Trust Region Policy Optimization & The Natural Policy Gradient

Lucas Janson and Sham Kakade

CS/Stat 184: Introduction to Reinforcement Learning
Fall 2023

Today

Ethics Lecture Mon!

- Recap
- The Performance Difference Lemma
- Algorithms:
  - ~~Conservative Policy Iteration (CPI)~~
  - Trust Region Policy Optimization (TRPO)
  - The Natural Policy Gradient (NPG)
  - Proximal Policy Optimization (PPO)

Recap++

Optimization Objective

- Consider a parameterize 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

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

- 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:

$$J(\theta) \ = E_{\tau \sim \rho_\theta}\Big[R(\tau)\Big]$$

- The REINFORCE Policy Gradient expression:

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

Proof

- From the likelihood ratio method, we have:

$$\nabla_\theta J(\theta) := \mathbb{E}_{\tau \sim \rho_\theta} \left[\nabla_\theta \ln \rho_\theta(\tau)\ R(\tau) \right]$$

- We have:

$$\nabla_\theta \ln \rho_\theta(\tau) = \nabla_\theta \Big(\ln \mu(s_0) + \ln \pi_\theta(a_0 \,|\, s_0) + \ln P(s_1 \,|\, s_0, a_0) + \ldots \Big)$$

$$= \nabla_\theta \Big(\ln \pi_\theta(a_0 \,|\, s_0) + \ln \pi_\theta(a_1 \,|\, s_1) \ldots \Big)$$

$$= \left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \,|\, s_h) \right)$$

PG with REINFORCE:

1. Initialize θ_0, parameters: $\eta^1, \eta^2, \ldots$
2. For k = 0, … :
   1. Obtain a trajectory $\tau \sim \rho_{\theta^k}$

      Set $\widetilde{\nabla}_\theta J(\theta^k) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta^k}(a_h | s_h) R(\tau)$

   2. Update: $\theta^{k+1} = \theta^k + \eta^k \widetilde{\nabla}_\theta J(\theta^k)$

Other PG formulas
(that are lower variance for sampling)

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

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

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

Intuition: Change 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, $b_h : S \to R$, we have:

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

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

This is (basically) the method of control variates.

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

The Advantage Function (finite horizon)

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

The Advantage Function (finite horizon)

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

- 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)

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

- The Advantage function is defined as:

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

- We have that:

$$E_{a\sim\pi(\cdot|s)}\left[A_h^\pi(s,a) \middle| s, h\right] = \sum_a \pi(a \mid s) A_h^\pi(s,a) = \ ??$$

The Advantage Function (finite horizon)

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

- The Advantage function is defined as:

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

- We have that:

$$E_{a\sim\pi(\cdot|s)}\left[A_h^\pi(s,a) \middle| s, h\right] = \sum_a \pi(a\,|\,s) A_h^\pi(s,a) = \ ??$$

$\forall s, a$

- What do we know about $A_h^{\pi^\star}(s,a)$?

$A_h^\pi(s,a) \leq 0$

iff π is optimal.

The Advantage Function (finite horizon)

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

- The Advantage function is defined as:

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

- We have that:

$$E_{a\sim\pi(\cdot|s)}\left[A_h^\pi(s,a) \middle| s, h\right] = \sum_a \pi(a \mid s) A_h^\pi(s,a) = \ ??$$

- What do we know about $A_h^{\pi^\star}(s,a)$?

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

The Advantage-based PG:

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

The Advantage-based PG:

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

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

The Advantage-based PG:

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

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

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

The Advantage-based PG:

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

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

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

(M=1) PG with a Learned Baseline:

(M=1) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$

(M=1) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For k = 0, … :

(M=1) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For k = 0, … :
    1. Sup. Learning: Using N trajectories sampled under π_{θ^k}, estimate a baseline $\widetilde{b}_h$
       $\widetilde{b}(s) \approx V_h^{\theta^k}(s)$

(M=1) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For k = 0, … :
    1. Sup. Learning: Using N trajectories sampled under π_{θ^k}, estimate a baseline $\widetilde{b}_h$
       $\widetilde{b}(s) \approx V_h^{\theta^k}(s)$
    2. Obtain a trajectory $\tau \sim \rho_{\theta^k}$
       Set $\widetilde{\nabla}_\theta J(\theta^k) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta^k}(a_h | s_h)\left(R_h(\tau) - \widetilde{b}(s_h)\right)$

(M=1) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For k = 0, … :
   1. Sup. Learning: Using N trajectories sampled under π_{θ^k}, estimate a baseline $\widetilde{b}_h$
      $\widetilde{b}(s) \approx V_h^{\theta^k}(s)$
   2. Obtain a trajectory $\tau \sim \rho_{\theta^k}$
      Set $\widetilde{\nabla}_\theta J(\theta^k) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta^k}(a_h \mid s_h)\left(R_h(\tau) - \widetilde{b}(s_h)\right)$
   3. Update: $\theta^{k+1} = \theta^k + \eta^k \widetilde{\nabla}_\theta J(\theta^k)$

(M=1) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For k = 0, … :
    1. Sup. Learning: Using N trajectories sampled under π_{θ^k}, estimate a baseline $\widetilde{b}_h$
       $\widetilde{b}(s) \approx V_h^{\theta^k}(s)$
    2. Obtain a trajectory $\tau \sim \rho_{\theta^k}$
       Set $\widetilde{\nabla}_\theta J(\theta^k) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta^k}(a_h | s_h) \left(R_h(\tau) - \widetilde{b}(s_h) \right)$
    3. Update: $\theta^{k+1} = \theta^k + \eta^k \widetilde{\nabla}_\theta J(\theta^k)$

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

(minibatch) PG with a Learned Baseline:

(minibatch) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For k = 0, … :
    1. Sup. Learning: Using N trajectories sampled under π_{θ^k}, estimate a baseline $\widetilde{b}_h$
       $\widetilde{b}(s) \approx V_h^{\theta^k}(s)$

(minibatch) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For k = 0, … :
    1. Sup. Learning: Using N trajectories sampled under π_{θ^k}, estimate a baseline $\widetilde{b}_h$
       $\widetilde{b}(s) \approx V_h^{\theta^k}(s)$
    2. Obtain M trajectories $\tau_1, \ldots \tau_M \sim \rho_{\theta^k}$

       Set $\widetilde{\nabla}_\theta J(\theta^k) = \frac{1}{M} \sum_{m=1}^{M} \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta^k}(a_h^m \mid s_h^m) \Big(R_h(\tau^m) - \widetilde{b}(s_h) \Big)$

(minibatch) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For k = 0, … :
    1. Sup. Learning: Using N trajectories sampled under π_{θ^k}, estimate a baseline $\widetilde{b}_h$
       $\widetilde{b}(s) \approx V_h^{\theta^k}(s)$
    2. Obtain M trajectories $\tau_1, \ldots \tau_M \sim \rho_{\theta^k}$

       Set $\widetilde{\nabla}_\theta J(\theta^k) = \frac{1}{M} \sum_{m=1}^{M} \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta^k}(a_h^m | s_h^m) \left(R_h(\tau^m) - \widetilde{b}(s_h) \right)$

       $\hat{A}_h(s_h, a_h)$
    3. Update: $\theta^{k+1} = \theta^k + \eta^k \widetilde{\nabla}_\theta J(\theta^k)$

Today:

Today

- Recap++
- ✓ Softmax Example
- The Performance Difference Lemma
- Algorithms:
  - Trust Region Policy Optimization (TRPO)
  - The Natural Policy Gradient (NPG)
  - Proximal Policy Optimization (PPO)

Policy Parameterizations

Recall that we consider parameterized policy $\pi_\theta(\cdot \mid s) \in \Delta(A), \forall s$

1. Softmax linear Policy

Feature vector $\phi(s, a) \in \mathbb{R}^d$, and parameter $\theta \in \mathbb{R}^d$

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s, a))}{\sum_{a'} \exp(\theta^\top \phi(s, a'))}$$

2. Neural Policy:

Neural network $f_\theta : S \times A \mapsto \mathbb{R}$

$$\pi_\theta(a \mid s) = \frac{\exp(f_\theta(s, a))}{\sum_{a'} \exp(f_\theta(s, a'))}$$

Softmax Policy Properties

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s, a))}{\sum_{a'} \exp(\theta^\top \phi(s, a'))}$$

Softmax Policy Properties

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s, a))}{\sum_{a'} \exp(\theta^\top \phi(s, a'))}$$

Two properties (see HW):

Softmax Policy Properties

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s, a))}{\sum_{a'} \exp(\theta^\top \phi(s, a'))}$$

Two properties (see HW):

- More probable actions have features which align with θ. Precisely,
  $\pi_\theta(a \mid s) \geq \pi_\theta(a' \mid s)$ if and only if $\theta^\top \phi(s, a) \geq \theta^\top \phi(s, a')$

Softmax Policy Properties

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$$

Two properties (see HW):

- More probable actions have features which align with θ. Precisely,

  $\pi_\theta(a \mid s) \geq \pi_\theta(a' \mid s)$ if and only if $\theta^\top \phi(s,a) \geq \theta^\top \phi(s,a')$

- The gradient of the log policy is:

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

Softmax Policy Properties

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$$

Two properties (see HW):

- More probable actions have features which align with θ. Precisely,
  $\pi_\theta(a \mid s) \geq \pi_\theta(a' \mid s)$ if and only if $\theta^\top \phi(s,a) \geq \theta^\top \phi(s,a')$

- The gradient of the log policy is:
  $\nabla_\theta \log(\pi_\theta(a \mid s)) = \phi(s,a) - \mathbb{E}_{a' \sim \pi_\theta(\cdot \mid s)}[\phi(s,a')]$

- We have:

$A_h^{\pi_\theta}(s_h, a_h)$

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

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

Today

- Recap++
- Softmax Example
- ✓ The Performance Difference Lemma
- Algorithms:
  - Trust Region Policy Optimization (TRPO)
  - The Natural Policy Gradient (NPG)
  - Proximal Policy Optimization (PPO)

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)

$\rho_{\pi,s}(\tau)$

The Performance Difference Lemma (PDL)

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

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

The Performance Difference Lemma (PDL)

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

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

Comments:

The Performance Difference Lemma (PDL)

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

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

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 π. (we are making the starting distribution explicit now).
- For any two policies π and $\widetilde{\pi}$ and any state s,

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

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 π. (we are making the starting distribution explicit now).
- For any two policies π and $\widetilde{\pi}$ and any state s,

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

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 Approximate Policy Iteration (API)

Back to Approximate Policy Iteration (API)

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

Back to Approximate Policy Iteration (API)

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?
- Suppose at some state s, π^{k+1} choose an action which has a negative advantage for π^k.

Back to Approximate Policy Iteration (API)

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?
- Suppose at some state s, π^{k+1} choose an action which has a negative advantage for π^k.
  - Since $\widetilde{A}^k(s, a, h) \approx A_h^{\pi^k}(s, a, h)$, we expect some error.

Back to Approximate Policy Iteration (API)

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?
- Suppose at some state s, π^{k+1} choose an action which has a negative advantage for π^k.
  - Since $\widetilde{A}^k(s, a, h) \approx A_h^{\pi^k}(s, a, h)$, we expect some error.
  - In the worst case, let us consider the most negative advantage:

$$\Delta_\infty := \min_{s \in S} A_h^{\pi^k}(s, \pi^{k+1}(s))$$

Back to Approximate Policy Iteration (API)

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?
- Suppose at some state s, π^{k+1} choose an action which has a negative advantage for π^k.
  - Since $\widetilde{A}^k(s, a, h) \approx A_h^{\pi^k}(s, a, h)$, we expect some error.
  - In the worst case, let us consider the most negative advantage:

$$\Delta_\infty := \min_{s \in S} A_h^{\pi^k}(s, \pi^{k+1}(s))$$

  - Here, if $\Delta_\infty < 0$, it is possible that degradation may occur:

Back to Approximate Policy Iteration (API)

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?
- Suppose at some state s, π^{k+1} choose an action which has a negative advantage for π^k.
  - Since $\widetilde{A}^k(s, a, h) \approx A_h^{\pi^k}(s, a, h)$, we expect some error.
  - In the worst case, let us consider the most negative advantage:

$$\Delta_\infty := \min_{s \in S} A_h^{\pi^k}(s, \pi^{k+1}(s))$$

  - Here, if $\Delta_\infty < 0$, it is possible that degradation may occur:

$$V^{\pi^{k+1}}(s_0) \geq V^{\pi^k}(s_0) - H \cdot |\Delta_\infty|$$

Back to Approximate Policy Iteration (API)

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?
- Suppose at some state s, π^{k+1} choose an action which has a negative advantage for π^k.
  - Since $\widetilde{A}^k(s, a, h) \approx A_h^{\pi^k}(s, a, h)$, we expect some error.
  - In the worst case, let us consider the most negative advantage:

$$\Delta_\infty := \min_{s \in S} A_h^{\pi^k}(s, \pi^{k+1}(s))$$

  - Here, if $\Delta_\infty < 0$, it is possible that degradation may occur:

$$V^{\pi^{k+1}}(s_0) \geq V^{\pi^k}(s_0) - H \cdot |\Delta_\infty|$$

Proof sketch:

Back to Approximate Policy Iteration (API)

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?
- Suppose at some state s, π^{k+1} choose an action which has a negative advantage for π^k.
  - Since $\widetilde{A}^k(s, a, h) \approx A_h^{\pi^k}(s, a, h)$, we expect some error.
  - In the worst case, let us consider the most negative advantage:

$$\Delta_\infty := \min_{s \in S} A_h^{\pi^k}(s, \pi^{k+1}(s))$$

  - Here, if $\Delta_\infty < 0$, it is possible that degradation may occur:

$$V^{\pi^{k+1}}(s_0) \geq V^{\pi^k}(s_0) - H \cdot |\Delta_\infty|$$

Proof sketch:

- Fitted PI does not enforce that the trajectory distributions, ρ_{π^k} and $\rho_{\pi^{k+1}}$, be close to each other.

Back to Approximate Policy Iteration (API)

- Suppose π^k gets updated to π^{k+1}. How much worse could π^{k+1} be?
- Suppose at some state s, π^{k+1} choose an action which has a negative advantage for π^k.
  - Since $\widetilde{A}^k(s, a, h) \approx A_h^{\pi^k}(s, a, h)$, we expect some error.
  - In the worst case, let us consider the most negative advantage:

$$\Delta_\infty := \min_{s \in S} A_h^{\pi^k}(s, \pi^{k+1}(s))$$

  - Here, if $\Delta_\infty < 0$, it is possible that degradation may occur:

$$V^{\pi^{k+1}}(s_0) \geq V^{\pi^k}(s_0) - H \cdot |\Delta_\infty|$$

Proof sketch:

- Fitted PI does not enforce that the trajectory distributions, ρ_{π^k} and $\rho_{\pi^{k+1}}$, be close to each other.
- Suppose the $\rho_{\pi^{k+1}}$ has full support on these worst case states s (i.e. we get trapped at this state where we made a bad choice).

Today

- Recap++
- Softmax Example
- The Performance Difference Lemma
- Algorithms:
    - ✓ Trust Region Policy Optimization (TRPO)
    - The Natural Policy Gradient (NPG)
    - Proximal Policy Optimization (PPO)

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. Init π_0
2. For $k = 0, \ldots K$:
   try to approximately solve:

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

s.t. ρ_θ is "close" to ρ_{θ^k}

3. Return π_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. Init π_0
2. For $k = 0, \ldots K$:
   try to approximately solve:

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

s.t. ρ_θ is "close" to ρ_{θ^k}

3. Return π_K

- How should we define "close"?

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:

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

Examples:

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

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 being 0 if and only if $P = Q$

Trust Region Policy Optimization (TRPO)

1. Init π_0
2. For $k = 0, \ldots K$:

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

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

3. Return π_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 staring policy π_0, samples size M
2. For $k = 0, \ldots K$:
    1. [A-Evaluation Subroutine]

       Using M sampled trajectories, $\tau_1, \ldots \tau_M \sim \rho_{\pi_k}$,

       $\widetilde{A}_k(s, a) \approx A_h^{\pi_k}(s, a)$
    2. Solve the following optimization problem to obtain π_{k+1}:

$$\max_{\theta} \sum_{m=1}^{M} \sum_{h=0}^{H-1} \mathbb{E}_{a \sim \pi_\theta(s_h^m)} \widetilde{A}_k(s_h^m, a)$$

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

Today

- Recap++
- Softmax Example
- The Performance Difference Lemma
- Algorithms:
  - Trust Region Policy Optimization (TRPO)
  - ✓ The Natural Policy Gradient (NPG)
  - Proximal Policy Optimization (PPO)

TRPO is locally equivalent to the NPG

TRPO at iteration k:

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

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

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

TRPO is locally equivalent to the NPG

TRPO at iteration k:

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

→ First-order Taylor expansion at θ^k

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

→ second-order Taylor expansion at θ^k

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

TRPO is locally equivalent to the NPG

TRPO at iteration k:

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

→ First-order Taylor expansion at θ^k

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

→ second-order Taylor expansion at θ^k

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

TRPO is locally equivalent to the NPG

TRPO at iteration k:

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

→ First-order Taylor expansion at θ^k

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

→ second-order Taylor expansion at θ^k

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

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

TRPO is locally equivalent to the NPG

TRPO at iteration k:

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

→ First-order Taylor expansion at θ^k

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

→ second-order Taylor expansion at θ^k

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

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

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

TRPO is locally equivalent to the NPG

TRPO at iteration k:

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

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

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

→ First-order Taylor expansion at θ^k

→ second-order Taylor expansion at θ^k

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

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

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

NPG: A "leading order" equivalent program to TRPO:

1. Init π_0
2. For $k = 0, \ldots K$:

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

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

3. Return π_K

NPG: A "leading order" equivalent program to TRPO:

1. Init π_0
2. For $k = 0, \ldots K$:

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

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

3. Return π_K

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

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

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

There is a closed form update:

1. Init π_0
2. For $k = 0, \ldots K$:

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

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

3. Return π_K

There is a closed form update:

1. Init π_0
2. For $k = 0, \ldots K$:

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

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

3. Return π_K

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

There is a closed form update:

1. Init π_0
2. For $k = 0, \ldots K$:

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

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

3. Return π_K

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

Indeed this gives us:

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

There is a closed form update:

1. Init π_0
2. For $k = 0, \ldots K$:

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

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

3. Return π_K

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

Indeed this gives us:

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

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

Summary:

1. Variance Reduction: with baselines
2. Perf. Diff Lemma/ TRPO/ NPG

Attendance:
bit.ly/3RcTC9T

KL divergence

Feedback:
bit.ly/3RHtlxy

An Implementation: Sample Based NPG

1. Init π_0
2. For $k = 0, \ldots K$:
   - Estimate PG $\nabla_\theta J(\pi_{\theta^k})$
   - Estimate Fisher info-matrix: $F_{\theta^k} = \mathbb{E}_{\tau \sim \rho_{\theta^k}} \left[\sum_{h=0}^{H-1} \nabla \ln \pi_{\theta^k}(a_h | s_h) \left(\nabla \ln \pi_{\theta^k}(a_h | s_h) \right)^\top \right]$
   - Natural Gradient Ascent: $\theta^{k+1} = \theta^k + \eta \widehat{F_{\theta^k}}^{-1} \widehat{\nabla_\theta J(\pi_{\theta^k})}$
3. Return π_K

An Implementation: Sample Based NPG

1. Init π_0
2. For $k = 0, \ldots K$:
   - Estimate PG $\nabla_\theta J(\pi_{\theta^k})$
   - Estimate Fisher info-matrix: $F_{\theta^k} = \mathbb{E}_{\tau \sim \rho_{\theta^k}} \left[\sum_{h=0}^{H-1} \nabla \ln \pi_{\theta^k}(a_h | s_h) \left(\nabla \ln \pi_{\theta^k}(a_h | s_h) \right)^\top \right]$
   - Natural Gradient Ascent: $\theta^{k+1} = \theta^k + \eta \widehat{F_{\theta^k}}^{-1} \widehat{\nabla_\theta J(\pi_{\theta^k})}$
3. Return π_K

(We will implement it in HW4 on Cartpole)

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

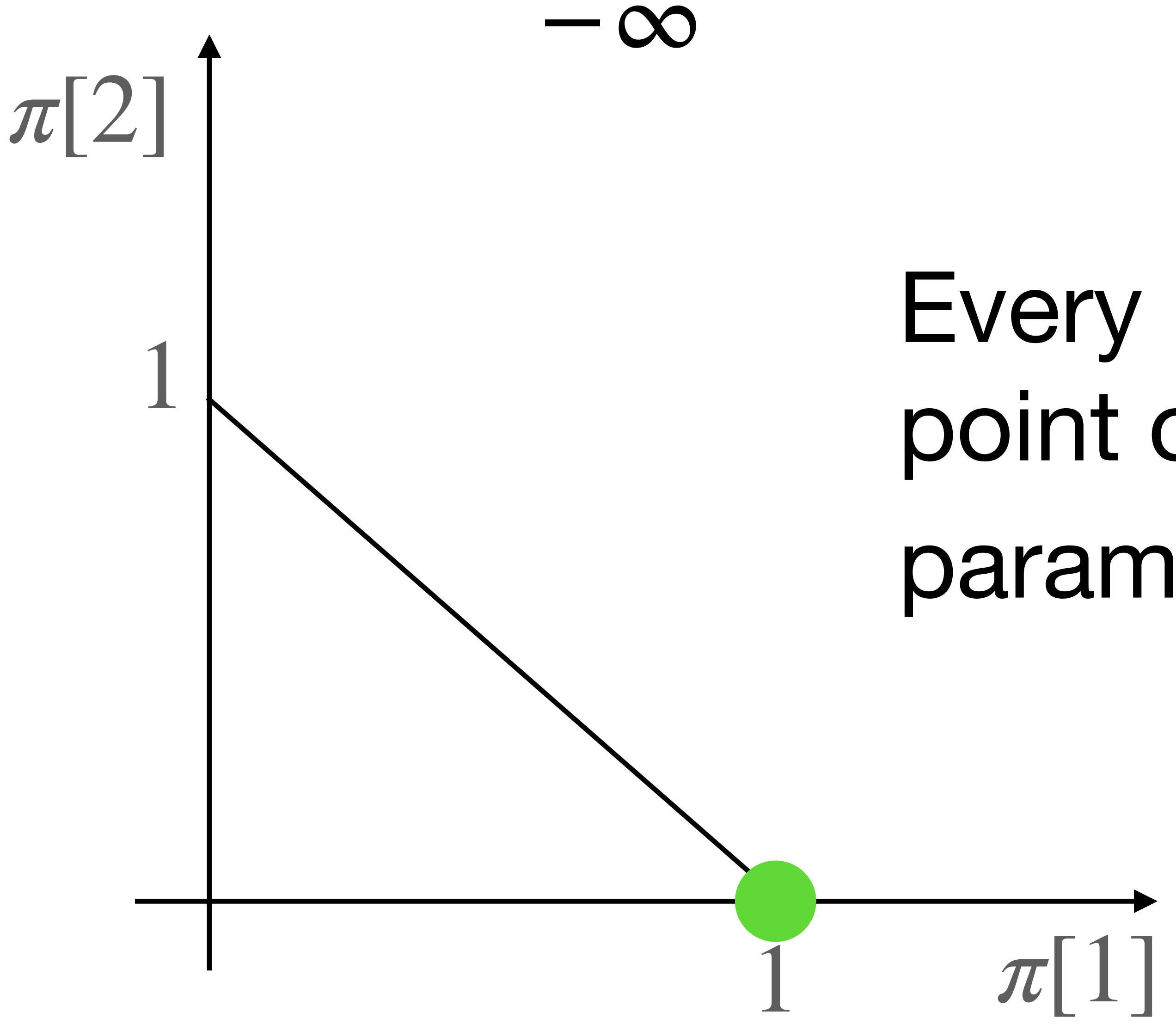
$$(\pi_\theta[1], \pi_\theta[2]) := \left(\frac{\exp(\theta)}{1+\exp(\theta)}, \frac{1}{1+\exp(\theta)} \right)$$

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

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

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

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

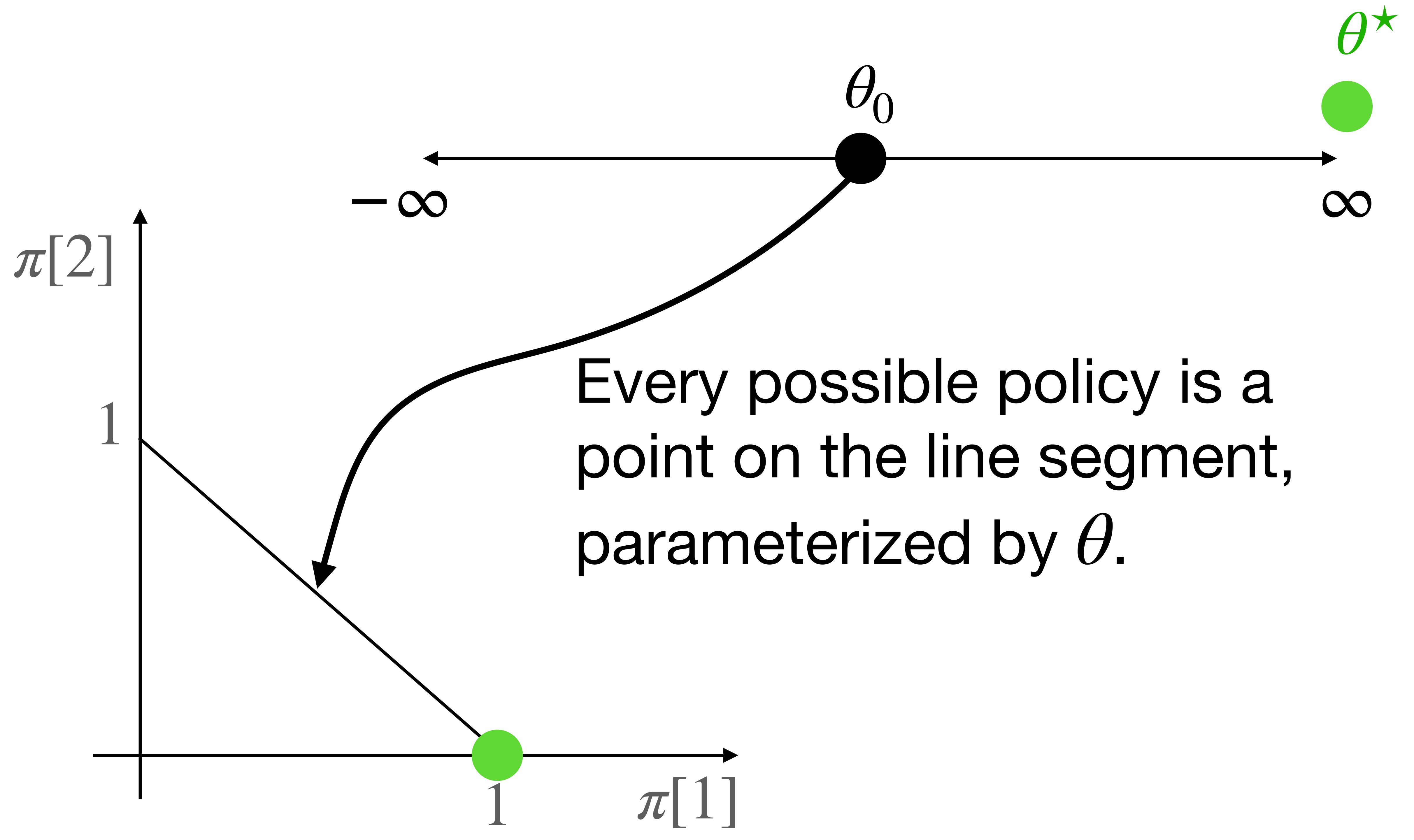


Every possible policy is a point on the line segment, parameterized by θ.

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

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

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



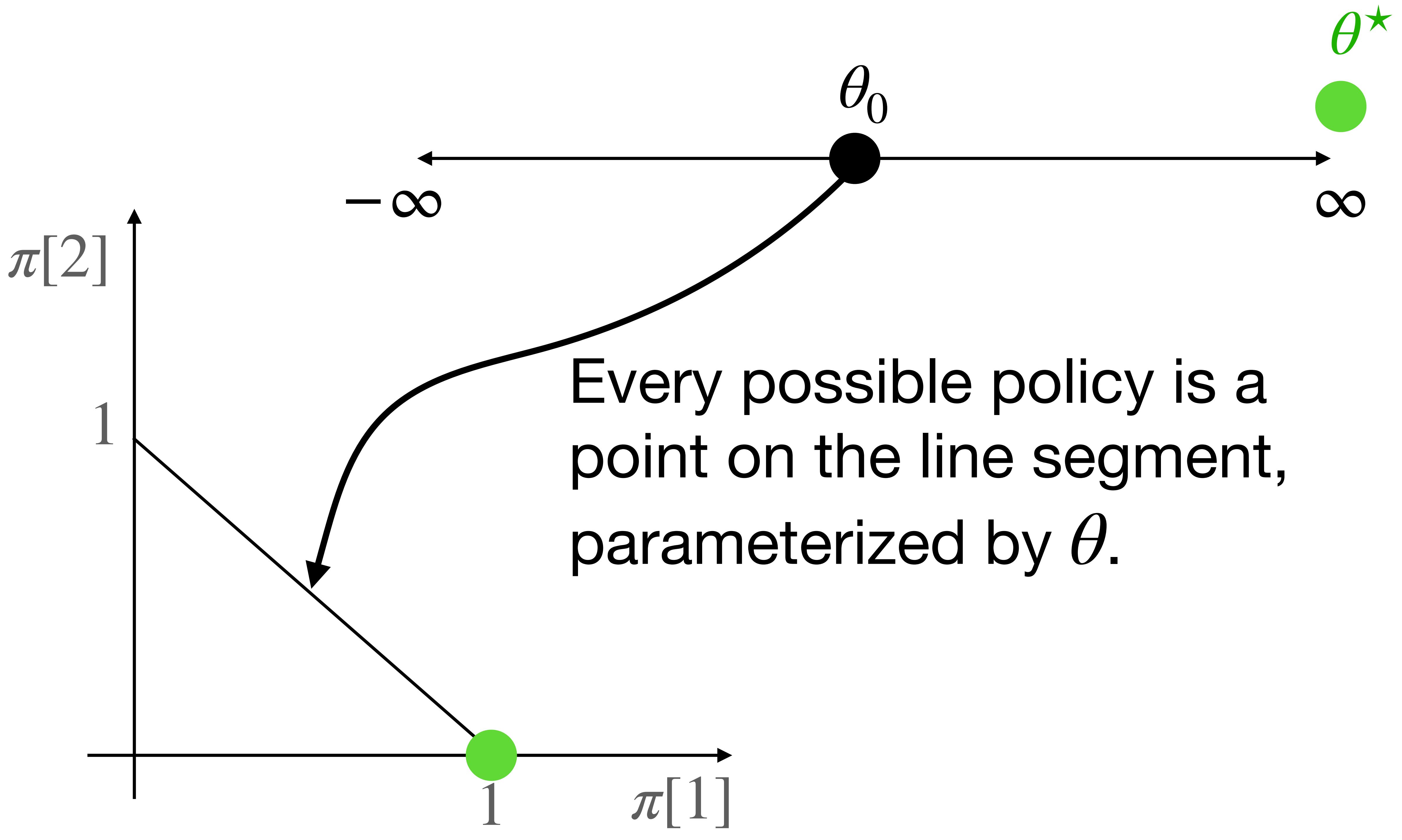
Every possible policy is a point on the line segment, parameterized by θ.

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

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



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

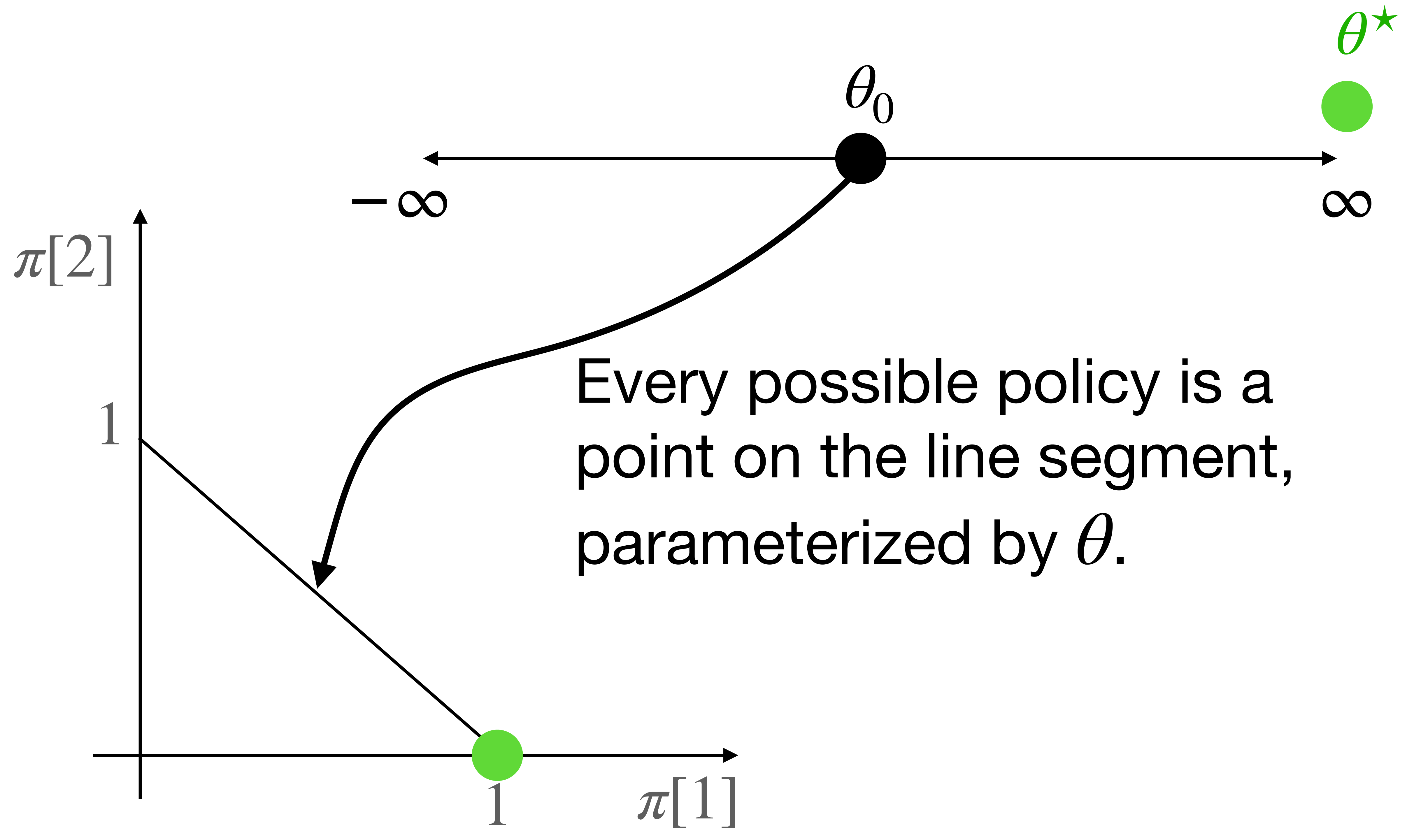
Example of Natural Gradient on 1-d problem: 2 actions, 1 state

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

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

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

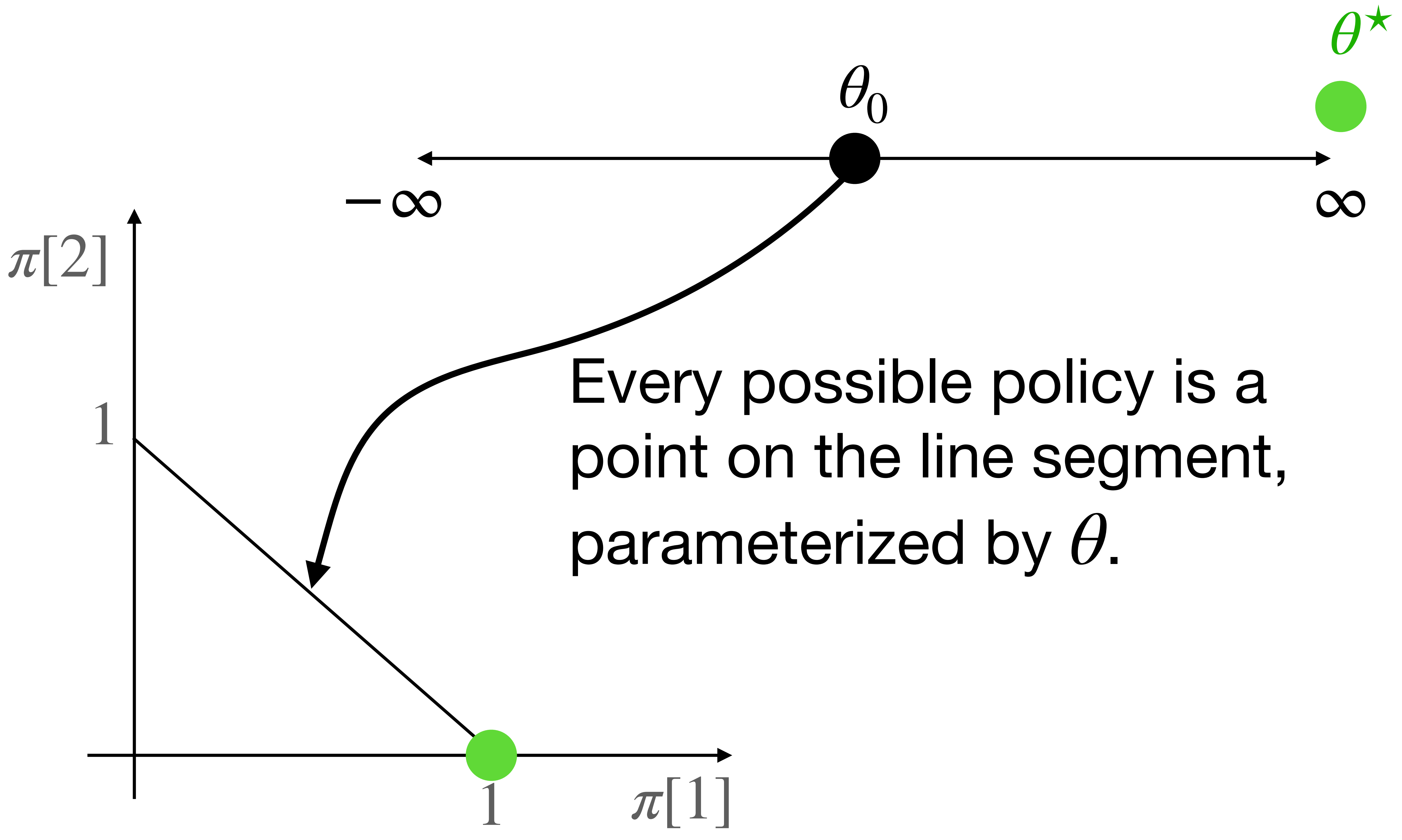
Exact PG: $\theta^{k+1} = \theta^k + \eta \dfrac{99\exp(\theta^k)}{(1+\exp(\theta^k))^2}$



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

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

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



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

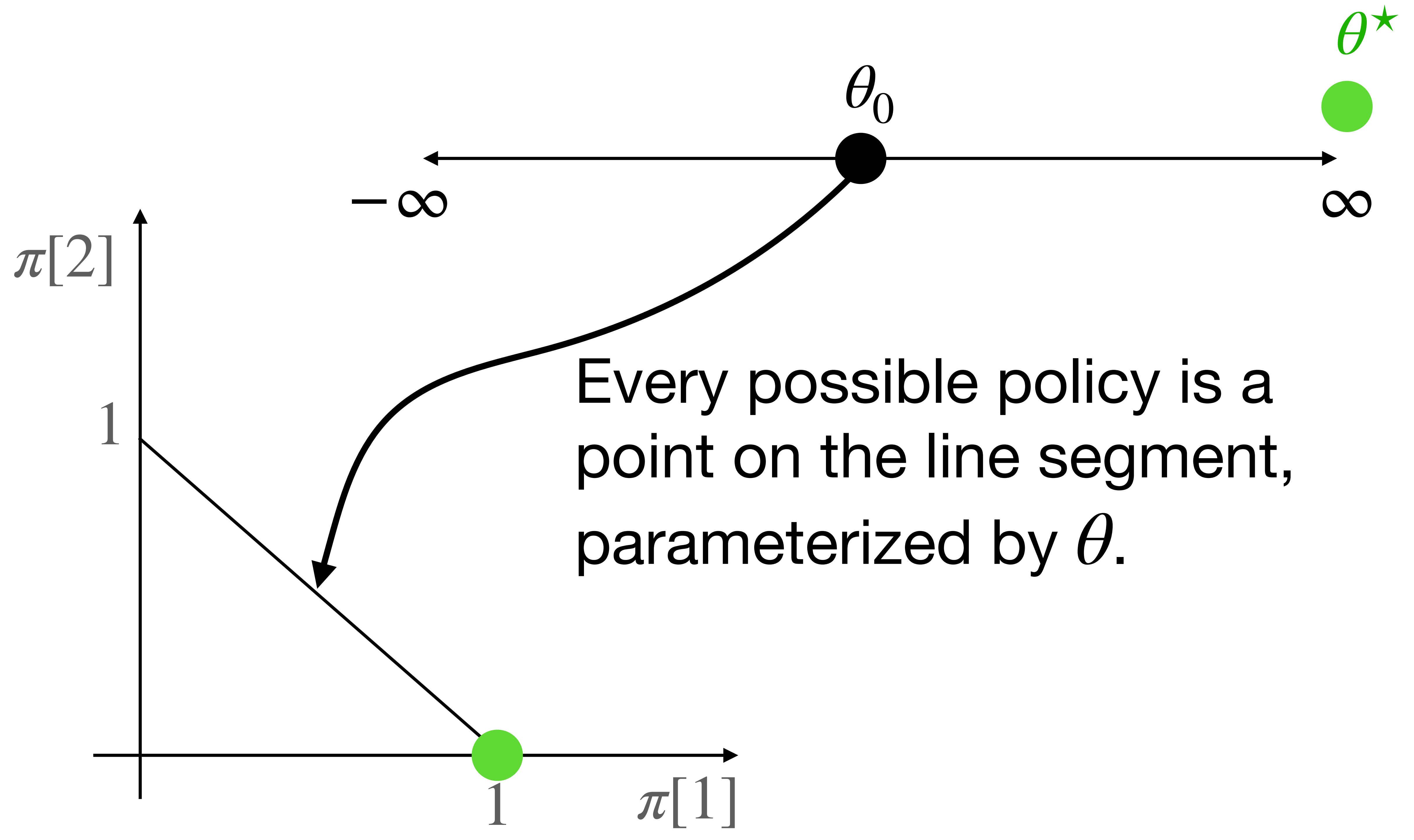
Exact PG: $\theta^{k+1} = \theta^k + \eta \dfrac{99\exp(\theta^k)}{(1+\exp(\theta^k))^2}$

i.e., vanilla GA moves to $\theta = \infty$ with smaller and smaller steps, since $J'(\theta) \to 0$ 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: $J'(\theta) = \dfrac{99\exp(\theta)}{(1+\exp(\theta))^2}$

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

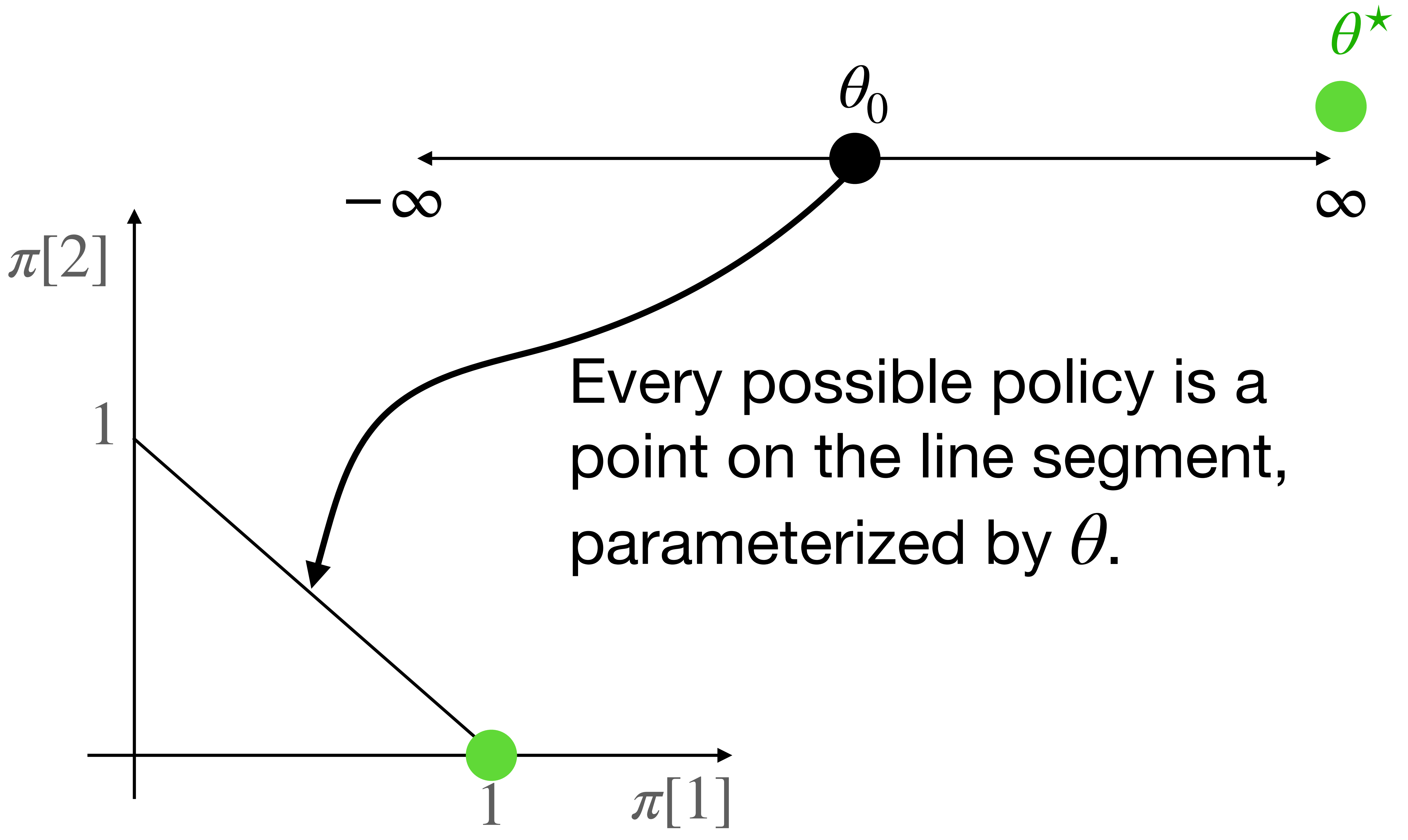
i.e., vanilla GA moves to $\theta = \infty$ with smaller and smaller steps, since $J'(\theta) \to 0$ as $\theta \to \infty$

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

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 $J'(\theta) \to 0$ as $\theta \to \infty$

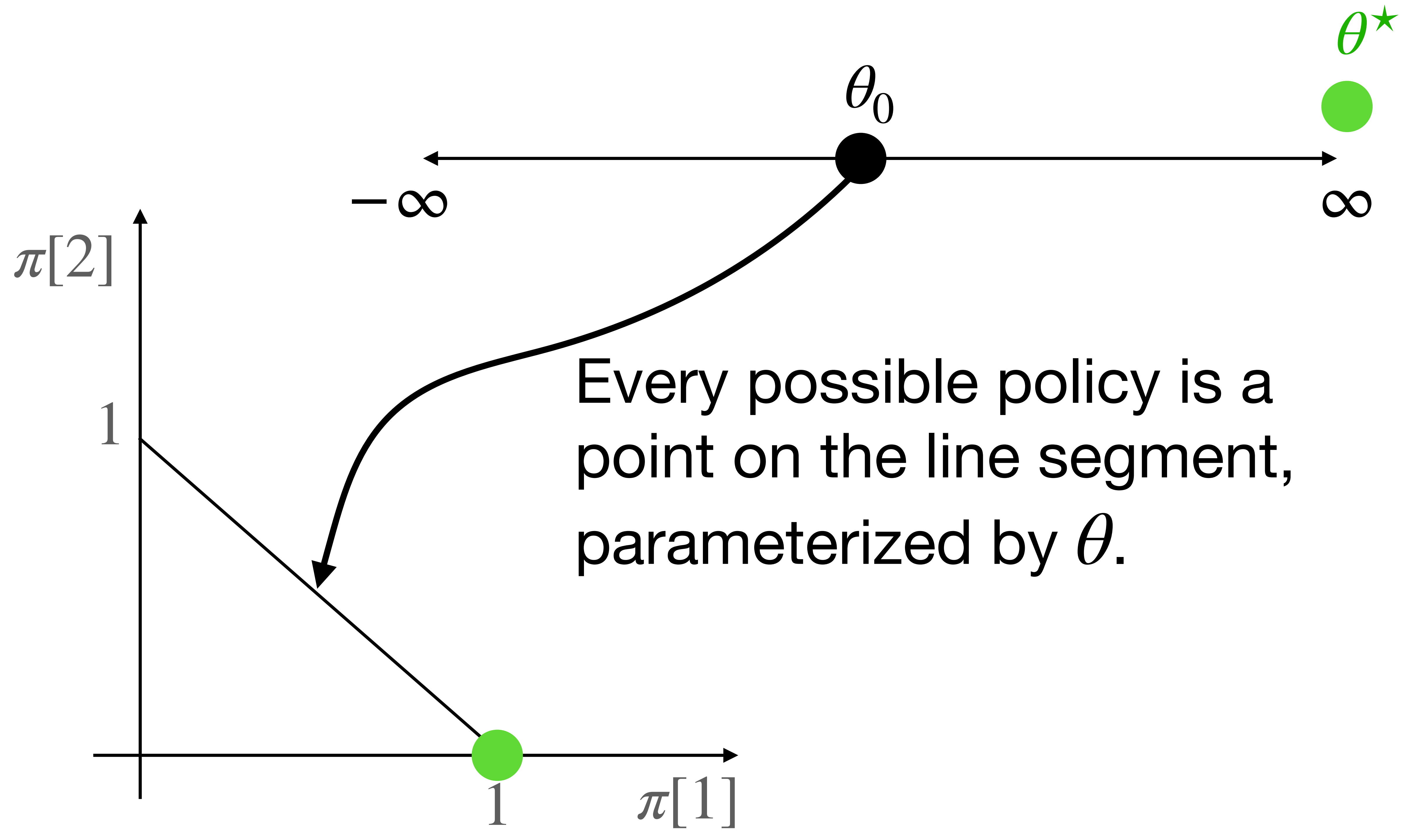
Fisher information scalar: $F_\theta = \dfrac{\exp(\theta)}{(1+\exp(\theta))^2}$

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

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

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 $J'(\theta) \to 0$ as $\theta \to \infty$

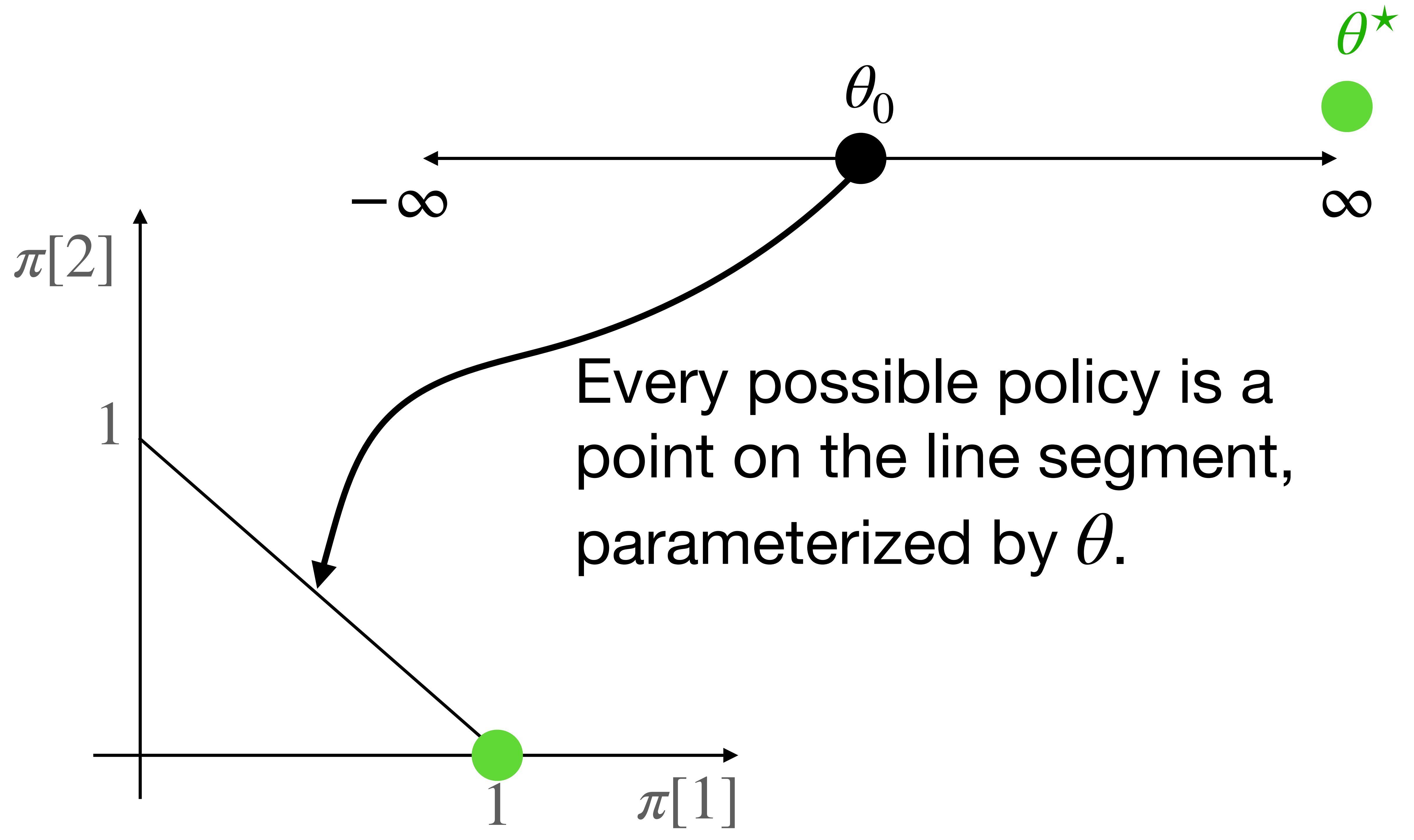
Fisher information scalar: $F_\theta = \dfrac{\exp(\theta)}{(1+\exp(\theta))^2}$

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

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

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 $J'(\theta) \to 0$ as $\theta \to \infty$

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

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

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