Trust Region Policy Optimization & The Natural Policy Gradient

Lucas Janson

CS/Stat 184(0): Introduction to Reinforcement Learning
Fall 2024

Today

- Feedback from last lecture
- Recap
- The Performance Difference Lemma
- Trust Region Policy Optimization (TRPO)
- The Natural Policy Gradient (NPG)

Feedback from feedback forms

Feedback from feedback forms

1. Thank you to everyone who filled out the forms!

Feedback from feedback forms

1. Thank you to everyone who filled out the forms!
2. Discuss projects!

Today

- Feedback from last lecture
- Recap
- The Performance Difference Lemma
- Trust Region Policy Optimization (TRPO)
- The Natural Policy Gradient (NPG)

Optimization Objective

- Consider a parameterized class of policies:
  $$\{\pi_\theta(a \mid s) \mid \theta \in \mathbb{R}^d\}$$
  (why do we make it stochastic?)

- Objective $\max_\theta J(\theta)$, where
  $$J(\theta) := \mathbb{E}_{s_0 \sim \mu}\left[V^{\pi_\theta}(s_0)\right] = \mathbb{E}_{\tau \sim \rho_{\pi_\theta}}\left[\sum_{h=0}^{H-1} r(s_h, a_h)\right]$$

- Policy Gradient Descent:
  $$\theta^{k+1} = \theta^k + \eta \nabla J(\theta^k)$$

REINFORCE: A Policy Gradient Algorithm

- Let $\rho_\theta(\tau)$ be the probability of a trajectory $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}\}$, i.e.
$$\rho_\theta(\tau) = \mu(s_0)\pi_\theta(a_0 \mid s_0)P(s_1 \mid s_0, a_0)\ldots P(s_{H-1} \mid s_{H-2}, a_{H-2})\pi_\theta(a_{H-1} \mid s_{H-1})$$

- Let $R(\tau)$ be the cumulative reward on trajectory τ, i.e. $R(\tau) := \sum_{h=0}^{H-1} r(s_h, a_h)$

- Our objective function is:
$$J(\theta) \;= E_{\tau\sim\rho_\theta}\Big[R(\tau)\Big]$$

- From the likelihood ratio method, we have:
$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\rho_\theta}\left[\nabla_\theta \ln \rho_\theta(\tau)\; R(\tau)\right]$$

- The REINFORCE Policy Gradient expression:
$$\nabla_\theta \ln \rho_\theta(\tau)\; R(\tau) = \left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\right) R(\tau)$$

Obtaining an Unbiased Gradient Estimate at θ

$$\nabla_\theta J(\theta) := \mathbb{E}_{\tau \sim \rho_\theta}\left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\right) R(\tau)\right]$$

Obtaining an Unbiased Gradient Estimate at θ

$$\nabla_\theta J(\theta) := \mathbb{E}_{\tau \sim \rho_\theta}\left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\right) R(\tau)\right]$$

1. Obtain a trajectory $\tau \sim \rho_\theta$
   (which we can do in our learning setting)

Obtaining an Unbiased Gradient Estimate at θ

$$\nabla_\theta J(\theta) := \mathbb{E}_{\tau \sim \rho_\theta}\left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\right) R(\tau)\right]$$

1. Obtain a trajectory $\tau \sim \rho_\theta$
   (which we can do in our learning setting)
2. Set:

$$g(\theta, \tau) := \left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\right) R(\tau)$$

Obtaining an Unbiased Gradient Estimate at θ

$$\nabla_\theta J(\theta) := \mathbb{E}_{\tau \sim \rho_\theta}\left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\right) R(\tau)\right]$$

1. Obtain a trajectory $\tau \sim \rho_\theta$
   (which we can do in our learning setting)
2. Set:

$$g(\theta, \tau) := \left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\right) R(\tau)$$

We have: $\mathbb{E}[g(\theta, \tau)] = \nabla_\theta J(\theta)$

PG with REINFORCE:

PG with REINFORCE:

1. Initialize θ^0, step size parameters: $\eta^1, \eta^2, \ldots$

PG with REINFORCE:

1. Initialize θ^0, step size parameters: $\eta^1, \eta^2, \ldots$
2. For $k = 0, \ldots$:

PG with REINFORCE:

1. Initialize θ^0, step size parameters: $\eta^1, \eta^2, \ldots$
2. For $k = 0, \ldots$:
    1. Obtain a trajectory $\tau \sim \rho_{\theta^k}$
       Compute $g(\theta^k, \tau)$

PG with REINFORCE:

1. Initialize θ^0, step size parameters: $\eta^1, \eta^2, \ldots$
2. For $k = 0, \ldots$:
    1. Obtain a trajectory $\tau \sim \rho_{\theta^k}$

       Compute $g(\theta^k, \tau)$

    2. Update: $\theta^{k+1} = \theta^k + \eta^k g(\theta^k, \tau)$

Other PG formulas
(that are lower variance for sampling)

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta}\left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\right) R(\tau)\right]$$ (REINFORCE)

Other PG formulas
(that are lower variance for sampling)

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta} \left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) \right) R(\tau) \right]$$

(REINFORCE)

$$= \mathbb{E}_{\tau \sim \rho_\theta} \left[\sum_{h=0}^{H-1} \left(\nabla_\theta \ln \pi_\theta(a_h \mid s_h) \sum_{t=h}^{H-1} r(s_t, a_t) \right) \right]$$

Other PG formulas
(that are lower variance for sampling)

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta}\left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\right) R(\tau)\right]$$

(REINFORCE)

$$= \mathbb{E}_{\tau \sim \rho_\theta}\left[\sum_{h=0}^{H-1}\left(\nabla_\theta \ln \pi_\theta(a_h \mid s_h) \sum_{t=h}^{H-1} r(s_t, a_t)\right)\right]$$

$$= \mathbb{E}_{\tau \sim \rho_\theta}\left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) Q_h^{\pi_\theta}(s_h, a_h)\right]$$

Other PG formulas
(that are lower variance for sampling)

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\rho_\theta}\left[\left(\sum_{h=0}^{H-1}\nabla_\theta \ln \pi_\theta(a_h \mid s_h)\right)R(\tau)\right]$$ (REINFORCE)

$$= \mathbb{E}_{\tau\sim\rho_\theta}\left[\sum_{h=0}^{H-1}\left(\nabla_\theta \ln \pi_\theta(a_h \mid s_h)\sum_{t=h}^{H-1} r(s_t, a_t)\right)\right]$$

$$= \mathbb{E}_{\tau\sim\rho_\theta}\left[\sum_{h=0}^{H-1}\nabla_\theta \ln \pi_\theta(a_h \mid s_h) Q_h^{\pi_\theta}(s_h, a_h)\right]$$

Intuition: Changing the action distribution at h only affects rewards later on…

HW: You will show these simplified version are also valid PG expressions

With a "baseline" function:

For any function only of the state, $b_h : S \to \mathbb{R}$, we have:

This is (basically) the method of control variates.

- For the proof, it was helpful to note:
  $\mathbb{E}_{x \sim P_\theta}\left[\nabla_\theta \log P_\theta(x)\ c\right] = 0$

With a "baseline" function:

For any function only of the state, $b_h : S \to \mathbb{R}$, we have:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta}\left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\left(R_h(\tau) - b_h(s_h)\right)\right]$$

$$= \mathbb{E}_{\tau \sim \rho_\theta}\left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\left(Q_h^{\pi_\theta}(s_h, a_h) - b_h(s_h)\right)\right]$$

This is (basically) the method of control variates.

- For the proof, it was helpful to note:
  $\mathbb{E}_{x \sim P_\theta}\left[\nabla_\theta \log P_\theta(x)\ c\right] = 0$

The Advantage Function (finite horizon)

$$V_h^\pi(s) = \mathbb{E}\left[\sum_{t=h}^{H-1} r(s_t, a_t) \,\middle|\, s_h = s\right] \qquad Q_h^\pi(s,a) = \mathbb{E}\left[\sum_{t=h}^{H-1} r(s_t, a_t) \,\middle|\, (s_h, a_h) = (s,a)\right]$$

The Advantage Function (finite horizon)

$$V_h^\pi(s) = \mathbb{E}\left[\sum_{t=h}^{H-1} r(s_t, a_t) \,\middle|\, s_h = s\right] \qquad Q_h^\pi(s, a) = \mathbb{E}\left[\sum_{t=h}^{H-1} r(s_t, a_t) \,\middle|\, (s_h, a_h) = (s, a)\right]$$

- The Advantage function is defined as:

$$A_h^\pi(s, a) = Q_h^\pi(s, a) - V_h^\pi(s)$$

The Advantage Function (finite horizon)

$$V_h^\pi(s) = \mathbb{E}\left[\sum_{t=h}^{H-1} r(s_t, a_t) \,\middle|\, s_h = s\right] \qquad Q_h^\pi(s,a) = \mathbb{E}\left[\sum_{t=h}^{H-1} r(s_t, a_t) \,\middle|\, (s_h, a_h) = (s,a)\right]$$

- The Advantage function is defined as:

$$A_h^\pi(s,a) = Q_h^\pi(s,a) - V_h^\pi(s)$$

- We have that:

$$\mathbb{E}_{a\sim\pi(\cdot|s)}\left[A_h^\pi(s,a) \,\middle|\, s, h\right] = \sum_a \pi(a\,|\,s) A_h^\pi(s,a) = 0$$

The Advantage Function (finite horizon)

$$V_h^\pi(s) = \mathbb{E}\left[\sum_{t=h}^{H-1} r(s_t, a_t) \,\middle|\, s_h = s\right] \qquad Q_h^\pi(s,a) = \mathbb{E}\left[\sum_{t=h}^{H-1} r(s_t, a_t) \,\middle|\, (s_h, a_h) = (s, a)\right]$$

- The Advantage function is defined as:

$$A_h^\pi(s,a) = Q_h^\pi(s,a) - V_h^\pi(s)$$

- We have that:

$$\mathbb{E}_{a\sim\pi(\cdot|s)}\left[A_h^\pi(s,a) \,\middle|\, s, h\right] = \sum_a \pi(a\,|\,s) A_h^\pi(s,a) = 0$$

- We know $A_h^{\pi^\star}(s,a) \leq 0 \quad \forall s, a$

The Advantage Function (finite horizon)

$$V_h^\pi(s) = \mathbb{E}\left[\sum_{t=h}^{H-1} r(s_t, a_t) \,\middle|\, s_h = s\right] \qquad Q_h^\pi(s,a) = \mathbb{E}\left[\sum_{t=h}^{H-1} r(s_t, a_t) \,\middle|\, (s_h, a_h) = (s,a)\right]$$

- The Advantage function is defined as:

$$A_h^\pi(s,a) = Q_h^\pi(s,a) - V_h^\pi(s)$$

- We have that:

$$\mathbb{E}_{a\sim\pi(\cdot|s)}\left[A_h^\pi(s,a) \,\middle|\, s, h\right] = \sum_a \pi(a\,|\,s) A_h^\pi(s,a) = 0$$

- We know $A_h^{\pi^\star}(s,a) \le 0 \quad \forall s, a$

- For the discounted case, $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$

The Advantage-based PG:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) \left(Q_h^{\pi_\theta}(s_h, a_h) - b_h(s_h) \right) \right]$$

The Advantage-based PG:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) \left(Q_h^{\pi_\theta}(s_h, a_h) - b_h(s_h) \right) \right]$$

$$= \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) A_h^{\pi_\theta}(s_h, a_h) \right]$$

The Advantage-based PG:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) \left(Q_h^{\pi_\theta}(s_h, a_h) - b_h(s_h) \right) \right]$$

$$= \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) A_h^{\pi_\theta}(s_h, a_h) \right]$$

- The second step follows by choosing $b_h(s) = V_h^\pi(s)$.

The Advantage-based PG:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) \left(Q_h^{\pi_\theta}(s_h, a_h) - b_h(s_h) \right) \right]$$

$$= \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) A_h^{\pi_\theta}(s_h, a_h) \right]$$

- The second step follows by choosing $b_h(s) = V_h^\pi(s)$.
- In practice, the most common approach is to use $b_h(s)$ that's an estimate of $V_h^\pi(s)$.

PG with a Learned Baseline:

PG with a Learned Baseline:

Let $g'(\theta, \tau, b()) := \sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\big(R_h(\tau) - b(s_h, h)\big)$

PG with a Learned Baseline:

Let $g'(\theta, \tau, b()) := \sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\big(R_h(\tau) - b(s_h, h)\big)$

1. Initialize θ^0, parameters: $\eta^1, \eta^2, \ldots$

PG with a Learned Baseline:

Let $g'(\theta, \tau, b()) := \sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) \big(R_h(\tau) - b(s_h, h)\big)$

1. Initialize θ^0, parameters: $\eta^1, \eta^2, \ldots$
2. For $k = 0, \ldots$:

PG with a Learned Baseline:

$$\text{Let } g'(\theta, \tau, b()) := \sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \,|\, s_h)\big(R_h(\tau) - b(s_h, h)\big)$$

1. Initialize θ^0, parameters: $\eta^1, \eta^2, \ldots$
2. For $k = 0, \ldots$:
    1. Supervised Learning: Using N trajectories sampled under π_{θ^k}, estimate a baseline $\widetilde{b}$
       $\widetilde{b}(s, h) \approx V_h^{\theta^k}(s)$

PG with a Learned Baseline:

Let $g'(\theta, \tau, b()) := \sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h | s_h)\left(R_h(\tau) - b(s_h, h)\right)$

1. Initialize θ^0, parameters: $\eta^1, \eta^2, \ldots$
2. For $k = 0, \ldots$:
   1. Supervised Learning: Using N trajectories sampled under π_{θ^k}, estimate a baseline $\widetilde{b}$
      $\widetilde{b}(s, h) \approx V_h^{\theta^k}(s)$
   2. Obtain a trajectory $\tau \sim \rho_{\theta^k}$
      Compute $g'(\theta^k, \tau, \widetilde{b}())$

PG with a Learned Baseline:

Let $g'(\theta, \tau, b()) := \sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h | s_h) \left(R_h(\tau) - b(s_h, h) \right)$

1. Initialize θ^0, parameters: $\eta^1, \eta^2, \ldots$
2. For $k = 0, \ldots$:
    1. Supervised Learning: Using N trajectories sampled under π_{θ^k}, estimate a baseline $\widetilde{b}$
       $\widetilde{b}(s, h) \approx V_h^{\theta^k}(s)$
    2. Obtain a trajectory $\tau \sim \rho_{\theta^k}$
       Compute $g'(\theta^k, \tau, \widetilde{b}())$
    3. Update: $\theta^{k+1} = \theta^k + \eta^k g'(\theta^k, \tau, \widetilde{b}())$

PG with a Learned Baseline:

$$\text{Let } g'(\theta, \tau, b()) := \sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h | s_h)\big(R_h(\tau) - b(s_h, h)\big)$$

1. Initialize θ^0, parameters: $\eta^1, \eta^2, \ldots$
2. For $k = 0, \ldots$:
    1. Supervised Learning: Using N trajectories sampled under π_{θ^k}, estimate a baseline $\widetilde{b}$
       $\widetilde{b}(s, h) \approx V_h^{\theta^k}(s)$
    2. Obtain a trajectory $\tau \sim \rho_{\theta^k}$
       Compute $g'(\theta^k, \tau, \widetilde{b}())$
    3. Update: $\theta^{k+1} = \theta^k + \eta^k g'(\theta^k, \tau, \widetilde{b}())$

Note that regardless of our choice of $\widetilde{b}$, we still get unbiased gradient estimates.

(minibatch) PG with a Learned Baseline:

(minibatch) PG with a Learned Baseline:

1. Initialize θ^0, parameters: $\eta^1, \eta^2, \ldots$

(minibatch) PG with a Learned Baseline:

1. Initialize θ^0, parameters: $\eta^1, \eta^2, \ldots$
2. For $k = 0, \ldots$:

(minibatch) PG with a Learned Baseline:

1. Initialize θ^0, parameters: $\eta^1, \eta^2, \ldots$
2. For $k = 0, \ldots$:
    1. Supervised Learning: Using N trajectories sampled under π_{θ^k}, estimate a baseline $\widetilde{b}$
    $\widetilde{b}(s, h) \approx V_h^{\theta^k}(s)$

(minibatch) PG with a Learned Baseline:

1. Initialize θ^0, parameters: $\eta^1, \eta^2, \ldots$
2. For $k = 0, \ldots$:
    1. Supervised Learning: Using N trajectories sampled under π_{θ^k}, estimate a baseline $\widetilde{b}$
       $\widetilde{b}(s, h) \approx V_h^{\theta^k}(s)$
    2. Obtain M trajectories $\tau_1, \ldots \tau_M \sim \rho_{\theta^k}$

       Compute $g = \frac{1}{M} \sum_{m=1}^{M} g'(\theta^k, \tau_m, \widetilde{b}())$

(minibatch) PG with a Learned Baseline:

1. Initialize θ^0, parameters: $\eta^1, \eta^2, \ldots$
2. For $k = 0, \ldots$:
   1. Supervised Learning: Using N trajectories sampled under π_{θ^k}, estimate a baseline $\widetilde{b}$
      $\widetilde{b}(s, h) \approx V_h^{\theta^k}(s)$
   2. Obtain M trajectories $\tau_1, \ldots \tau_M \sim \rho_{\theta^k}$

      Compute $g = \frac{1}{M} \sum_{m=1}^{M} g'(\theta^k, \tau_m, \widetilde{b}())$
   3. Update: $\theta^{k+1} = \theta^k + \eta^k g$

Today

- Feedback from last lecture
- Recap
- The Performance Difference Lemma
- Trust Region Policy Optimization (TRPO)
- The Natural Policy Gradient (NPG)

Recall: Fitted Policy Iteration

- Initialization: choose a policy $\pi^0 : S \mapsto A$ and a sample size N
- For $k = 0,1,\ldots$
  1. **Fitted Policy Evaluation**: Using N sampled trajectories $\tau_1, \ldots \tau_N \sim \rho_{\pi^k}$, obtain approximation $\hat{Q}^{\pi^k} \approx Q^{\pi^k}$
  2. **Policy Improvement**: set $\pi_h^{k+1}(s) := \arg\max_a \hat{Q}^{\pi^k}(s, a, h)$

Fitted Policy Iteration: Advantage Version

- Initialization: choose a policy $\pi^0 : S \mapsto A$ and a sample size N
- For $k = 0,1,\ldots$
  1. **Fitted Policy Evaluation**: Using N sampled trajectories $\tau_1, \ldots \tau_N \sim \rho_{\pi^k}$, obtain approximation $\hat{A}^{\pi^k} \approx A^{\pi^k}$
  2. **Policy Improvement**: set $\pi_h^{k+1}(s) := \arg\max_a \hat{A}^{\pi^k}(s, a, h)$

The Performance Difference Lemma (PDL)

The Performance Difference Lemma (PDL)

- Let $\rho_{\widetilde{\pi},s}$ be the distribution of trajectories from starting state s acting under $\widetilde{\pi}$.
  (we are making the starting distribution explicit now).

The Performance Difference Lemma (PDL)

- Let $\rho_{\widetilde{\pi},s}$ be the distribution of trajectories from starting state s acting under $\widetilde{\pi}$. (we are making the starting distribution explicit now).
- For any two policies π and $\widetilde{\pi}$ and any state s,

$$V^{\widetilde{\pi}}(s) - V^{\pi}(s) = \mathbb{E}_{\tau \sim \rho_{\widetilde{\pi},s}} \left[\sum_{h=0}^{H-1} A^{\pi}(s_h, a_h, h) \right]$$

The Performance Difference Lemma (PDL)

- Let $\rho_{\widetilde{\pi},s}$ be the distribution of trajectories from starting state s acting under $\widetilde{\pi}$. (we are making the starting distribution explicit now).
- For any two policies π and $\widetilde{\pi}$ and any state s,

$$V^{\widetilde{\pi}}(s) - V^{\pi}(s) = \mathbb{E}_{\tau \sim \rho_{\widetilde{\pi},s}} \left[\sum_{h=0}^{H-1} A^{\pi}(s_h, a_h, h) \right]$$

Comments:

The Performance Difference Lemma (PDL)

- Let $\rho_{\widetilde{\pi},s}$ be the distribution of trajectories from starting state s acting under $\widetilde{\pi}$. (we are making the starting distribution explicit now).
- For any two policies π and $\widetilde{\pi}$ and any state s,

$$V^{\widetilde{\pi}}(s) - V^{\pi}(s) = \mathbb{E}_{\tau \sim \rho_{\widetilde{\pi},s}} \left[\sum_{h=0}^{H-1} A^{\pi}(s_h, a_h, h) \right]$$

Comments:

- Helps us think about error analysis, instabilities of fitted PI, and sub-optimality.

The Performance Difference Lemma (PDL)

- Let $\rho_{\widetilde{\pi},s}$ be the distribution of trajectories from starting state s acting under $\widetilde{\pi}$. (we are making the starting distribution explicit now).
- For any two policies π and $\widetilde{\pi}$ and any state s,

$$V^{\widetilde{\pi}}(s) - V^{\pi}(s) = \mathbb{E}_{\tau \sim \rho_{\widetilde{\pi},s}} \left[\sum_{h=0}^{H-1} A^{\pi}(s_h, a_h, h) \right]$$

Comments:

- Helps us think about error analysis, instabilities of fitted PI, and sub-optimality.
- Helps to understand algorithm design (TRPO, NPG, PPO)

The Performance Difference Lemma (PDL)

- Let $\rho_{\widetilde{\pi},s}$ be the distribution of trajectories from starting state s acting under $\widetilde{\pi}$. (we are making the starting distribution explicit now).
- For any two policies π and $\widetilde{\pi}$ and any state s,

$$V^{\widetilde{\pi}}(s) - V^{\pi}(s) = \mathbb{E}_{\tau \sim \rho_{\widetilde{\pi},s}} \left[\sum_{h=0}^{H-1} A^{\pi}(s_h, a_h, h) \right]$$

Comments:

- Helps us think about error analysis, instabilities of fitted PI, and sub-optimality.
- Helps to understand algorithm design (TRPO, NPG, PPO)
- This also motivates the use of “local” methods (e.g. policy gradient descent)

Back to Fitted Policy Iteration

Back to Fitted Policy Iteration

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?

Back to Fitted Policy Iteration

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?
- In Fitted Policy Iteration, $\hat{A}^{\pi^k} \approx A^{\pi^k}$ is achieved via supervised learning on $\tau_1, \ldots \tau_N \sim \rho_{\pi^k}$

Back to Fitted Policy Iteration

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?
- In Fitted Policy Iteration, $\hat{A}^{\pi^k} \approx A^{\pi^k}$ is achieved via supervised learning on $\tau_1, \ldots \tau_N \sim \rho_{\pi^k}$
- This means we expect $\mathbb{E}_{\tau \sim \rho_{\pi^k, s}}\left[\sum_{h=0}^{H-1} \hat{A}^{\pi^k}(s_h, a_h, h)\right] \approx \mathbb{E}_{\tau \sim \rho_{\pi^k, s}}\left[\sum_{h=0}^{H-1} A^{\pi^k}(s_h, a_h, h)\right]$

Back to Fitted Policy Iteration

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?
- In Fitted Policy Iteration, $\hat{A}^{\pi^k} \approx A^{\pi^k}$ is achieved via supervised learning on $\tau_1, \ldots \tau_N \sim \rho_{\pi^k}$
- This means we expect $\mathbb{E}_{\tau \sim \rho_{\pi^k,s}} \left[\sum_{h=0}^{H-1} \hat{A}^{\pi^k}(s_h, a_h, h) \right] \approx \mathbb{E}_{\tau \sim \rho_{\pi^k,s}} \left[\sum_{h=0}^{H-1} A^{\pi^k}(s_h, a_h, h) \right]$
- In particular, $\hat{A}^{\pi^k}$ should be close to A^{π^k} where π^k visits often…

Back to Fitted Policy Iteration

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?
- In Fitted Policy Iteration, $\hat{A}^{\pi^k} \approx A^{\pi^k}$ is achieved via supervised learning on $\tau_1, \ldots \tau_N \sim \rho_{\pi^k}$
- This means we expect $\mathbb{E}_{\tau \sim \rho_{\pi^k,s}} \left[\sum_{h=0}^{H-1} \hat{A}^{\pi^k}(s_h, a_h, h) \right] \approx \mathbb{E}_{\tau \sim \rho_{\pi^k,s}} \left[\sum_{h=0}^{H-1} A^{\pi^k}(s_h, a_h, h) \right]$
- In particular, $\hat{A}^{\pi^k}$ should be close to A^{π^k} where π^k visits often…
- But it could be very bad in places π^k visits rarely, and nothing stops π^{k+1} from visiting those bad places very often!

Back to Fitted Policy Iteration

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?
- In Fitted Policy Iteration, $\hat{A}^{\pi^k} \approx A^{\pi^k}$ is achieved via supervised learning on $\tau_1, \ldots \tau_N \sim \rho_{\pi^k}$
- This means we expect $\mathbb{E}_{\tau \sim \rho_{\pi^k,s}} \left[\sum_{h=0}^{H-1} \hat{A}^{\pi^k}(s_h, a_h, h) \right] \approx \mathbb{E}_{\tau \sim \rho_{\pi^k,s}} \left[\sum_{h=0}^{H-1} A^{\pi^k}(s_h, a_h, h) \right]$
- In particular, $\hat{A}^{\pi^k}$ should be close to A^{π^k} where π^k visits often…
- But it could be very bad in places π^k visits rarely, and nothing stops π^{k+1} from visiting those bad places very often!
- So π^{k+1} could end up being (much) worse than π^k

Back to Fitted Policy Iteration

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?
- In Fitted Policy Iteration, $\hat{A}^{\pi^k} \approx A^{\pi^k}$ is achieved via supervised learning on $\tau_1, \ldots \tau_N \sim \rho_{\pi^k}$
- This means we expect $\mathbb{E}_{\tau \sim \rho_{\pi^k,s}} \left[\sum_{h=0}^{H-1} \hat{A}^{\pi^k}(s_h, a_h, h) \right] \approx \mathbb{E}_{\tau \sim \rho_{\pi^k,s}} \left[\sum_{h=0}^{H-1} A^{\pi^k}(s_h, a_h, h) \right]$
- In particular, $\hat{A}^{\pi^k}$ should be close to A^{π^k} where π^k visits often…
- But it could be very bad in places π^k visits rarely, and nothing stops π^{k+1} from visiting those bad places very often!
- So π^{k+1} could end up being (much) worse than π^k

- Problem is a mismatch between expectations: what we really want is

$$\mathbb{E}_{\tau \sim \rho_{\pi^{k+1},s}} \left[\sum_{h=0}^{H-1} \hat{A}^{\pi^k}(s_h, a_h, h) \right] \approx \mathbb{E}_{\tau \sim \rho_{\pi^{k+1},s}} \left[\sum_{h=0}^{H-1} A^{\pi^k}(s_h, a_h, h) \right]$$

Back to Fitted Policy Iteration

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?
- In Fitted Policy Iteration, $\hat{A}^{\pi^k} \approx A^{\pi^k}$ is achieved via supervised learning on $\tau_1, \ldots \tau_N \sim \rho_{\pi^k}$
- This means we expect $\mathbb{E}_{\tau \sim \rho_{\pi^k,s}} \left[\sum_{h=0}^{H-1} \hat{A}^{\pi^k}(s_h, a_h, h) \right] \approx \mathbb{E}_{\tau \sim \rho_{\pi^k,s}} \left[\sum_{h=0}^{H-1} A^{\pi^k}(s_h, a_h, h) \right]$
- In particular, $\hat{A}^{\pi^k}$ should be close to A^{π^k} where π^k visits often…
- But it could be very bad in places π^k visits rarely, and nothing stops π^{k+1} from visiting those bad places very often!
- So π^{k+1} could end up being (much) worse than π^k

- Problem is a mismatch between expectations: what we really want is

$$\mathbb{E}_{\tau \sim \rho_{\pi^{k+1},s}} \left[\sum_{h=0}^{H-1} \hat{A}^{\pi^k}(s_h, a_h, h) \right] \approx \mathbb{E}_{\tau \sim \rho_{\pi^{k+1},s}} \left[\sum_{h=0}^{H-1} A^{\pi^k}(s_h, a_h, h) \right]$$

- One way to ensure this: keep $\pi^{k+1} \approx \pi^k$

Today

- Feedback from last lecture
- Recap
- The Performance Difference Lemma
- Trust Region Policy Optimization (TRPO)
- The Natural Policy Gradient (NPG)

A trust region formulation for policy update:

A trust region formulation for policy update:

- What's bad about fitted PI?
  even if we pick better actions "on average", the trajectory updates are unstable

A trust region formulation for policy update:

- What's bad about fitted PI?
  even if we pick better actions "on average", the trajectory updates are unstable
- Can we fix this?
  Let's look at an incremental policy updating approach

A trust region formulation for policy update:

- What's bad about fitted PI?
  even if we pick better actions "on average", the trajectory updates are unstable
- Can we fix this?
  Let's look at an incremental policy updating approach

1. Initialize θ^0
2. For $k = 0, \ldots, K$:
   try to approximately solve:
   $$\theta^{k+1} = \arg\max_{\theta} \mathbb{E}_{s_0, \ldots, s_{H-1} \sim \rho_{\pi_{\theta^k}}} \left[\sum_{h=0}^{H-1} \mathbb{E}_{a_h \sim \pi_\theta(\cdot|s_h)} \left[A^{\pi_{\theta^k}}(s_h, a_h, h) \right] \right]$$
   s.t. ρ_θ is "close" to $\rho_{\pi_{\theta^k}}$
3. Return π_{θ^K}

A trust region formulation for policy update:

- What's bad about fitted PI?
  even if we pick better actions "on average", the trajectory updates are unstable
- Can we fix this?
  Let's look at an incremental policy updating approach

1. Initialize θ^0
2. For $k = 0, \ldots, K$:
   try to approximately solve:

$$\theta^{k+1} = \arg\max_{\theta} \mathbb{E}_{s_0,\ldots,s_{H-1}\sim\rho_{\pi_{\theta^k}}} \left[\sum_{h=0}^{H-1} \mathbb{E}_{a_h\sim\pi_\theta(\cdot|s_h)} \left[A^{\pi_{\theta^k}}(s_h, a_h, h) \right] \right]$$

s.t. ρ_θ is "close" to $\rho_{\pi_{\theta^k}}$

3. Return π_{θ^K}

- How should we define "close", i.e., what is our "trust" region?

KL-divergence: measures the distance between two distributions

Given two distributions P & Q, where $P \in \Delta(X), Q \in \Delta(X)$,
KL Divergence is defined as:

$$KL(P \mid Q) = \mathbb{E}_{x \sim P}\left[\ln \frac{P(x)}{Q(x)}\right]$$

KL-divergence: measures the distance between two distributions

Given two distributions P & Q, where $P \in \Delta(X), Q \in \Delta(X)$,
KL Divergence is defined as:

$$KL(P \mid Q) = \mathbb{E}_{x \sim P}\left[\ln \frac{P(x)}{Q(x)}\right]$$

Examples:

If $Q = P$, then $KL(P \mid Q) = KL(Q \mid P) = 0$

KL-divergence: measures the distance between two distributions

Given two distributions P & Q, where $P \in \Delta(X), Q \in \Delta(X)$,
KL Divergence is defined as:

$$KL(P|Q) = \mathbb{E}_{x \sim P}\left[\ln \frac{P(x)}{Q(x)}\right]$$

Examples:

If $Q = P$, then $KL(P|Q) = KL(Q|P) = 0$

If $P = \mathcal{N}(\mu_1, \sigma^2 I), Q = \mathcal{N}(\mu_2, \sigma^2 I)$, then $KL(P|Q) = \frac{1}{2\sigma^2}\|\mu_1 - \mu_2\|^2$

KL-divergence: measures the distance between two distributions

Given two distributions P & Q, where $P \in \Delta(X), Q \in \Delta(X)$,
KL Divergence is defined as:

$$KL(P \mid Q) = \mathbb{E}_{x \sim P} \left[\ln \frac{P(x)}{Q(x)} \right]$$

Examples:

If $Q = P$, then $KL(P \mid Q) = KL(Q \mid P) = 0$

If $P = \mathcal{N}(\mu_1, \sigma^2 I), Q = \mathcal{N}(\mu_2, \sigma^2 I)$, then $KL(P \mid Q) = \frac{1}{2\sigma^2} \|\mu_1 - \mu_2\|^2$

Fact:

$KL(P \mid Q) \geq 0$, and is 0 if and only if $P = Q$

Trust Region Policy Optimization (TRPO)

1. Initialize θ^0 , δ
2. For $k = 0, \ldots, K$:
   try to approximately solve:

$$\theta^{k+1} = \arg\max_{\theta} \mathbb{E}_{s_0,\ldots,s_{H-1}\sim\rho_{\pi_{\theta^k}}} \left[\sum_{h=0}^{H-1} \mathbb{E}_{a_h\sim\pi_\theta(\cdot|s_h)} \left[A^{\pi_{\theta^k}}(s_h, a_h, h)\right] \right]$$

$$\text{s.t. } KL\left(\rho_{\pi_{\theta^k}} | \rho_{\pi_\theta}\right) \leq \delta$$

3. Return π_{θ^K}

- We want to maximize local advantage against π_{θ^k},
  but we want the new policy to be close to π_{θ^k} (in the KL sense)
- How do we implement this with sampled trajectories?)

How do we implement TRPO with samples?

1. Initialize parameter θ^0, sample size M, and tolerance δ
2. For $k = 0, \ldots, K$:
   1. [Advantage-Evaluation Subroutine]
      Using M sampled trajectories $\tau_1, \ldots \tau_M \sim \rho_{\pi_{\theta^k}}$, obtain approximation $\hat{A}^{\pi_{\theta^k}} \approx A^{\pi_{\theta^k}}$
   2. Solve the following optimization problem to obtain θ^{k+1}:

$$\max_{\theta} \sum_{m=1}^{M} \sum_{h=0}^{H-1} \mathbb{E}_{a \sim \pi_\theta(\cdot | s_h^m)} \left[\hat{A}^{\pi_{\theta^k}}(s_h^m, a, h) \right]$$

$$\text{s.t. } \sum_{m=1}^{M} \sum_{h=0}^{H-1} \ln \frac{\pi_{\theta^k}(a_h^m \mid s_h^m)}{\pi_\theta(a_h^m \mid s_h^m)} \leq \delta$$

How do we implement TRPO with samples?

1. Initialize parameter θ^0, sample size M, and tolerance δ
2. For $k = 0,\ldots,K$:
    1. [Advantage-Evaluation Subroutine]
       Using M sampled trajectories $\tau_1, \ldots \tau_M \sim \rho_{\pi_{\theta^k}}$, obtain approximation $\hat{A}^{\pi_{\theta^k}} \approx A^{\pi_{\theta^k}}$
    2. Solve the following optimization problem to obtain θ^{k+1}:

$$\max_{\theta} \sum_{m=1}^{M} \sum_{h=0}^{H-1} \mathbb{E}_{a \sim \pi_\theta(\cdot|s_h^m)} \left[\hat{A}^{\pi_{\theta^k}}(s_h^m, a, h) \right]$$

$$\text{s.t.} \sum_{m=1}^{M} \sum_{h=0}^{H-1} \ln \frac{\pi_{\theta^k}(a_h^m | s_h^m)}{\pi_\theta(a_h^m | s_h^m)} \leq \delta$$

Approximate expectation by importance sampling:

$$\mathbb{E}_{a \sim \pi_\theta(\cdot|s_h^m)} \left[\hat{A}^{\pi_{\theta^k}}(s_h^m, a, h) \right]$$

$$= \mathbb{E}_{a \sim \pi_{\theta^k}(\cdot|s_h^m)} \left[\frac{\pi_\theta(a | s_h^m)}{\pi_{\theta^k}(a | s_h^m)} \hat{A}^{\pi_{\theta^k}}(s_h^m, a, h) \right]$$

Today

- Feedback from last lecture
- Recap
- The Performance Difference Lemma
- Trust Region Policy Optimization (TRPO)
- The Natural Policy Gradient (NPG)

TRPO is locally equivalent to a much simpler algorithm

TRPO at iteration k:

$$\max_{\theta} \mathbb{E}_{s_0,\ldots,s_{H-1}\sim\rho_{\pi_{\theta^k}}} \left[\sum_{h=0}^{H-1} \mathbb{E}_{a_h\sim\pi_\theta(\cdot|s_h)} \left[A^{\pi_{\theta^k}}(s_h, a_h, h) \right] \right]$$

$$\text{s.t. } KL\left(\rho_{\pi_{\theta^k}} | \rho_{\pi_\theta}\right) \leq \delta$$

Intuition: maximize local advantage subject to being incremental (in KL)

TRPO is locally equivalent to a much simpler algorithm

TRPO at iteration k:

$$\max_{\theta} \mathbb{E}_{s_0,\ldots,s_{H-1}\sim\rho_{\pi_{\theta^k}}}\left[\sum_{h=0}^{H-1}\mathbb{E}_{a_h\sim\pi_\theta(\cdot|s_h)}\left[A^{\pi_{\theta^k}}(s_h,a_h,h)\right]\right]$$ → First-order Taylor expansion at θ^k

s.t. $KL\left(\rho_{\pi_{\theta^k}} | \rho_{\pi_\theta}\right) \leq \delta$ → second-order Taylor expansion at θ^k

Intuition: maximize local advantage
subject to being incremental (in KL)

TRPO is locally equivalent to a much simpler algorithm

TRPO at iteration k:

$$\max_{\theta} \mathbb{E}_{s_0,\ldots,s_{H-1}\sim\rho_{\pi_{\theta^k}}}\left[\sum_{h=0}^{H-1}\mathbb{E}_{a_h\sim\pi_\theta(\cdot|s_h)}\left[A^{\pi_{\theta^k}}(s_h,a_h,h)\right]\right]$$ → First-order Taylor expansion at θ^k

$$\text{s.t. } KL\left(\rho_{\pi_{\theta^k}} | \rho_{\pi_\theta}\right) \leq \delta$$ → second-order Taylor expansion at θ^k

Intuition: maximize local advantage subject to being incremental (in KL)

TRPO is locally equivalent to a much simpler algorithm

TRPO at iteration k:

$$\max_{\theta} \mathbb{E}_{s_0,\ldots,s_{H-1}\sim\rho_{\pi_{\theta^k}}}\left[\sum_{h=0}^{H-1}\mathbb{E}_{a_h\sim\pi_\theta(\cdot|s_h)}\left[A^{\pi_{\theta^k}}(s_h,a_h,h)\right]\right]$$

→ First-order Taylor expansion at θ^k

$$\text{s.t. } KL\left(\rho_{\pi_{\theta^k}} | \rho_{\pi_\theta}\right) \le \delta$$

→ second-order Taylor expansion at θ^k

Intuition: maximize local advantage subject to being incremental (in KL)

$$\max_{\theta} \nabla_\theta J(\theta^k)^\top(\theta - \theta^k)$$

TRPO is locally equivalent to a much simpler algorithm

TRPO at iteration k:

$$\max_{\theta} \mathbb{E}_{s_0,\ldots,s_{H-1}\sim\rho_{\pi_{\theta^k}}} \left[\sum_{h=0}^{H-1} \mathbb{E}_{a_h\sim\pi_\theta(\cdot|s_h)} \left[A^{\pi_{\theta^k}}(s_h, a_h, h) \right] \right]$$

→ First-order Taylor expansion at θ^k

$$\text{s.t. } KL\left(\rho_{\pi_{\theta^k}} | \rho_{\pi_\theta}\right) \le \delta$$

→ second-order Taylor expansion at θ^k

Intuition: maximize local advantage subject to being incremental (in KL)

$$\max_{\theta} \nabla_\theta J(\theta^k)^\top (\theta - \theta^k)$$

$$\text{s.t. } (\theta - \theta^k)^\top F_{\theta^k} (\theta - \theta^k) \le \delta$$

TRPO is locally equivalent to a much simpler algorithm

TRPO at iteration k:

$$\max_{\theta} \mathbb{E}_{s_0,\ldots,s_{H-1}\sim\rho_{\pi_{\theta^k}}} \left[\sum_{h=0}^{H-1} \mathbb{E}_{a_h\sim\pi_\theta(\cdot|s_h)} \left[A^{\pi_{\theta^k}}(s_h, a_h, h) \right] \right]$$

→ First-order Taylor expansion at θ^k

$$\text{s.t. } KL\left(\rho_{\pi_{\theta^k}} | \rho_{\pi_\theta}\right) \leq \delta$$

→ second-order Taylor expansion at θ^k

Intuition: maximize local advantage subject to being incremental (in KL)

$$\max_{\theta} \nabla_\theta J(\theta^k)^\top (\theta - \theta^k)$$

$$\text{s.t. } (\theta - \theta^k)^\top F_{\theta^k} (\theta - \theta^k) \leq \delta$$

(Where F_{θ^k} is the "Fisher Information Matrix")

Natural Policy Gradient (NPG): A "leading order" equivalent program to TRPO:

1. Initialize θ^0
2. For $k = 0, \ldots, K$:

$$\theta^{k+1} = \arg\max_{\theta} \nabla_\theta J(\theta^k)^\top (\theta - \theta^k)$$

$$\text{s.t. } (\theta - \theta^k)^\top F_{\theta^k} (\theta - \theta^k) \leq \delta$$

3. Return π_{θ^K}

Natural Policy Gradient (NPG): A "leading order" equivalent program to TRPO:

1. Initialize θ^0
2. For $k = 0, \ldots, K$:
$$\theta^{k+1} = \arg\max_{\theta} \nabla_\theta J(\theta^k)^\top (\theta - \theta^k)$$
$$\text{s.t. } (\theta - \theta^k)^\top F_{\theta^k} (\theta - \theta^k) \le \delta$$
3. Return π_{θ^K}

- Where $\nabla_\theta J(\theta^k)$ is the gradient of $J(\theta)$ evaluated at θ^k, and
- F_θ is (basically) the Fisher information matrix at $\theta \in \mathbb{R}^d$, defined as:

$$F_\theta := \mathbb{E}_{\tau \sim \rho_{\pi_\theta}} \left[\nabla_\theta \ln \rho_\theta(\tau) \left(\nabla_\theta \ln \rho_\theta(\tau) \right)^\top \right] \in \mathbb{R}^{d \times d}$$

$$= \mathbb{E}_{\tau \sim \rho_{\pi_\theta}} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) \left(\nabla_\theta \ln \pi_\theta(a_h \mid s_h) \right)^\top \right]$$

NPG has a closed form update!

1. Initialize θ^0
2. For $k = 0, \ldots, K$:

$$\theta^{k+1} = \arg\max_{\theta} \nabla_\theta J(\theta^k)^\top (\theta - \theta^k)$$

$$\text{s.t. } (\theta - \theta^k)^\top F_{\theta^k} (\theta - \theta^k) \leq \delta$$

3. Return π_{θ^K}

NPG has a closed form update!

1. Initialize θ^0
2. For $k = 0, \ldots, K$:

$$\theta^{k+1} = \arg\max_{\theta} \nabla_\theta J(\theta^k)^\top (\theta - \theta^k)$$

$$\text{s.t. } (\theta - \theta^k)^\top F_{\theta^k} (\theta - \theta^k) \le \delta$$

3. Return π_{θ^K}

Linear objective and quadratic convex constraint: we can solve it optimally!

NPG has a closed form update!

1. Initialize θ^0
2. For $k = 0, \ldots, K$:

$$\theta^{k+1} = \arg\max_{\theta} \nabla_\theta J(\theta^k)^\top (\theta - \theta^k)$$

$$\text{s.t. } (\theta - \theta^k)^\top F_{\theta^k} (\theta - \theta^k) \leq \delta$$

3. Return π_{θ^K}

Linear objective and quadratic convex constraint: we can solve it optimally!

Indeed this gives us:

$$\theta^{k+1} = \theta^k + \eta F_{\theta^k}^{-1} \nabla_\theta J(\theta^k)$$

NPG has a closed form update!

1. Initialize θ^0
2. For $k = 0, \ldots, K$:

$$\theta^{k+1} = \arg\max_{\theta} \nabla_\theta J(\theta^k)^\top (\theta - \theta^k)$$

$$\text{s.t. } (\theta - \theta^k)^\top F_{\theta^k} (\theta - \theta^k) \leq \delta$$

3. Return π_{θ^K}

Linear objective and quadratic convex constraint: we can solve it optimally!

Indeed this gives us:

$$\theta^{k+1} = \theta^k + \eta F_{\theta^k}^{-1} \nabla_\theta J(\theta^k)$$

Where $\eta = \sqrt{\dfrac{\delta}{\nabla_\theta J(\theta^k)^\top F_{\theta^k}^{-1} \nabla_\theta J(\theta^k)}}$

An Implementation: Sample Based NPG

1. Initialize θ^0
2. For $k = 0, \ldots, K$:
   - Obtain approximation of Policy Gradient: $\hat{g} \approx \nabla_\theta J(\theta^k)$
   - Obtain approximation of Fisher information: $\hat{F} \approx F_{\theta^k}$
   - Natural Gradient Ascent: $\theta^{k+1} = \theta^k + \eta \hat{F}^{-1} \hat{g}$
3. Return π_{θ^K}

An Implementation: Sample Based NPG

1. Initialize θ^0
2. For $k = 0,\ldots,K$:
   - Obtain approximation of Policy Gradient: $\hat{g} \approx \nabla_\theta J(\theta^k)$
   - Obtain approximation of Fisher information: $\hat{F} \approx F_{\theta^k}$
   - Natural Gradient Ascent: $\theta^{k+1} = \theta^k + \eta \hat{F}^{-1} \hat{g}$
3. Return π_{θ^K}

(We will implement it in HW4 on Cartpole)

Today

- Feedback from last lecture
- Recap
- The Performance Difference Lemma
- Trust Region Policy Optimization (TRPO)
- The Natural Policy Gradient (NPG)

Summary:

1. Performance Difference Lemma tells us we need to stay local
2. TRPO and NPG ensure we don't move too much each step

Attendance:
bit.ly/3RcTC9T

Feedback:
bit.ly/3RHtlxy

Example of Natural Gradient on 1-d problem: 2 actions, 1 state

$$(\pi_\theta[1], \pi_\theta[2]) := \left(\frac{\exp(\theta)}{1 + \exp(\theta)}, \frac{1}{1 + \exp(\theta)} \right)$$

$$J(\theta) = 100 \cdot \pi_\theta[1] + 1 \cdot \pi_\theta[2]$$

Example of Natural Gradient on 1-d problem: 2 actions, 1 state

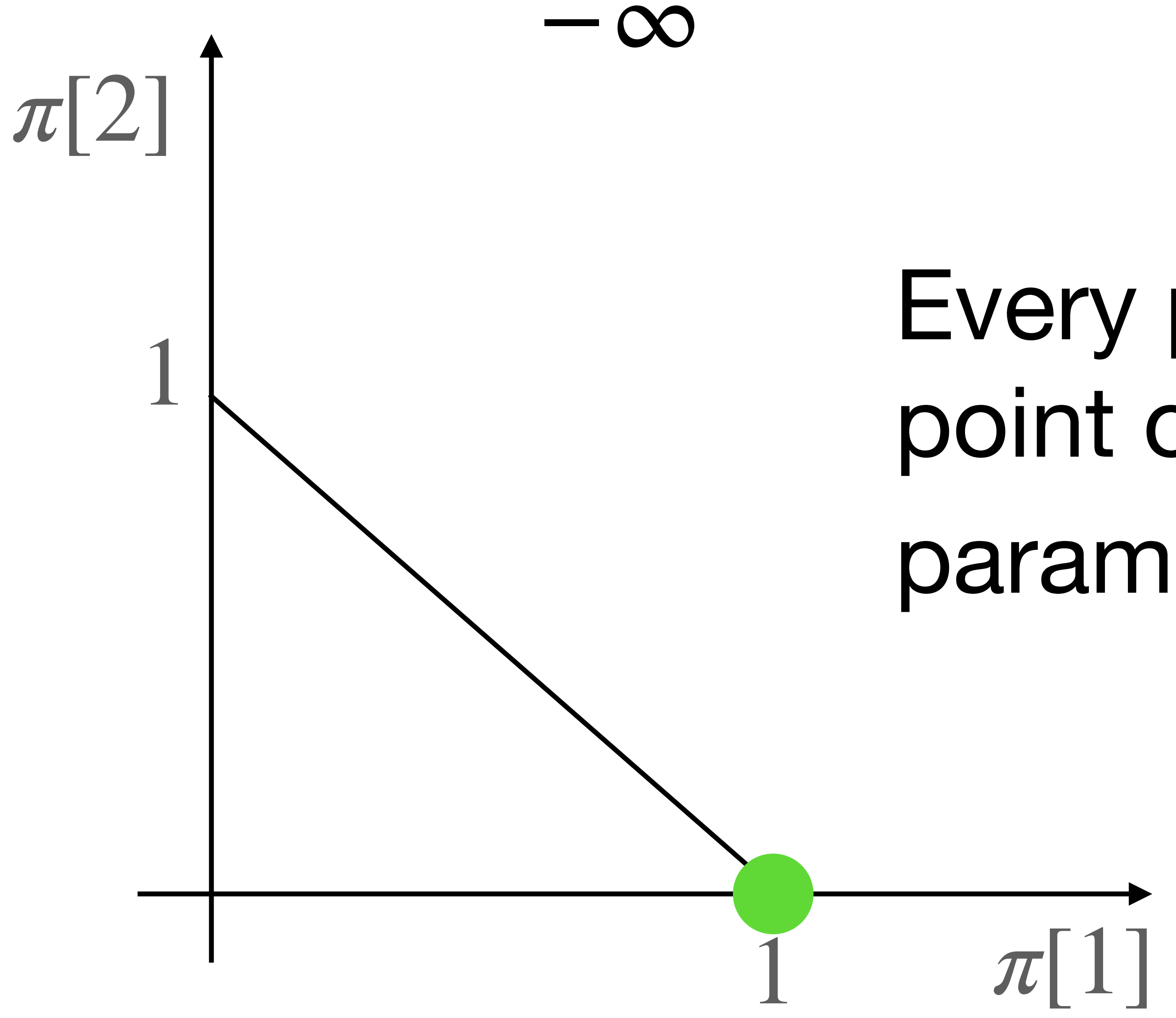
$$(\pi_\theta[1], \pi_\theta[2]) := \left(\frac{\exp(\theta)}{1+\exp(\theta)}, \frac{1}{1+\exp(\theta)} \right)$$

$$J(\theta) = 100 \cdot \pi_\theta[1] + 1 \cdot \pi_\theta[2]$$

Example of Natural Gradient on 1-d problem: 2 actions, 1 state

$$(\pi_\theta[1], \pi_\theta[2]) := \left(\frac{\exp(\theta)}{1+\exp(\theta)}, \frac{1}{1+\exp(\theta)} \right)$$

$$J(\theta) = 100 \cdot \pi_\theta[1] + 1 \cdot \pi_\theta[2]$$

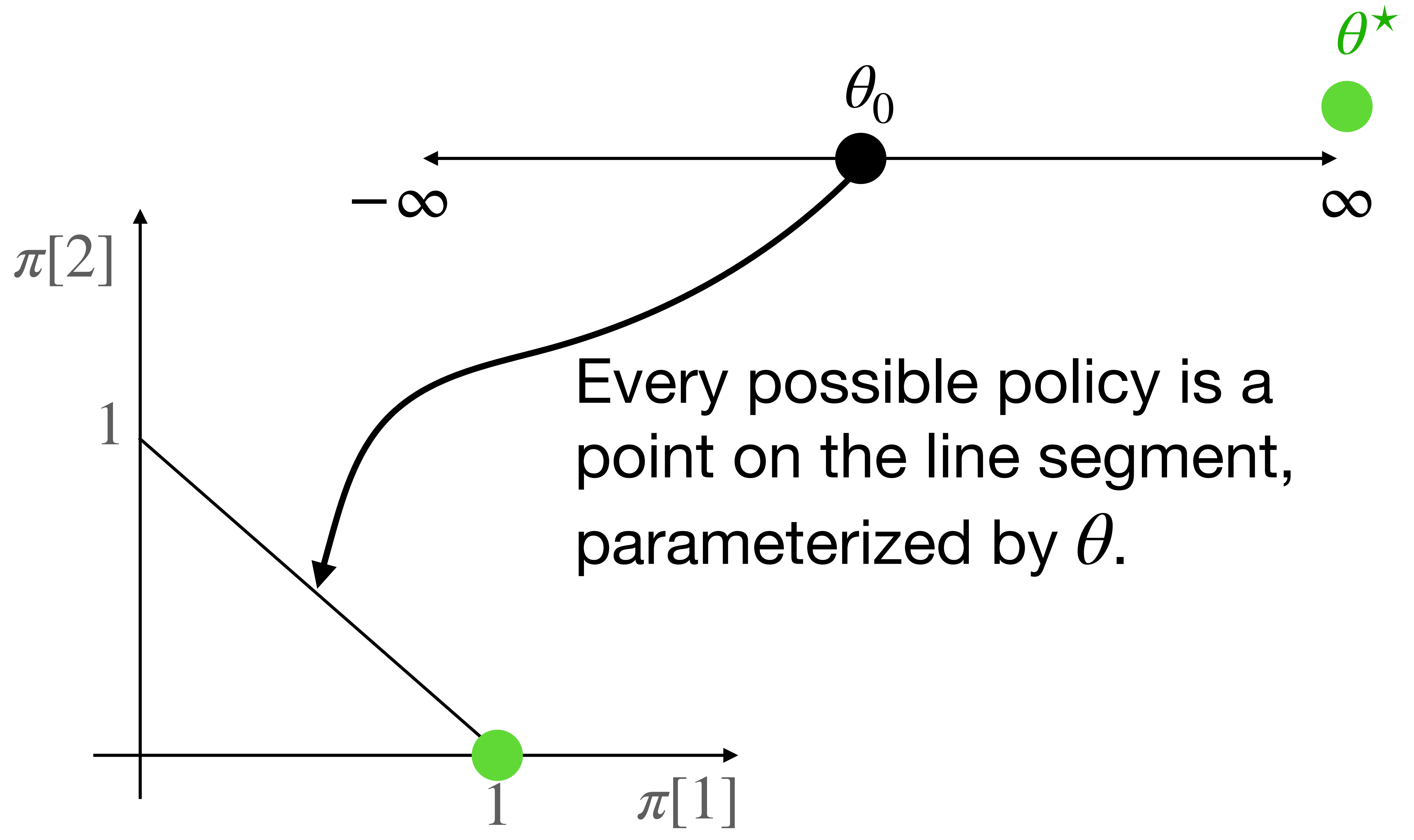


Every possible policy is a point on the line segment, parameterized by θ.

Example of Natural Gradient on 1-d problem: 2 actions, 1 state

$$(\pi_\theta[1], \pi_\theta[2]) := \left(\frac{\exp(\theta)}{1+\exp(\theta)}, \frac{1}{1+\exp(\theta)}\right)$$

$$J(\theta) = 100 \cdot \pi_\theta[1] + 1 \cdot \pi_\theta[2]$$



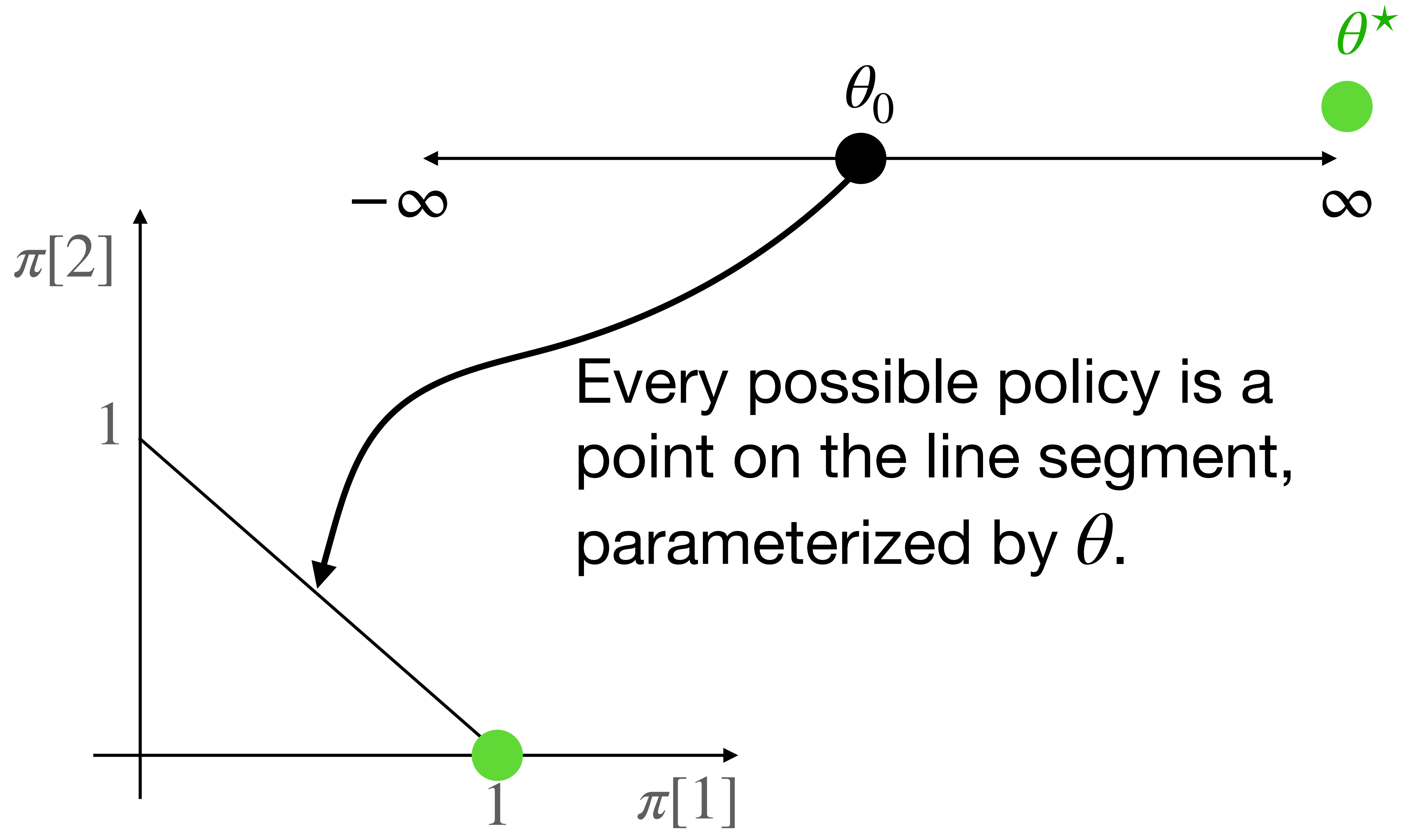
Every possible policy is a point on the line segment, parameterized by θ.

Example of Natural Gradient on 1-d problem: 2 actions, 1 state

$$(\pi_\theta[1], \pi_\theta[2]) := \left(\frac{\exp(\theta)}{1+\exp(\theta)}, \frac{1}{1+\exp(\theta)} \right)$$

Gradient: $\nabla_\theta J(\theta) = \dfrac{99\exp(\theta)}{(1+\exp(\theta))^2}$

$$J(\theta) = 100 \cdot \pi_\theta[1] + 1 \cdot \pi_\theta[2]$$



Every possible policy is a point on the line segment, parameterized by θ.

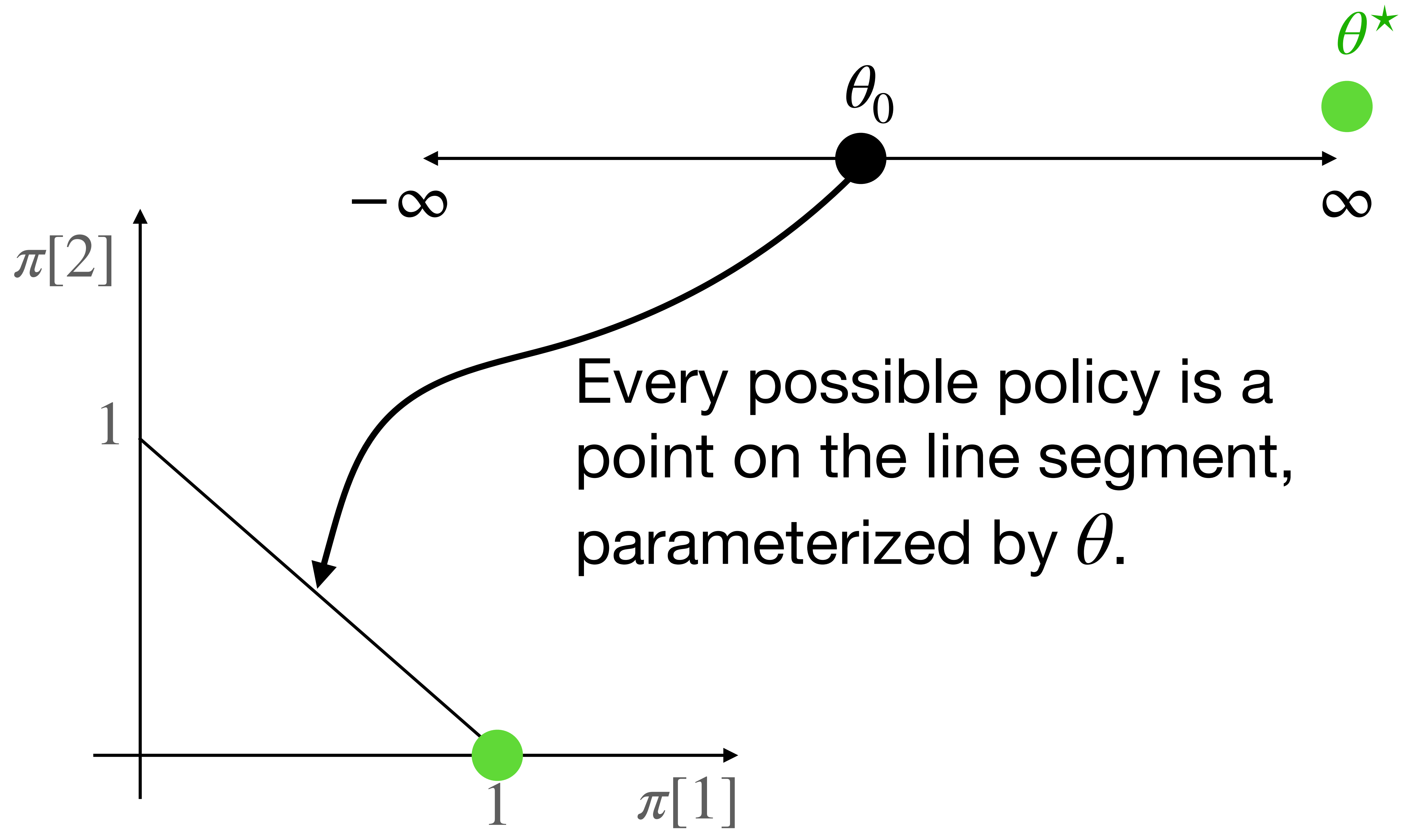
Example of Natural Gradient on 1-d problem: 2 actions, 1 state

$$(\pi_\theta[1], \pi_\theta[2]) := \left(\frac{\exp(\theta)}{1+\exp(\theta)}, \frac{1}{1+\exp(\theta)} \right)$$

$$J(\theta) = 100 \cdot \pi_\theta[1] + 1 \cdot \pi_\theta[2]$$

Gradient: $\nabla_\theta J(\theta) = \dfrac{99 \exp(\theta)}{(1+\exp(\theta))^2}$

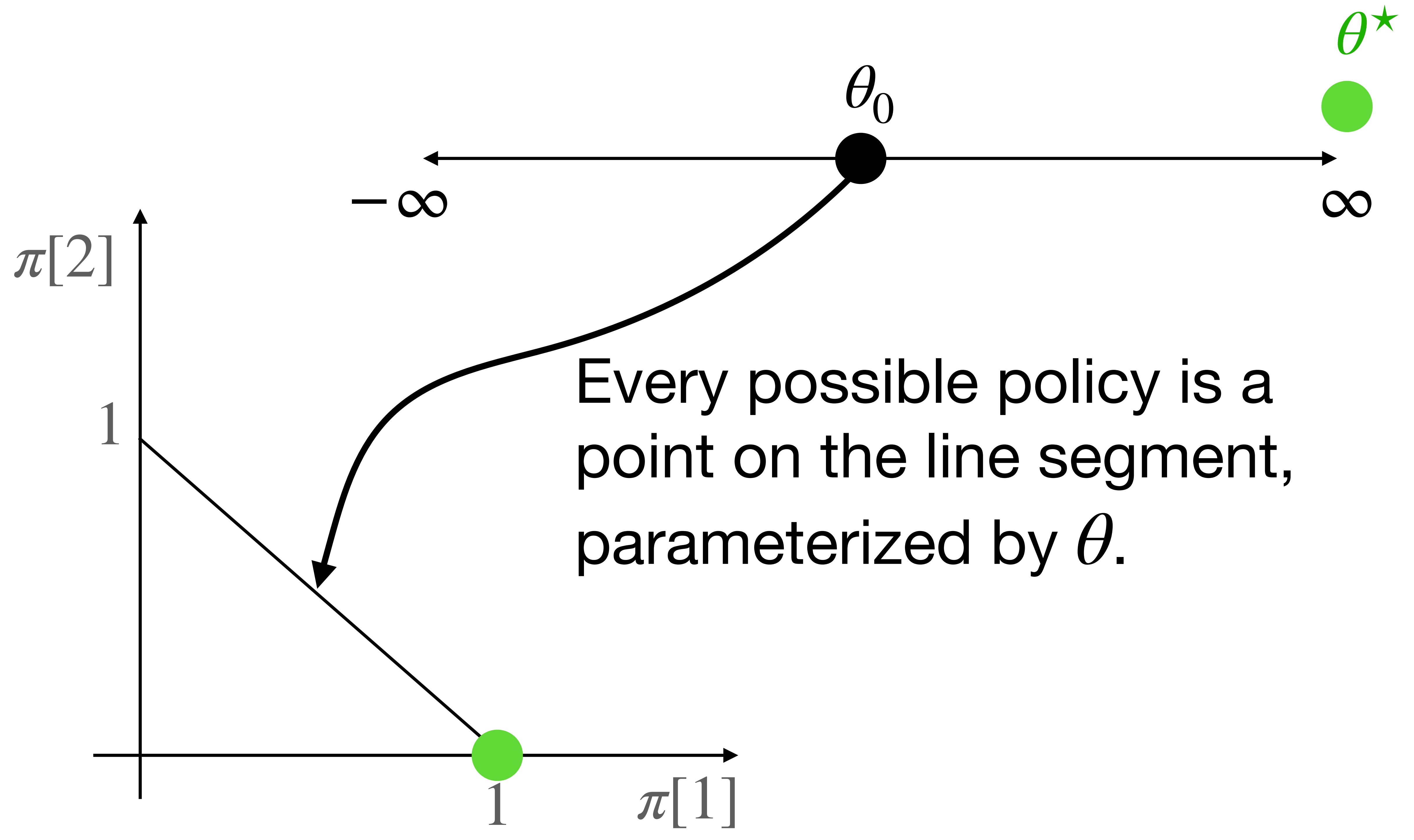
Exact PG: $\theta^{k+1} = \theta^k + \eta \dfrac{99 \exp(\theta^k)}{(1+\exp(\theta^k))^2}$



Example of Natural Gradient on 1-d problem: 2 actions, 1 state

$$(\pi_\theta[1], \pi_\theta[2]) := \left(\frac{\exp(\theta)}{1+\exp(\theta)}, \frac{1}{1+\exp(\theta)} \right)$$

$$J(\theta) = 100 \cdot \pi_\theta[1] + 1 \cdot \pi_\theta[2]$$



Every possible policy is a point on the line segment, parameterized by θ.

Gradient: $\nabla_\theta J(\theta) = \dfrac{99\exp(\theta)}{(1+\exp(\theta))^2}$

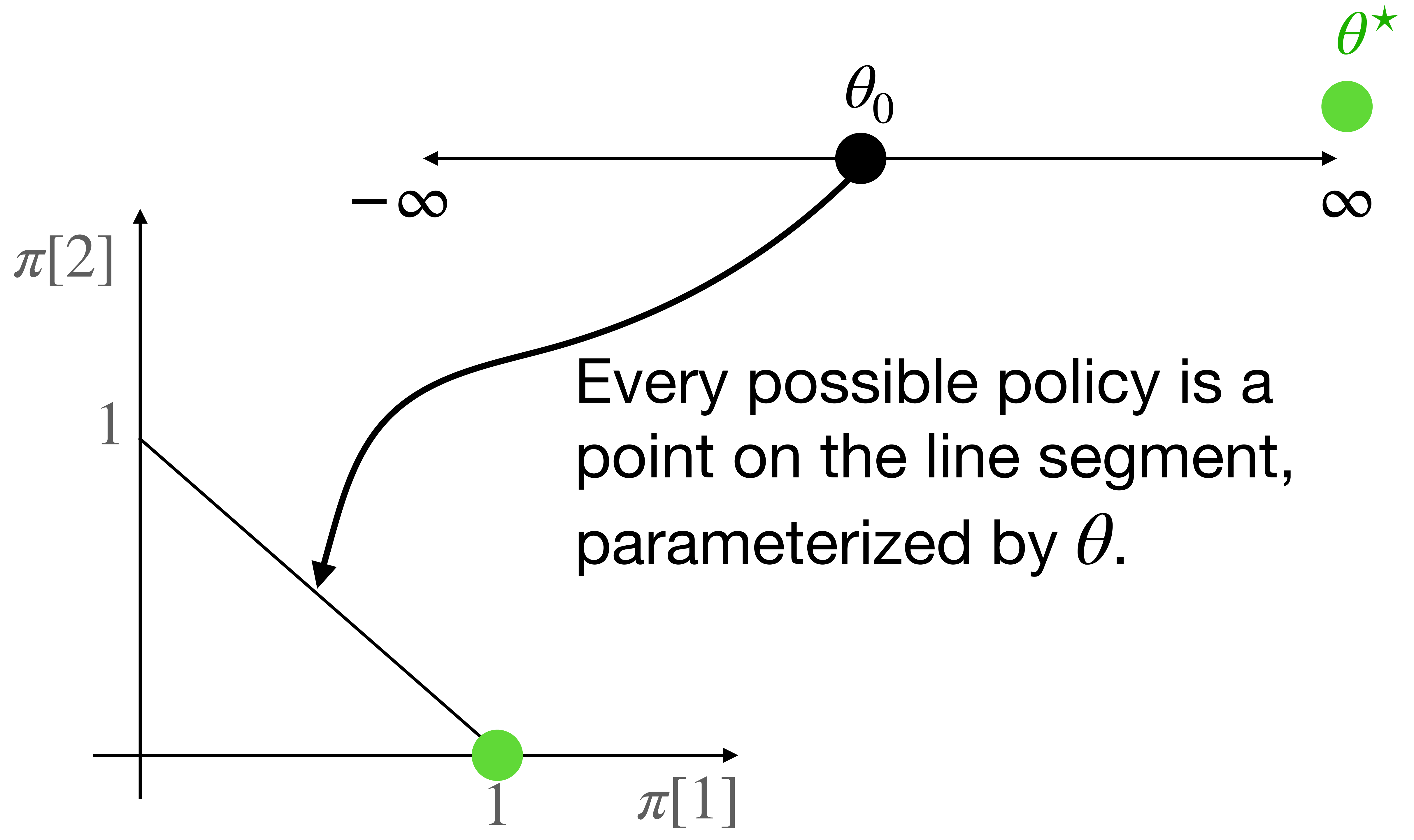
Exact PG: $\theta^{k+1} = \theta^k + \eta \dfrac{99\exp(\theta^k)}{(1+\exp(\theta^k))^2}$

i.e., vanilla GA moves to $\theta = \infty$ with smaller and smaller steps, since $\nabla_\theta J(\theta) \to 0$ as $\theta \to \infty$

Example of Natural Gradient on 1-d problem: 2 actions, 1 state

$$(\pi_\theta[1], \pi_\theta[2]) := \left(\frac{\exp(\theta)}{1+\exp(\theta)}, \frac{1}{1+\exp(\theta)} \right)$$

$$J(\theta) = 100 \cdot \pi_\theta[1] + 1 \cdot \pi_\theta[2]$$



Gradient: $\nabla_\theta J(\theta) = \dfrac{99\exp(\theta)}{(1+\exp(\theta))^2}$

Exact PG: $\theta^{k+1} = \theta^k + \eta \dfrac{99\exp(\theta^k)}{(1+\exp(\theta^k))^2}$

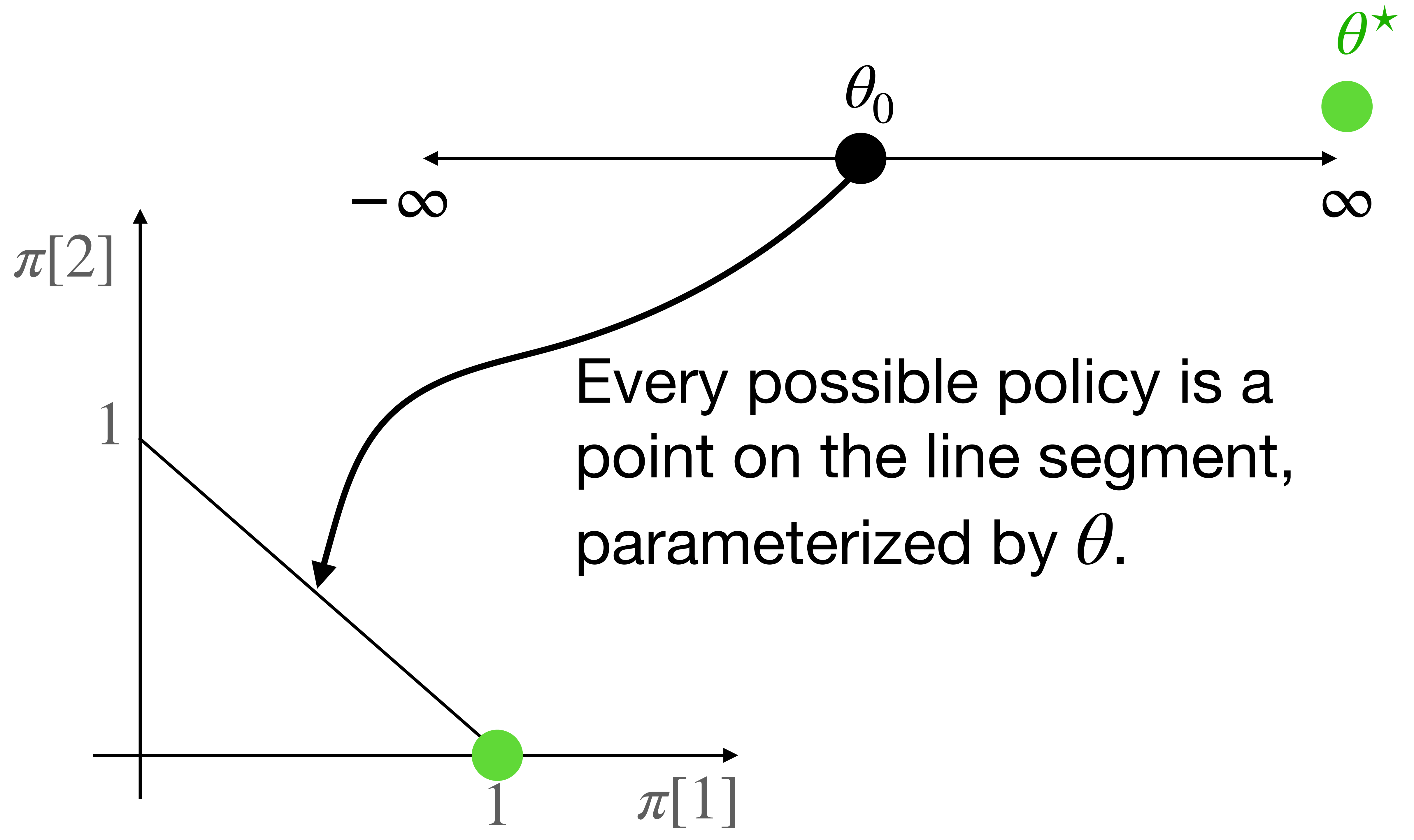
i.e., vanilla GA moves to $\theta = \infty$ with smaller and smaller steps, since $\nabla_\theta J(\theta) \to 0$ as $\theta \to \infty$

Fisher information scalar: $F_\theta = \dfrac{\exp(\theta)}{(1+\exp(\theta))^2}$

Example of Natural Gradient on 1-d problem: 2 actions, 1 state

$$(\pi_\theta[1], \pi_\theta[2]) := \left(\frac{\exp(\theta)}{1+\exp(\theta)}, \frac{1}{1+\exp(\theta)}\right)$$

$$J(\theta) = 100 \cdot \pi_\theta[1] + 1 \cdot \pi_\theta[2]$$



Gradient: $\nabla_\theta J(\theta) = \dfrac{99\exp(\theta)}{(1+\exp(\theta))^2}$

Exact PG: $\theta^{k+1} = \theta^k + \eta \dfrac{99\exp(\theta^k)}{(1+\exp(\theta^k))^2}$

i.e., vanilla GA moves to $\theta = \infty$ with smaller and smaller steps, since $\nabla_\theta J(\theta) \to 0$ as $\theta \to \infty$

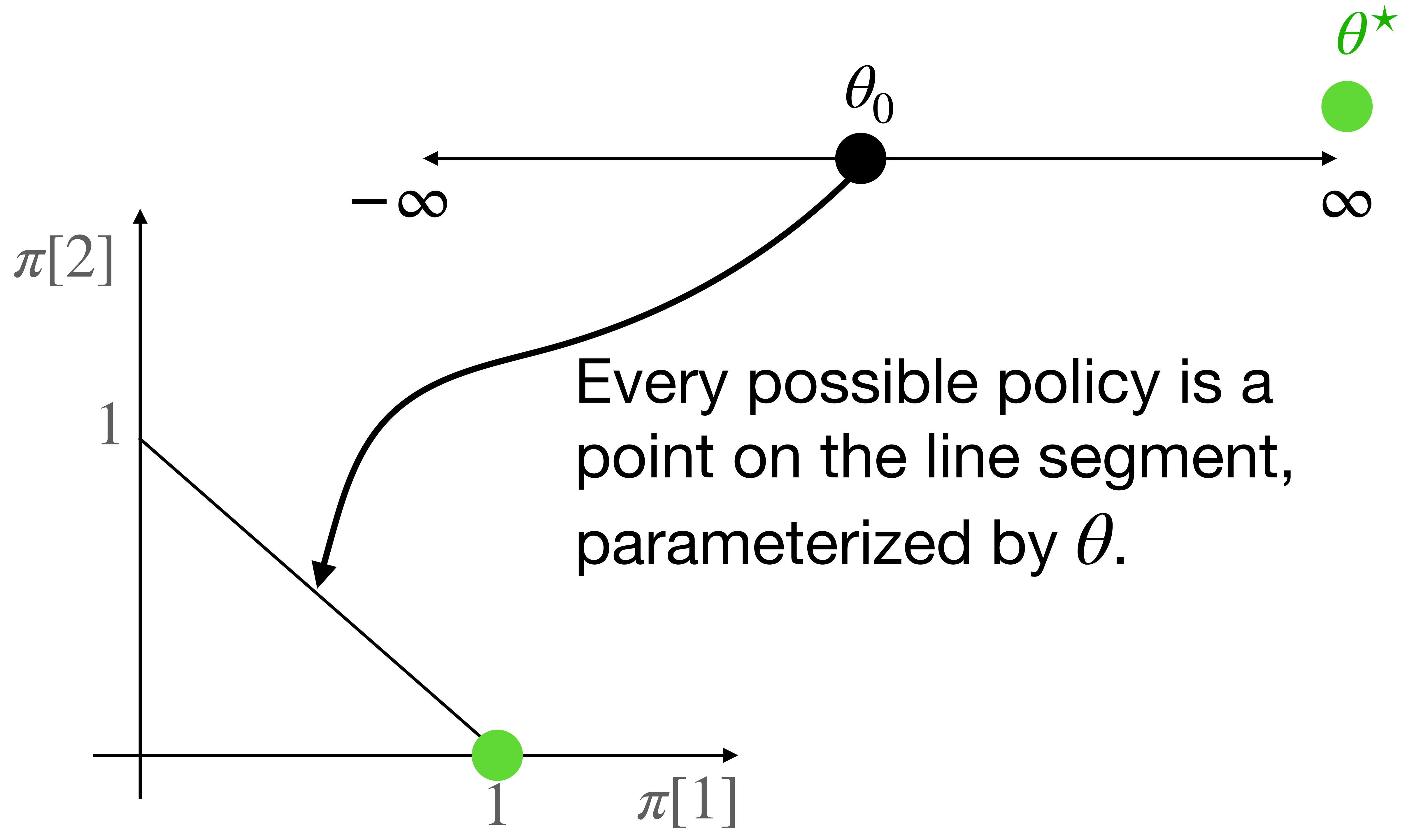
Fisher information scalar: $F_\theta = \dfrac{\exp(\theta)}{(1+\exp(\theta))^2}$

NPG: $\theta^{k+1} = \theta^k + \eta \dfrac{\nabla_\theta J(\theta^k)}{F_{\theta^k}}$

Example of Natural Gradient on 1-d problem: 2 actions, 1 state

$$(\pi_\theta[1], \pi_\theta[2]) := \left(\frac{\exp(\theta)}{1+\exp(\theta)}, \frac{1}{1+\exp(\theta)} \right)$$

$$J(\theta) = 100 \cdot \pi_\theta[1] + 1 \cdot \pi_\theta[2]$$



Gradient: $\nabla_\theta J(\theta) = \dfrac{99\exp(\theta)}{(1+\exp(\theta))^2}$

Exact PG: $\theta^{k+1} = \theta^k + \eta \dfrac{99\exp(\theta^k)}{(1+\exp(\theta^k))^2}$

i.e., vanilla GA moves to $\theta = \infty$ with smaller and smaller steps, since $\nabla_\theta J(\theta) \to 0$ as $\theta \to \infty$

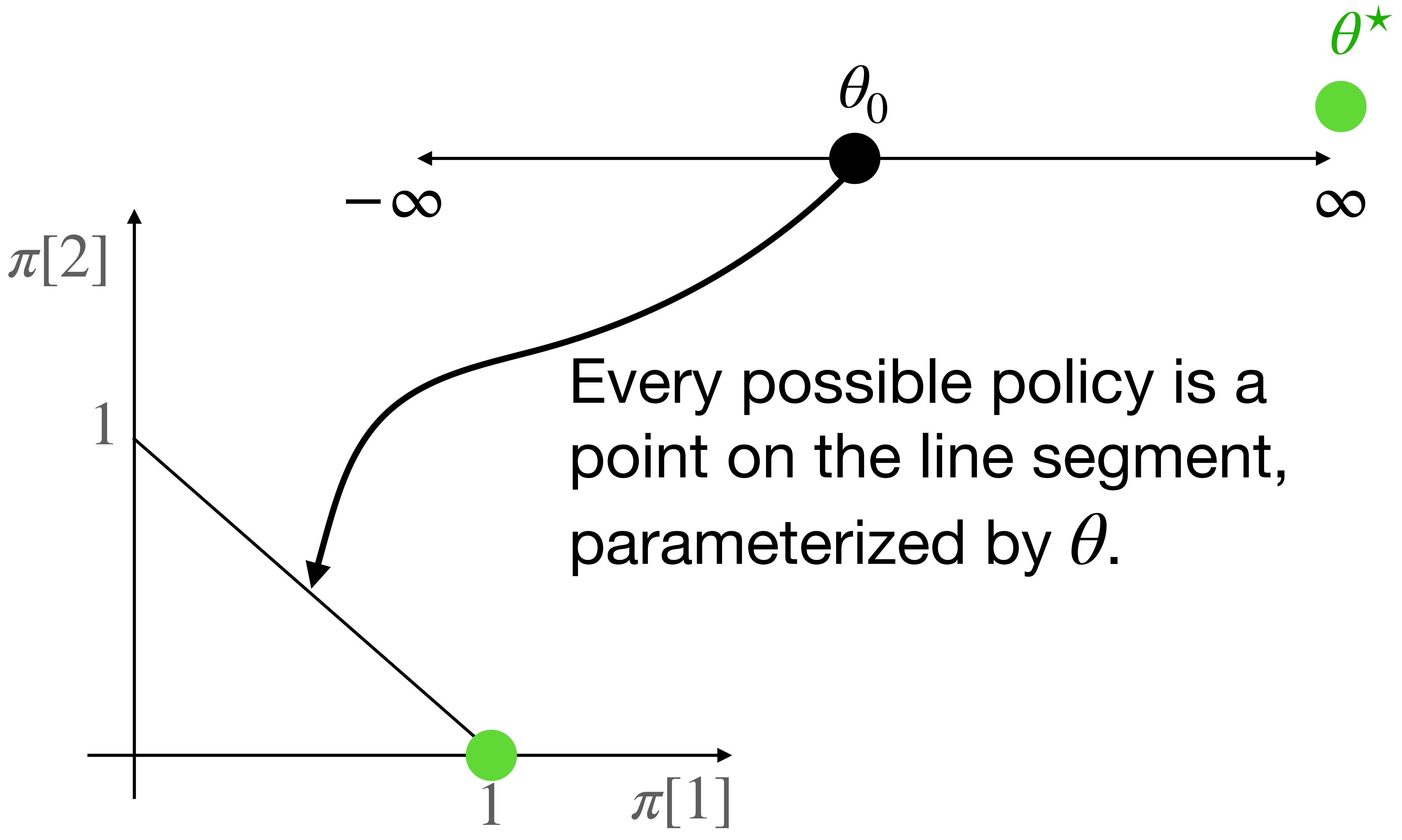
Fisher information scalar: $F_\theta = \dfrac{\exp(\theta)}{(1+\exp(\theta))^2}$

NPG: $\theta^{k+1} = \theta^k + \eta \dfrac{\nabla_\theta J(\theta^k)}{F_{\theta^k}} = \theta_t + \eta \cdot 99$

Example of Natural Gradient on 1-d problem: 2 actions, 1 state

$$(\pi_\theta[1], \pi_\theta[2]) := \left(\frac{\exp(\theta)}{1 + \exp(\theta)}, \frac{1}{1 + \exp(\theta)} \right)$$

$$J(\theta) = 100 \cdot \pi_\theta[1] + 1 \cdot \pi_\theta[2]$$



Gradient: $\nabla_\theta J(\theta) = \dfrac{99 \exp(\theta)}{(1 + \exp(\theta))^2}$

Exact PG: $\theta^{k+1} = \theta^k + \eta \dfrac{99 \exp(\theta^k)}{(1 + \exp(\theta^k))^2}$

i.e., vanilla GA moves to $\theta = \infty$ with smaller and smaller steps, since $\nabla_\theta J(\theta) \to 0$ as $\theta \to \infty$

Fisher information scalar: $F_\theta = \dfrac{\exp(\theta)}{(1 + \exp(\theta))^2}$

NPG: $\theta^{k+1} = \theta^k + \eta \dfrac{\nabla_\theta J(\theta^k)}{F_{\theta^k}} = \theta_t + \eta \cdot 99$

NPG moves to $\theta = \infty$ much more quickly (for a fixed η)