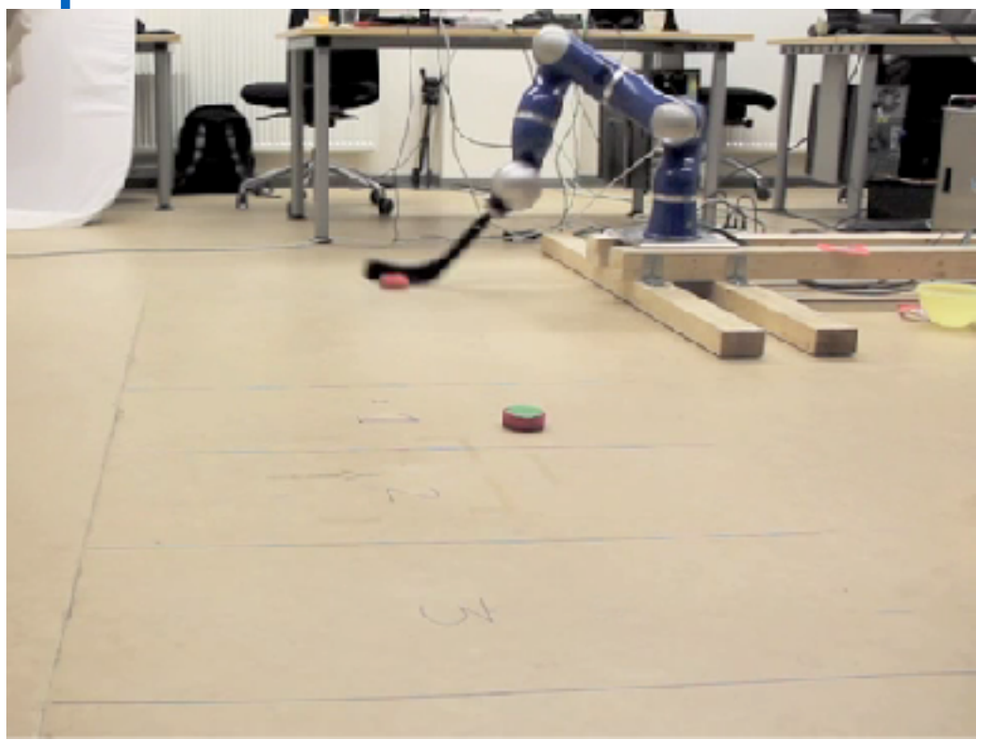
Robot Autonomy

Lecture 17:
Imitation and Skill Learning

Oliver Kroemer

Motivation

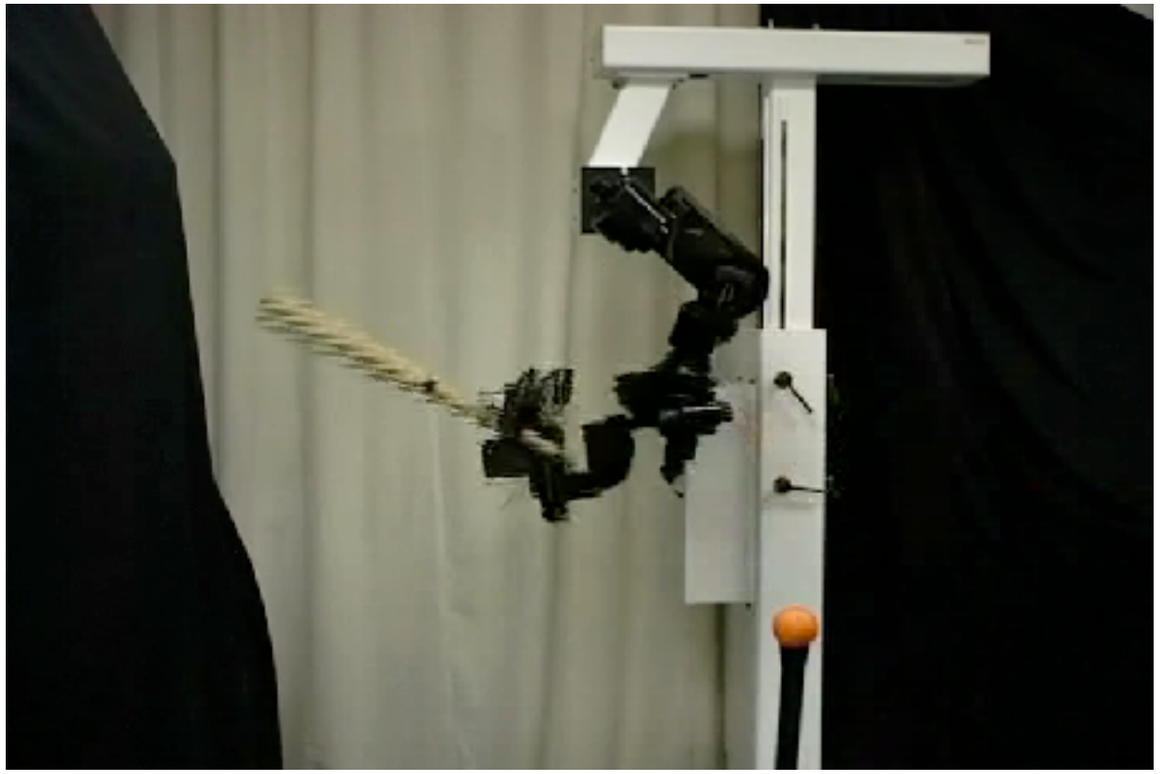
- Want robots to acquire complex motor skills



- Humans are experts at manipulation tasks
- Difficult to program skills by hand
- Learn skills efficiently from human demonstrations
- Master skills autonomously through trial and error

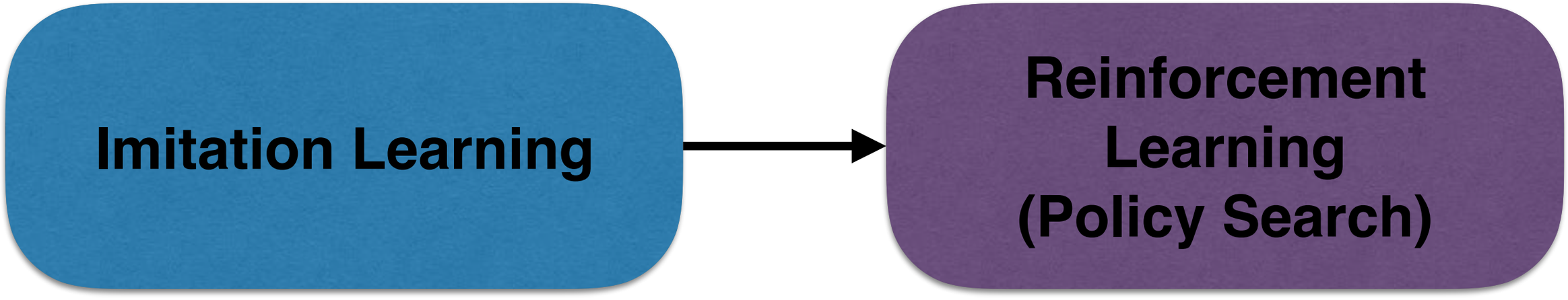
Skill Learning

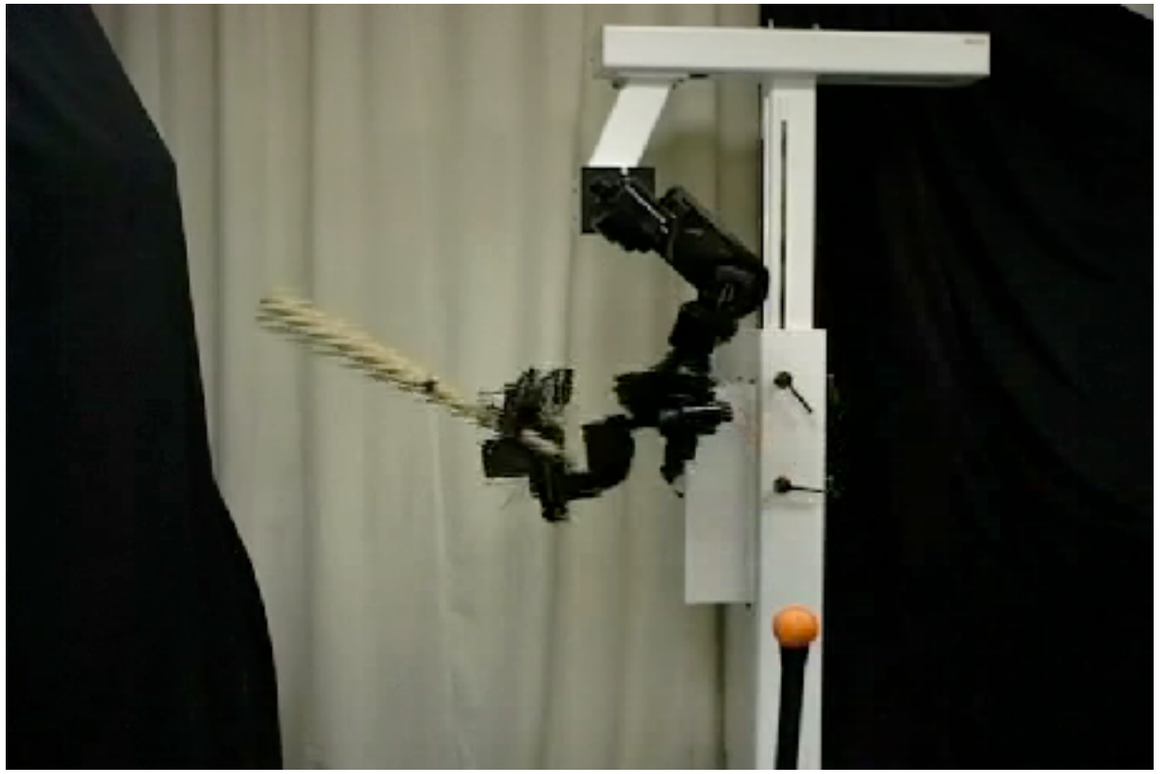
Imitation Learning → **Reinforcement Learning (Policy Search)**



(Peters et al. 2003, 2005)

Skill Learning





(Peters et al. 2003, 2005)

Markov Decision Process

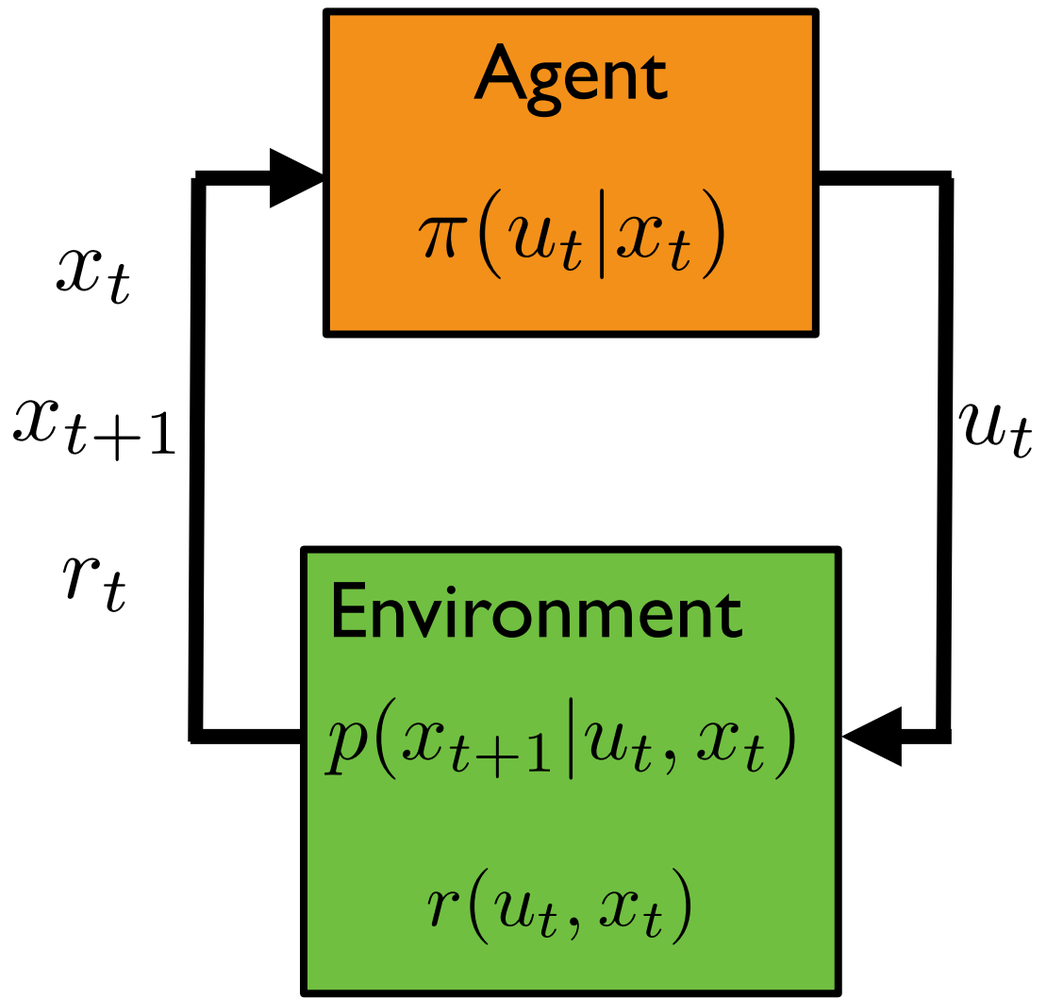
- Goal:
  learn a policy π^* that maximises the expected return
- Policy

$$\pi(u_t|x_t)$$

- Expected Return

$$J(\pi) = E\left[\sum_{t=0}^{\infty} \gamma^t r(u_t, x_t)\right]$$

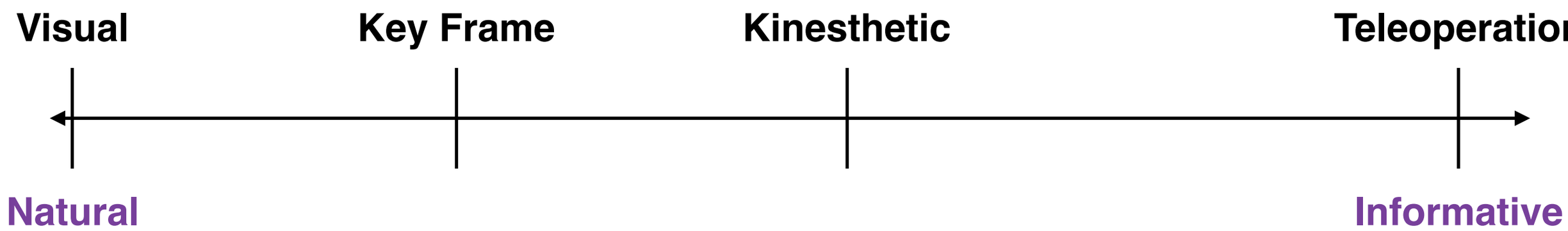
- Imitation Learning
  - Policy from demonstrations
- Reinforcement Learning
  - Policy from trial and error reward



Imitation Learning

Demonstration Sources

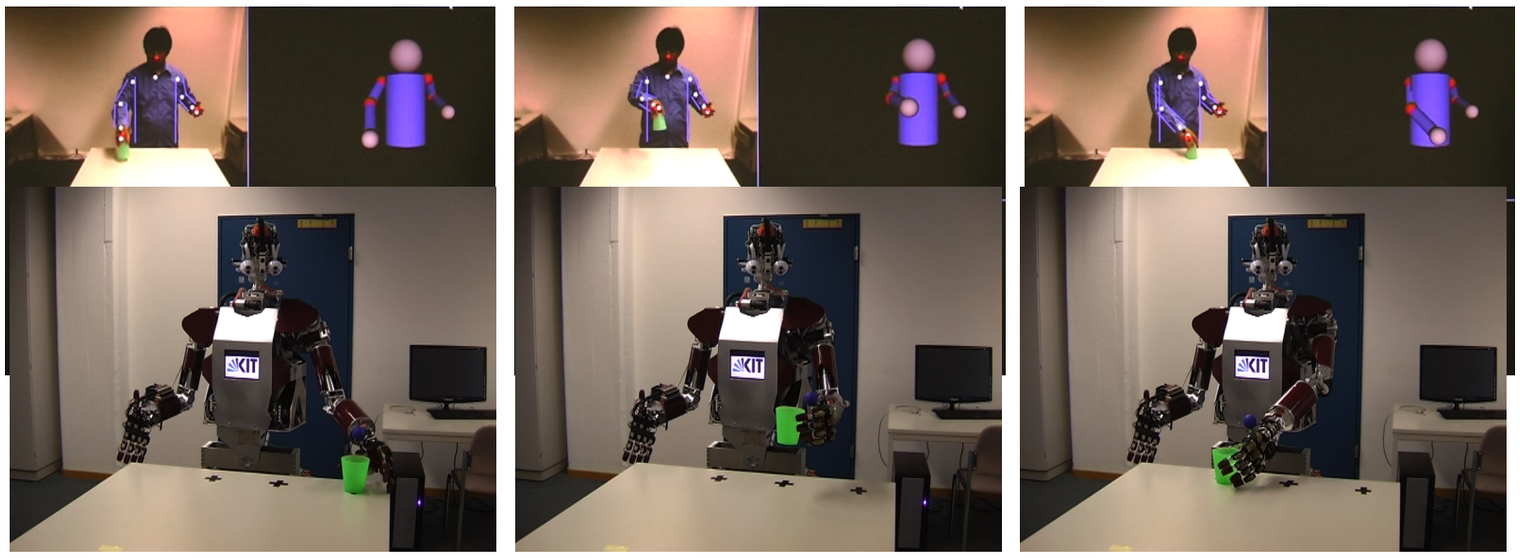
- Demonstrations are acquired using different techniques
- Trade-off ease of demonstration vs informativeness



- Correspondence Problem
  - Human and robot have different embodiment (sense, kin, dyn)
  - Need to define a human2robot mapping (e.g. hand pos, not joints)

Visual Demonstrations

- The human performs the task and the robot observes
  - A lot of potential data available (e.g., youtube)
  - The most natural form of demonstrations



- Correspondence problem is a major issue
  - Robot may not be able to perform task like a human
  - Do not observe forces, sensor readings, etc.
  - Often used to demonstrate high-level task

Dillman et al, Advances in Robot Programming by Demonstrations,

Teleoperation

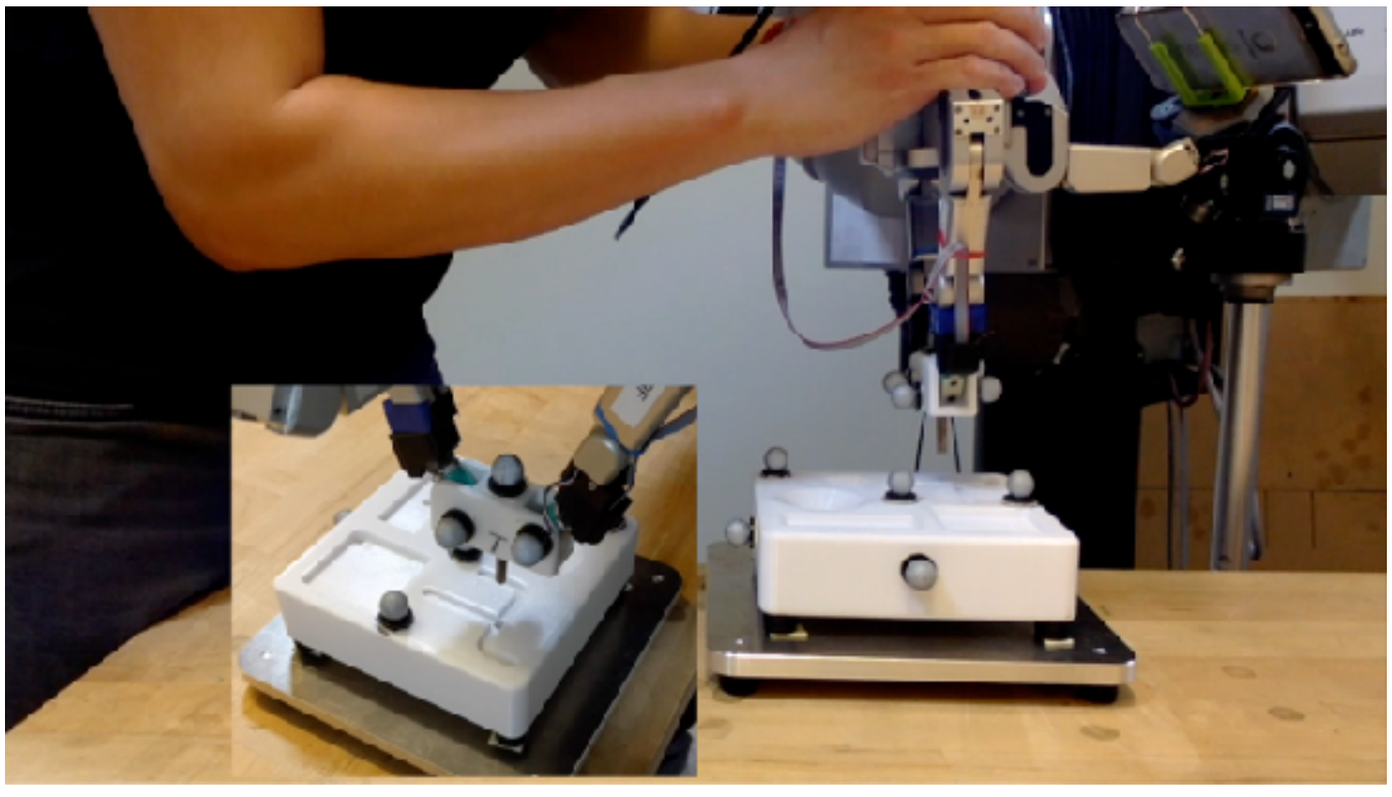
- Control the robot using a haptic/computer interface
- Directly provide actions for states

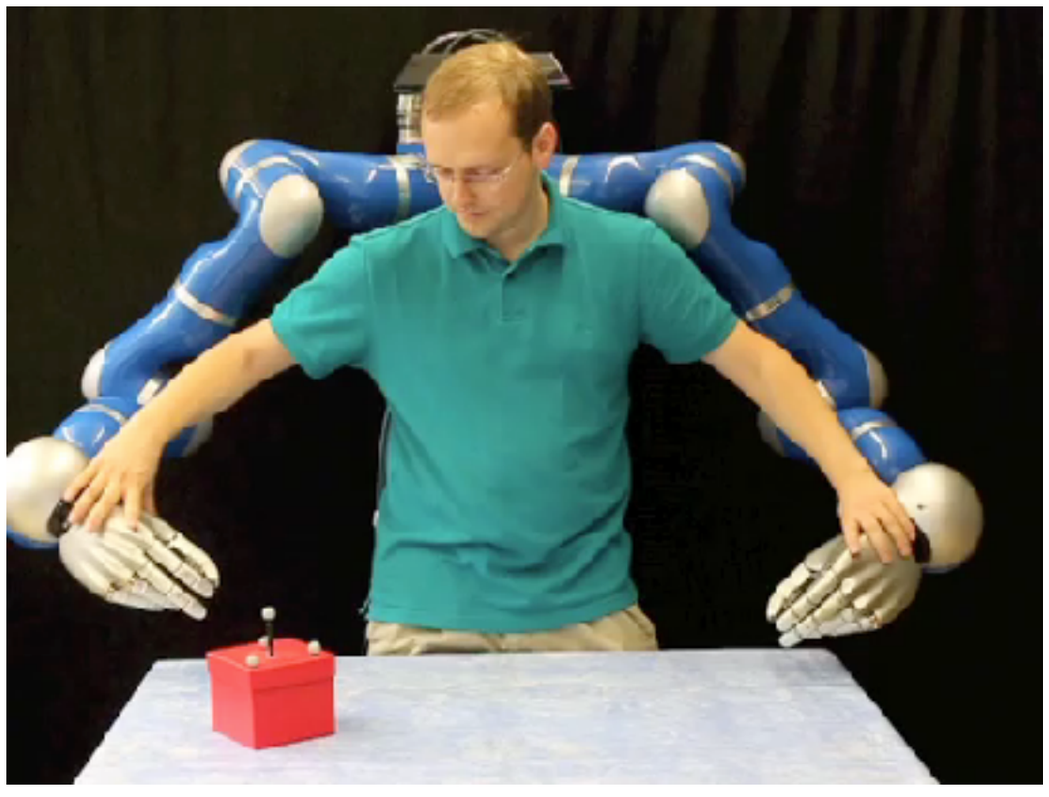
- Most informative and best for learning policies
- Human often needs to relearn task for teleoperation

Hertkorn et al., Virtual reality support for teleoperation using online grasp planning, IROS 2013

Kinaesthetic Teaching

- Take the robot by the hand to demonstrate a task
- Compromise between human and tele-operation



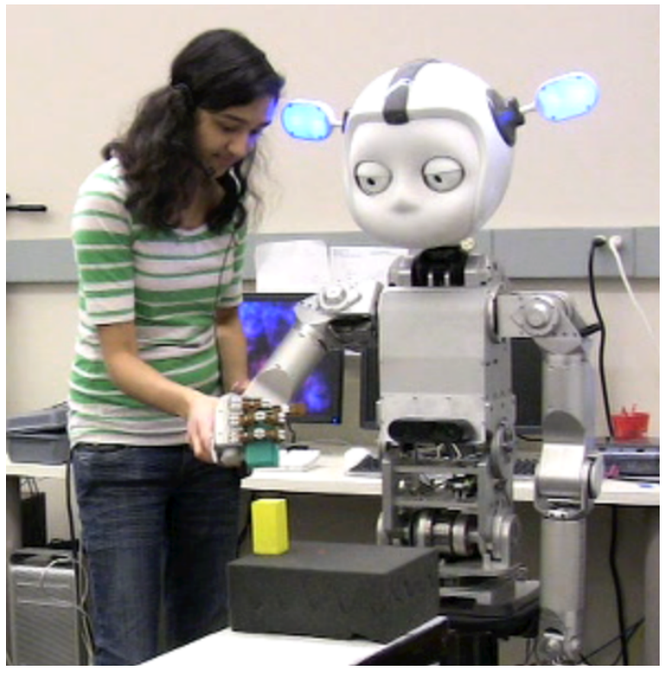


- Still some correspondence problems (e.g, senses, forces)
- Not as intuitive as human performing task (eg. dexterity)

Su et al., Learning to Switch Between Sensorimotor Primitives Using Multimodal Haptic Signals, SAB 2016

Key Frames

- Define a set of key frames rather than entire trajectory

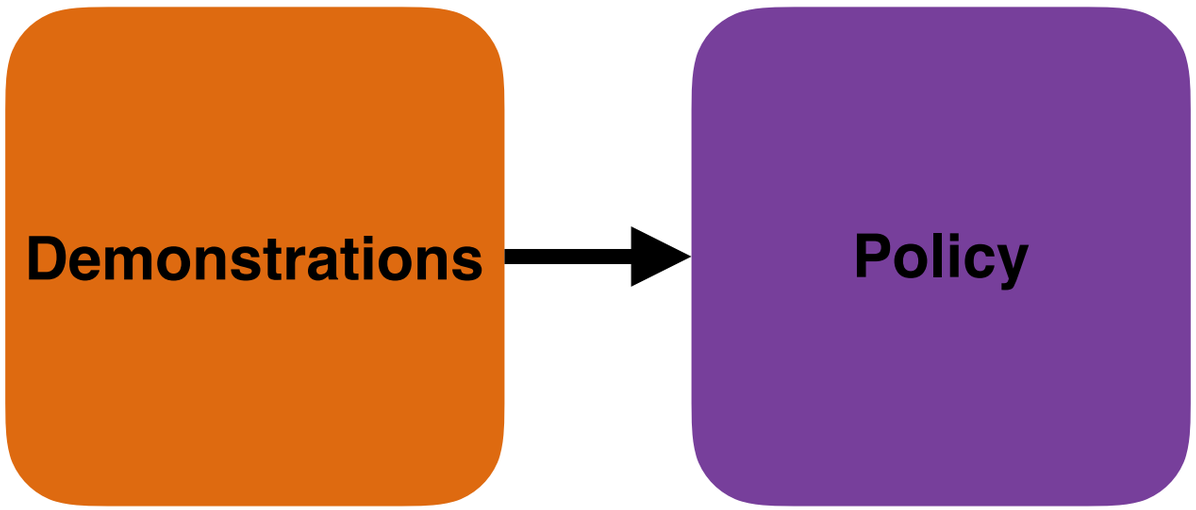


- Allows human to control robots with many DoFs
  - Control a subset of DoFs at a time to position robot
  - Avoid learning useless parts of the demonstration
- Need to ensure important frames are included

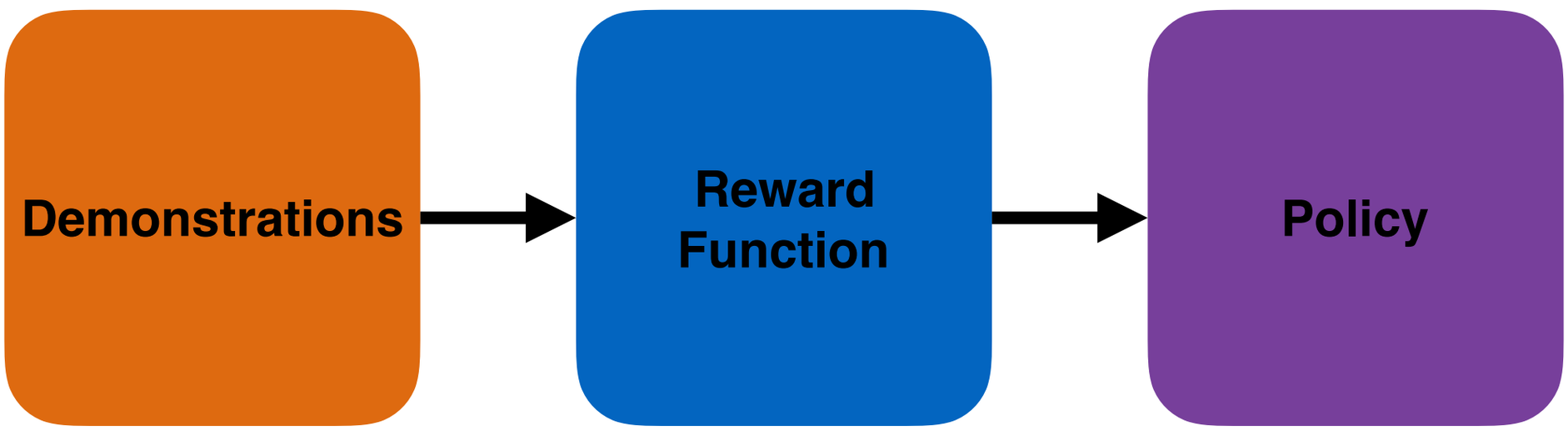
Akgun et al., Trajectories and Keyframes for Kinesthetic Teaching: A Human-Robot Interaction Perspective, HRI 2012

Main Types of Imitation Learning

- Behavioural Cloning



- Apprenticeship Learning / Inverse RL

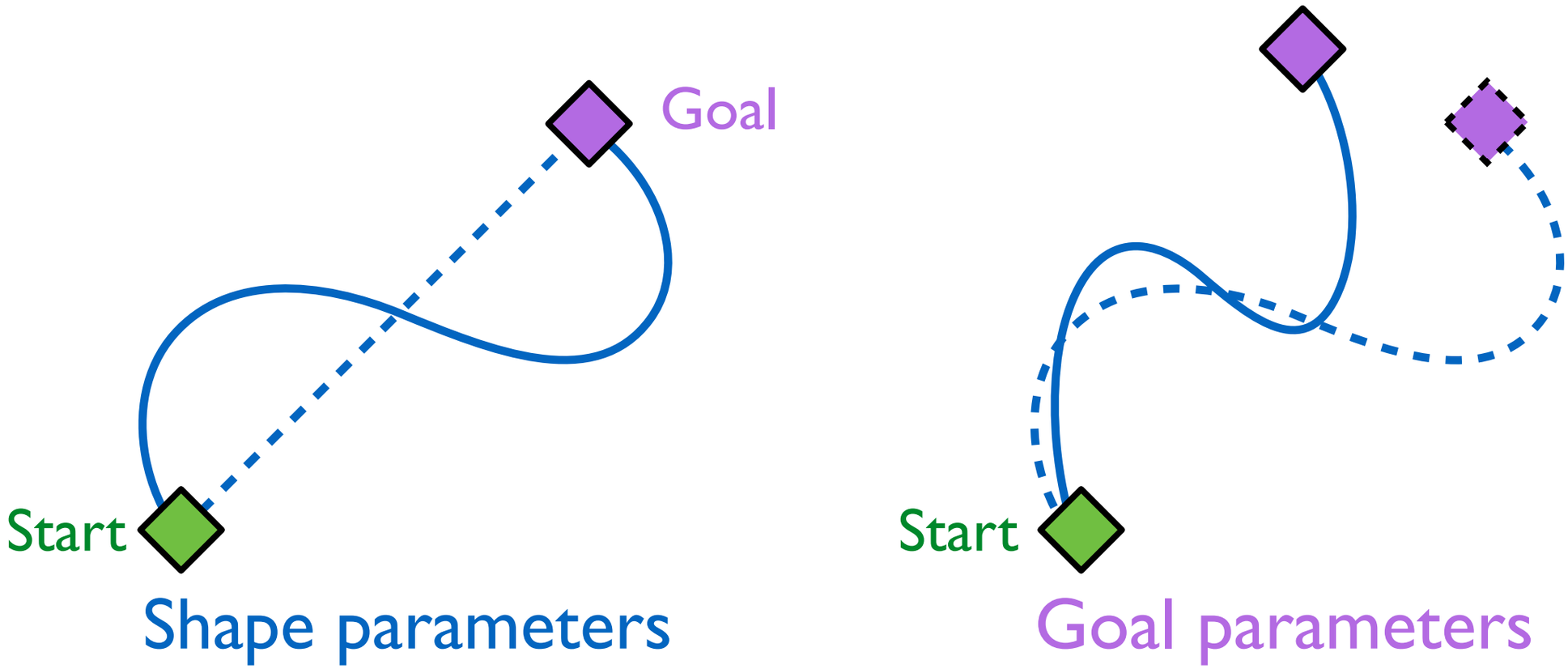


Representations

- Supervised learning from demonstrations
- Learn policy or desired trajectory generator
- Representations trade-off flexibility and prior knowledge
  - Way points
  - Trajectories with (predefined) feedback controllers
  - Neural network feedback controllers
  - Non-parametric policies (e.g., Gaussian Policies)
- Generalization often based on meta parameter inputs
  - Low-dimensional task parameterisation
  - Goal states are a common type of meta parameter

Dynamic Motor Primitives

- Dynamic motor primitives represent trajectory families*
  - Smooth, flexible, generalize well, and easy to learn



- Can be defined in joint or Cartesian spaces

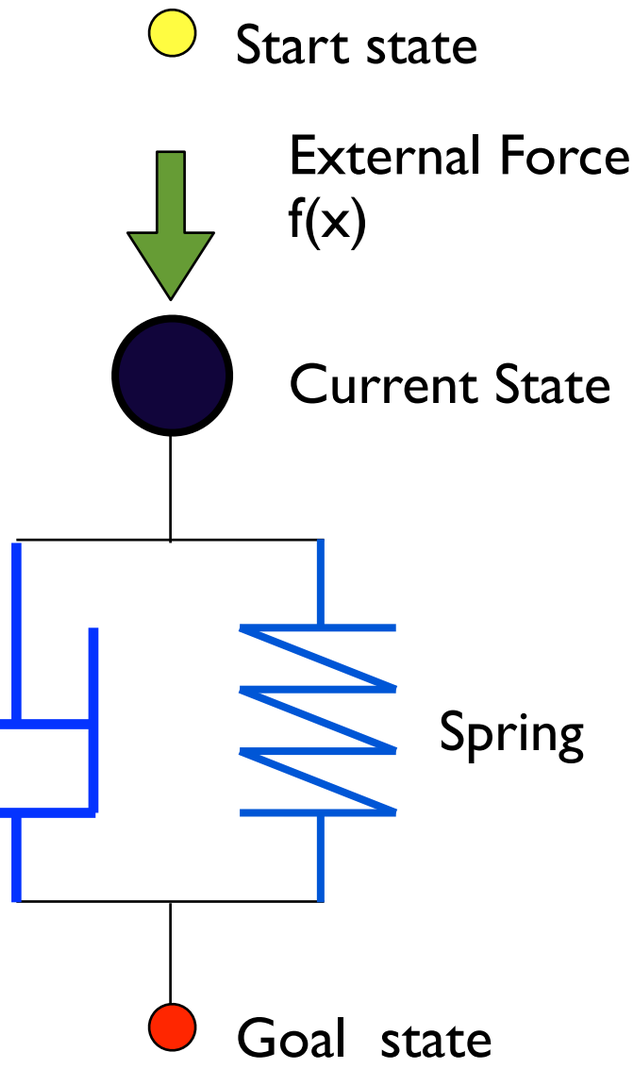
Dynamic Motor Primitives

- Transformed System:

Spring and Damper External Force

$$\ddot{y} = \alpha_z(\beta_z\tau^{-2}(g - y) - \tau^{-1}\dot{y}) + a\tau^{-2}f(x)$$

Goal State Current State Amplitude



- Canonical System:

$$\dot{x} = -\tau x$$

Time Constant Canonical State

  - Phase shifts from 1 to 0

- Each dimension has own transformed system
- Onec canonical system is shared to synchronise DMPs

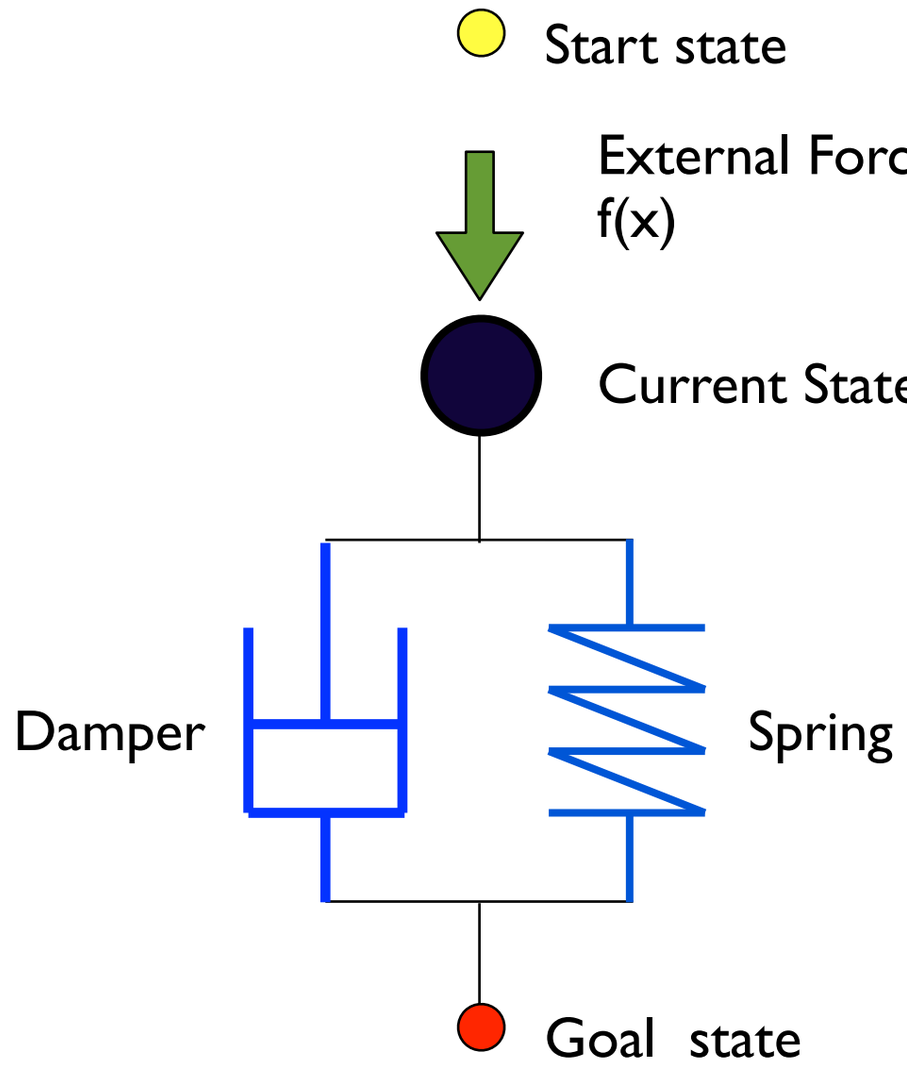
Dynamic Motor Primitives

- Critically damped passive system
- Arbitrary trajectory with force

$$f(x) = \frac{\sum_{j=1}^{M} \psi_j(x)\, w_j x}{\sum_{i=1}^{M} \psi_i(x)}$$

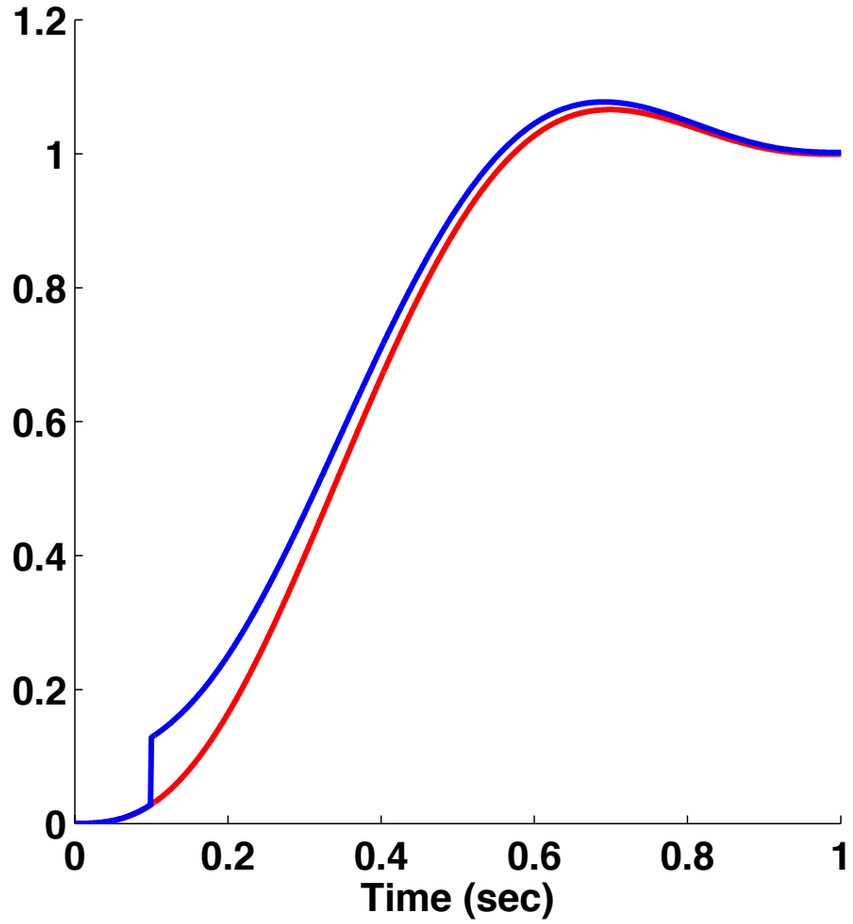
  - Easy to learn by imitation
  - Linear regression of force
- Adaptable to new situations
- Amplitude set to maintain shape

$$a = g - y_0$$



Dynamic Motor Primitives

- DMPs are inherently stable
  - Passive elements
  - External force decays to zero
- Safe to use on a robot
- Always reach goal if possible
- Reject additional external forces
- Act as attractor fields...



DMP Extensions

- Detractor fields
  - DMP represents a time-varying attractor field
  - Add forces that push DMP away from obstacles
- Sensory coupling
  - Adapt trajectory to sensory feedback

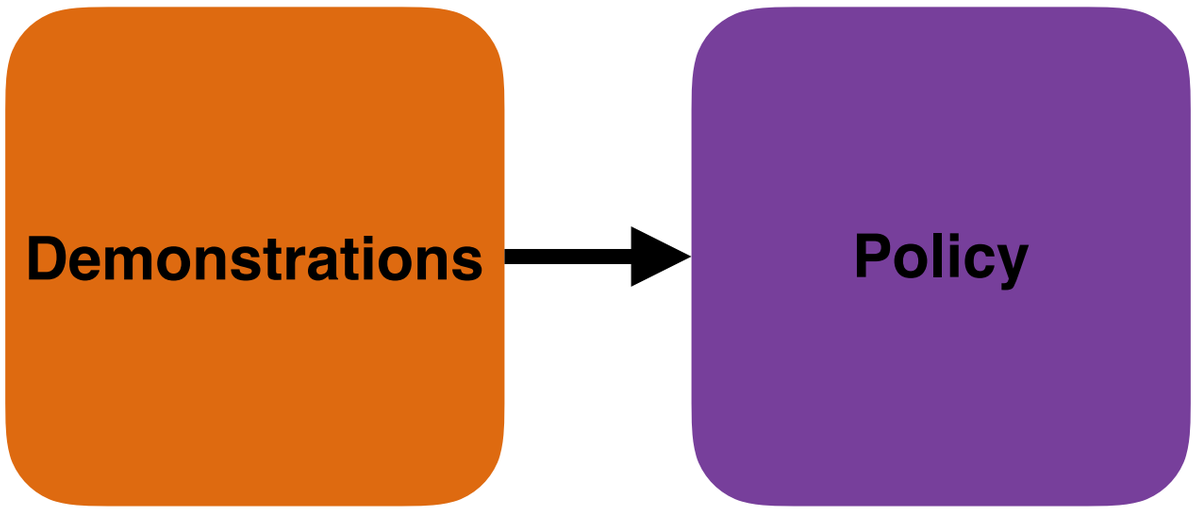
$$+a\tau^{-2}f(x) + \tau^{-2}C(x, s)$$

sensor data

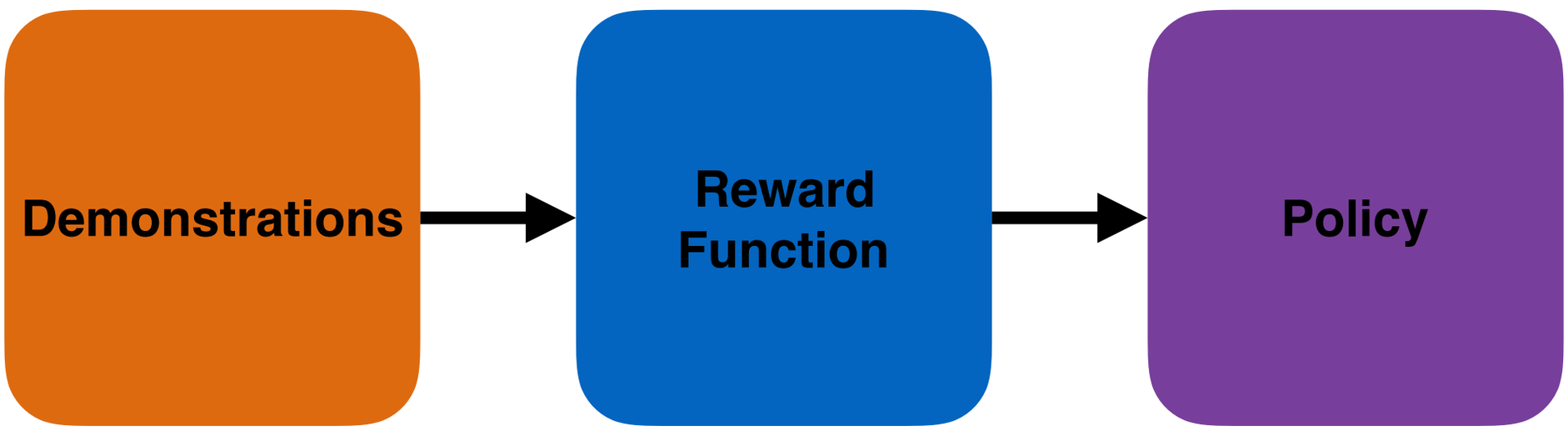
  - Use additional low-level feedback to follow DMP trajectory
- Moving goals for striking
  - Standard DMP always converges to goal state
  - Can define a moving goal state for striking movements

Main Types of Imitation Learning

- Behavioural Cloning



- Apprenticeship Learning / Inverse RL



Terrain Traversal Example

- Simply scaling the trajectory does not always work

Demonstrated Behavior

Novel Scene

Terrain Traversal Example

- Follow the path

Demonstrated Behavior

Learned Behavior

Terrain Traversal Example

- Move through the trees

Demonstrated Behavior

Learned Behavior

- Different rewards for traversing path vs grass vs trees
- Capture the task by extracting the reward function

Inverse Reinforcement Learning

- Assume we have an MDP/R without a reward function
- Given demonstrations from expert policy π^E
- Find a reward function such that for all policies:

$$J(\pi^E) \geq J(\pi)$$

- Many challenges
  - Reward ambiguity - many rewards will have same optimal policy
  - Only observe traces of expert policy, not full policy
  - Assumes demonstrator is expert, otherwise infeasible

Feature Expectation/Count

- Each state and action is associated to a set of features
- Assume that reward is linear in the features

$$r(u, x) = w^T f(u, x)$$

- Hence we can rewrite $J(\pi^E) \geq J(\pi)$ as:

$$\mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i w^T f(u_i, x_i) | \pi^E\right] \geq \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i w^T f(u_i, x_i) | \pi\right]$$

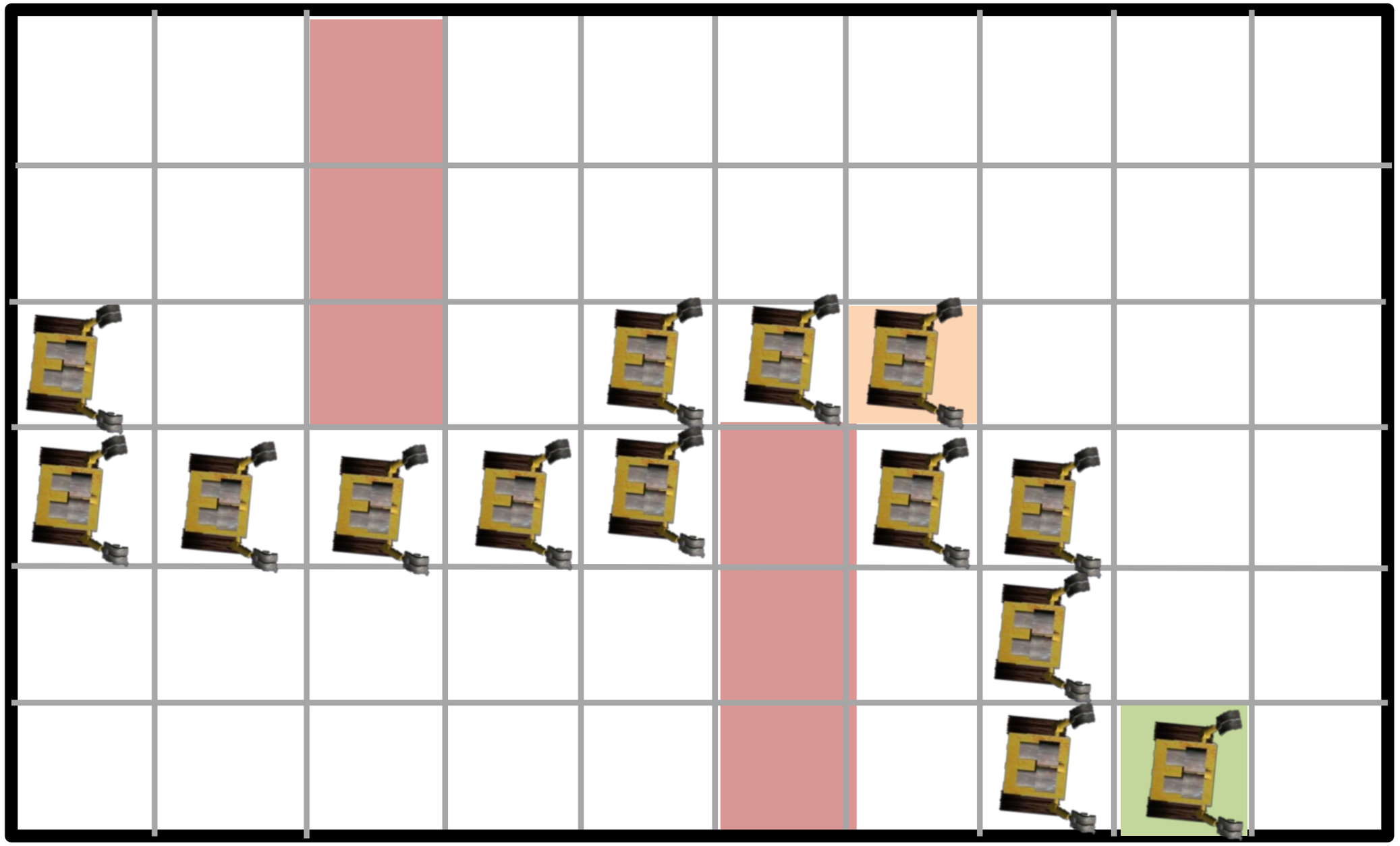
$$w^T \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i f(u_i, x_i) | \pi^E\right] \geq w^T \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i f(u_i, x_i) | \pi\right]$$

$$w^T \mu(\pi^E) \geq w^T \mu(\pi)$$

Feature expectation/count

Grid World Example

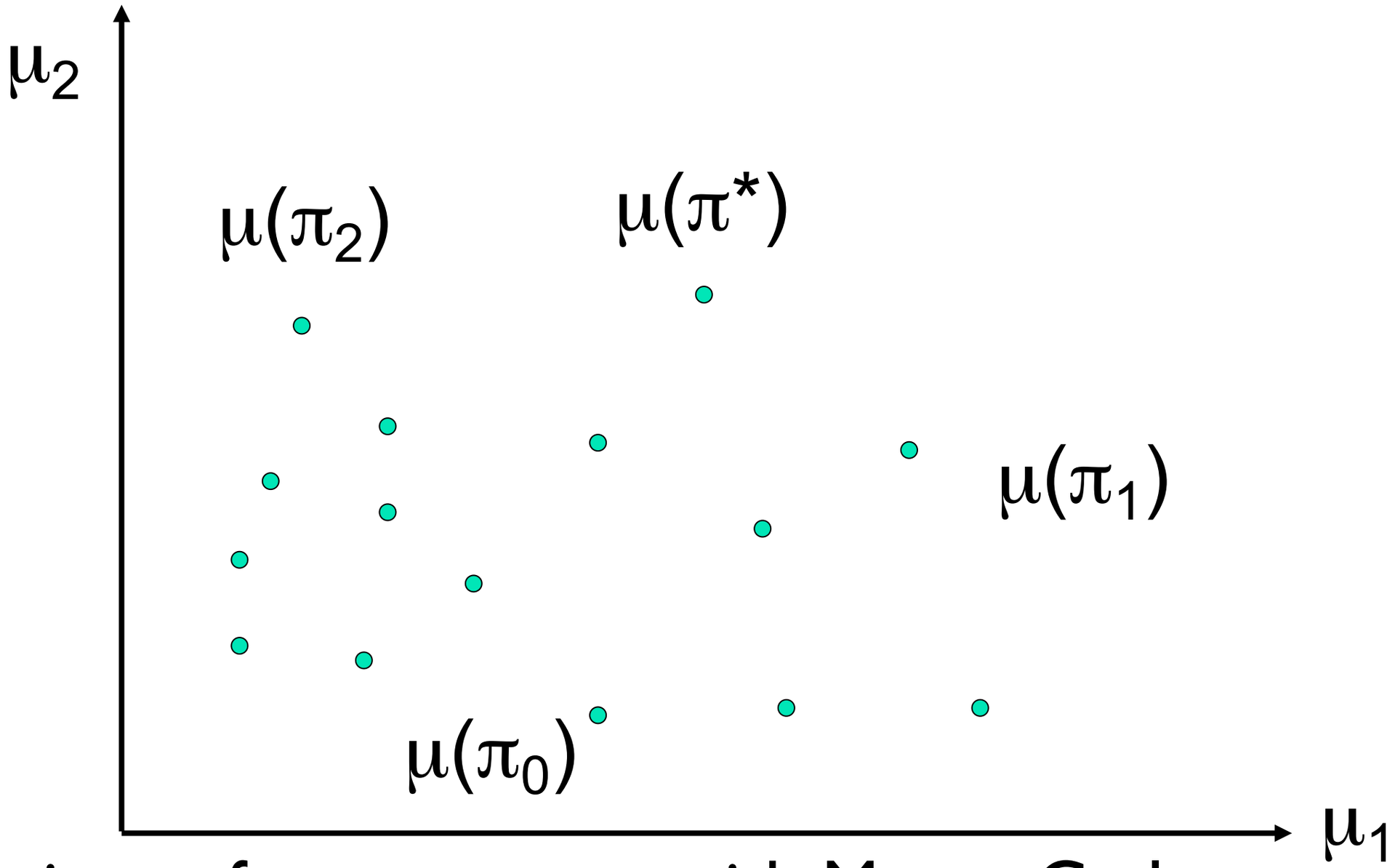
Grid World Example



Grid World Example

Feature Counts

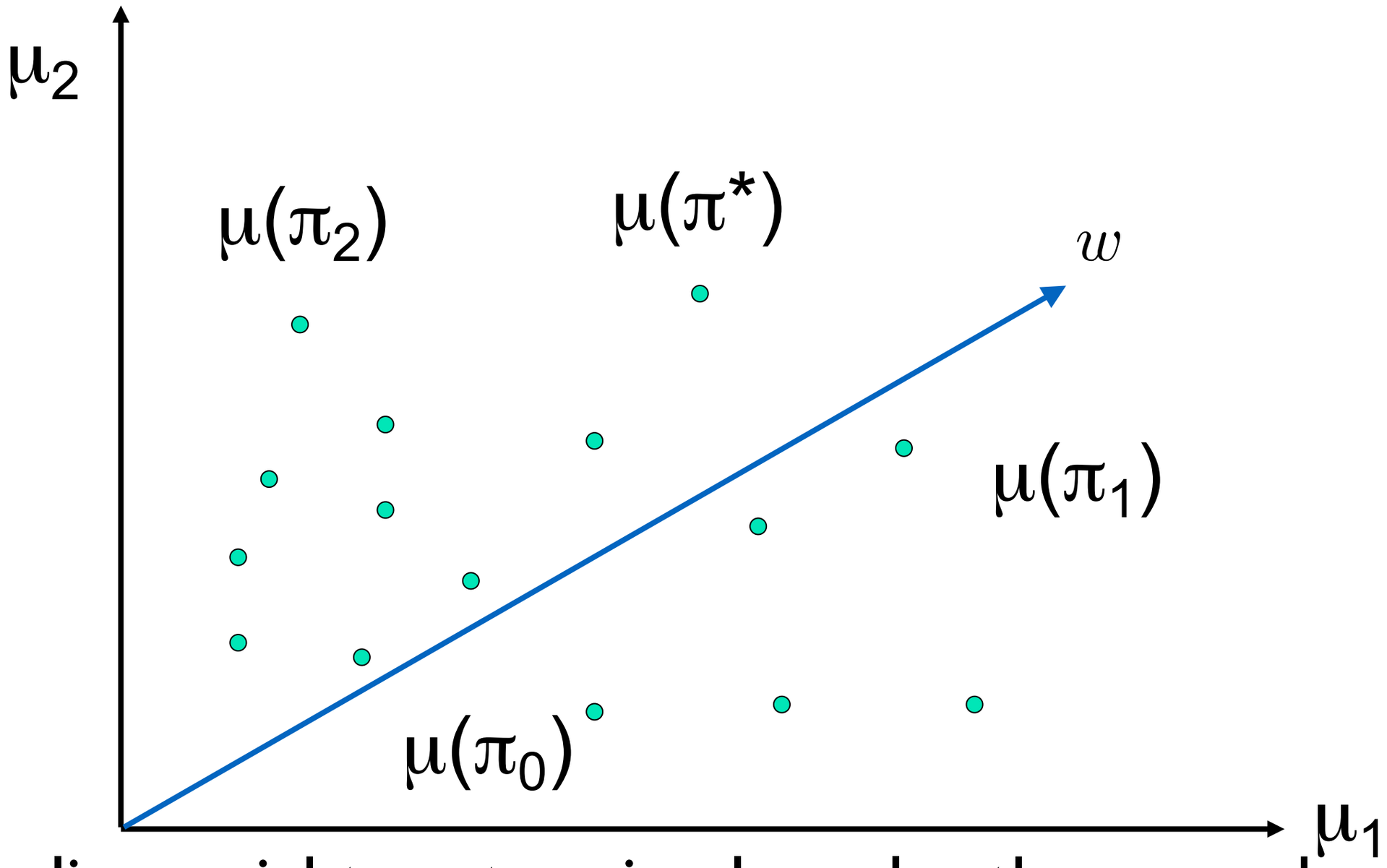
- Each policy corresponds to an expected feature vector



- Estimate feature counts with Monte Carlo approach
- Similar feature counts will give similar rewards

Feature Counts

- Expected return determined by reward weight vector



- Scaling weight vector simply scales the rewards
- Bound the weight magnitude, e.g., $|w|_1 \leq 1$

Apprenticeship Learning via Inverse RL

- Goal:
  Find a policy $\tilde{\pi}$ whose performance is close to π^E

$$|w^T \mu(\tilde{\pi}) - w^T \mu(\pi^E)| \leq \epsilon$$

- Reduces to finding policy with similar feature counts

$$|w^T(\mu(\tilde{\pi}) - \mu(\pi^E))| \leq ||w||_2 ||\mu(\tilde{\pi}) - \mu(\pi^E)||_2$$

$$||w||_2 \leq ||w||_1 \leq 1$$

$$||\mu(\tilde{\pi}) - \mu(\pi^E)||_2 \leq \epsilon$$

Abbeel and Ng,Apprenticeship Learning via Inverse Reinforcement Learning, ICML 04

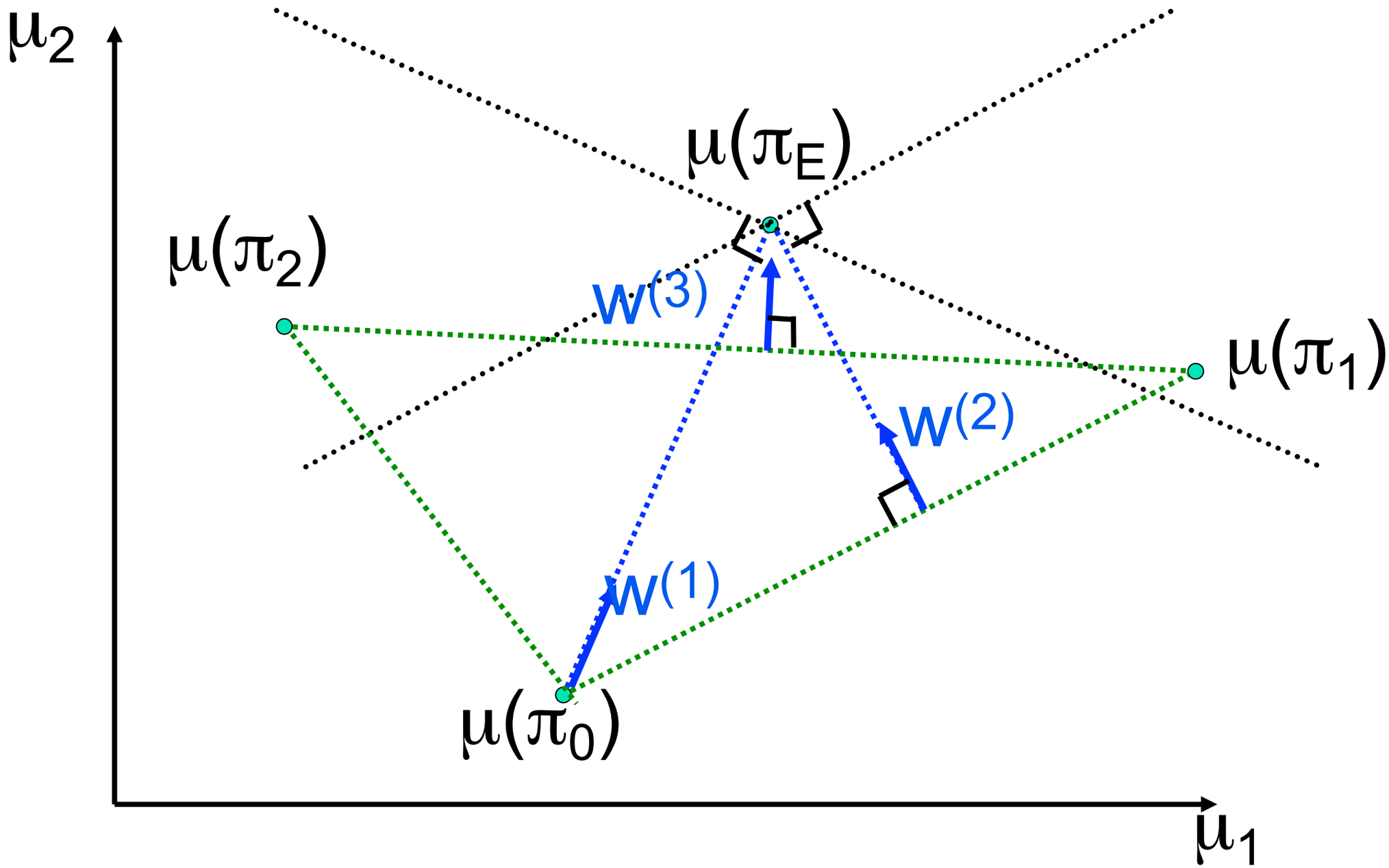
Apprenticeship Learning via Inverse RL

1. Randomly pick some policy $\pi^{(0)}$, compute (or approximate via Monte Carlo) $\mu^{(0)} = \mu(\pi^{(0)})$, and set $i = 1$.
2. Compute $t^{(i)} = \max_{w:\|w\|_2 \leq 1} \min_{j \in \{0..(i-1)\}} w^T(\mu_E - \mu^{(j)})$, and let $w^{(i)}$ be the value of w that attains this maximum.

Maximize margin to "second best" policy

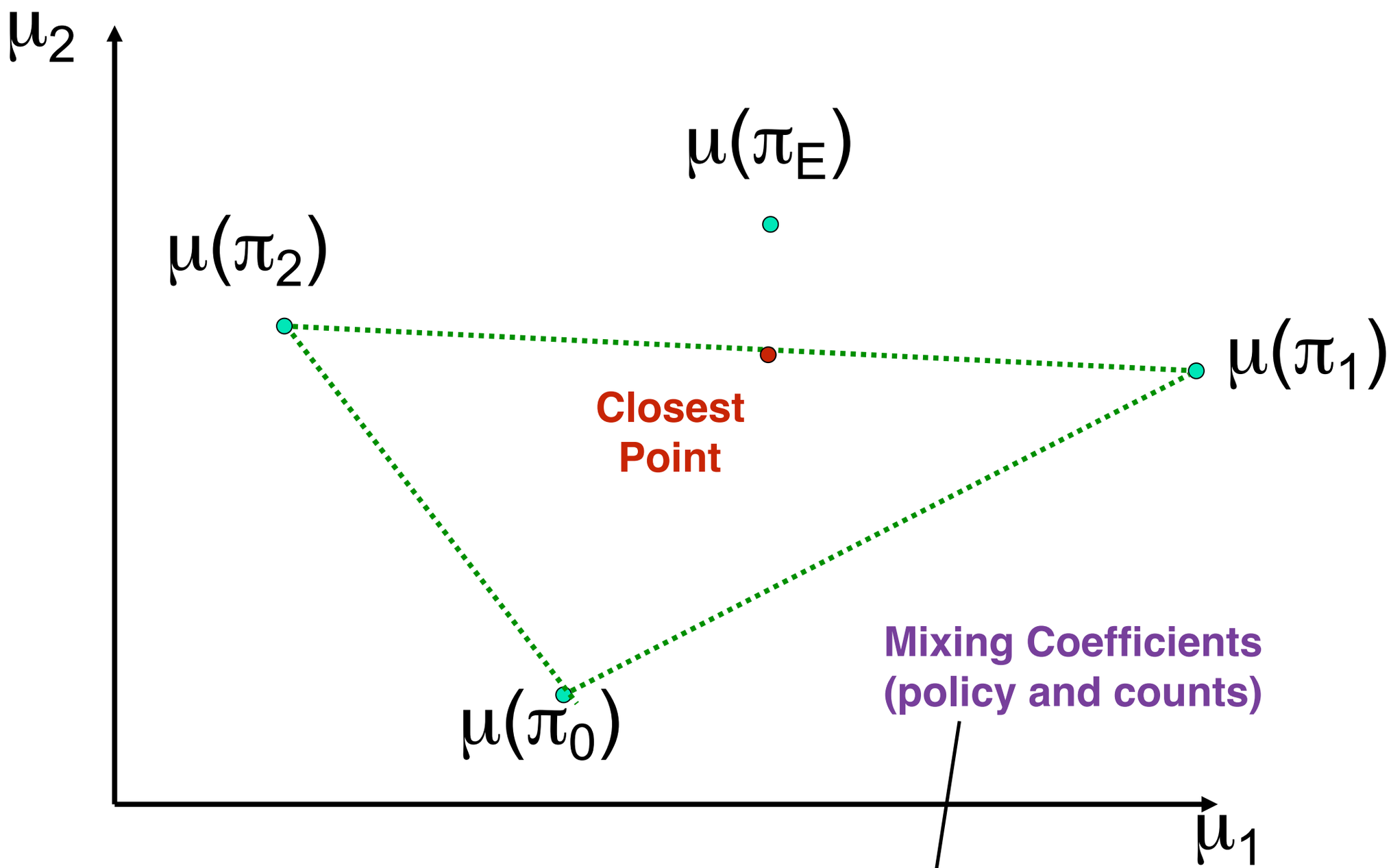
3. If $t^{(i)} \leq \epsilon$, then terminate.
4. Using the RL algorithm, compute the optimal policy $\pi^{(i)}$ for the MDP using rewards $R = (w^{(i)})^T \phi$.
5. Compute (or estimate) $\mu^{(i)} = \mu(\pi^{(i)})$.
6. Set $i = i + 1$, and go back to step 2.

Apprenticeship Learning via Inverse RL



Apprenticeship Learning via Inverse RL

- Select closest policy, or stochastically mix policies



$$\min ||\mu_E - \mu||_2, \text{s.t. } \mu = \sum_i \lambda_i \mu^{(i)}, \lambda_i \geq 0, \sum_i \lambda_i = 1.$$

Further Reading and Applications

- Route Planning

  Ratliff et al., 2006

- Parking lot navigation

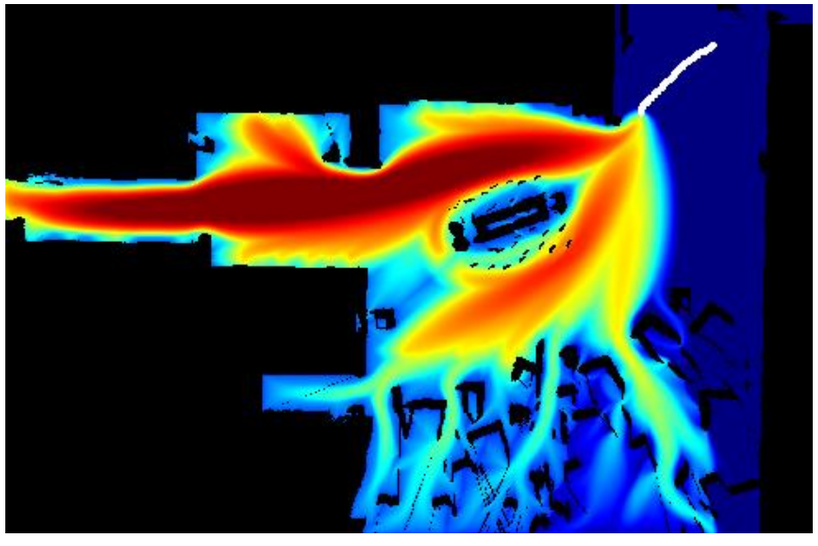
  Abbeel et al., 2008

- Quadruped locomotion

  Kolter et al. 2008

- Pedestrian Prediction

  Ziebart et al., 2009



“good” Parking Examples

Abbeel et al.Apprenticeship learning for motion planning with application to *parking* lot navigation, AAAI 08

“sloppy” Parking Example

Abbeel et al.Apprenticeship learning for motion planning with application to *parking* lot navigation, AAAI 08

“don’t mind reverse” Parking Example

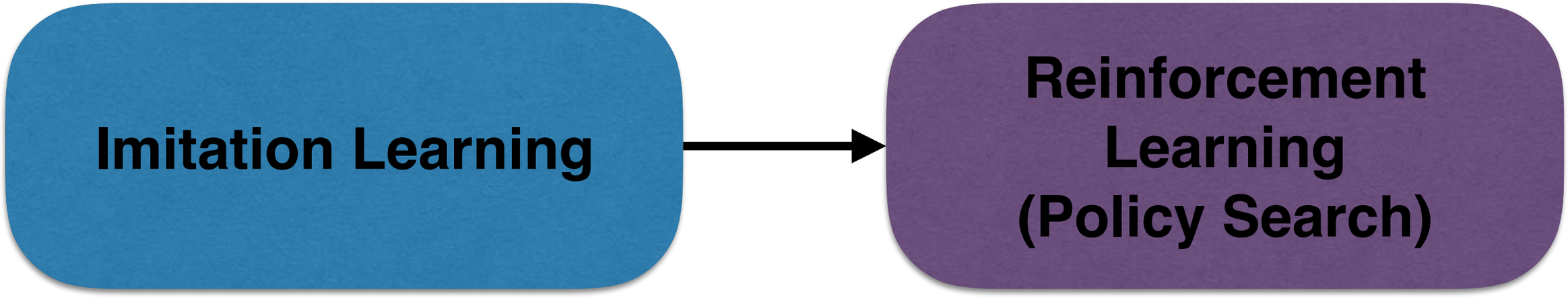
Helicopter Example

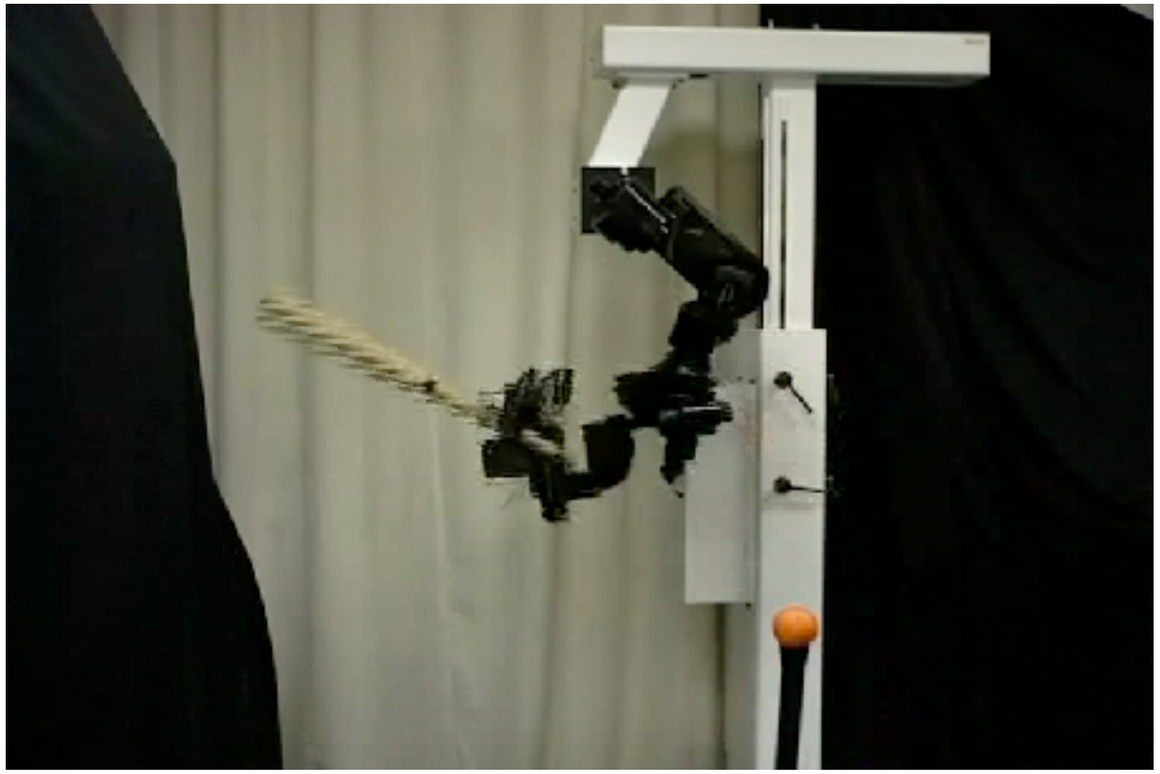
- Apprenticeship learning and model learning:

Coates et al. , Apprenticeship Learning for Helicopter Control, 2008

Policy Search Reinforcement Learning (a brief intro to finite differences)

Skill Learning





(Peters et al. 2003, 2005)

Problem Statement

- Executing a policy results in a sample trajectory

$$\tau = \{x_0, u_0, ..., x_T, u_T, x_{T+1}\}$$

- Each trajectory provides the robot with a return

$$R(\tau) = \sum_{t=0}^{T} r(u_t, x_t)$$

- Goal: find policy that maximises the expected return

$$J(\theta) = \int_{\mathbb{T}} p_\theta(\tau|\pi) R(\tau) \mathrm{d}\tau$$

- The goal is therefore to learn the policy parameters θ

$$\pi_\theta(u_t|x_t)$$

Problem Statement

- The goal is to compute

$$\arg\max_\theta J(\theta)$$

- Finding the global optimum is intractable
  - Complicated dynamics and policies
  - Non-convex reward landscape
  - High-dimensional state space
  - No closed-form solutions
- Use gradient ascent to find (good) local optimum

$$\theta_{i+1} = \theta_i + \beta \nabla_\theta J(\theta)$$

- How to compute estimate of the gradient?

Finite Difference Method

- Finite difference method procedure:

1. Initialize policy with some initial parameters $\theta_{i=0}$
2. Sample neighbourhood of parameters $\tilde{\theta}_{ij} \quad j \in \{1, ..., n\}$
3. Evaluate parameter settings to estimate returns $J(\theta_i) \quad J(\tilde{\theta}_{ij})$
4. Compute finite difference gradient estimate g_{fd}
5. Update parameters using gradient ascent $\theta_{i+1} = \theta_i + \beta g_{\text{fd}}$
6. Repeat from step 2

- Final performance depends on good initial parameters

Finite Difference Method

- Sample n local parameters from grid or stochastically

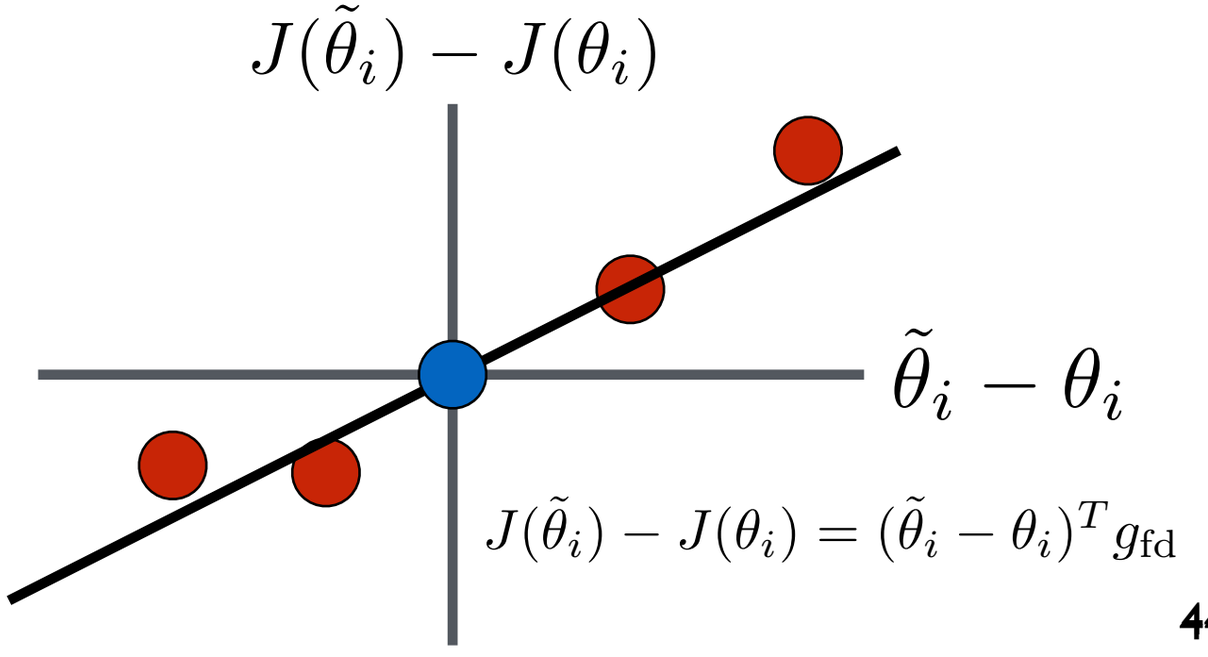
$$\tilde{\theta}_{ij} = \theta_i + \delta \qquad \tilde{\theta}_{ij} \sim \mathcal{N}(\theta_i, \Sigma)$$

- Estimate gradient using linear regression on differences

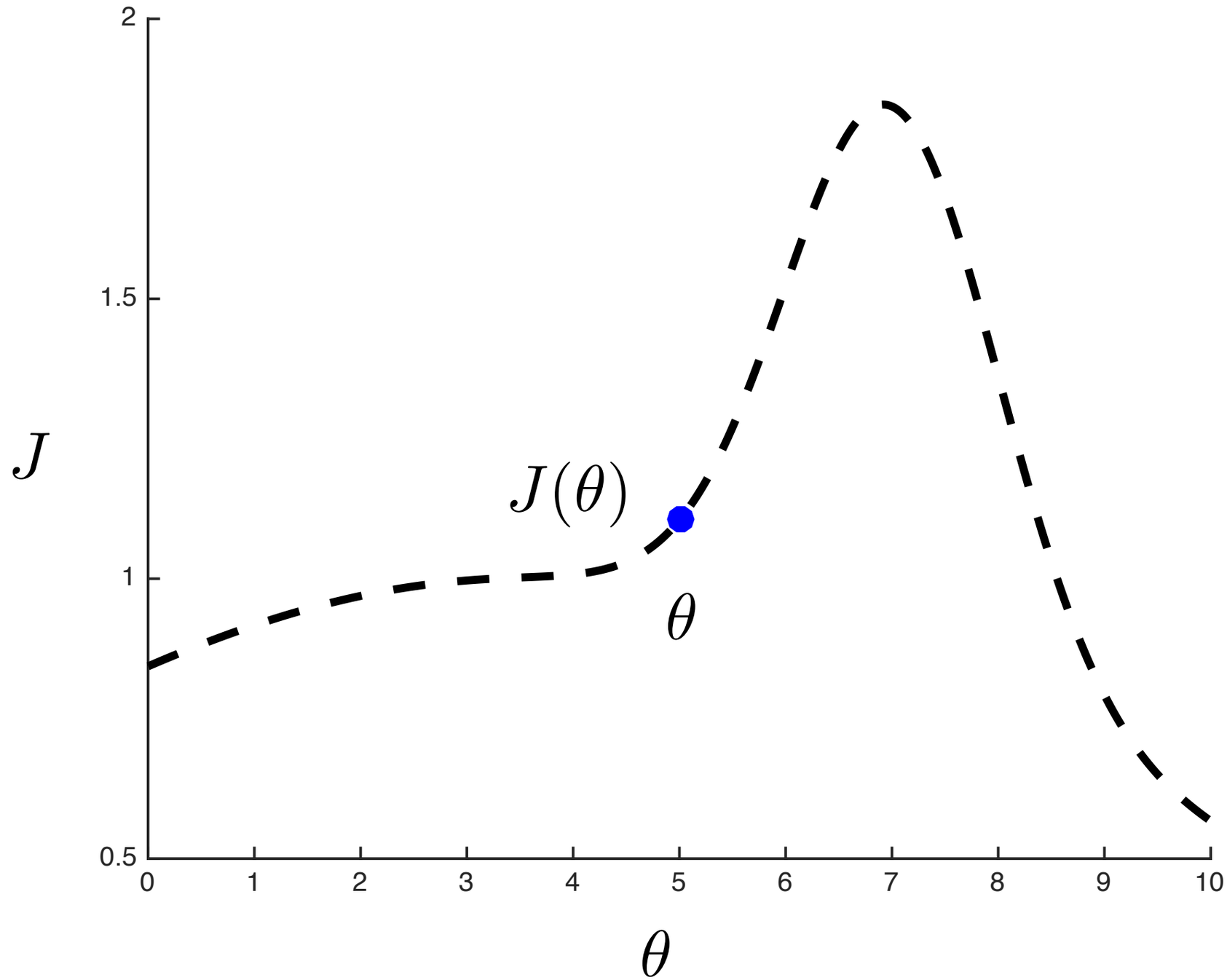
$$\widehat{\Theta} = \begin{bmatrix} | & & | \\ \tilde{\theta}_{i1} - \theta_i & \dots & \tilde{\theta}_{in} - \theta_i \\ | & & | \end{bmatrix} \qquad \widehat{J} = \begin{bmatrix} J(\tilde{\theta}_{i1}) - J(\theta_i) \\ \vdots \\ J(\tilde{\theta}_{in}) - J(\theta_i) \end{bmatrix}$$

$$g_{\text{fd}} = (\widehat{\Theta}\widehat{\Theta}^T + \lambda I)^{-1}\widehat{\Theta}\widehat{J}$$

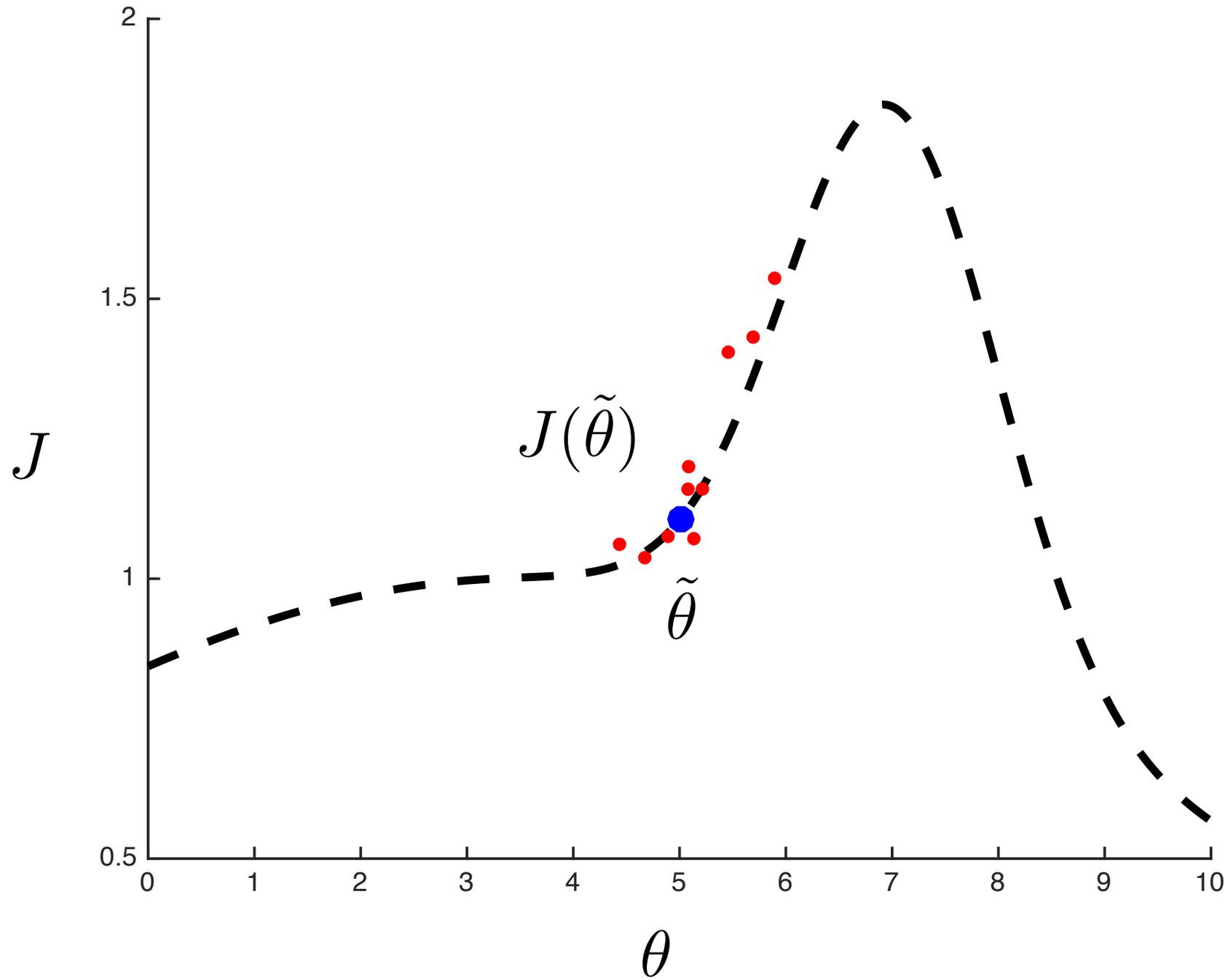
$$\theta_{i+1} = \theta_i + \beta g_{\text{fd}}$$



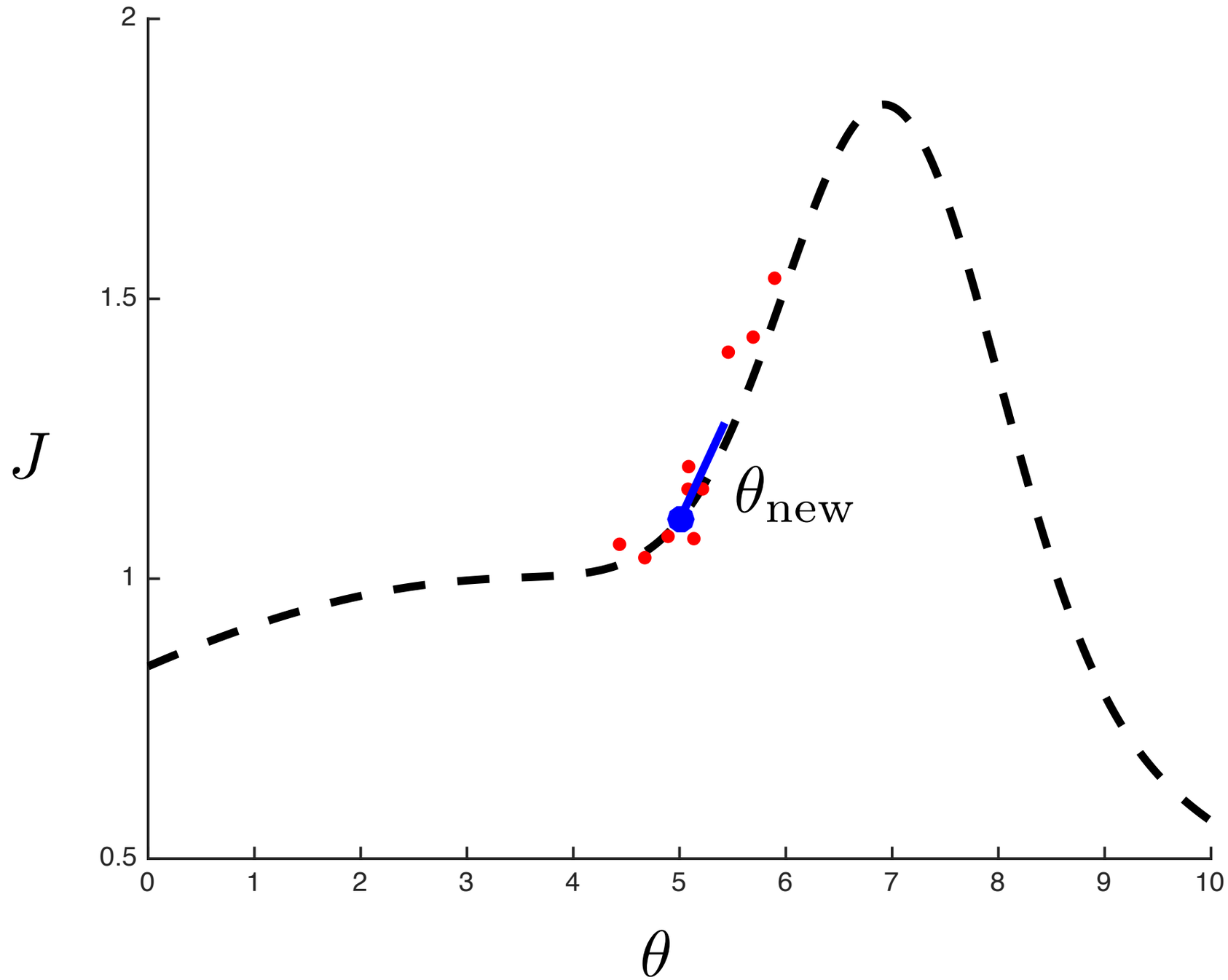
Finite Difference Example



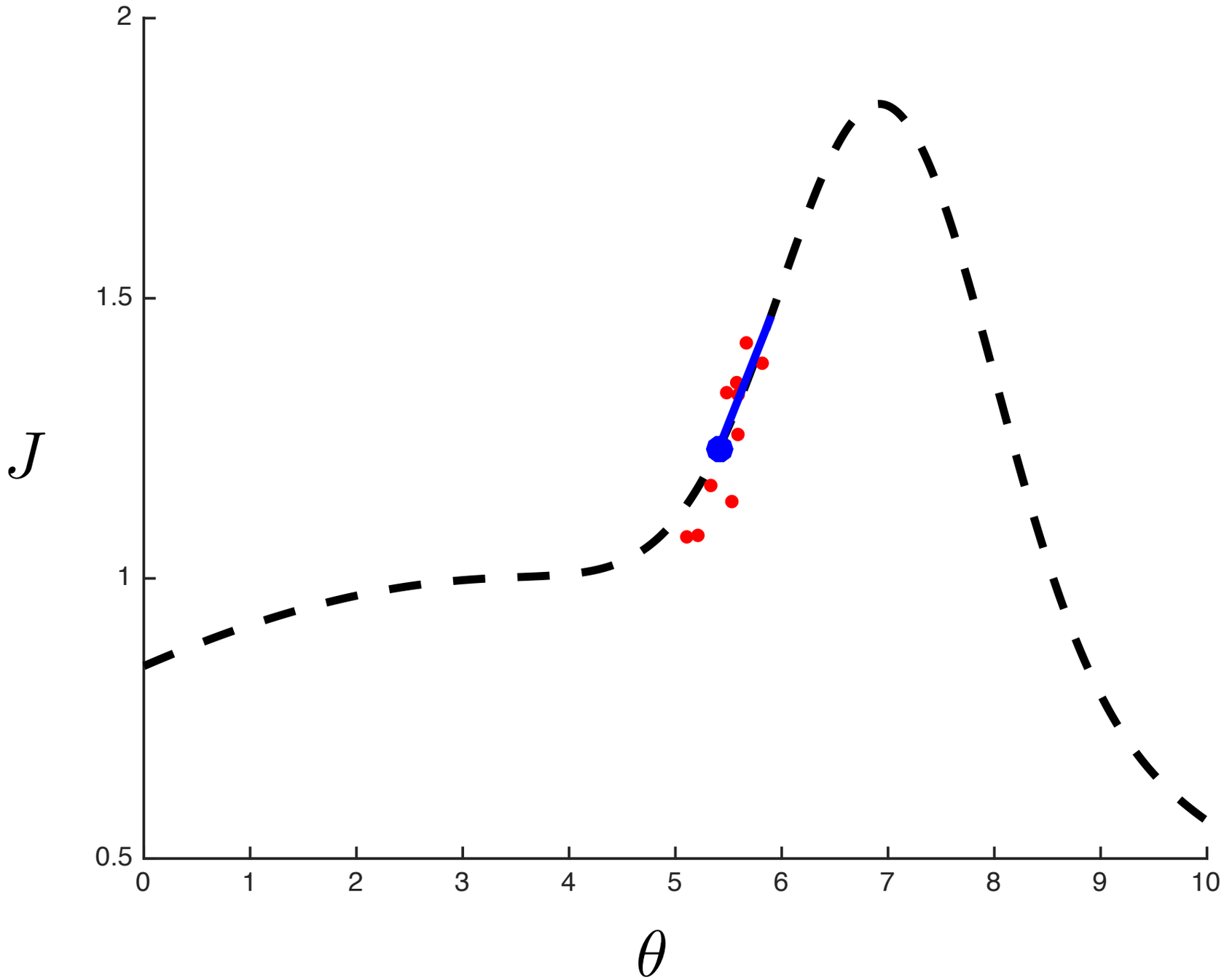
Finite Difference Example



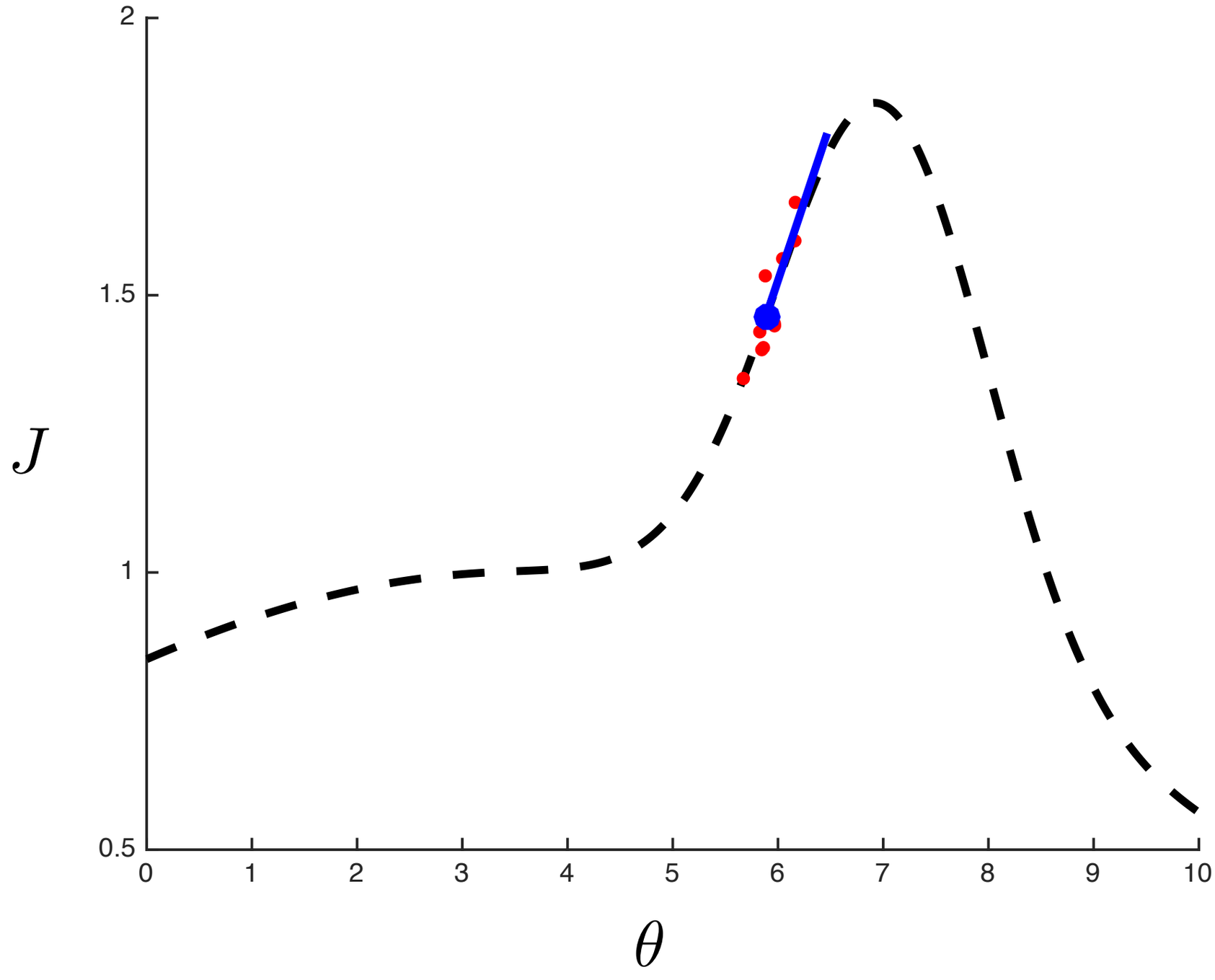
Finite Difference Example



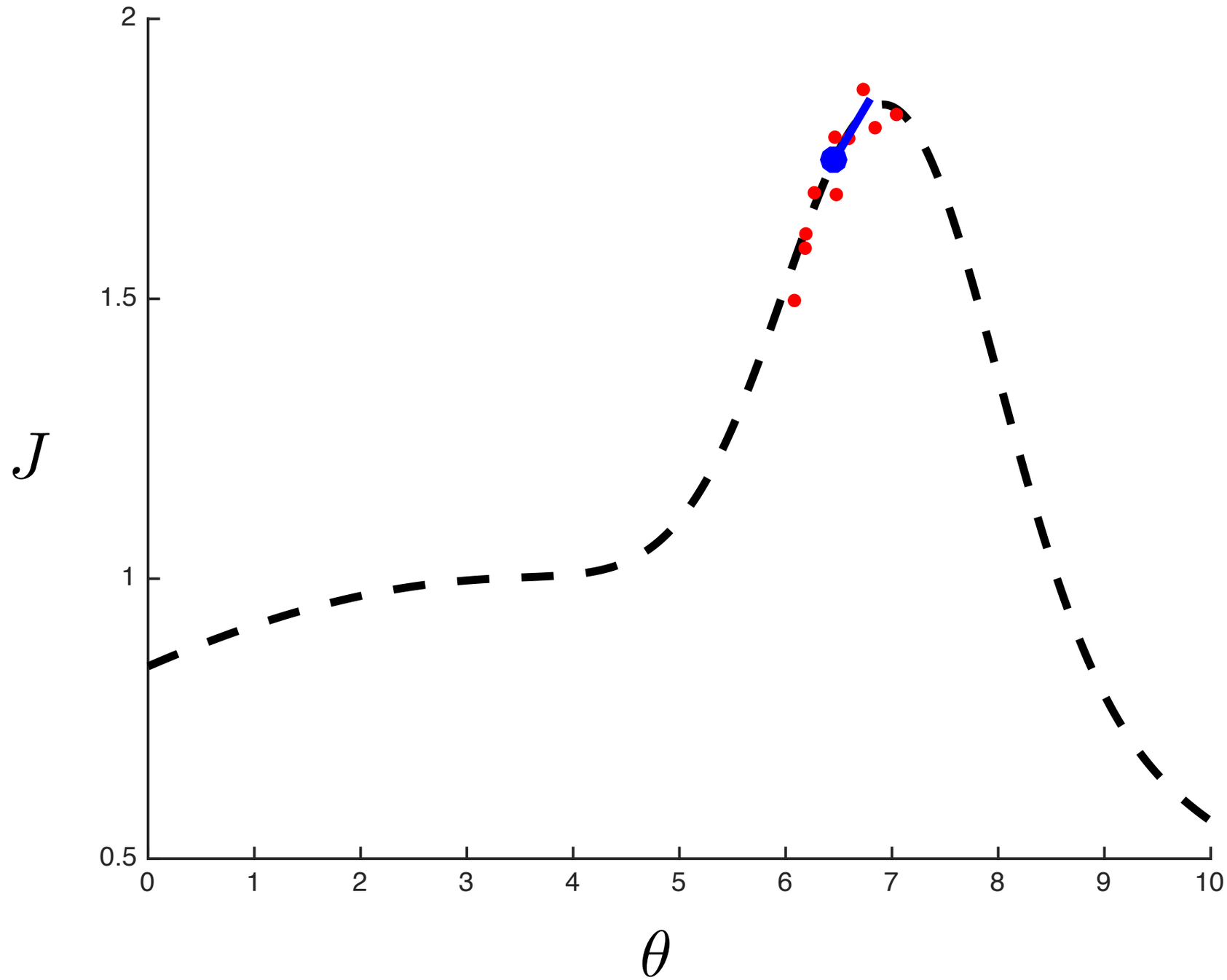
Finite Difference Example



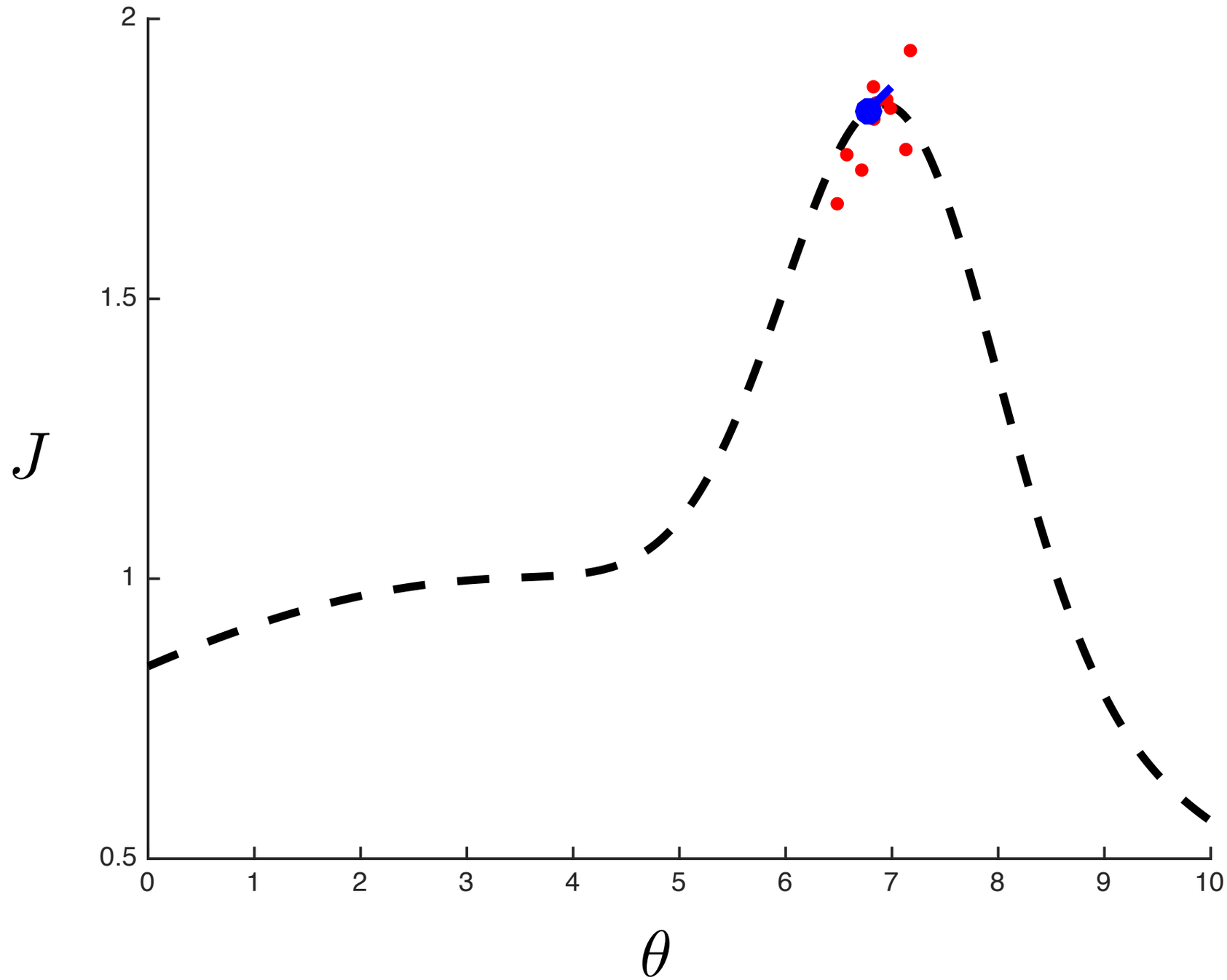
Finite Difference Example



Finite Difference Example



Finite Difference Example



Policy Search RL

- Finite differences is a rudimentary black-box method
  - Treats trajectory as a whole rather than individual steps
  - Does not exploit knowledge of policy's form
  - Requires many samples and is not very efficient

- Many (more advanced) policy search methods exist
  - Similar over all structure of sample-and-improve
  - Beyond the scope of this course…

Questions?