step.

To see this most cleanly, consider an exponential-family policy:

$$\pi_\theta(a|s) = \frac{\exp(\theta^\top \phi_{s,a})}{\sum_b \exp(\theta^\top \phi_{s,b})}.$$

Question 6: Computing the Score Feature**[4 Points]**

Considering the exponential-family policy, show that the score feature is

$$\psi^\pi(s, a) = \phi_{s,a} - \mathbb{E}_{b \sim \pi_\theta(\cdot|s)} [\phi_{s,b}].$$

In the next question we show that for a very large natural-gradient step size, the dominant term is exactly the one that ranks actions by the compatible approximation.

In other words, the natural gradient moves toward the greedy action choices under the compatible function approximator.

Question 7: Large-Step Greedy Limit**[4 Points]**

Define

$$\pi_\infty(a|s) = \lim_{\alpha \rightarrow \infty} \pi_{\theta + \alpha \tilde{\nabla} \eta(\theta)}(a|s).$$

Show that

$$\pi_\infty(a|s) \neq 0$$

if and only if

$$a \in \arg \max_{a'} f^\pi(s, a'; w).$$

4.8 Local Policy Improvement for General Policies

The previous section used exponential-family policies and a large step size. For a general smooth policy class, we can still obtain a local statement.

Let

$$\theta' = \theta + \alpha \tilde{\nabla} \eta(\theta).$$

For a general smooth policy, the infinite-step argument no longer applies. However, for small α , the natural-gradient update still has a local policy-improvement interpretation: it changes action probabilities according to the compatible approximation $f^\pi(s, a; w)$.

Question 8: Local Natural-Gradient Update**[4 Points]**

For a general smooth policy, prove the local first-order relationship between $\pi_{\theta'}(a|s)$ and $\pi_\theta(a|s)$ under the update $\theta' = \theta + \alpha \tilde{\nabla} \eta(\theta)$. Then, show that for sufficiently small α , to first order, actions with larger values of $f^\pi(s, a; w)$ receive larger multiplicative increases in probability.

4.9 Why This is Not Newton's Method

The Fisher matrix is not generally the Hessian of the reinforcement-learning objective.

The Fisher matrix is

$$F(\theta) = \mathbb{E}_{s,a} [\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^\top].$$

It measures curvature of the policy distribution.

Question 9: Conceptual Comparison**[2 Points]**

Explain why the natural gradient is not the same as Newton's method. What does the Fisher matrix measure, and what additional quantities appear in the Hessian of the RL objective (write the additional terms)?

4.10 Natural Gradient as a Small Policy-Distribution Step

The previous sections showed that the Fisher matrix defines a geometry on policy space. We now make this more precise using KL divergence.

A parameter update can be small in Euclidean norm but still cause a large change in the policy distribution. Conversely, a parameter update can look large in Euclidean norm but barely change the action probabilities. Therefore, the natural question is not only “how large is $\Delta\theta$?”, but rather “How much does the policy distribution change?”

For a fixed state s , consider the KL divergence between the old policy and the new policy:

$$D_{\text{KL}}(\pi_{\theta}(\cdot|s) \parallel \pi_{\theta+\Delta\theta}(\cdot|s)).$$

Averaging over states gives the local policy-distribution change $\bar{D}_{\text{KL}}(\theta, \theta + \Delta\theta)$.

Question 10: KL Divergence and the Fisher Matrix

[2 Points]

Write the Taylor expansion of $\bar{D}_{\text{KL}}(\theta, \theta + \Delta\theta)$, around $\Delta\theta = 0$ up to second order. Then explain why the natural-gradient direction is the steepest ascent direction under a local KL constraint.

Thus the Fisher matrix is the local quadratic approximation to KL divergence between nearby policies.

This explains the difference between ordinary gradient ascent and natural gradient ascent. Ordinary gradient ascent chooses the direction that gives the largest first-order improvement for a fixed Euclidean parameter step:

$$\Delta\theta^{\top} \Delta\theta = \epsilon.$$

Natural gradient ascent instead chooses the direction that gives the largest first-order improvement for a fixed local policy-distribution step:

$$\frac{1}{2} \Delta\theta^{\top} F(\theta) \Delta\theta = \epsilon.$$

So the natural gradient is trying to keep the actual change in the policy distribution small.

Ordinary PG controls parameter movement.

Natural PG controls policy-distribution movement.

4.11 MiniTetris

The previous subsections developed the natural policy gradient from a theoretical point of view. To make these ideas more concrete, you will now implement the accompanying notebook. The goal of the notebook is not only to train a policy, but also to visualize the geometry behind natural policy gradient.

Implementation 2: PG and NPG in MiniTetris

[15 Points]

The notebook implements a small Tetris-like control problem in which the policy is a softmax over afterstate features:

$$\pi_{\theta}(a|s) = \frac{\exp(\theta^{\top} \phi_{s,a})}{\sum_b \exp(\theta^{\top} \phi_{s,b})}.$$

Your implementation should help you understand the following ideas more clearly:

- how the score feature $\nabla_{\theta} \log \pi_{\theta}(a|s)$ is computed for a softmax policy;
- how the policy-gradient vector g and the Fisher matrix F can be estimated from rollouts;
- how the natural-gradient direction $d_{\text{NPG}} = (F + \lambda I)^{-1} g$ differs from the ordinary policy-gradient

direction $d_{\text{PG}} = g$;

- why comparing PG and NPG using the same Euclidean step size is different from comparing them using the same local KL step size;
- how the compatible function approximation explains the greedy-action behavior of large natural-gradient updates.

Question 11: PG vs. NPG**[1 Points]**

In the notebook we compare ordinary policy gradient and natural policy gradient under both fixed-learning-rate updates and KL-scaled updates. Explain whether the observed behavior agrees with the theoretical interpretation developed in this section.

References

- [1] Based on INF8250AE – Introduction to Reinforcement Learning. Fall 2025.
- [2] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, 2nd Edition, MIT Press, 2018.
- [3] Williams, R. J. (1992). *Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning*. Machine Learning, 8, 229–256.
- [4] Greensmith, E., Bartlett, P. L., & Baxter, J. (2004). *Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning*. Journal of Machine Learning Research (JMLR).