Carnegie Mellon
School of Computer Science

Deep Reinforcement Learning and Control

Imitation Learning II

Lecture 15, CMU 10703

Katerina Fragkiadaki

So far in the course

Reinforcement Learning: Learning policies guided by sparse rewards, e.g., win or not the game.

- Good: simplest, cheapest form of supervision
- Bad: High sample complexity

Where is it successful so far?

- in simulation, where we can afford a lot of trials, easy to parallelize
- not in robotic systems:
  1. action execution takes long
  2. we cannot afford to fail
  3. safety concerns

Crusher robot

Learning from Demonstration for Autonomous Navigation in Complex Unstructured

Reward shaping

Ideally we want dense in time rewards to closely guide the agent closely along the way.

Who will supply those shaped rewards?

1. We will manually design them: *“cost function design by hand remains one of the ’black arts’ of mobile robotics, and has been applied to untold numbers of robotic systems”*
2. We will learn them from demonstrations: *“rather than having a human expert tune a system to achieve desired behavior, the expert can demonstrate desired behavior and the robot can tune itself to match the demonstration”*

Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain, Silver et al. 2010

The Imitation Learning problem

The agent (learner) needs to come up with a policy whose resulting state, action trajectory distribution matches the expert trajectory distribution.

Imitation Learning

For taking this action interdependence into account, numerous formulations have been proposed:

- Direct: Supervised learning for policy (mapping states to actions) using the demonstration trajectories as ground-truth(a.k.a. behavior cloning) + ways to handle the neglect of action interdependence.

- Indirect: Learning the latent rewards/goals of the teacher and planning under those rewards to get the policy, a.k.a. Inverse Reinforcement Learning

Experts can be:

- Humans

- Optimal or near Optimal Planners/Controllers

Outline

Last lecture

- Behavior Cloning: Imitation learning as supervised learning
- Compounding errors
- Demonstration augmentation techniques
- DAGGER
- Structured prediction as Decision Making (learning to search)
- Imitating MCTS

This lecture:

- Inverse reinforcement learning
- Max margin planning
- Maximum entropy IRL
- Adversarial Imitation learning
- Value Iteration Networks

Inverse Reinforcement Learning

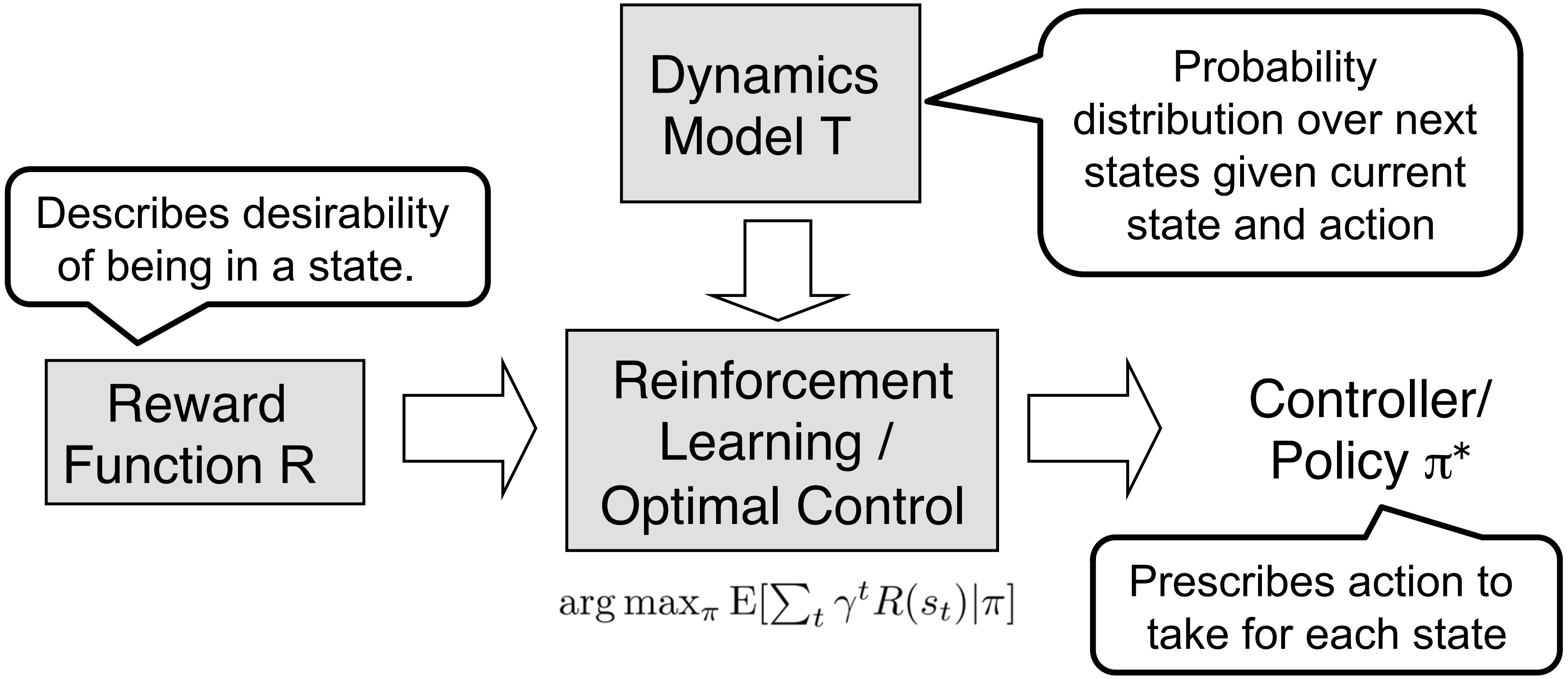

Dynamics Model T
Probability distribution over next states given current state and action
Describes desirability of being in a state.
Reward Function R
Reinforcement Learning / Optimal Control
Controller/ Policy π*
Prescribes action to take for each state
$\arg\max_\pi \mathrm{E}[\sum_t \gamma^t R(s_t)|\pi]$

Diagram: Pieter Abbeel

Given π, let's recover R!

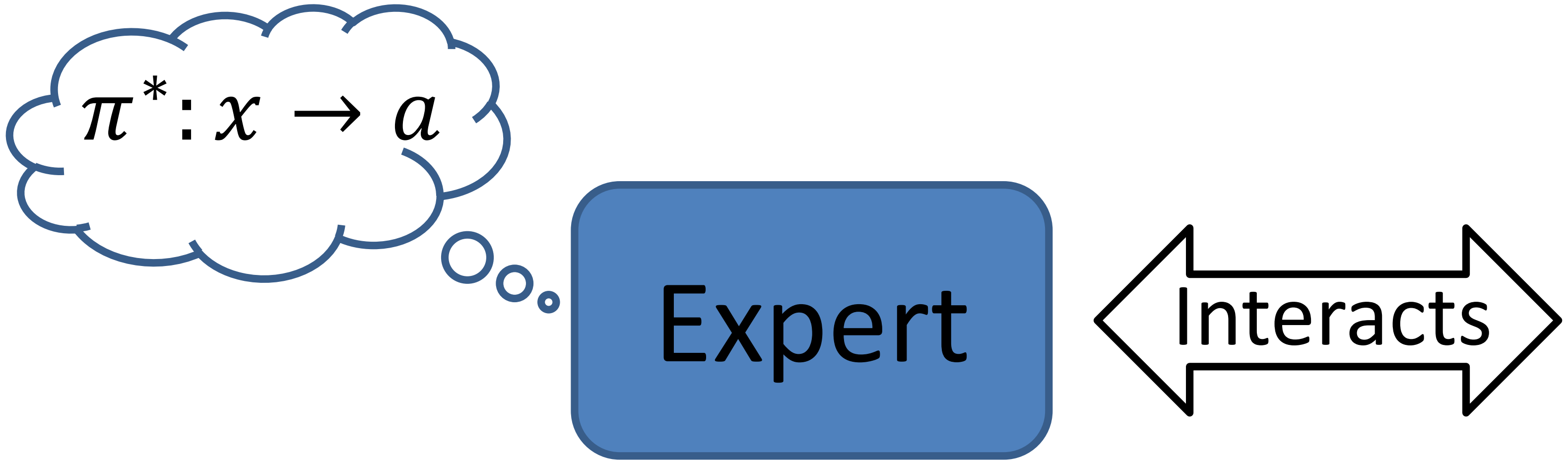
Problem Setup

- **Given**:
  - State space, action space
  - *No* reward function
  - Dynamics (sometimes) $T_{s,a}[s_{t+1}|s_t, a_t]$
  - Teacher's demonstration: $s_0, a_0, s_1, a_1, s_2, a_2, \ldots$ $(= \text{trace of the teacher's policy } \pi^*)$
- **Inverse RL**
  - Can we recover R?
- **Apprenticeship learning via inverse RL**
  - Can we then use this R to find a good policy?
- **Behavioral cloning (last lecture)**
  - Can we directly learn the teacher's policy using supervised learning?

Assumptions (for now)

- Known Dynamics (transition model) T
- Reward is a linear function over fixed state features ϕ

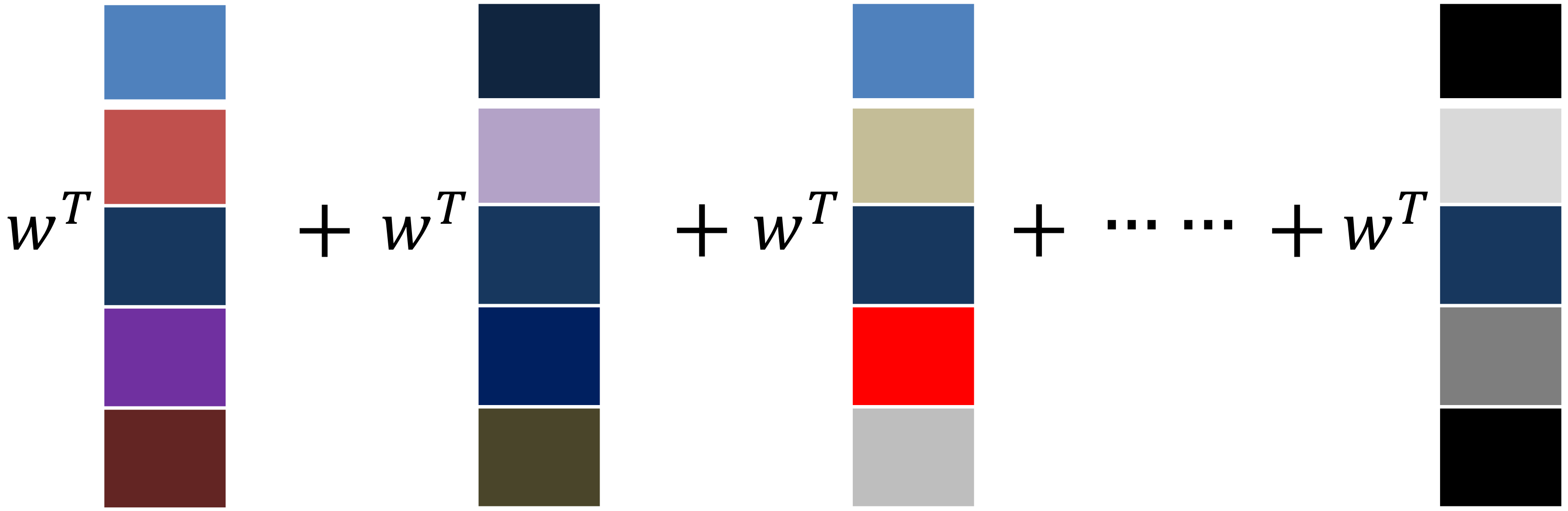
Inverse RL with linear costs/rewards



Expert trajectory

$$(x_1, a_1) \to (x_2, a_2) \to (x_3, a_3) \to \cdots \to (x_n, a_n)$$

Demonstration



Expert trajectory cost

Jain, Hu

Principle: Expert is optimal

- Find a reward function R^* which explains the expert behavior
- Find R^* such that

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi^*] \geq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi] \quad \forall \pi$$

Feature Based Reward Function

(We assume reward is linear over features)

Let $R(s) = w^T \phi(s)$, where $w \in \mathbb{R}^n$, and $\phi : S \to \mathbb{R}^n$

$$\begin{aligned}\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi] &= \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t w^T \phi(s_t)|\pi] \\ &= w^T \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] \\ &= w^T \boxed{\mu(\pi)}\end{aligned}$$

expected discounted sum of feature values or feature expectations—dependent on state visitation distributions

Sub/ting into $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi^*] \geq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi] \quad \forall \pi$

gives us: Find w^* such that ${w^*}^T \mu(\pi^*) \geq {w^*}^T \mu(\pi) \quad \forall \pi$

Challenges: Reward function is ambiguous

(We assume reward is linear over features)

Let $R(s) = w^T \phi(s)$, where $w \in \mathbb{R}^n$, and $\phi : S \rightarrow \mathbb{R}^n$

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi] = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t w^T \phi(s_t) | \pi]$$

$$= w^T \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | \pi]$$

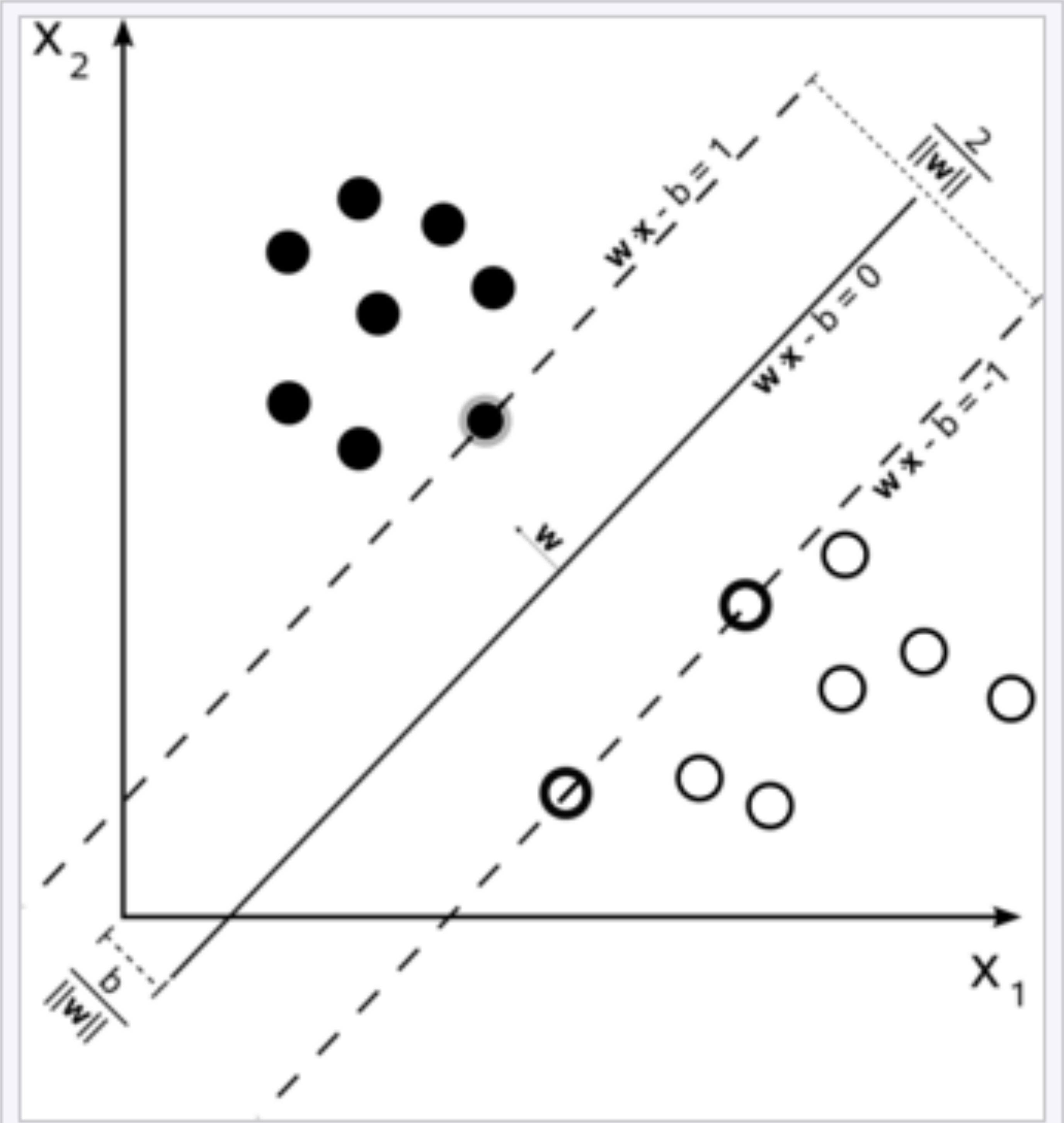
$$= w^T \boxed{\mu(\pi)}$$

expected discounted sum of feature values or feature expectations—dependent on state visitation distributions

Sub/ting into $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) | \pi^*] \geq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) | \pi] \quad \forall \pi$

gives us: Find w^* such that $w^{*T} \mu(\pi^*) \geq w^{*T} \mu(\pi) \quad \forall \pi$

Max-margin Classifiers


X_2
$\mathbf{w}\,\mathbf{x} - b = 1$
$\frac{2}{\|\mathbf{w}\|}$
$\mathbf{w}\,\mathbf{x} - b = 0$
$\mathbf{w}\,\mathbf{x} - b = -1$
$\mathbf{w}$
$\frac{b}{\|\mathbf{w}\|}$
X_1

"Minimize $\|\vec{w}\|$ subject to $y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1$, for $i = 1, \ldots, n$"

Max-margin Classifiers

- We are given a training dataset of n points of the form

$$(\vec{x}_1, y_1), ..., (\vec{x}_n, y_n)$$

- Where the y_i are either 1 or -1, each indicating the class to which the point $\vec{x}_i$ belongs. Each $\vec{x}_i$ is a p-dimensional real vector. We want to find the “maximum-margin hyperplane” that divides the group of points $\vec{x}_i$, for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point $\vec{x}_i$ from either group is maximized.

- Any hyperplane can be written as the set of points $\vec{x}$ satisfying

$$\vec{w} \cdot \vec{x} - b = 0$$

where $\vec{w}$ is the normal vector the the hyperplane

Max-margin Classifiers

- Let x_1 be any point in the first hyperplane and consider the line L that passes through x_1 in the direction of the normal vector a. An equation for L is given by $x_1 + at$ for all $t \in \mathbb{R}$. Now find the intersection of L and the second hyperplane:

$$a^T(x + at) = b_2 \iff t = (b_2 - a^T x_1)/a^T a = (b_2 - b_1)/a^T a$$

- Therefore the intersection point is $x_2 = x_1 + a(b_2 - b_1)/a^T a$. The distance between these two points is the distance between the hyperplanes:

$$\|x_1 - x_2\| = \frac{|b_2 - b_1|}{a^T a}\|a\| = \frac{|b_2 - b_1|}{\|a\|}$$

Max Margin Planning

- Standard max margin:

$$\min_{w} \|w\|_2^2$$
$$\text{s.t.} \quad w^T \mu(\pi^*) \geq w^T \mu(\pi) + 1 \quad \forall \pi$$

Maximum Margin Planning, Ratliff et al. 2006

Max Margin Planning

- Standard max margin:

$$\min_w \|w\|_2^2$$
$$\text{s.t.} \quad w^T \mu(\pi^*) \geq w^T \mu(\pi) + 1 \quad \forall \pi$$

- "Structured prediction" max margin:

$$\min_w \|w\|_2^2$$
$$\text{s.t.} \quad w^T \mu(\pi^*) \geq w^T \mu(\pi) + m(\pi^*, \pi) \quad \forall \pi$$

- Justification: margin should be larger for policies that are very different from π^*

- Example: $m(\pi^*, \pi) =$ number of states in which $\pi^*$and π disagree

Maximum Margin Planning, Ratliff et al. 2006

Expert Suboptimality

- Structured prediction max margin with slack variables:

$$\min_{w,\xi} \|w\|_2^2 + C\xi$$

$$\text{s.t.} \quad w^T\mu(\pi^*) \geq w^T\mu(\pi) + m(\pi^*, \pi) - \xi \quad \forall \pi$$

- Can be generalized to multiple MDPs (could also be same MDP with different initial state)

$$\min_{w,\xi^{(i)}} \|w\|_2^2 + C\sum_i \xi^{(i)}$$

$$\text{s.t.} \quad w^T\mu(\pi^{(i)*}) \geq w^T\mu(\pi^{(i)}) + m(\pi^{(i)*}, \pi^{(i)}) - \xi^{(i)} \quad \boxed{\forall i, \pi^{(i)}}$$

Complete Max-margin Formulation

$$\min_w \|w\|_2^2 + C \sum_i \xi^{(i)}$$

$$\text{s.t.} \quad w^T \mu(\pi^{(i)*}) \geq w^T \mu(\pi^{(i)}) + m(\pi^{(i)*}, \pi^{(i)}) - \xi^{(i)} \quad \forall i, \pi^{(i)}$$

- Challenge: very large number of constraints. Solutions:
  - iterative constraint generation

Maximum Margin Planning, Ratliff et al. 2006

Constraint Generation

- Iterate $\Pi^{(i)} = \{\}$ for all i and then iterate
  - Solve $$\min_w \|w\|_2^2 + C \sum_i \xi^{(i)}$$

$$\text{s.t.} \quad w^T \mu(\pi^{(i)*}) \geq w^T \mu(\pi^{(i)}) + m(\pi^{(i)*}, \pi^{(i)}) - \xi^{(i)} \quad \forall i, \pi^{(i)} \in \Pi^{(i)}$$

- For current w, find most violated constraint for all i by solving:

$$\boxed{\max_{\pi^{(i)}} w^T \mu(\pi^{(i)}) + m(\pi^{(i)*}, \pi^{(i)})}$$

- for all i add $\pi^{(i)}$ to $\Pi^{(i)}$

- If no constraint violations were found, we are done

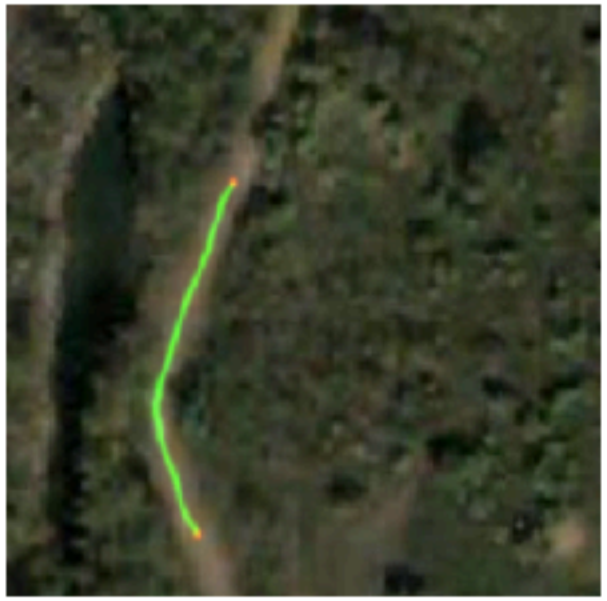
Assumes an RL algorithm that can find the optimal policy for a given reward function! Nested RL problem. However, as we assumed dynamics known, is more like a nested planning problem.

Maximum Margin Planning, Ratliff et al. 2006

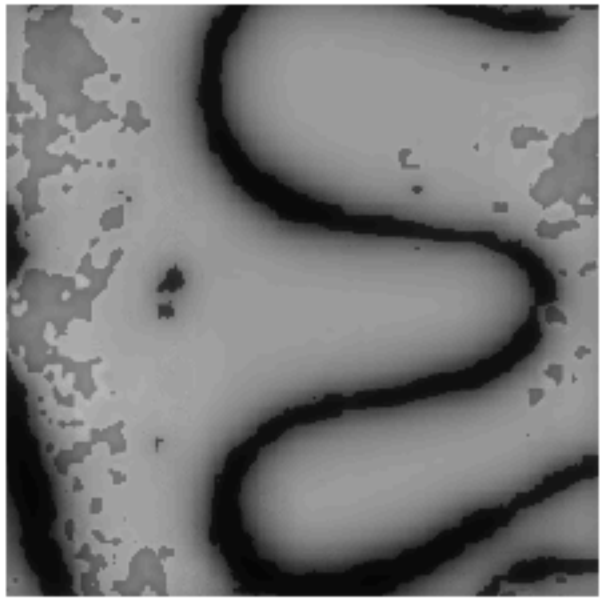
Max Margin Planning

trained to follow roads

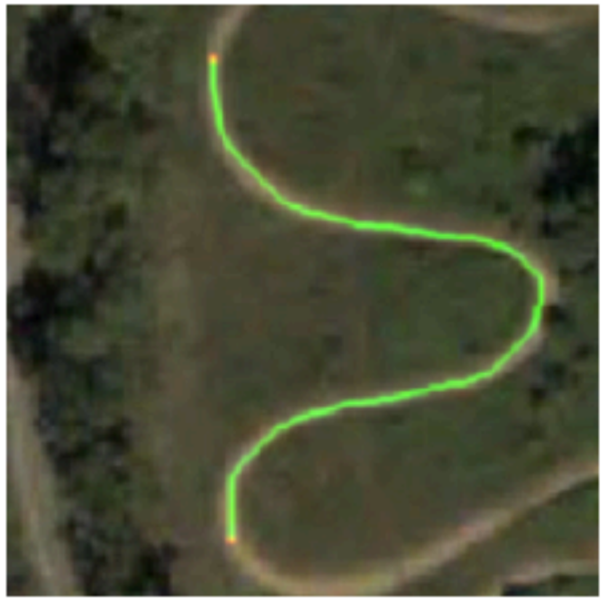
mode 1 - training



mode 1 - learned cost map over novel region

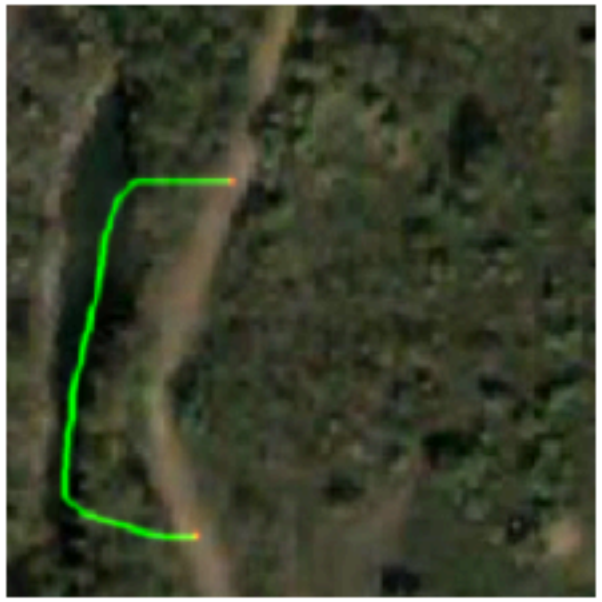


mode 1 - learned path over novel region

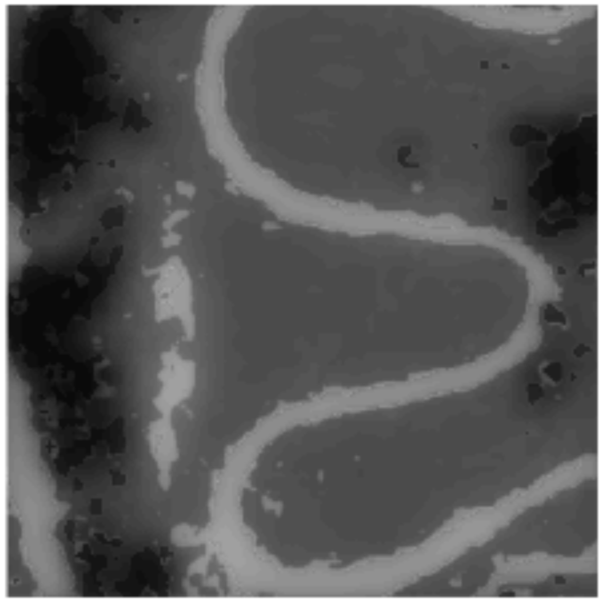


trained to hide in the trees

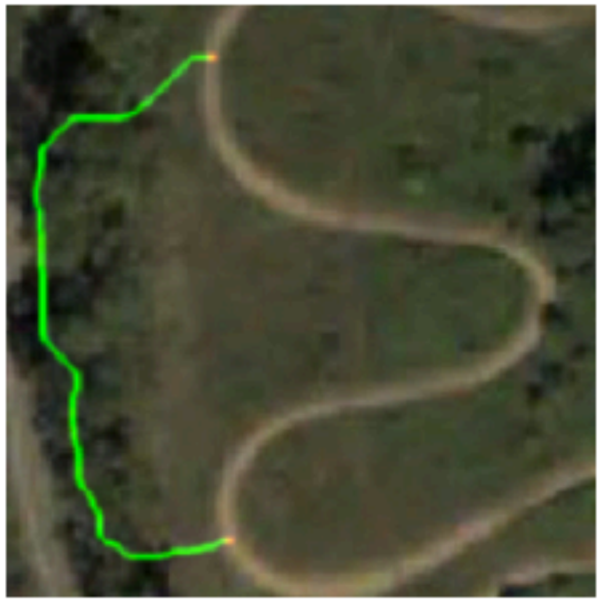
mode 2 - training



mode 2 - learned cost map over novel region



mode 2 - learned path over novel region

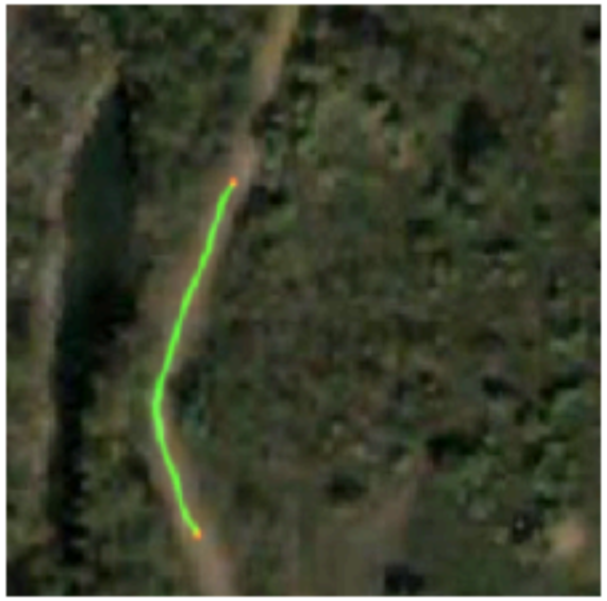


Maximum Margin Planning, Ratliff et al. 2006

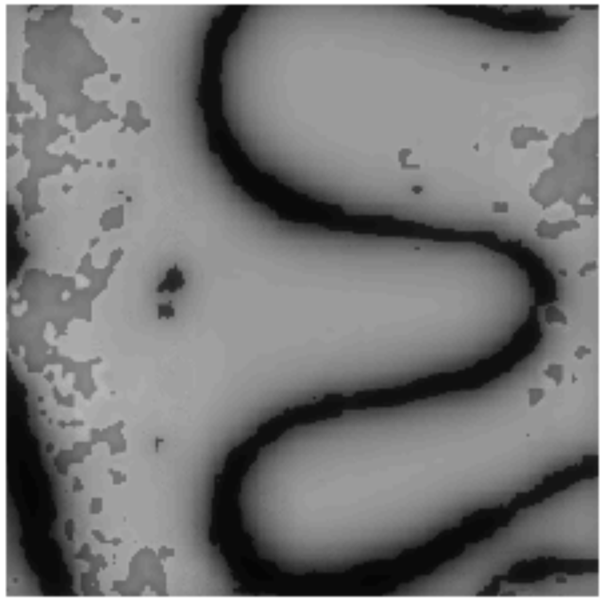
Q: Why don't we train a CNN given a patch to predict the trajectory?

trained to follow roads

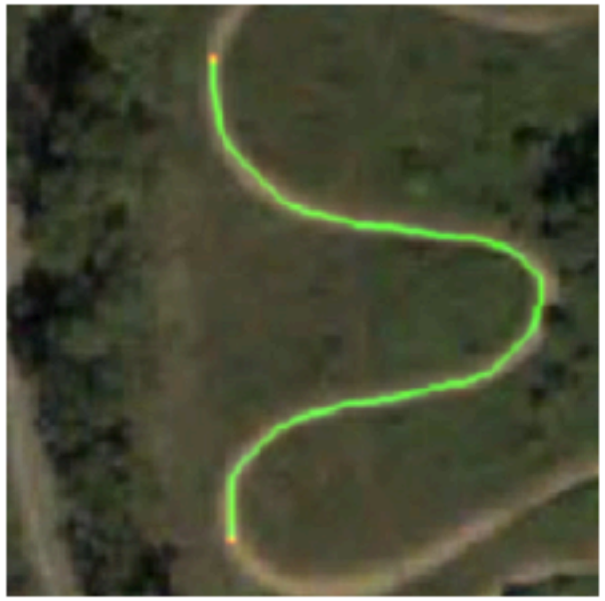
mode 1 - training



mode 1 - learned cost map over novel region

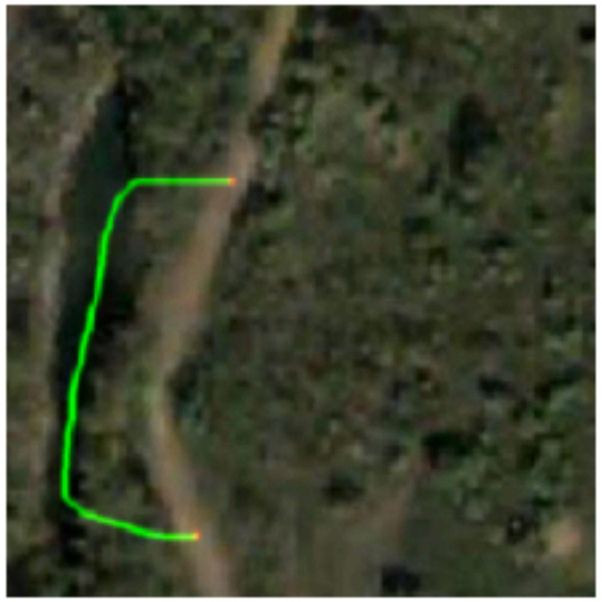


mode 1 - learned path over novel region

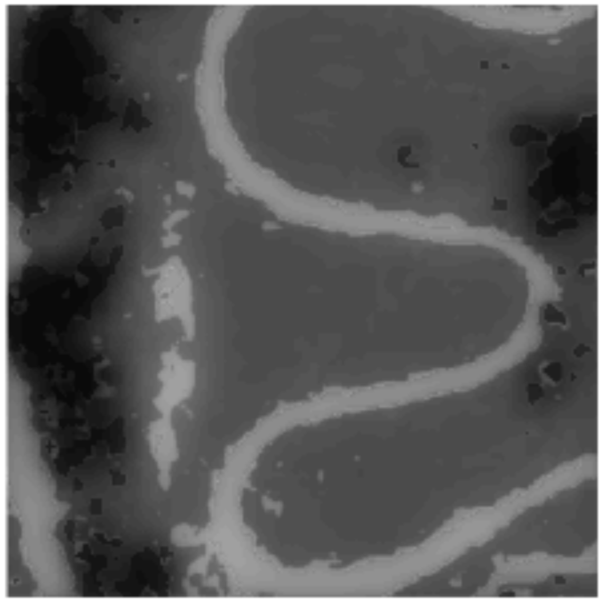


trained to hide in the trees

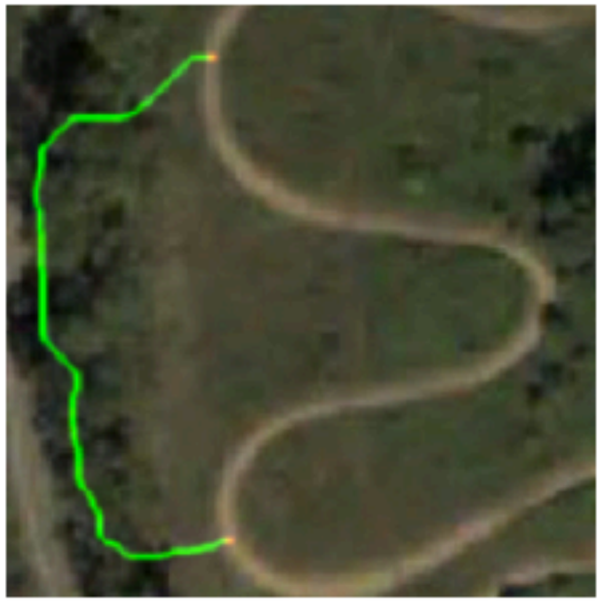
mode 2 - training



mode 2 - learned cost map over novel region



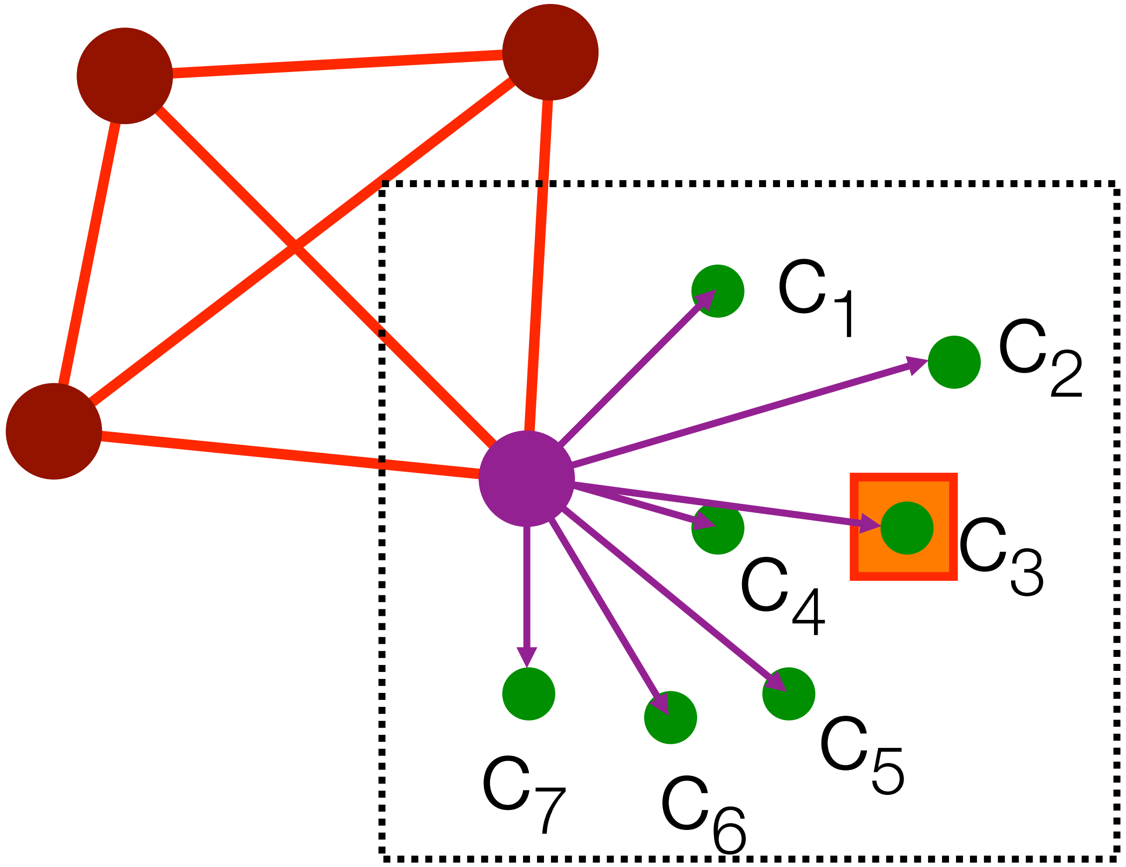
mode 2 - learned path over novel region

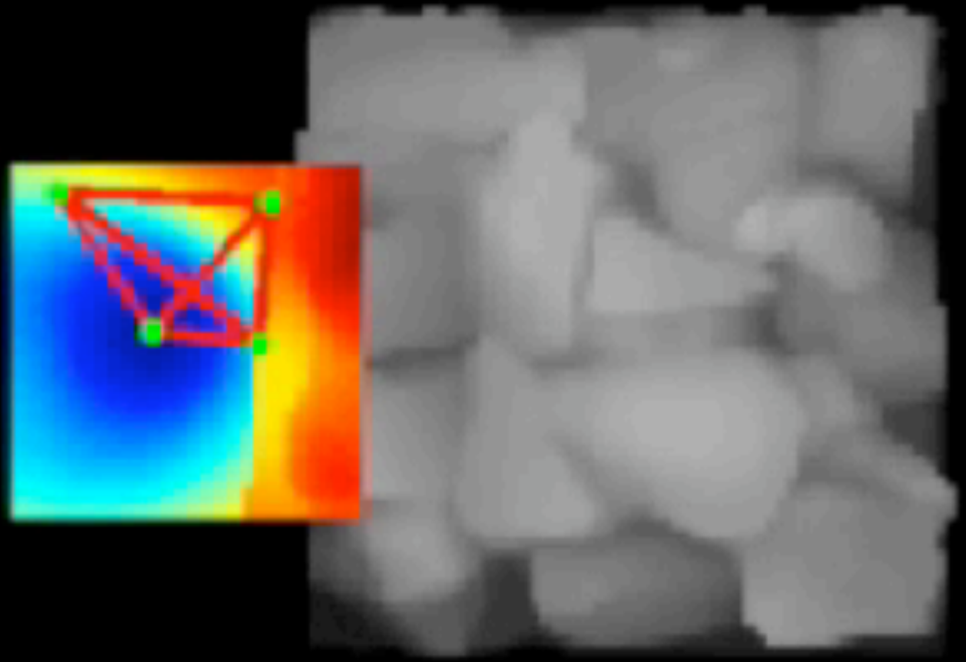


Maximum Margin Planning, Ratliff et al. 2006

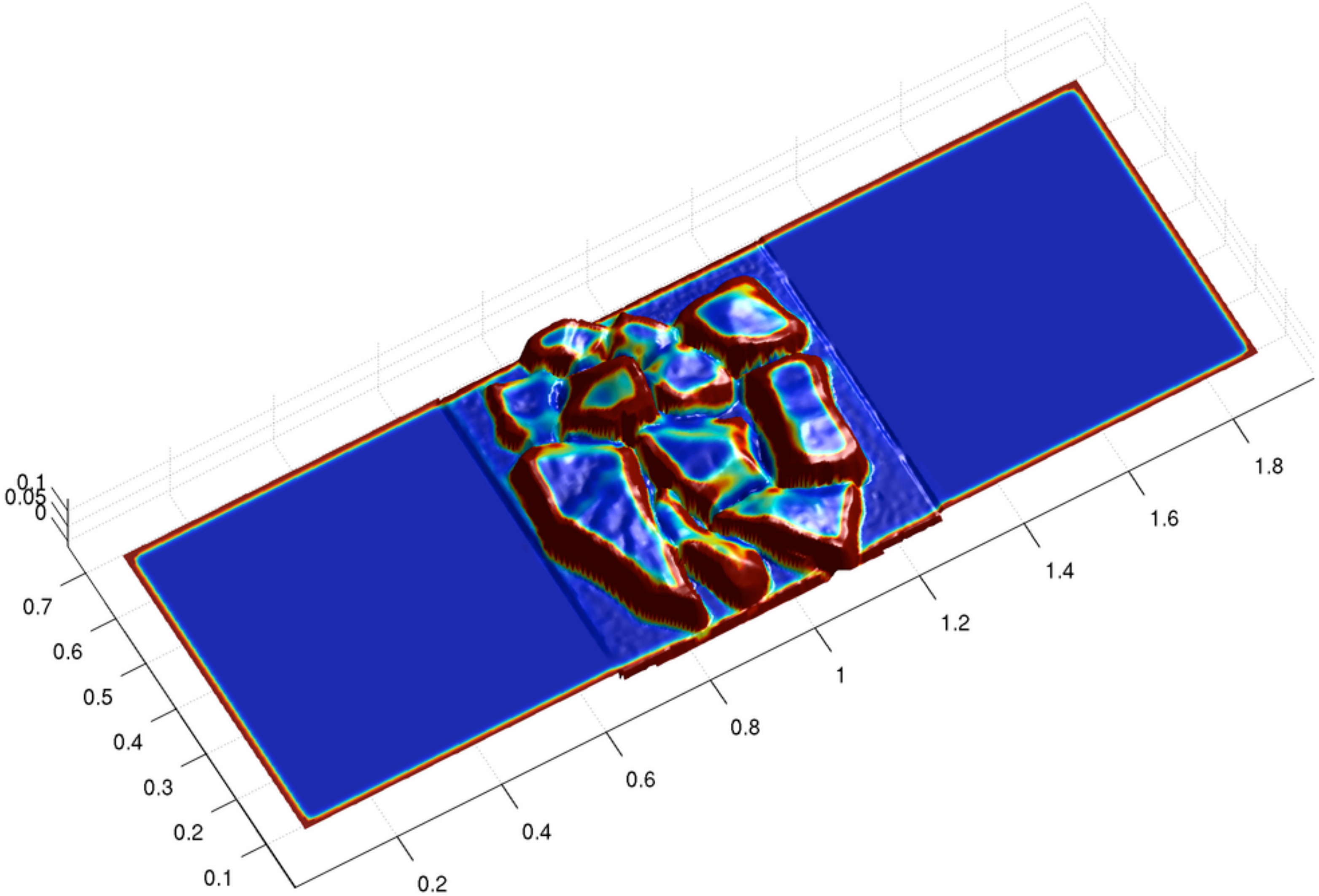
Learning to step (mimicking footsteps)

Where should we place the foot next?

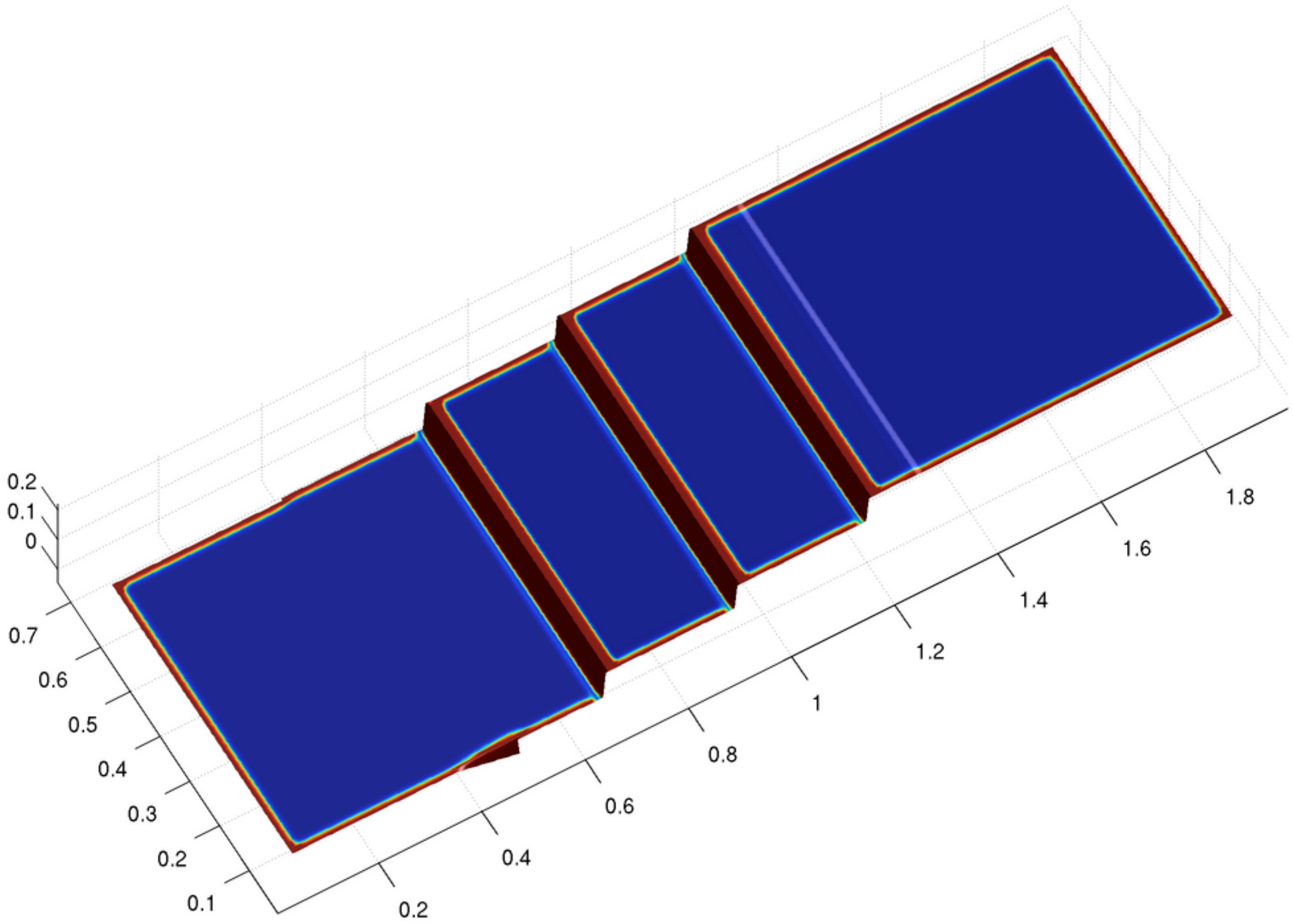




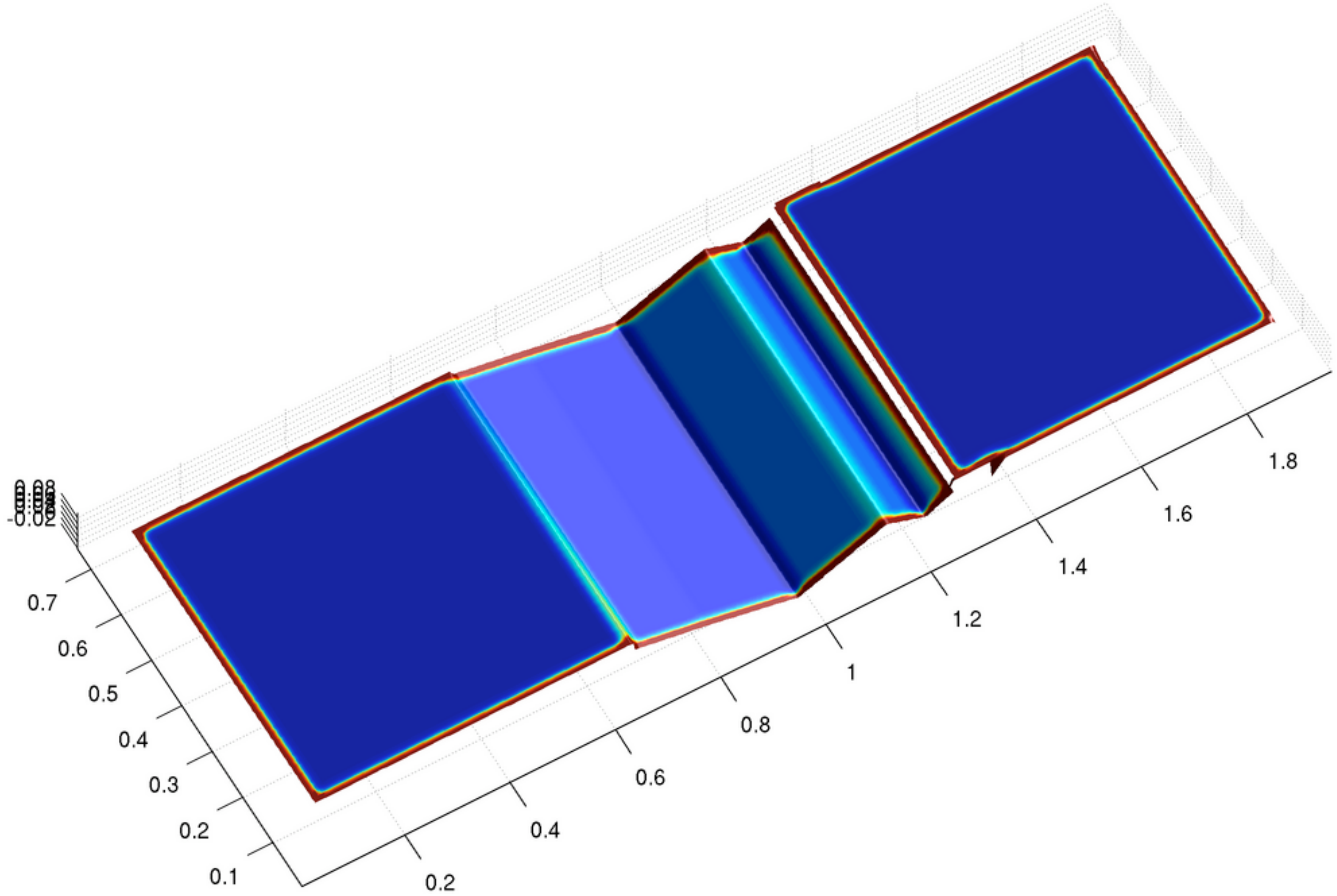
Learned Cost Function Examples



Learned Cost Function Examples



Learned Cost Function Examples



https://www.youtube.com/watch?v=mKLRNIlChrk

Feature Matching

- Inverse RL starting point: find a reward function such that the expert outperforms other policies

$$\text{Let } R(s) = w^T\phi(s), \text{ where } w \in \mathbb{R}^n, \text{ and } \phi : S \to \mathbb{R}^n$$

$$\text{Find } w^* \text{ such that } {w^*}^T\mu(\pi^*) \geq {w^*}^T\mu(\pi) \quad \forall\pi$$

- Observation in Abbeel and Ng, 2004: for a policy π to be guaranteed to perform as well as the expert policy $\mu*$, it suffices that the feature expectations match:

$$\|\mu(\pi) - \mu(\pi^*)\|_1 \leq \epsilon$$

Implies that for all w with $\|w\|_\infty \leq 1$:

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

Abbeel and Ng 2004

Apprenticeship Learning [Abbeel & Ng, 2004]

- Assume $R_w(s) = w^T\phi(s)$ for a feature map $\phi : S \to \mathbb{R}^n$
- Initialize: pick some policy π_0
- Iterate for $i = 1, 2, \ldots$:
  - **“Guess” the reward function:**

    Find a reward function such that the teacher maximally outperforms all previously found policies

$$\max_{\gamma, w : \|w\|_2 \leq 1} \gamma$$

$$\text{s.t.} \quad w^T\mu(\pi^*) \geq w^T\mu(\pi) + \gamma \quad \forall \pi \in \{\pi_0, \pi_1, \ldots, \pi_{i-1}\}$$

  - **Find optimal control policy** π for the current guess of the reward function R_w
  - $\gamma \leq \varepsilon/2$ exit the algorithm

Hmmmm….. Ambiguity again

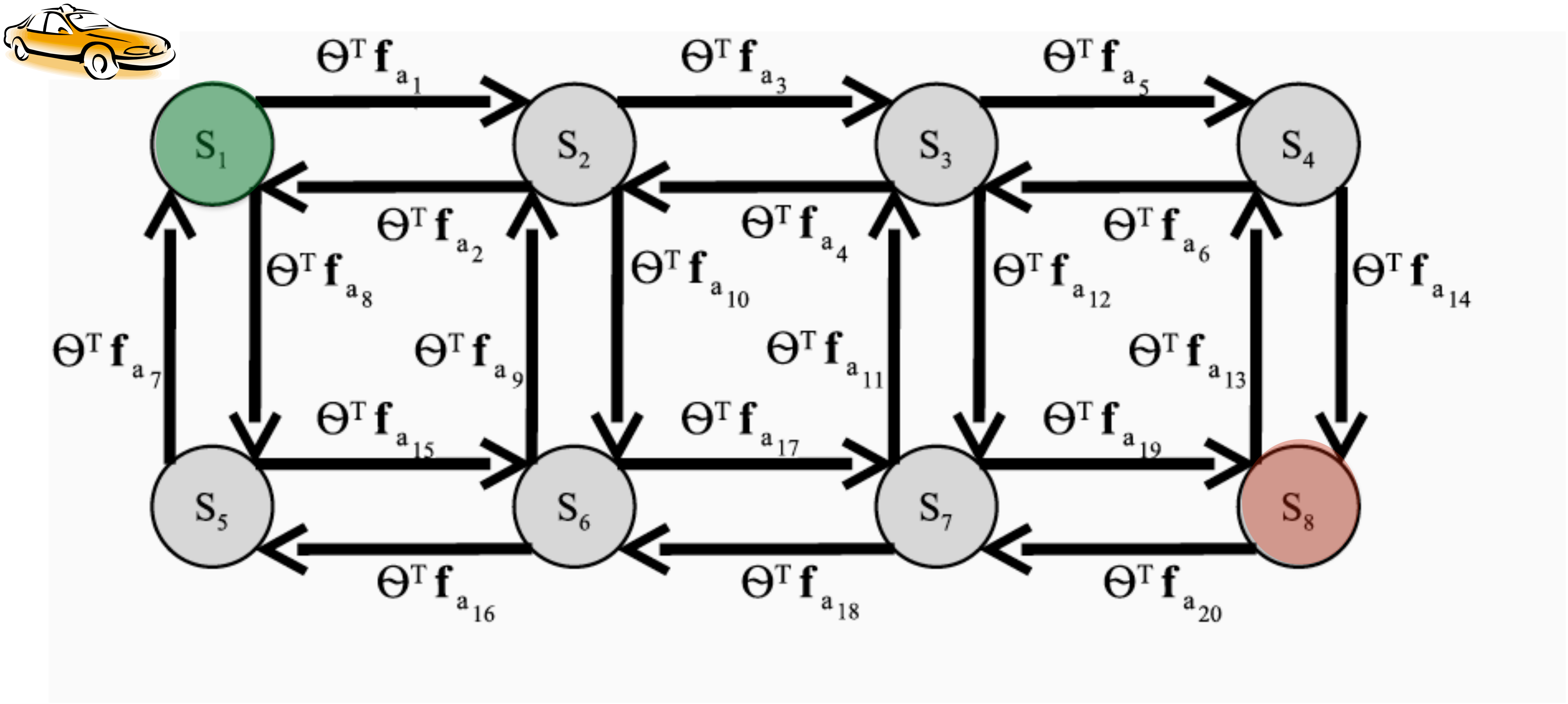
- There is **no reward function and no optimal policy** that matches that matches almost all behavior

- There are **infinitely many** stochastic behaviors (policies or mixtures of policies) that can match feature counts….

- **How can we possibly pick a good one?**

Maximum Entropy Inverse Optimal Control

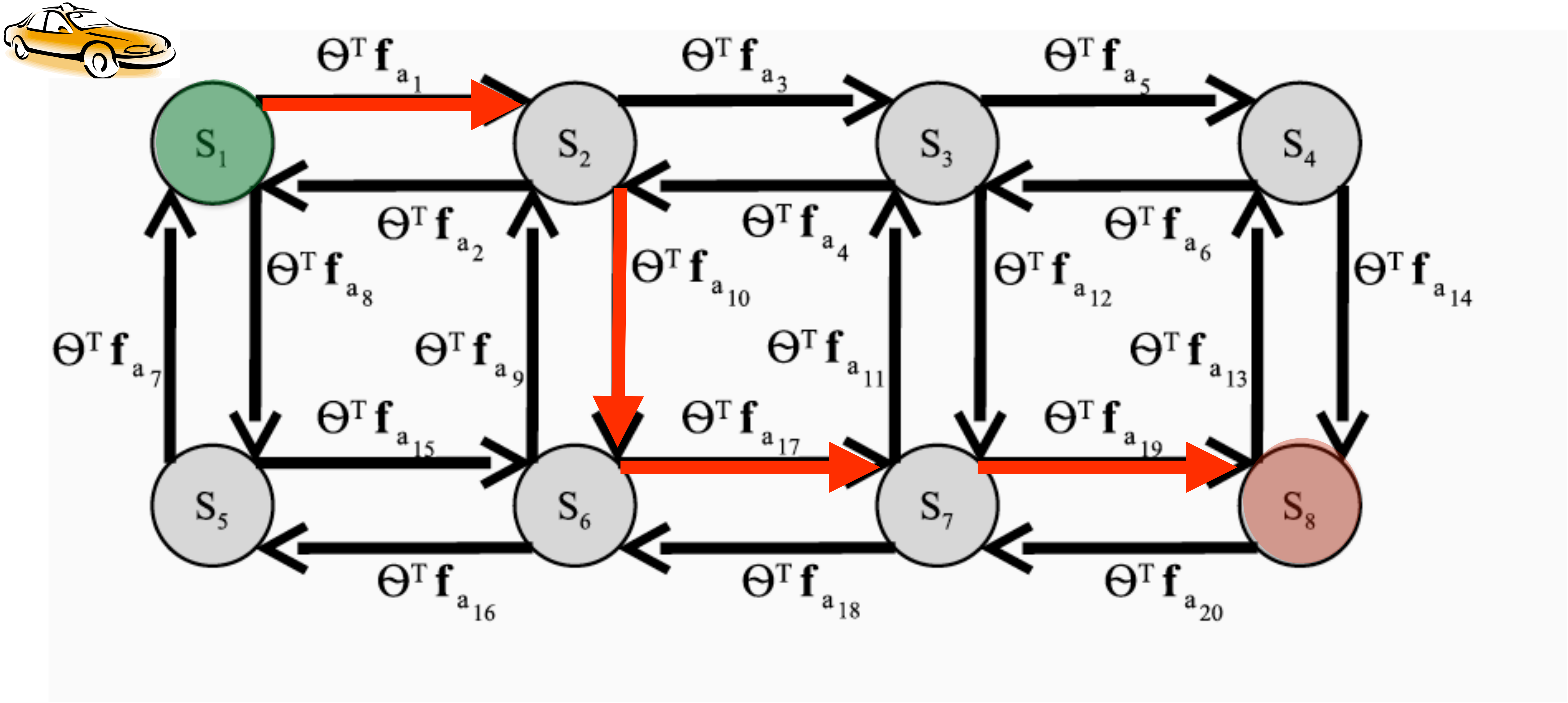
Roads have unknown costs linear in features



Maximum Entropy Inverse Optimal Control

Roads have unknown costs linear in features

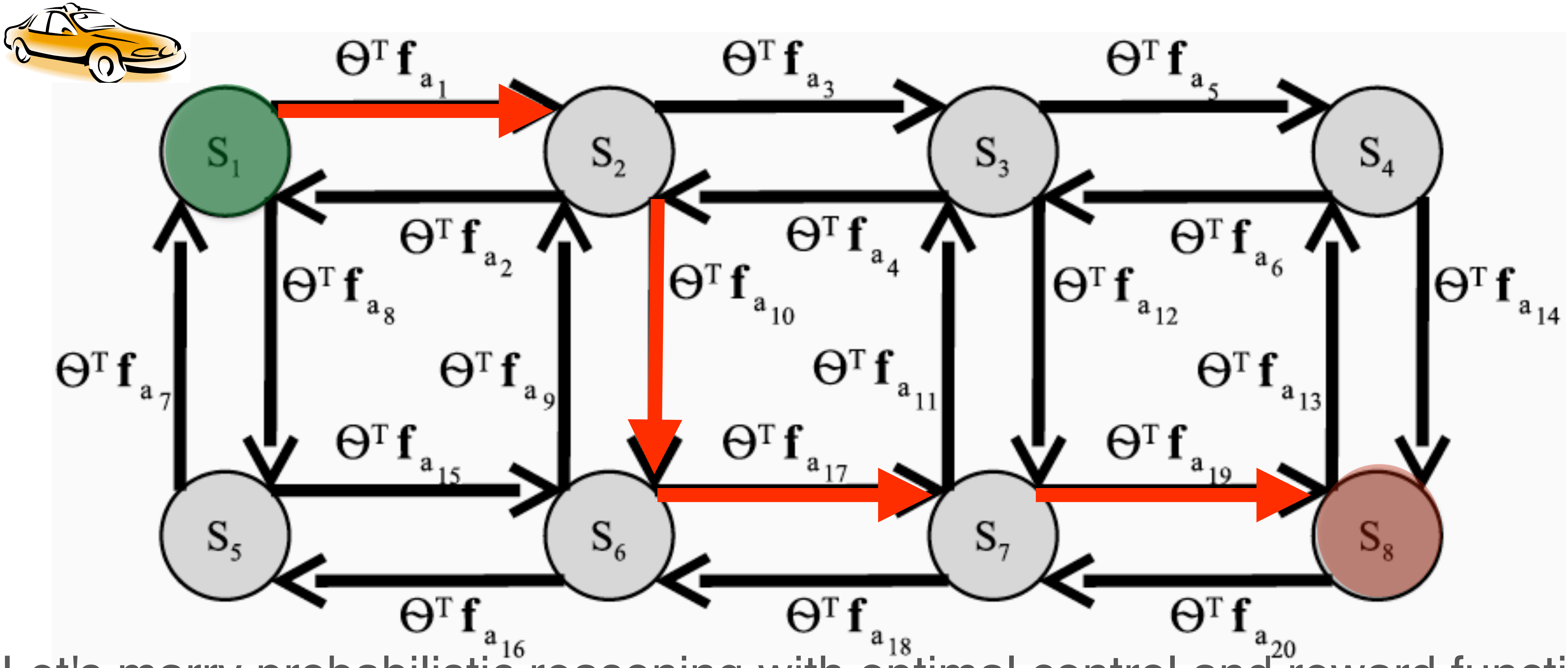
Paths have unknown costs, sum of road costs



Maximum Entropy Inverse Optimal Control

Roads have unknown costs linear in features

Paths have unknown costs, sum of road costs



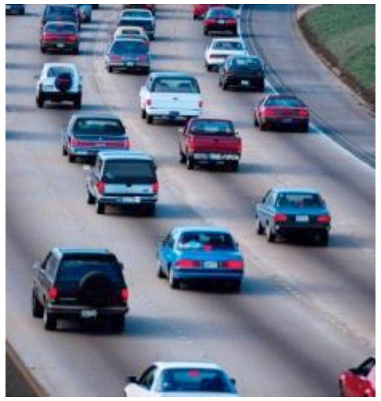
Let's marry probabilistic reasoning with optimal control and reward functions:

- the costs induce a distribution over paths! P(\tau)
- path probability based on unknown cost

Feature matching using path probabilities

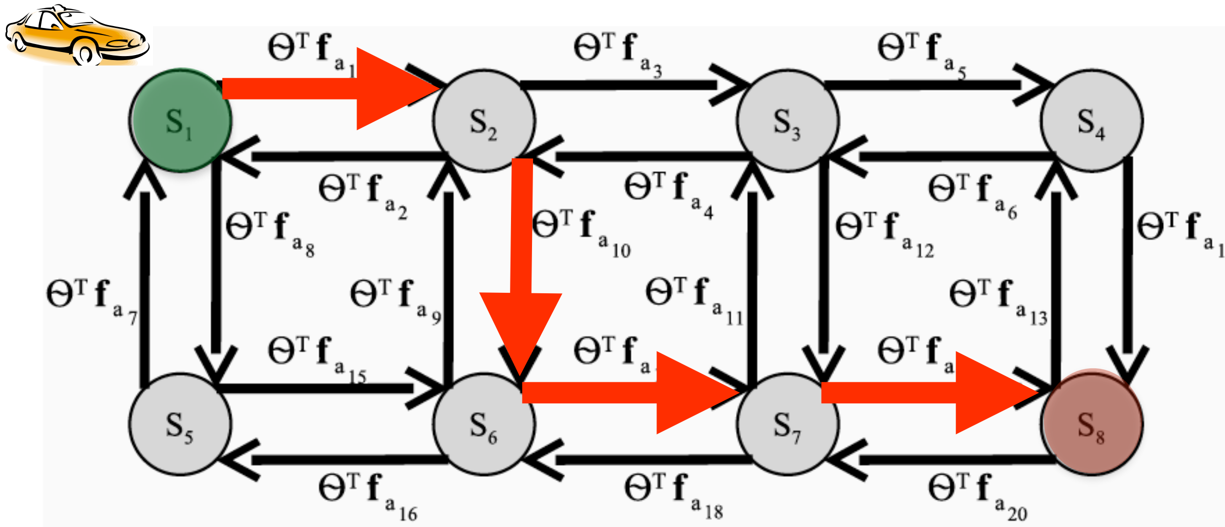
Features f can be:

# Bridges crossed



# Miles of interstate

# Stoplights


S_1
S_2
S_3
S_4
S_5
S_6
S_7
S_8
$\Theta^T \mathbf{f}_{a_1}$
$\Theta^T \mathbf{f}_{a_3}$
$\Theta^T \mathbf{f}_{a_5}$
$\Theta^T \mathbf{f}_{a_2}$
$\Theta^T \mathbf{f}_{a_4}$
$\Theta^T \mathbf{f}_{a_6}$
$\Theta^T \mathbf{f}_{a_8}$
$\Theta^T \mathbf{f}_{a_{10}}$
$\Theta^T \mathbf{f}_{a_{12}}$
$\Theta^T \mathbf{f}_{a_1}$
$\Theta^T \mathbf{f}_{a_7}$
$\Theta^T \mathbf{f}_{a_9}$
$\Theta^T \mathbf{f}_{a_{11}}$
$\Theta^T \mathbf{f}_{a_{13}}$
$\Theta^T \mathbf{f}_{a_{15}}$
$\Theta^T \mathbf{f}_{a}$
$\Theta^T \mathbf{f}_{a}$
$\Theta^T \mathbf{f}_{a_{16}}$
$\Theta^T \mathbf{f}_{a_{18}}$
$\Theta^T \mathbf{f}_{a_{20}}$

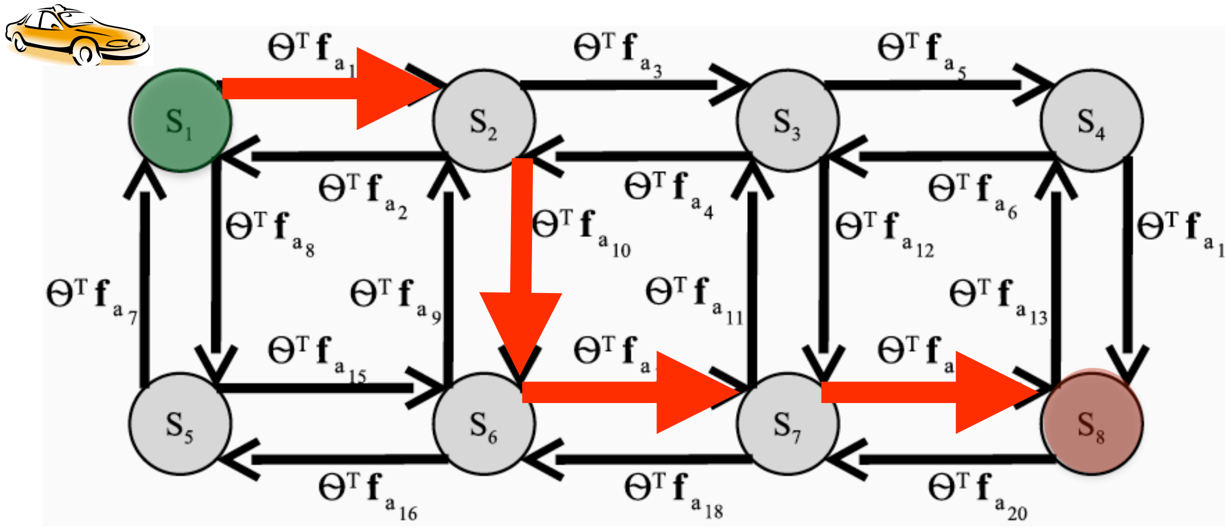
Feature matching:

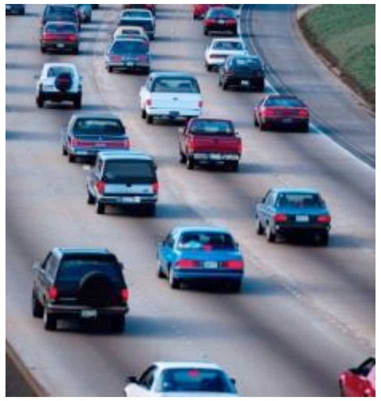
$$\sum_{\text{Path}\,\tau_i} P(\tau_i) f_{\tau_i} = \tilde{\text{f}}$$

Which path distribution to pick?

Features f can be:

# Bridges crossed





# Miles of interstate

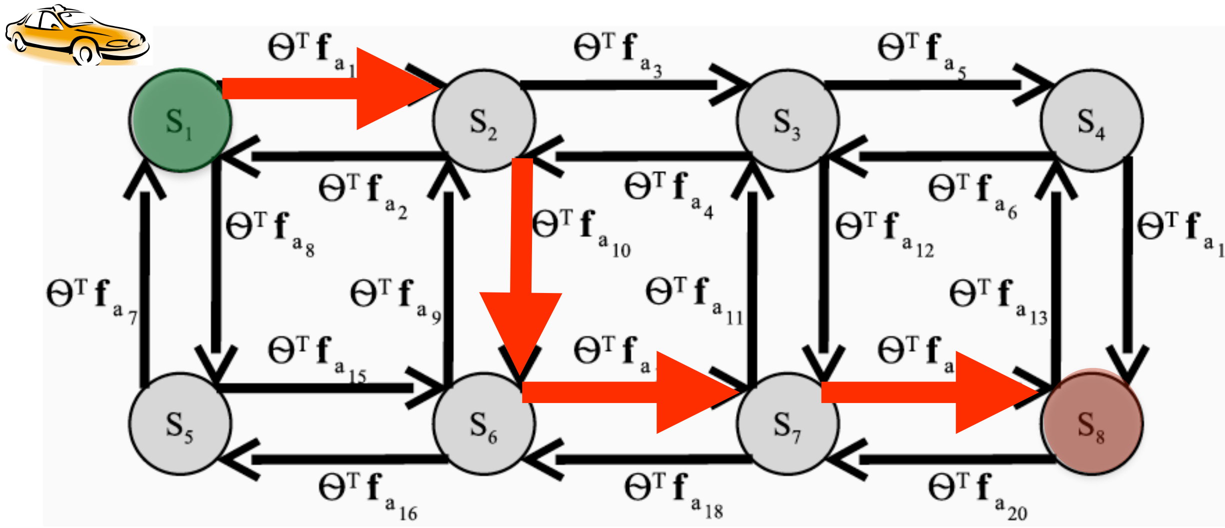
Feature matching:

$$\sum_{\text{Path}\tau_i} P(\tau_i) f_{\tau_i} = \tilde{\text{f}}$$

# Stoplights

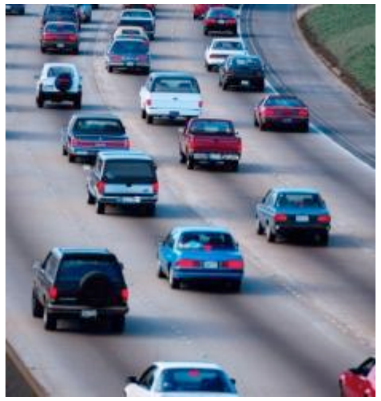
"If a driver uses136.3 miles of interstate and crosses 12 bridges in a month's worth of trips, the model should also use 136.3 miles of interstate and 12 bridges in expectation for those same start-destination pairs."

Which path distribution to pick?



Features f can be:

# Bridges crossed



# Miles of interstate

Feature matching:

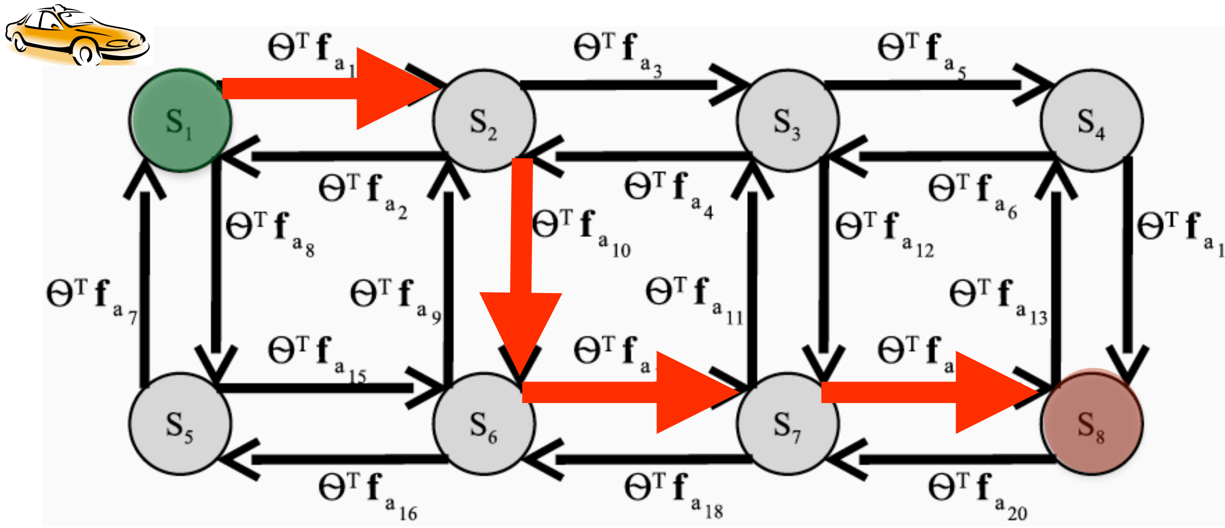
$$\sum_{\text{Path}\tau_i} P(\tau_i) f_{\tau_i} = \tilde{\text{f}}$$

# Stoplights

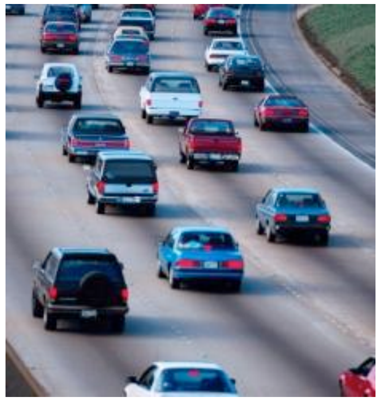
“Many distributions over paths can match feature counts, and some will be very different from observed behavior. In our simple example, the model could produce plans that avoid the interstate and bridges for all routes except one, which drives in circles on the interstate for 136 miles and crosses 12 bridges”

Which path distribution to pick?

Features f can be:



# Bridges crossed



# Miles of interstate

Feature matching:

$$\sum_{\text{Path}\,\tau_i} P(\tau_i) f_{\tau_i} = \tilde{\mathrm{f}}$$

# Stoplights

The one that satisfies feature count constraints without over-committing!

Maximum Entropy Inverse Optimal Control

- Maximizing the entropy over paths:

As Uniform As possible

$$\boxed{\max_P - \sum_\tau P(\tau) \log P(\tau)}$$

- While matching feature counts (and being a probability distribution):

$$\sum_\tau P(\tau) f_\tau = f_{\text{dem}}$$

$$\sum_\tau P(\tau) = 1$$

Maximum Entropy Principle

- Maximizing the entropy of the distribution over paths subject to the feature constraints from observed data implies that we maximize the likelihood of the observed data under the maximum entropy (exponential family) distribution (Jaynes 1957)

$$P(\tau_i|\theta) = \frac{1}{Z(\theta)} e^{\theta^T f_{\tau_i}} = \frac{1}{Z(\theta)} e^{\sum_{s_j \in \tau_i} \theta^T f_{s_j}}$$

$$Z(\theta, s) = \sum_{\tau_S} e^{\theta^T f_{\tau_S}}$$

Strong Preference for Low Cost Paths
Equal Cost Paths Equally Probable

MaxEntIOC: Learning \theta

- Maximizing the entropy of the distribution over paths subject to the feature constraints from observed data implies that we maximize the likelihood of the observed data under the maximum entropy (exponential family) distribution (Jaynes 1957)

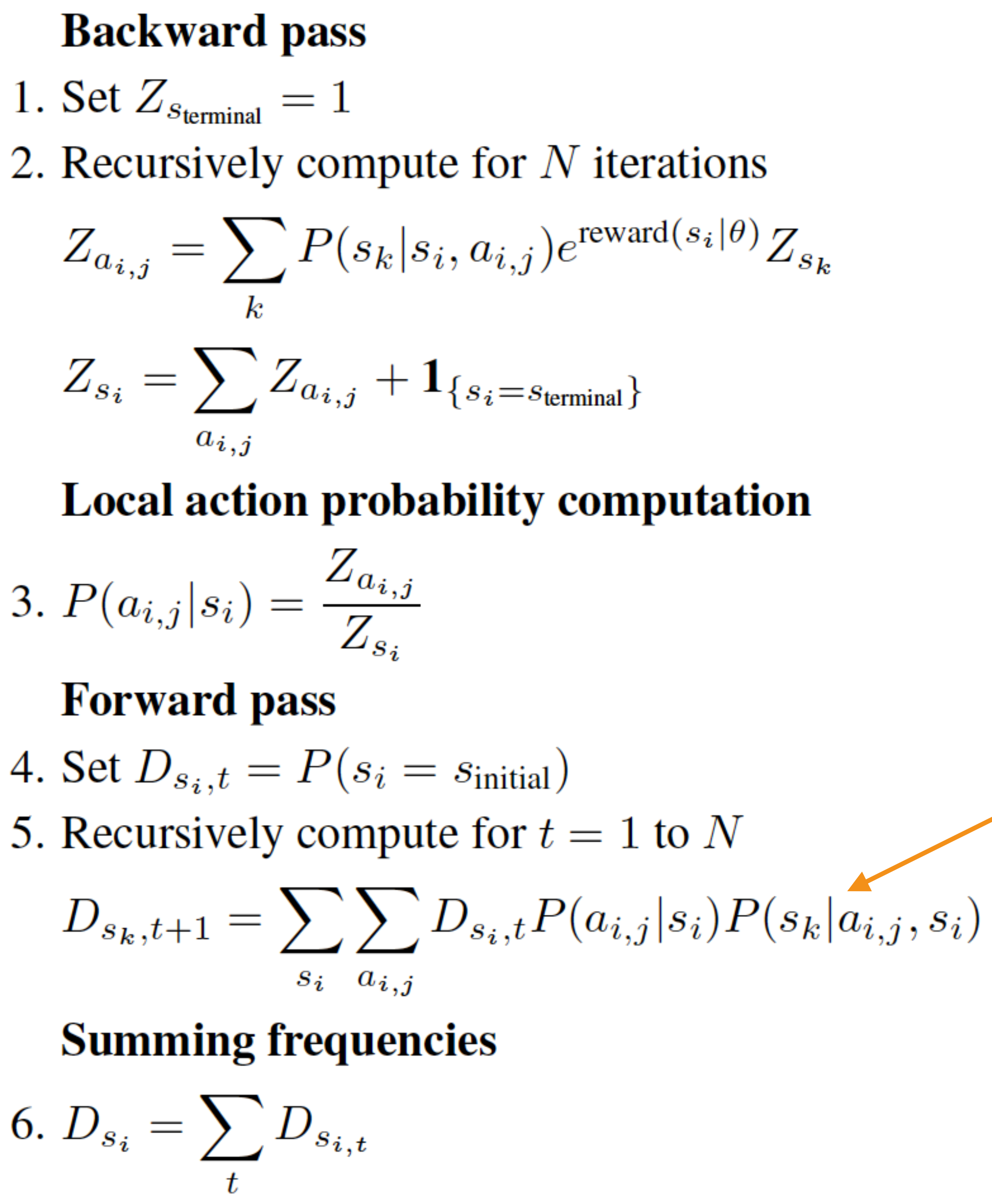
$$\theta^* = \arg\max_\theta L(\theta) = \arg\max_\theta \sum_{\text{examples}} \log P(\tilde{\tau}|\theta)$$

- The gradient is the difference between expected empirical feature counts and the learner's expected feature counts, which can be expressed in terms of expected state visitation frequencies,

$$\nabla L(\theta) = \tilde{f} - \sum_{\tau} P(\tau|\theta) f_\tau = \tilde{f} - \sum_{s_i} D_{s_i} f_{s_i}$$

state visitation frequencies!

Dynamic Programming for State visitation frequencies



Backward pass

1. Set $Z_{s_{\text{terminal}}} = 1$

2. Recursively compute for N iterations

$$Z_{a_{i,j}} = \sum_k P(s_k | s_i, a_{i,j}) e^{\text{reward}(s_i|\theta)} Z_{s_k}$$

$$Z_{s_i} = \sum_{a_{i,j}} Z_{a_{i,j}} + \mathbf{1}_{\{s_i = s_{\text{terminal}}\}}$$

Local action probability computation

3. $P(a_{i,j}|s_i) = \dfrac{Z_{a_{i,j}}}{Z_{s_i}}$

Forward pass

4. Set $D_{s_i,t} = P(s_i = s_{\text{initial}})$

5. Recursively compute for $t = 1$ to N

$$D_{s_k,t+1} = \sum_{s_i} \sum_{a_{i,j}} D_{s_i,t} P(a_{i,j}|s_i) P(s_k|a_{i,j}, s_i)$$

Summing frequencies

6. $D_{s_i} = \sum_t D_{s_i,t}$

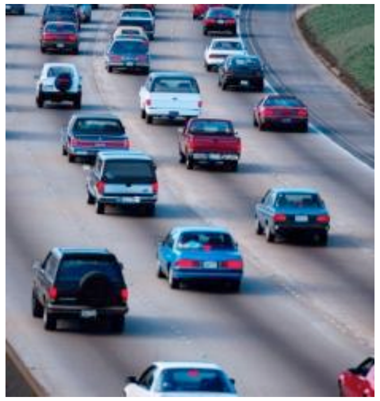
Dynamics are Known!

Learning from Demonstration

Demonstrated Behavior

Bridges crossed: **3**

Miles of interstate: **20.7**



Stoplights: **10**

Model Behavior (Expectation)

Bridges crossed: **?**

Miles of interstate: **?**

Cost Weight: 5.0

Cost Weight: 3.0

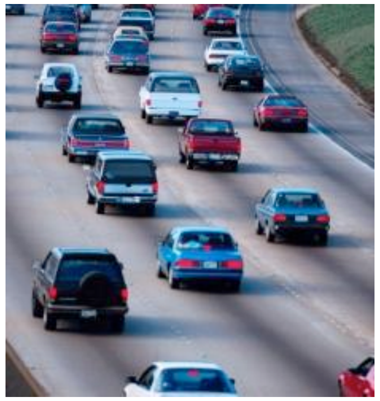
Stoplights
:
?

Learning from Demonstration

Demonstrated Behavior

Bridges crossed: **3**

Miles of interstate: **20.7**



Stoplights: **10**

Model Behavior (Expectation)

Bridges crossed: **4.7**

+1.7

Miles of interstate: **16.2**

-4.5

Cost Weight: 5.0

Cost Weight: 3.0

Stoplights: **7.4**

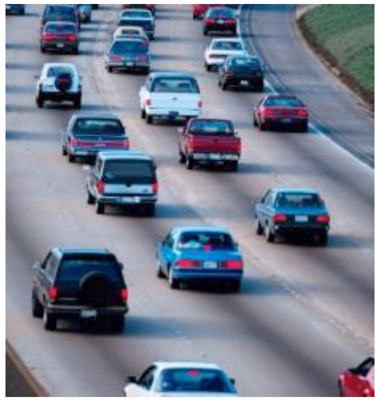
-2.6

Learning from Demonstration

Demonstrated Behavior

Bridges crossed: **3**

Miles of interstate: **20.7**



Stoplights: **10**

Model Behavior (Expectation)

Bridges crossed: **4.7**

Miles of interstate: **16.2**

7.2

Cost Weight: 5.0

Stoplights : **7.4**

1.1

Cost Weight:

Limitations of MaxEntIOC

- Cost was assumed linear over features f
- Dynamics T were assumed known

Next:

- General function approximations for cost c_θ: Finn et al. 2016
- Unknown Dynamics -> sample based approximations for the partition function Z: Boularias et al. 2011, Kalakrishnan et al. 2013, Finn et al. 2016

MaxEnt IOC general cost function

$$\max_{\theta} \sum_{\tau \in \mathcal{D}} \log p_{c_\theta}(\tau)$$

$$p(\tau) = \frac{1}{Z} \exp(-C_\theta(\tau))$$

$$Z = \int \exp(-C_\theta(\tau)) d\tau$$

Cost of a trajectory is decomposed over costs of individual states

$$C_\theta(\tau) = \sum_t c_\theta(x_t, u_t)$$

MaxEnt IOC general cost function

$$\max_{\theta} \sum_{\tau \in \mathcal{D}} \log p_{c_\theta}(\tau)$$

$$p(\tau) = \frac{1}{Z} \exp(-C_\theta(\tau))$$

$$Z = \int \exp(-C_\theta(\tau)) d\tau$$

Cost of a trajectory is decomposed over costs of individual states

$$C_\theta(\tau) = \sum_t c_\theta(x_t, u_t)$$

Before:

$$c_\theta(\mathbf{u}_t, \mathbf{u}_t) = \theta^{\mathrm{T}} \mathbf{f}(\mathbf{u}_t, \mathbf{x}_t)$$

MaxEnt IOC general cost function

$$\max_{\theta} \sum_{\tau \in \mathcal{D}} \log p_{c_\theta}(\tau)$$

$$p(\tau) = \frac{1}{Z} \exp(-C_\theta(\tau))$$

$$Z = \int \exp(-C_\theta(\tau)) d\tau$$

Cost of a trajectory is decomposed over costs of individual states

$$C_\theta(\tau) = \sum_t c_\theta(x_t, u_t)$$

Before:

$$c_\theta(\mathbf{u}_t, \mathbf{u}_t) = \theta^{\mathrm{T}} \mathbf{f}(\mathbf{u}_t, \mathbf{x}_t)$$

In the form of a loss function

$$\mathcal{L}_{\mathrm{IOC}}(\theta) = \frac{1}{N} \sum_{\tau_i \in \mathcal{D}_{\mathrm{demo}}} c_\theta(\tau_i) + \log Z$$

Approximating Z with Importance Sampling

$$\mu = \int_{\mathcal{D}} f(\boldsymbol{x}) p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \int_{\mathcal{D}} \frac{f(\boldsymbol{x}) p(\boldsymbol{x})}{q(\boldsymbol{x})} q(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \mathbb{E}_q \left(\frac{f(\boldsymbol{X}) p(\boldsymbol{X})}{q(\boldsymbol{X})} \right)$$

$$\log Z \;\approx\; \log \frac{1}{M} \sum_{\tau_j \in \mathcal{D}_{\text{samp}}} \frac{\exp(-c_\theta(\tau_j))}{q(\tau_j)}$$

MaxEntIOC with Importance Sampling

$$\mathcal{L}_{\mathrm{IOC}}(\theta) = \frac{1}{N} \sum_{\tau_i \in \mathcal{D}_{\mathrm{demo}}} c_\theta(\tau_i) + \log Z$$

MaxEntIOC with Importance Sampling

$$\begin{aligned}\mathcal{L}_{\text{IOC}}(\theta) &= \frac{1}{N}\sum_{\tau_i \in \mathcal{D}_{\text{demo}}} c_\theta(\tau_i) + \log Z \\ &\approx \frac{1}{N}\sum_{\tau_i \in \mathcal{D}_{\text{demo}}} c_\theta(\tau_i) + \log \frac{1}{M}\sum_{\tau_j \in \mathcal{D}_{\text{samp}}} \frac{\exp(-c_\theta(\tau_j))}{q(\tau_j)}\end{aligned}$$

MaxEntIOC with Importance Sampling

$$\mathcal{L}_{\text{IOC}}(\theta) = \frac{1}{N} \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} c_\theta(\tau_i) + \log Z$$

$$\approx \frac{1}{N} \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} c_\theta(\tau_i) + \log \frac{1}{M} \sum_{\tau_j \in \mathcal{D}_{\text{samp}}} \frac{\exp(-c_\theta(\tau_j))}{q(\tau_j)}$$

$$w_j = \frac{\exp(-c_\theta(\tau_j))}{q(\tau_j)}$$

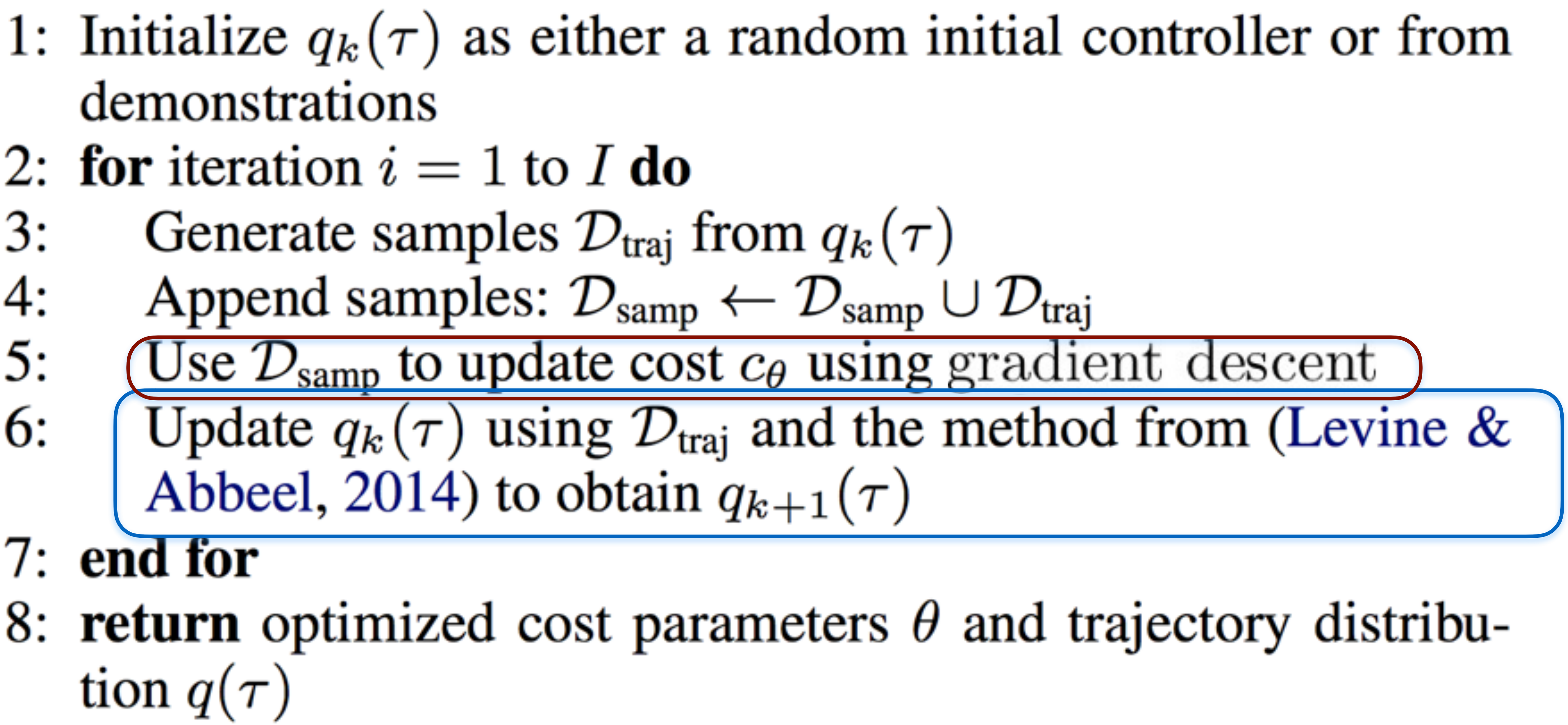
$$\frac{d\mathcal{L}_{\text{IOC}}}{d\theta} = \frac{1}{N} \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \frac{dc_\theta}{d\theta}(\tau_i) - \frac{1}{Z} \sum_{\tau_j \in \mathcal{D}_{\text{samp}}} w_j \frac{dc_\theta}{d\theta}(\tau_j)$$

Adapting the sampling distribution q

What should be the background sampling distribution q?

- Uniform: Boularias et al. 2011
- In the vicinity of demonstrations: Kalakrishnan et al. 2013
- Refine it over time! Finn at al. 2016: Interleave IOC with policy optimization, then sample trajectories according to the policy -> better trajectories (have much higher likelihood) guided by your current estimate of the cost c_theta

MaxEntIOC with Adaptive Importance Sampling

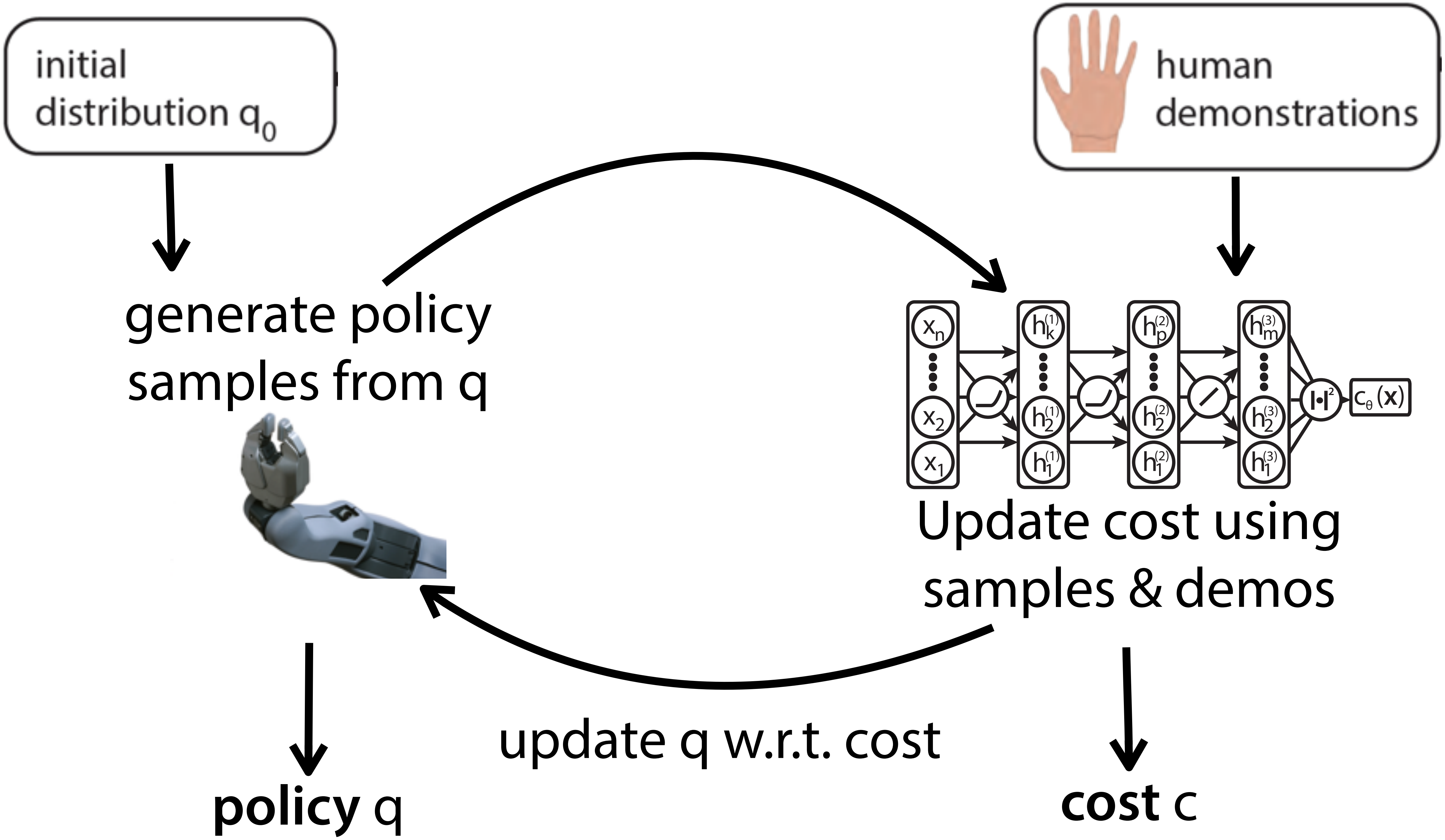


```
Initialize $q_k(\tau)$ as either a random initial controller or from demonstrations
for iteration $i = 1$ to $I$ do
    Generate samples $\mathcal{D}_{\text{traj}}$ from $q_k(\tau)$
    Append samples: $\mathcal{D}_{\text{samp}} \leftarrow \mathcal{D}_{\text{samp}} \cup \mathcal{D}_{\text{traj}}$
    Use $\mathcal{D}_{\text{samp}}$ to update cost $c_\theta$ using gradient descent
    Update $q_k(\tau)$ using $\mathcal{D}_{\text{traj}}$ and the method from (Levine & Abbeel, 2014) to obtain $q_{k+1}(\tau)$
end for
return optimized cost parameters $\theta$ and trajectory distribution $q(\tau)$
```

This can be any RL, planning algorithm that given rewards computes a policy (the forward RL problem), e.g. Ho and Ermon 2016 used TRPO

Given expert demonstrations and policy sampled trajectories improve rewards/costs (Inverse RL)

MaxEntIOC with Adaptive Importance Sampling



initial distribution q_0

human demonstrations

generate policy samples from q

Update cost using samples & demos

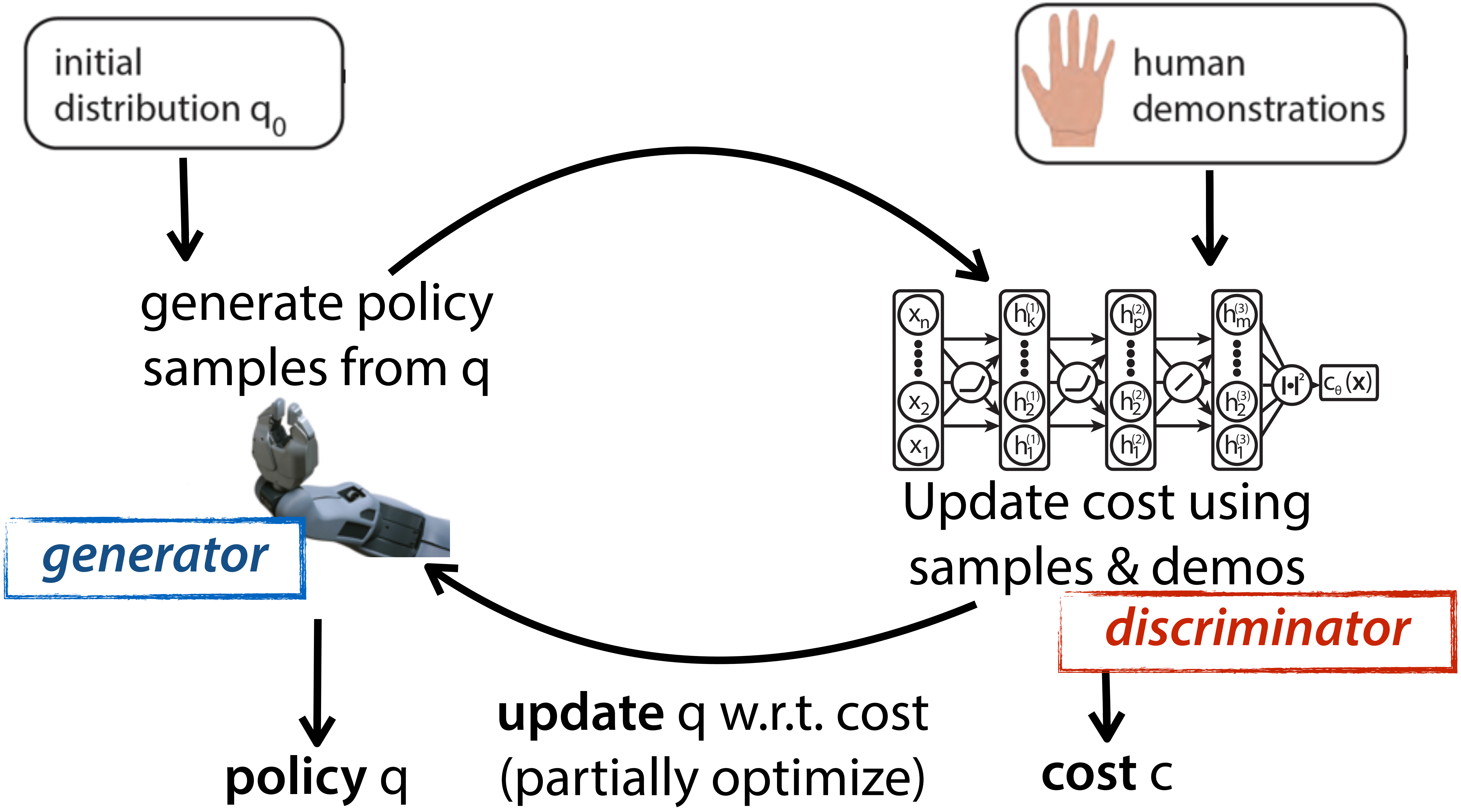
update q w.r.t. cost

policy q

cost c

Diagram from Chelsea Finn

MaxEntIOC with Adaptive Importance Sampling



initial distribution q_0

human demonstrations

generate policy samples from q

Update cost using samples & demos

generator

discriminator

policy q

update q w.r.t. cost (partially optimize)

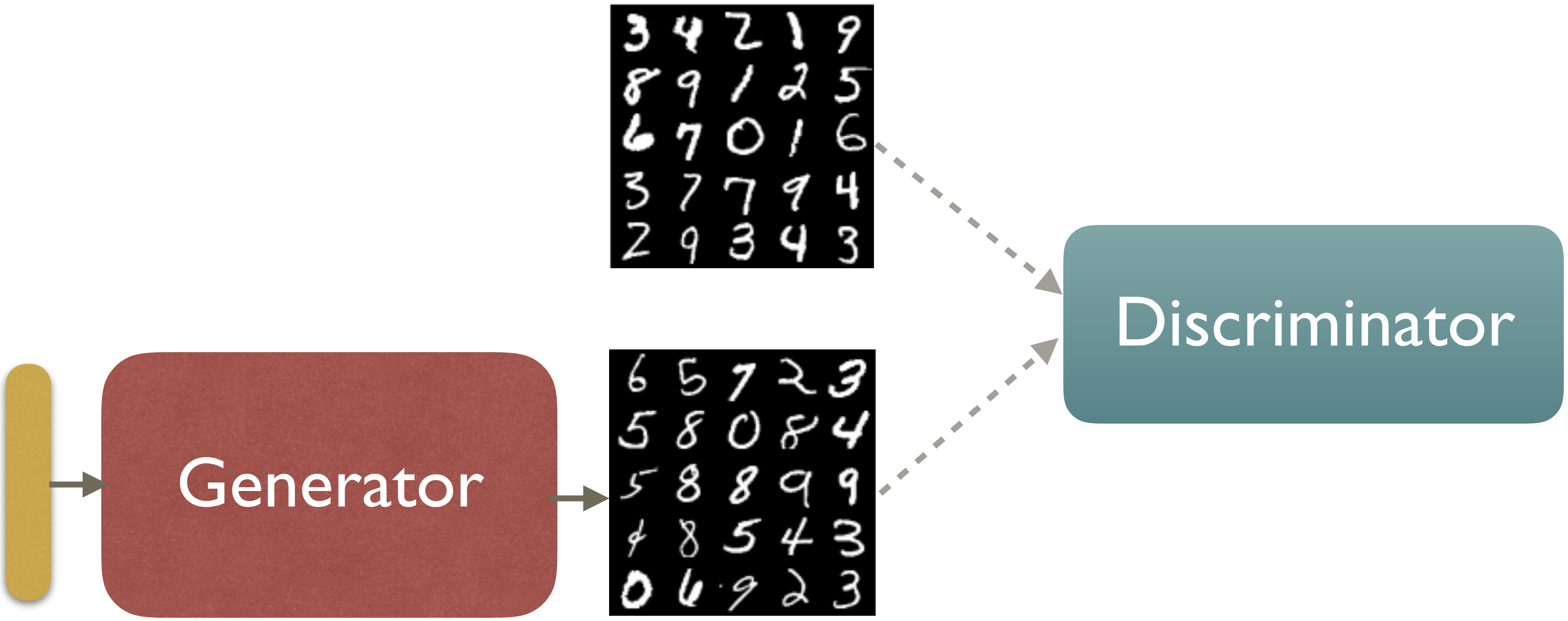
cost c

Diagram from Chelsea Finn

Generative Adversarial Networks

D(x): the probability that x came from the data rather than the generator

min max E_{x\sim p_data(x)}[log D(x)] + E_{z\sim p_z(z)}[log(1-D(G(z)))]
G D



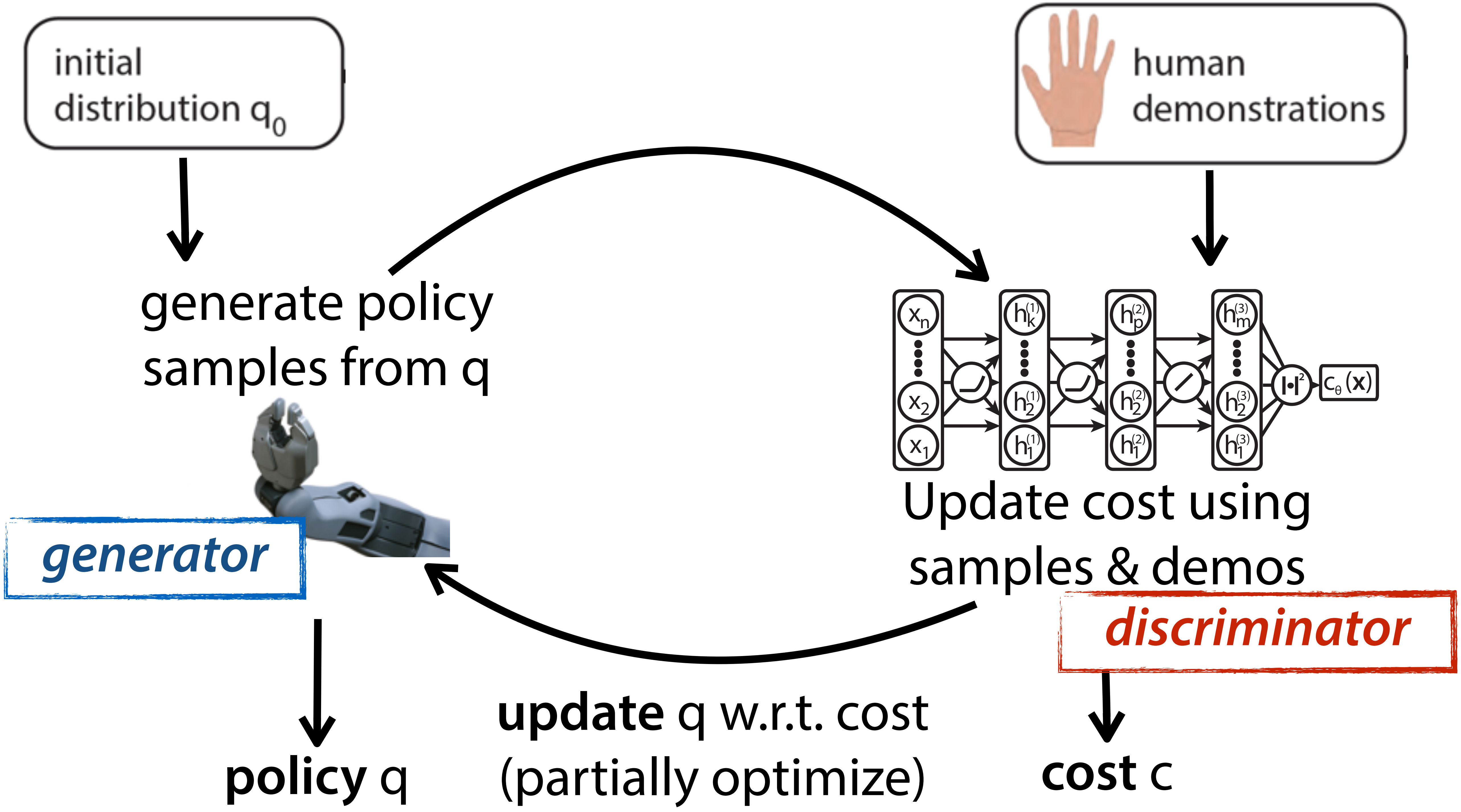
Real Data x

Generator

Discriminator

z ~ uniform([0, 1])

MaxEntIOC with Adaptive Importance Sampling



initial distribution q_0

human demonstrations

generate policy samples from q

Update cost using samples & demos

generator

discriminator

policy q

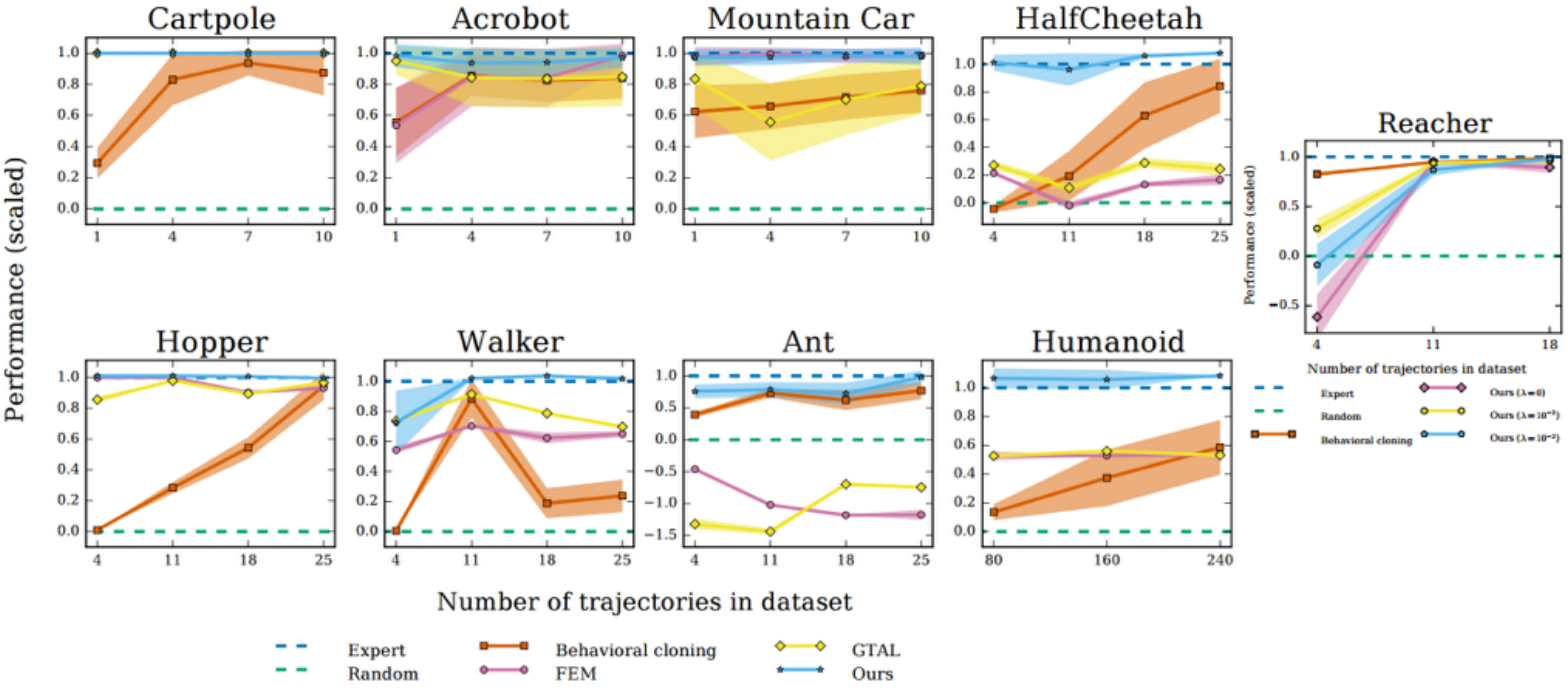
update q w.r.t. cost (partially optimize)

cost c

Diagram from Chelsea Finn

Case Study: Generative Adversarial Imitation Learning

- demonstrations from TRPO-optimized policy
- use TRPO as a policy optimizer
- OpenAI gym tasks



Q:

- Why we need to have this separate optimization over cost, and then separately planning/RL over this cost to find the policy? Because the dynamics where unknown..

- Let’s assume they are known.

- Can we imitate the expert and backdrop through all the way till the rewards simply by imitating expert behavior (e.g., through supervised learning!!), in an end-to-end fashion?

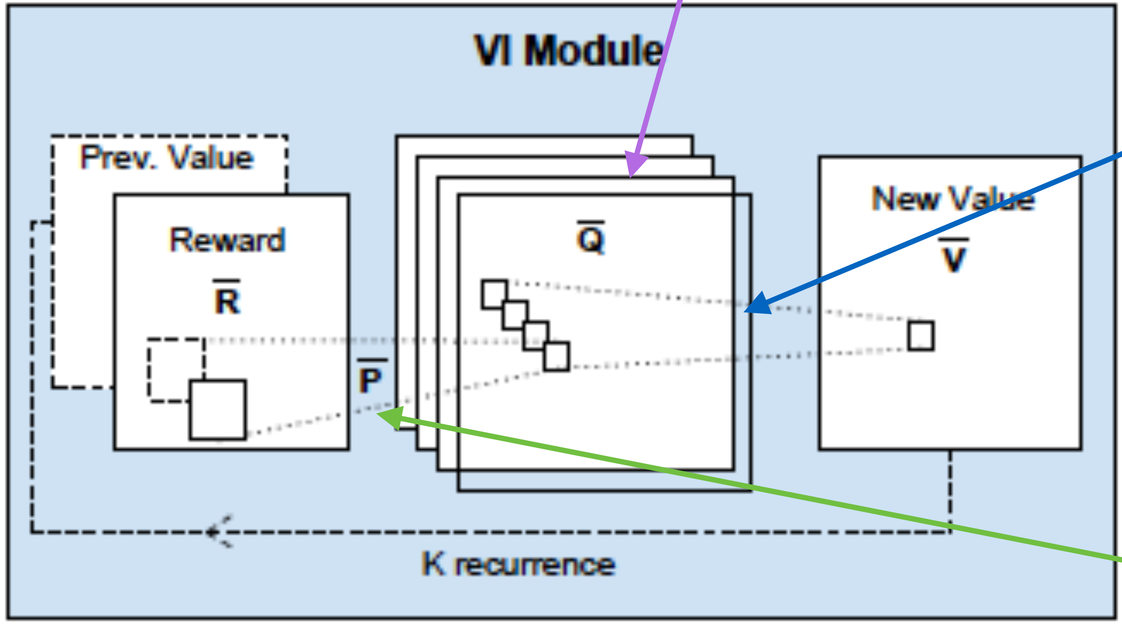
Value Iteration Sub-Network

Each iteration of VI may be seen as passing the previous value function V_n and reward function R through a convolution layer and max-pooling layer.

$$Q_n(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_n(s')$$

$$V_{n+1}(s) = \max_a Q_n(s,a)$$

Each channel in the convolution layer corresponds to the Q -function for a specific action



Max-pooling

convolution kernel weights correspond to the discounted transition probabilities

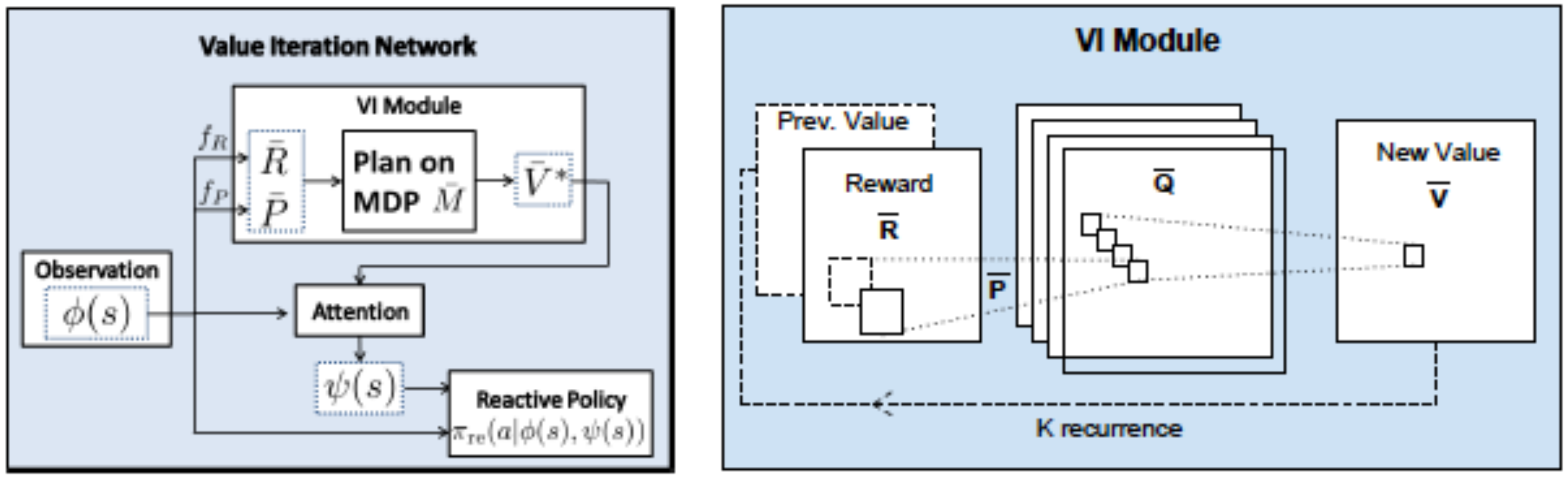
By recurrently applying a convolution layer K times, K iterations of VI are effectively performed.

Value Iteration Networks, Tamar et al. 2016

Value Iteration Network

- Notice that the optimal action at each state depends only on the value function of its immediate neighbors->locality->attention

$$\bar{\pi}^*(\bar{s}) = \arg\max_{\bar{a}} \bar{R}(\bar{s},\bar{a}) + \gamma \sum_{\bar{s}'} \bar{P}(\bar{s}'|\bar{s},\bar{a})\bar{V}^*(\bar{s}').$$

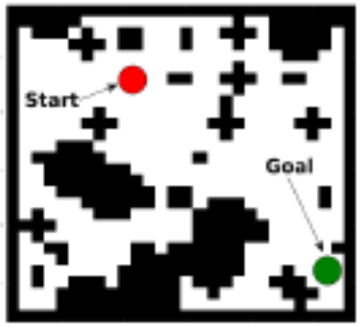


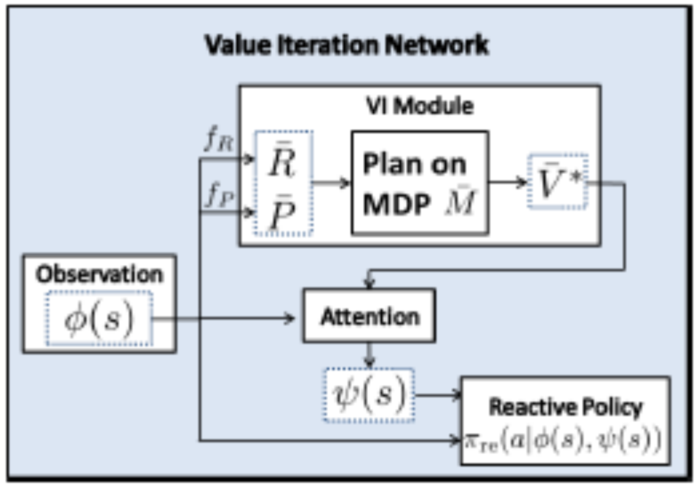
- To estimate the optimal action in each state, I only need to use the value functions in the vicinity of the state, then train and CNN with a standard architecture

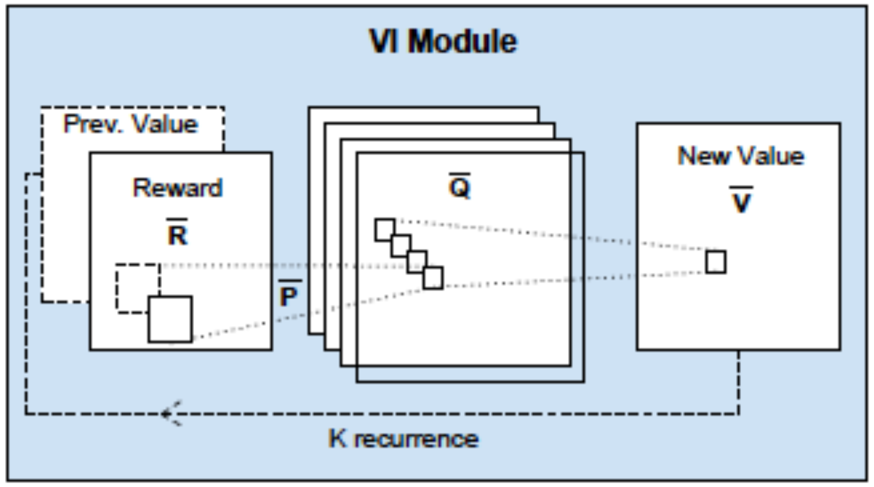
Value Iteration Networks, Tamar et al. 2016

Value Iteration Network

- This particular CNN architecture allows propagation of the state value far in space, as opposed to a standard CNN over the same state space: reactive policies, versus planning based policies







Domain	VIN			CNN			FCN		
	Prediction loss	Success rate	Traj. diff.	Pred. loss	Succ. rate	Traj. diff.	Pred. loss	Succ. rate	Traj. diff.
8 × 8	0.004	**99.6%**	0.001	0.02	97.9%	0.006	0.01	97.3%	0.004
16 × 16	0.05	**99.3%**	0.089	0.10	87.6%	0.06	0.07	88.3%	0.05
28 × 28	0.11	**97%**	0.086	0.13	74.2%	0.078	0.09	76.6%	0.08

Value Iteration Networks, Tamar et al. 2016