

Carnegie Mellon

School of Computer Science

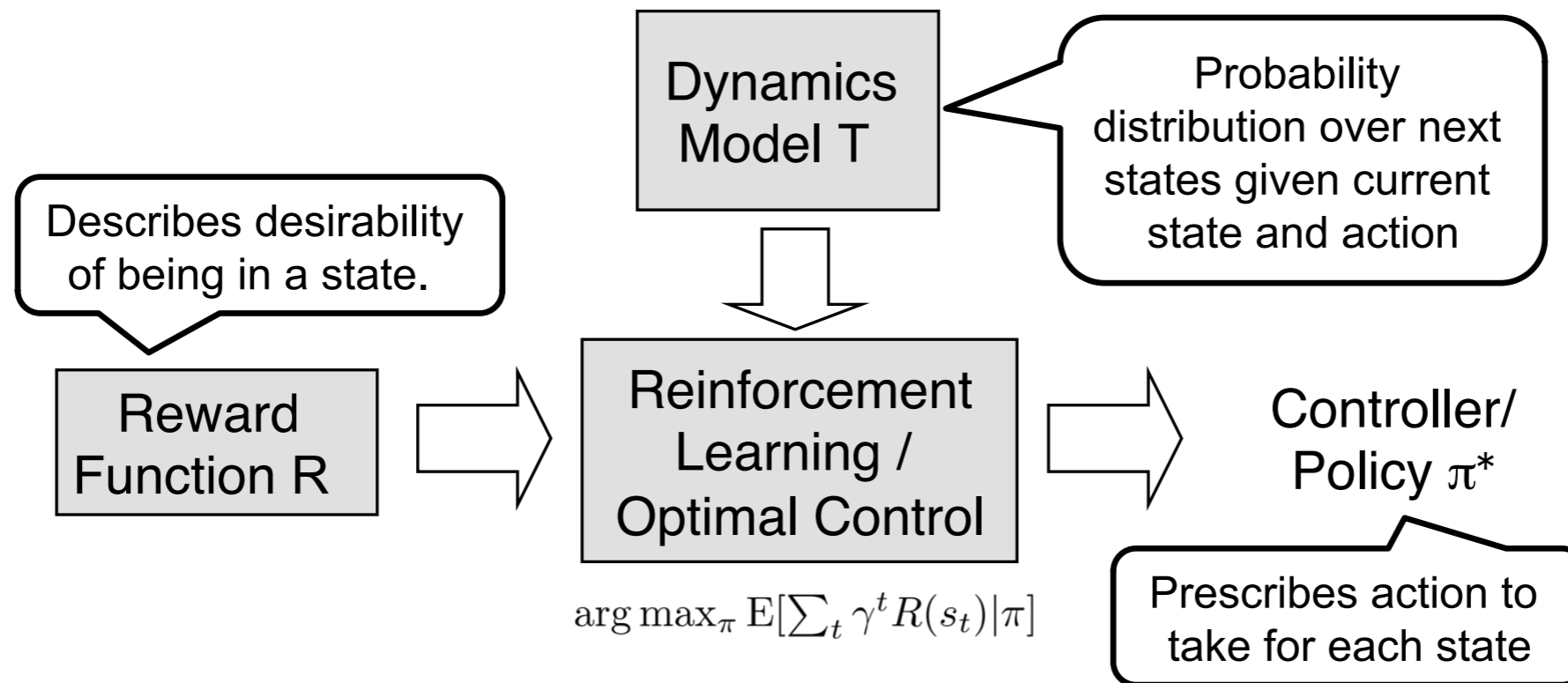
Deep Reinforcement Learning and Control

Maximum Entropy Inverse RL, Adversarial imitation learning

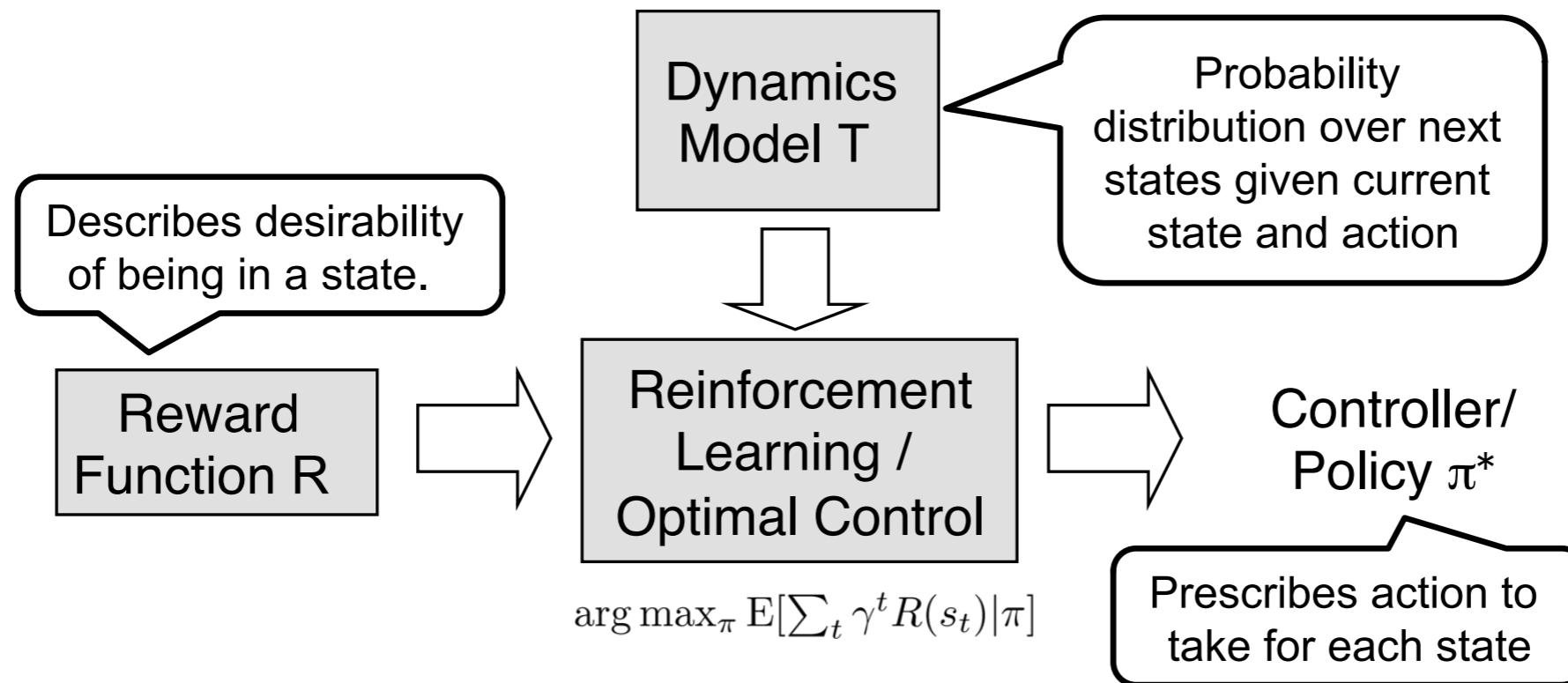
Katerina Fragkiadaki



Reinforcement Learning

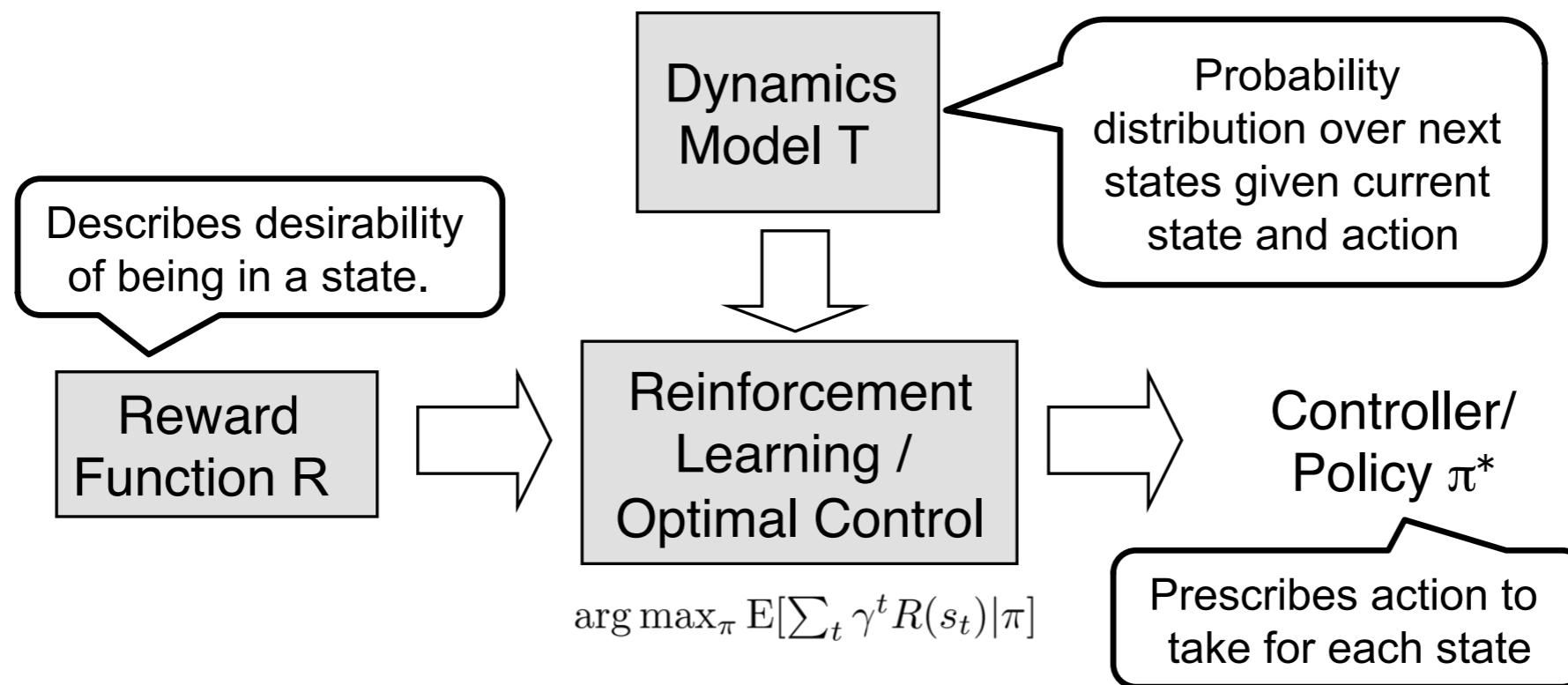


Inverse Reinforcement Learning



IRL reverses the diagram: **Given a finite set of demonstration trajectories, let's recover reward R and policy π^* !**

Inverse Reinforcement Learning



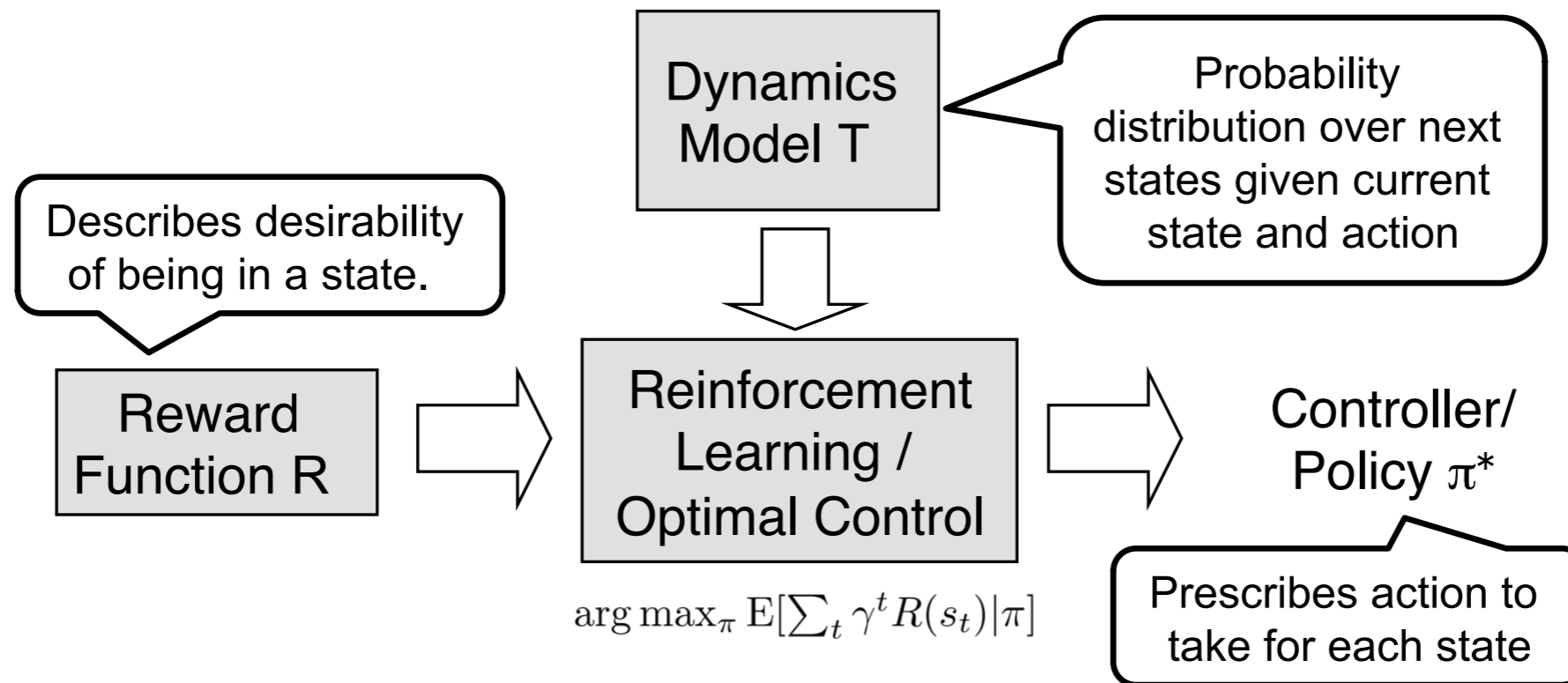
IRL reverses the diagram: **Given a finite set of demonstration trajectories, let's recover reward R and policy π^* !**

In contrast to the DAGGER setup, we cannot interactively query the expert for additional labels.

Inverse Reinforcement Learning

Q: Why inferring the reward is useful as opposed to learning a policy directly?

A: Because it can generalize better, e.g., if the dynamics of the environment change, you can use the reward to learn a policy that can handle those new dynamics

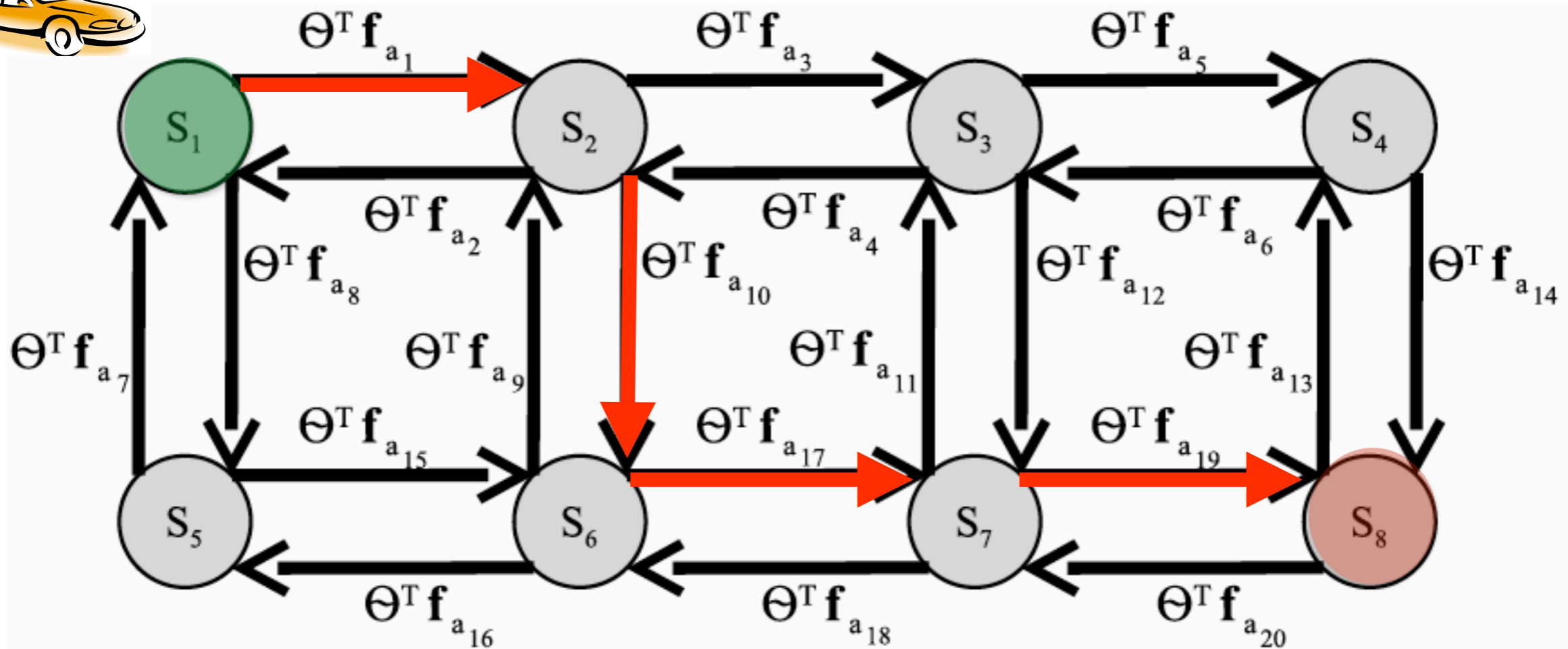


IRL reverses the diagram: **Given a finite set of demonstration trajectories, let's recover reward R and policy π^* !**

In contrast to the DAGGER setup, we cannot interactively query the expert for additional labels.

A simple example

- Roads have unknown costs linear in features
- **Paths (trajectories)** have unknown costs, sum of road (state) costs
- Experts (taxi-drivers) demonstrate Pittsburgh traveling behavior
- How can we learn to navigate Pitts like a taxi (or uber) driver?



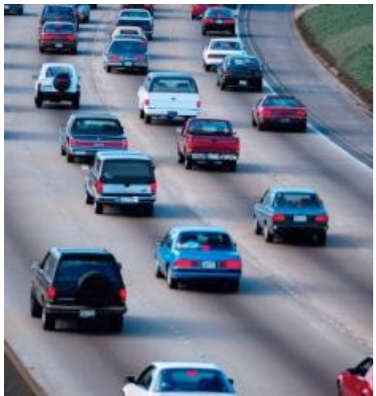
- Assumption: cost is independent of the goal state, so it only depends on road features, e.g., traffic width tolls etc.

State features

Features f can be:



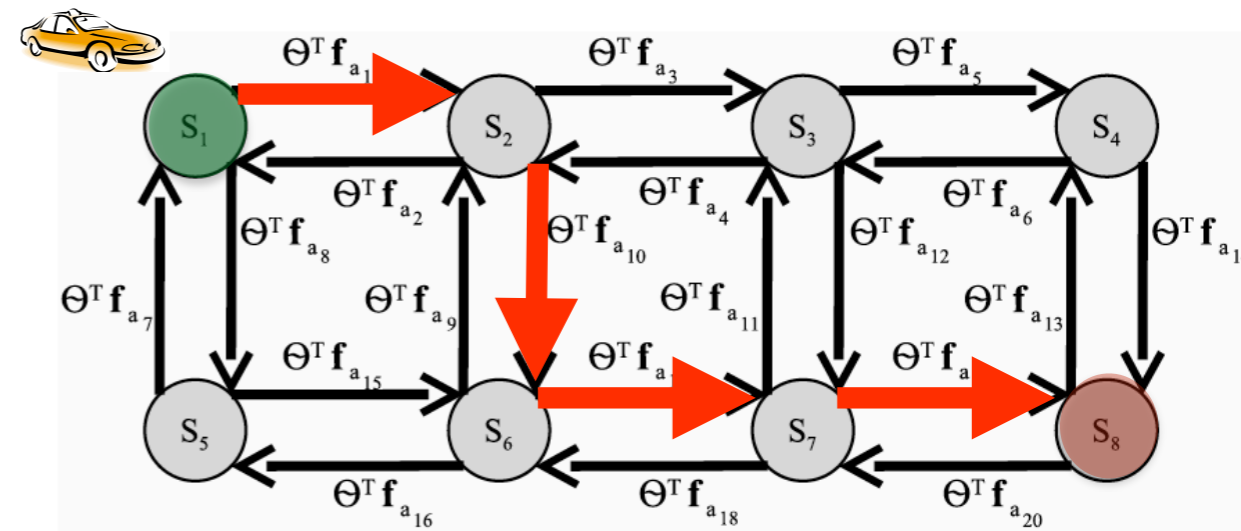
Bridges crossed



Miles of interstate



Stoplights

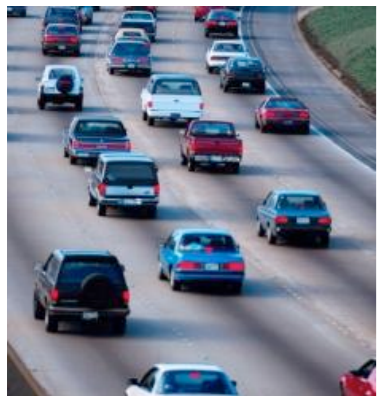


A good guess: Match expected features

Features f can be:



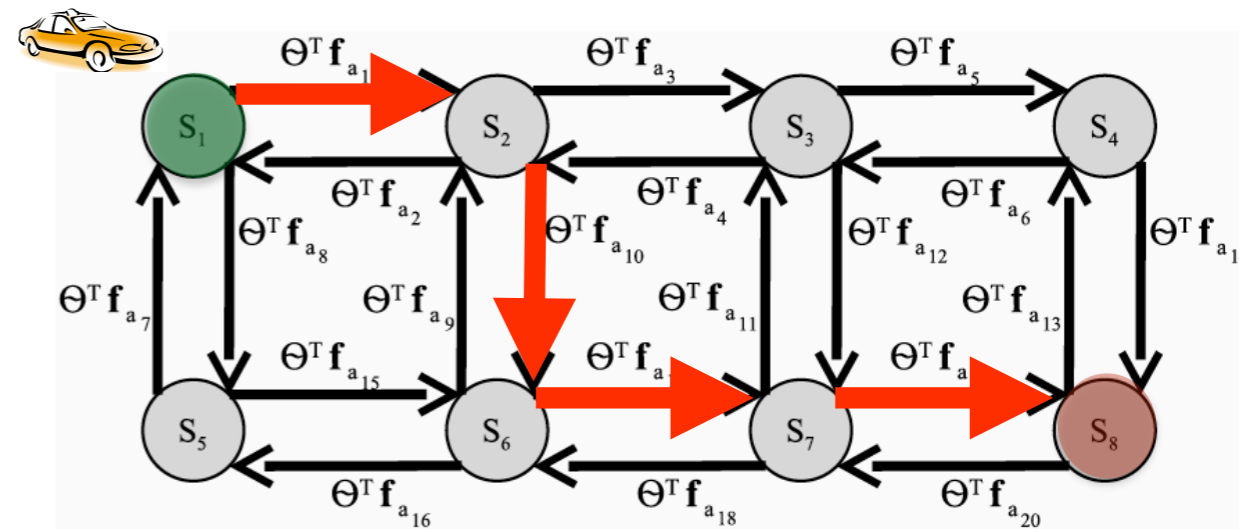
Bridges crossed



Miles of interstate



Stoplights



Feature matching:

$$\sum_{\tau_i} p(\tau_i) f_{\tau_i} = \tilde{f}$$

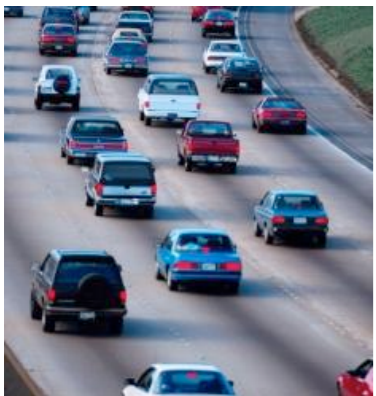
“If a driver uses 136.3 miles of interstate and crosses 12 bridges in a month’s worth of trips, the model should also use 136.3 miles of interstate and 12 bridges in expectation for those same start-destination pairs.”

A good guess: Match expected features

Features f can be:



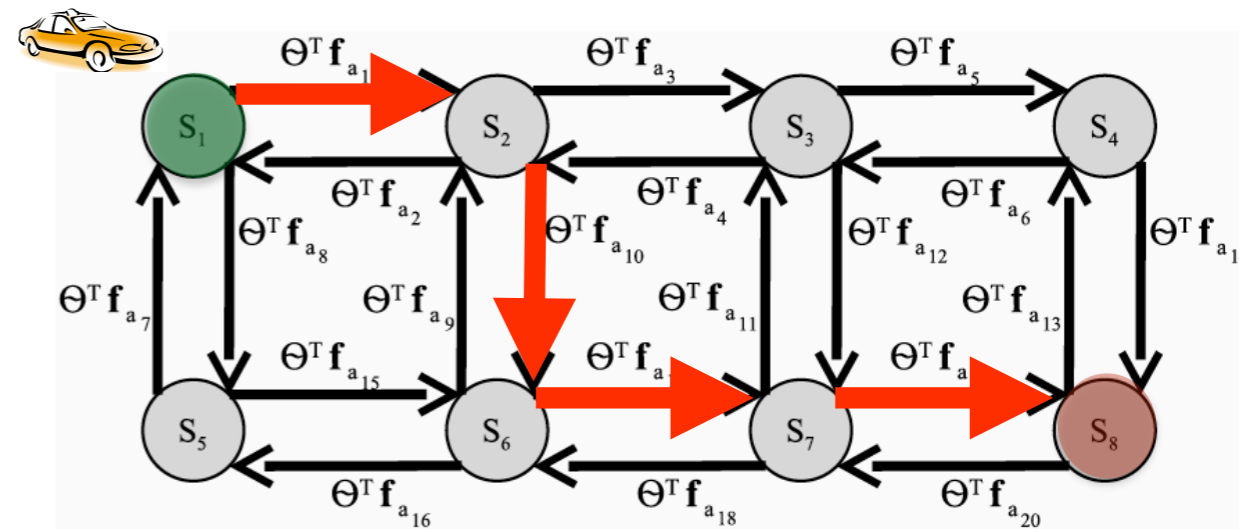
Bridges crossed



Miles of interstate



Stoplights



Feature matching:

$$\sum_{\tau_i} p(\tau_i) f_{\tau_i} = \tilde{f}$$

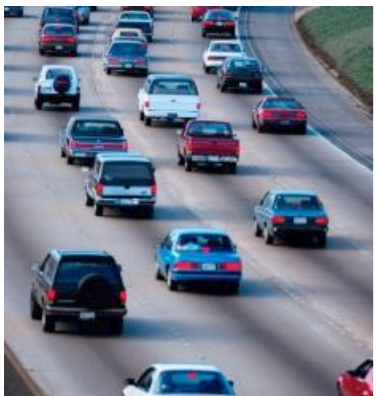
Demonstrated feature counts

A good guess: Match expected features

Features f can be:



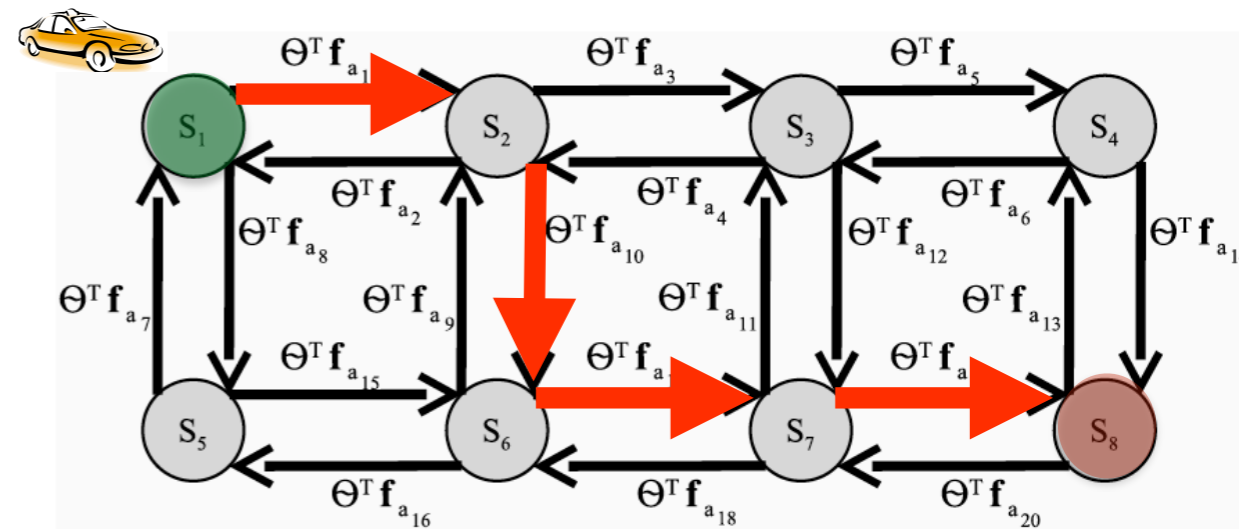
Bridges crossed



Miles of interstate



Stoplights



Feature matching:

$$\sum_{\tau_i} p(\tau_i) f_{\tau_i} = \tilde{f}$$

Demonstrated feature counts

a policy induces a distribution over trajectories

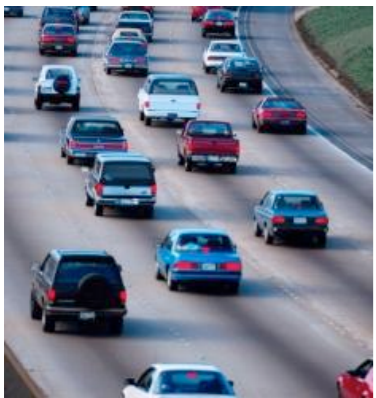
$$p(\tau) = p(s_1) \prod p(a_t | s_t) P(s_{t+1} | s_t, a_t)$$

Ambiguity

Features f can be:



Bridges crossed



Miles of interstate



Stoplights

However, many distributions over paths can match feature counts, and some will be very different from observed behavior. The model could produce a policy 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.

Feature matching:

$$\sum_{\text{path } \tau_i} p(\tau_i) f_{\tau_i} = \tilde{f}$$

Demonstrated feature counts

a policy induces a distribution over trajectories

$$p(\tau) = p(s_1) \prod p(a_t | s_t) P(s_{t+1} | s_t, a_t)$$

Principle of Maximum Entropy

The Principle of Maximum Entropy is based on the premise that when estimating the probability distribution, you should select that distribution which leaves you the largest remaining uncertainty (i.e., the maximum entropy) consistent with your constraints. That way you have not introduced any additional assumptions or biases into your calculations

$$H(x) = - \sum_{i=1}^n p(x_i) \log(p(x_i))$$

Resolve Ambiguity by Maximum Entropy

Features f can be:



Bridges crossed



Miles of interstate



Stoplights

Let's pick the policy that satisfies feature count constraints without over-committing!

$$\max_p . \quad - \sum_{\tau} p(\tau) \log p(\tau)$$

Feature matching constraint:

$$\sum_{\text{path } \tau_i} p(\tau_i) f_{\tau_i} = \tilde{f}$$

Demonstrated feature counts

a policy induces a distribution over trajectories

$$p(\tau) = p(s_1) \prod p(a_t | s_t) P(s_{t+1} | s_t, a_t)$$

From features to costs

Constraint: Match the cost of expert trajectories in expectation:

$$\int p(\tau) c_{\theta}(\tau) d\tau = \frac{1}{|\mathbf{D}_{\text{demo}}|} \sum_{\tau_i \in \mathbf{D}_{\text{demo}}} c_{\theta}(\tau_i) = \tilde{c}$$

Maximum Entropy Inverse Optimal Control

Optimization problem:

$$\begin{aligned} \min_p . \quad & -H(p(\tau)) = \sum_{\tau} p(\tau) \log p(\tau) \\ \text{s.t.} \quad & \int_{\tau} p(\tau) c_{\theta}(\tau) = \tilde{c}, \quad \int_{\tau} p(\tau) = 1 \end{aligned}$$

From maximum entropy to exponential family

$$\min_p. \quad -H(p(\tau)) = \sum_{\tau} p(\tau) \log p(\tau)$$

$$\text{s.t.} \quad \int_{\tau} p(\tau) c_{\theta}(\tau) = \tilde{c}, \quad \int_{\tau} p(\tau) = 1$$

$$\iff \mathcal{L}(p, \lambda) = \int p(\tau) \log(p(\tau)) d\tau + \lambda_1 \left(\int p(\tau) c_{\theta}(\tau) d\tau - \tilde{c} \right) + \lambda_0 \left(\int p(\tau) d\tau - 1 \right)$$

$$\frac{\partial \mathcal{L}}{\partial p} = \log p(\tau) + 1 + \lambda_1 c_{\theta}(\tau) + \lambda_0$$

$$\frac{\partial \mathcal{L}}{\partial p} = 0 \iff \log p(\tau) = -1 - \lambda_1 c_{\theta}(\tau) - \lambda_0$$

$$\iff p(\tau) = e^{-1 - \lambda_0 - \lambda_1 c_{\theta}(\tau)}$$

$$\rightarrow p(\tau) \propto e^{c_{\theta}(\tau)}$$

From maximum entropy to exponential family

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

$$p(\tau | \theta) = \frac{e^{-\text{cost}(\tau|\theta)}}{\sum_{\tau'} e^{-\text{cost}(\tau'|\theta)}}$$

- Strong preference for low cost trajectories
- Equal cost trajectories are equally probable

Maximum Likelihood

$$\max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} \log p(\tau_i)$$

Maximum Likelihood

$$\begin{aligned} & \max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} \log p(\tau_i) \\ \iff & \max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} \log \frac{e^{-c_{\theta}(\tau_i)}}{Z} \end{aligned}$$

Maximum Likelihood

$$\max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} \log p(\tau_i)$$

$$\iff \max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} \log \frac{e^{-c_{\theta}(\tau_i)}}{Z}$$

$$\iff \max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} -c_{\theta}(\tau_i) - \sum_{\tau_i \in D_{\text{demo}}} \log Z$$

Maximum Likelihood

$$\max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} \log p(\tau_i)$$

$$\iff \max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} \log \frac{e^{-c_{\theta}(\tau_i)}}{Z}$$

$$\iff \max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} -c_{\theta}(\tau_i) - \sum_{\tau_i \in D_{\text{demo}}} \log Z$$

$$\iff \max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} -c_{\theta}(\tau_i) - \sum_{\tau_i \in D_{\text{demo}}} \log \left(\sum_{\tau} e^{-c_{\theta}(\tau)} \right)$$

This is a huge sum, intractable to compute in large state spaces.

Maximum Likelihood

$$\max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} \log p(\tau_i)$$

$$\iff \max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} \log \frac{e^{-c_{\theta}(\tau_i)}}{Z}$$

$$\iff \max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} -c_{\theta}(\tau_i) - \sum_{\tau_i \in D_{\text{demo}}} \log Z$$

$$\iff \max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} -c_{\theta}(\tau_i) - \sum_{\tau_i \in D_{\text{demo}}} \log \left(\sum_{\tau} e^{-c_{\theta}(\tau)} \right)$$

$$\iff \max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} -c_{\theta}(\tau_i) - \log \left(\sum_{\tau} e^{-c_{\theta}(\tau)} \right) |D_{\text{demo}}|$$

This is a huge sum, intractable to compute in large state spaces.

Maximum Likelihood

$$\max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} \log p(\tau_i)$$

This is a huge sum, intractable to compute in large state spaces.

$$\iff \max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} \log \frac{e^{-c_{\theta}(\tau_i)}}{Z}$$

$$\iff \max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} -c_{\theta}(\tau_i) - \sum_{\tau_i \in D_{\text{demo}}} \log Z$$

$$\iff \max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} -c_{\theta}(\tau_i) - \sum_{\tau_i \in D_{\text{demo}}} \log \left(\sum_{\tau} e^{-c_{\theta}(\tau)} \right)$$

$$\iff \max_{\theta} \sum_{\tau_i \in D_{\text{demo}}} -c_{\theta}(\tau_i) - \log \left(\sum_{\tau} e^{-c_{\theta}(\tau)} \right) |D_{\text{demo}}|$$

$$\iff \min_{\theta} \sum_{\tau_i \in D_{\text{demo}}} c_{\theta}(\tau_i) + |D_{\text{demo}}| \log \left(\sum_{\tau} e^{-c_{\theta}(\tau)} \right) \rightarrow \mathcal{L}(\theta)$$

Maximum Likelihood

$$\nabla_{\theta} \mathcal{L}(\theta) = \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \frac{dc_{\theta}(\tau_i)}{d\theta} + |\mathcal{D}_{\text{demo}}| \frac{1}{\sum_{\tau} e^{-c_{\theta}(\tau)}} \sum_{\tau} (e^{-c_{\theta}(\tau)} \left(-\frac{dc_{\theta}(\tau)}{d\theta} \right))$$

Maximum Likelihood

$$\begin{aligned}\nabla_{\theta} \mathcal{L}(\theta) &= \sum_{\tau_i \in \mathbf{D}_{\text{demo}}} \frac{dc_{\theta}(\tau_i)}{d\theta} + |\mathbf{D}_{\text{demo}}| \frac{1}{\sum_{\tau} e^{-c_{\theta}(\tau)}} \sum_{\tau} (e^{-c_{\theta}(\tau)} \left(-\frac{dc_{\theta}(\tau)}{d\theta}\right)) \\ &= \sum_{\tau_i \in \mathbf{D}_{\text{demo}}} \frac{dc_{\theta}(\tau_i)}{d\theta} - |\mathbf{D}_{\text{demo}}| \sum_{\tau} p(\tau | \theta) \frac{dc_{\theta}(\tau)}{d\theta}\end{aligned}$$

Maximum Likelihood

$$\begin{aligned}\nabla_{\theta} \mathcal{L}(\theta) &= \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \frac{dc_{\theta}(\tau_i)}{d\theta} + |\mathcal{D}_{\text{demo}}| \frac{1}{\sum_{\tau} e^{-c_{\theta}(\tau)}} \sum_{\tau} (e^{-c_{\theta}(\tau)} \left(-\frac{dc_{\theta}(\tau)}{d\theta}\right)) \\ &= \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \frac{dc_{\theta}(\tau_i)}{d\theta} - |\mathcal{D}_{\text{demo}}| \sum_{\tau} p(\tau | \theta) \frac{dc_{\theta}(\tau)}{d\theta}\end{aligned}$$

Trajectory cost is additive over states: $c_{\theta}(\tau) = \sum_{s \in \tau} c_{\theta}(s)$

Maximum Likelihood

$$\begin{aligned}\nabla_{\theta} \mathcal{L}(\theta) &= \sum_{\tau_i \in D_{\text{demo}}} \frac{dc_{\theta}(\tau_i)}{d\theta} + |D_{\text{demo}}| \frac{1}{\sum_{\tau} e^{-c_{\theta}(\tau)}} \sum_{\tau} (e^{-c_{\theta}(\tau)} \left(-\frac{dc_{\theta}(\tau)}{d\theta}\right)) \\ &= \sum_{\tau_i \in D_{\text{demo}}} \frac{dc_{\theta}(\tau_i)}{d\theta} - |D_{\text{demo}}| \sum_{\tau} p(\tau | \theta) \frac{dc_{\theta}(\tau)}{d\theta}\end{aligned}$$

Trajectory cost is additive over states: $c_{\theta}(\tau) = \sum_{s \in \tau} c_{\theta}(s)$

$$\Rightarrow p(\tau) \propto e^{-\sum_{s \in \tau} c_{\theta}(s)}$$

Maximum Likelihood

$$\begin{aligned}\nabla_{\theta} \mathcal{L}(\theta) &= \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \frac{dc_{\theta}(\tau_i)}{d\theta} + |\mathcal{D}_{\text{demo}}| \frac{1}{\sum_{\tau} e^{-c_{\theta}(\tau)}} \sum_{\tau} (e^{-c_{\theta}(\tau)} \left(-\frac{dc_{\theta}(\tau)}{d\theta}\right)) \\ &= \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \frac{dc_{\theta}(\tau_i)}{d\theta} - |\mathcal{D}_{\text{demo}}| \sum_{\tau} p(\tau | \theta) \frac{dc_{\theta}(\tau)}{d\theta}\end{aligned}$$

Trajectory cost is additive over states: $c_{\theta}(\tau) = \sum_{s \in \tau} c_{\theta}(s)$

$$\Rightarrow p(\tau) \propto e^{-\sum_{s \in \tau} c_{\theta}(s)}$$

$$\nabla_{\theta} \mathcal{L}(\theta) = \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \frac{dc_{\theta}(\tau_i)}{d\theta} - |\mathcal{D}_{\text{demo}}| \sum_{\tau} p(\tau | \theta) \frac{dc_{\theta}(\tau)}{d\theta}$$

Maximum Likelihood

$$\begin{aligned}\nabla_{\theta} \mathcal{L}(\theta) &= \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \frac{dc_{\theta}(\tau_i)}{d\theta} + |\mathcal{D}_{\text{demo}}| \frac{1}{\sum_{\tau} e^{-c_{\theta}(\tau)}} \sum_{\tau} (e^{-c_{\theta}(\tau)} \left(-\frac{dc_{\theta}(\tau)}{d\theta}\right)) \\ &= \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \frac{dc_{\theta}(\tau_i)}{d\theta} - |\mathcal{D}_{\text{demo}}| \sum_{\tau} p(\tau | \theta) \frac{dc_{\theta}(\tau)}{d\theta}\end{aligned}$$

Trajectory cost is additive over states: $c_{\theta}(\tau) = \sum_{s \in \tau} c_{\theta}(s)$

$$\Rightarrow p(\tau) \propto e^{-\sum_{s \in \tau} c_{\theta}(s)}$$

$$\nabla_{\theta} \mathcal{L}(\theta) = \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \frac{dc_{\theta}(\tau_i)}{d\theta} - |\mathcal{D}_{\text{demo}}| \sum_{\tau} p(\tau | \theta) \frac{dc_{\theta}(\tau)}{d\theta}$$

$$\nabla_{\theta} \mathcal{L}(\theta) = \sum_{s \in \tau_i \in \mathcal{D}_{\text{demo}}} \frac{dc_{\theta}(s)}{d\theta} - |\mathcal{D}_{\text{demo}}| \sum_s p(s | \theta) \frac{dc_{\theta}(s)}{d\theta}$$

This is still an intractable sum, impossible to compute exactly in large state spaces.

Trajectory cost is additive over states

$$\nabla_{\theta} \mathcal{L}(\theta) = \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \frac{dc_{\theta}(\tau_i)}{d\theta} - |\mathcal{D}_{\text{demo}}| \sum_{\tau} p(\tau | \theta) \frac{dc_{\theta}(\tau)}{d\theta}$$

$$\nabla_{\theta} \mathcal{L}(\theta) = \sum_{s \in \tau_i \in \mathcal{D}_{\text{demo}}} \frac{dc_{\theta}(s)}{d\theta} - |\mathcal{D}_{\text{demo}}| \sum_s p(s | \theta) \frac{dc_{\theta}(s)}{d\theta}$$

State densities:
how much time
the policy
spends on each
state

For linear costs: $c_{\theta}(s) = \theta^{\top} \mathbf{f}_s$

$$\nabla_{\theta} \mathcal{L}(\theta) = \sum_{s \in \mathcal{D}_{\text{demo}}} \mathbf{f}_s - |\mathcal{D}_{\text{demo}}| \sum_s p(s | \theta) \mathbf{f}_s$$

State densities can be computed analytically in **small** MDPs
with **known dynamics**

State densities can be computed analytically in **small** MDPs with **known dynamics**

$\mu_t(s)$: time indexed state density

State densities can be computed analytically in **small** MDPs with **known dynamics**

$\mu_t(s)$: time indexed state density

initialize $\mu_1(s) \forall s$

State densities can be computed analytically in **small** MDPs with **known dynamics**

$\mu_t(s)$: time indexed state density

initialize $\mu_1(s) \forall s$

for $t = 1, \dots, T$

$$\mu_{t+1}(s) = \sum_a \sum_{s'} \mu_t(s') \pi(a | s') p(s | s', a)$$

State densities can be computed analytically in **small** MDPs with **known dynamics**

$\mu_t(s)$: time indexed state density

initialize $\mu_1(s) \forall s$

for $t = 1, \dots, T$

$$\mu_{t+1}(s) = \sum_a \sum_{s'} \mu_t(s') \pi(a | s') p(s | s', a)$$

$$p(s | \theta) = \sum_t \mu_t(s)$$

State densities can be computed analytically in **small** MDPs with **known dynamics**

$\mu_t(s)$: time indexed state density

initialize $\mu_1(s) \forall s$

for $t = 1, \dots, T$

$$\mu_{t+1}(s) = \sum_a \sum_{s'} \mu_t(s') \pi(a | s') p(s | s', a)$$

Known dynamics

$$p(s | \theta) = \sum_t \mu_t(s)$$

State densities can be computed analytically in **small** MDPs with **known dynamics**

$\mu_t(s)$: time indexed state density

initialize $\mu_1(s) \forall s$

for $t = 1, \dots, T$

$$\mu_{t+1}(s) = \sum_a \sum_{s'} \mu_t(s') \pi(a | s') p(s | s', a)$$

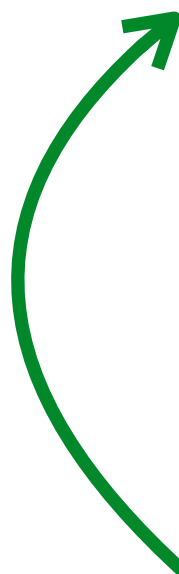
Known dynamics

Unknown policy

$$p(s | \theta) = \sum_t \mu_t(s)$$

Maximum entropy Inverse RL

Known dynamics, small state space, linear costs

0. Initialize θ , gather demonstrations \mathbf{D}_{demo}
 1. Solve for optimal policy $\pi(a|s)$ w.r.t. c_θ with value iteration
 2. Solve for state visitation frequencies $p(s|\theta)$
 3. Compute gradient $\nabla_\theta \mathcal{L}(\theta) = \sum_{s \in \mathbf{D}_{\text{demo}}} \mathbf{f}_s - |\mathbf{D}_{\text{demo}}| \sum_s p(s|\theta) \mathbf{f}_s$
 4. Update θ with one gradient step using $\nabla_\theta \mathcal{L}(\theta)$
- 

Maximum entropy Inverse RL

Demonstrated Behavior



Bridges
crossed: **3**

Miles of
interstate:
20.7



Stoplights:
10



Model Behavior (Expectation)



Bridges
crossed: ?

Miles of
interstate:
?



Stoplights
:
?



Maximum entropy Inverse RL

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



Cost Weight:
5.0



Cost Weight:
3.0

Stoplights
:
7.4
-2.6

-4.5

34

Maximum entropy Inverse RL

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



1.1
Cost
Weight:

Stoplights
:
7.4

Limitations of the formulation so far

- Cost was assumed linear over features f
- Dynamics were assumed known
- State space was small

Next:

- General function approximations for the cost: 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

Recall our maximum likelihood formulation

$$\begin{aligned} & \max_{\theta} \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \log p(\tau_i) \\ \iff & \max_{\theta} \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \log \frac{e^{-c_{\theta}(\tau_i)}}{Z} \\ \iff & \max_{\theta} \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} -c_{\theta}(\tau_i) - \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \log Z \end{aligned}$$

We need to minimize the following loss function:

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

Sample approximation for Z

This is a huge integral, intractable to compute.

$$Z = \int e^{-c_{\theta}(\tau)} d\tau$$

Sample approximation for Z

This is a huge integral, intractable to compute.

$$Z = \int e^{-c_{\theta}(\tau)} d\tau$$

$$Z = \int e^{-c_{\theta}(\tau)} d\tau = \int q(\tau) \frac{e^{-c_{\theta}(\tau)}}{q(\tau)} d\tau$$

Sample approximation for Z

This is a huge integral, intractable to compute.

$$Z = \int e^{-c_{\theta}(\tau)} d\tau$$

$$Z = \int e^{-c_{\theta}(\tau)} d\tau = \int q(\tau) \frac{e^{-c_{\theta}(\tau)}}{q(\tau)} d\tau \approx \frac{1}{|\mathbf{D}_{\text{samp}}|} \sum_{\tau_j \in \mathbf{D}_{\text{samp}}} \frac{e^{-c_{\theta}(\tau_j)}}{q(\tau_j)}$$

Sample approximation for Z

This is a huge integral, intractable to compute.

$$Z = \int e^{-c_{\theta}(\tau)} d\tau$$

$$Z = \int e^{-c_{\theta}(\tau)} d\tau = \int q(\tau) \frac{e^{-c_{\theta}(\tau)}}{q(\tau)} d\tau \approx \frac{1}{|\mathbf{D}_{\text{samp}}|} \sum_{\tau_j \in \mathbf{D}_{\text{samp}}} \frac{e^{-c_{\theta}(\tau_j)}}{q(\tau_j)}$$

$$\mathcal{L}(\theta) = \frac{1}{|\mathbf{D}_{\text{demo}}|} \sum_{\tau_i \in \mathbf{D}_{\text{demo}}} c_{\theta}(\tau_i) + \log \left(\frac{1}{|\mathbf{D}_{\text{samp}}|} \sum_{\tau_j \in \mathbf{D}_{\text{samp}}} \frac{e^{-c_{\theta}(\tau_j)}}{q(\tau_j)} \right)$$

What q shall we choose?

$$\nabla_{\theta} \mathcal{L}(\theta) = \frac{1}{|\mathcal{D}_{\text{demo}}|} \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \frac{dc_{\theta}}{d\theta}(\tau_i) - \log \left(\frac{1}{|\mathcal{D}_{\text{samp}}|} \sum_{\tau_j \in \mathcal{D}_{\text{samp}}} \frac{e^{-c_{\theta}(\tau_j)}}{q(\tau_j)} \frac{dc_{\theta}}{d\theta}(\tau_j) \right)$$

- When is this approximation good?
- When q samples highly probable trajectories..
- Whn q is the expert policy!!
- Finding a good q is a chicken and egg problem. If I knew the expert reward function, then I 'd compute the expert policy with RL, and I 'd sample highly likely trajectories with that policy!
- Solution: iteration. Refine the sampling distribution q (policy) over time. (Finn at al. 2016)

MaxEntIRL with Adaptive Importance Sampling

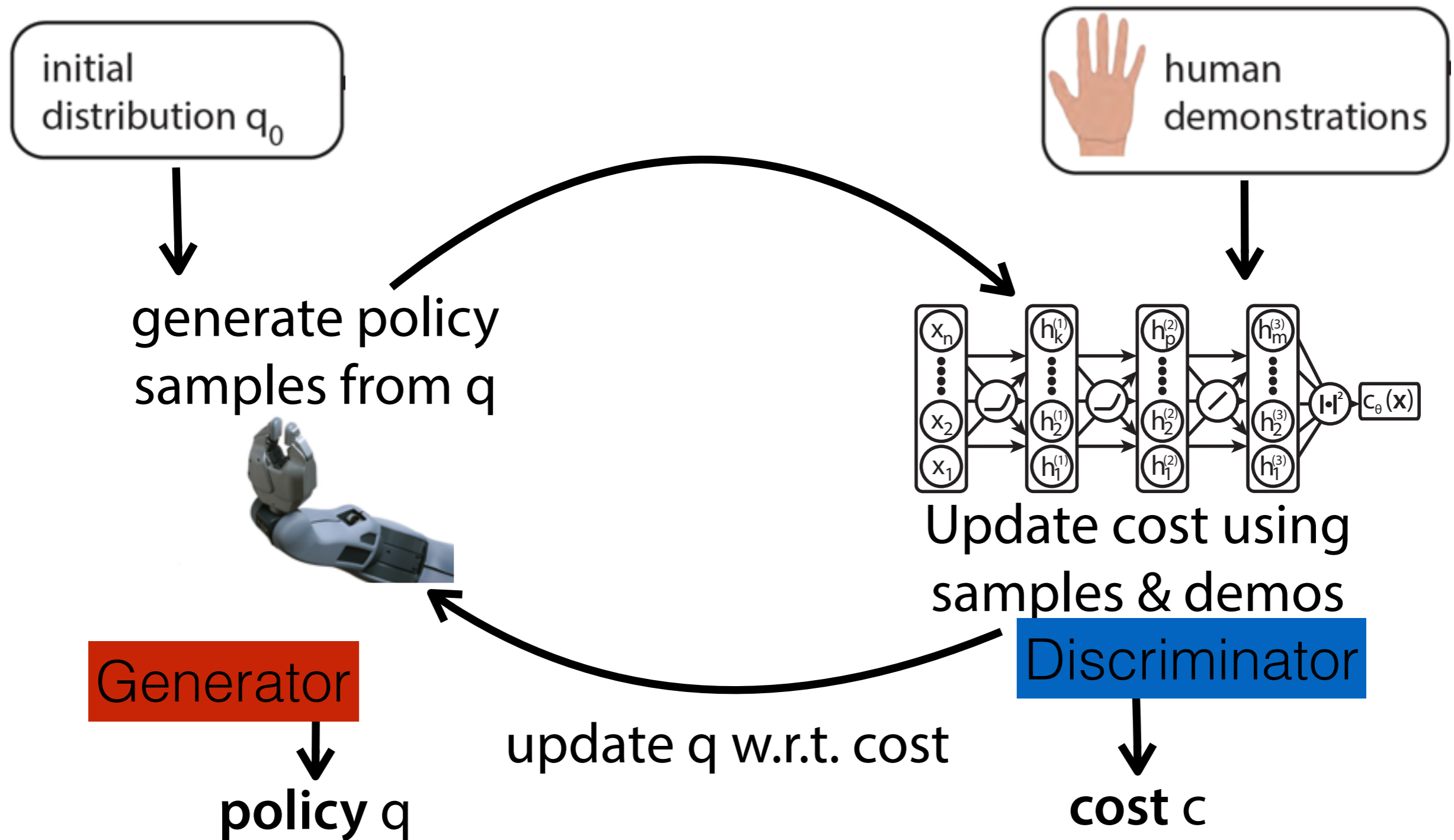
1. Initialize q_0 either from a random policy or using behavior cloning on expert demonstrations.

MaxEntIRL with Adaptive Importance Sampling

1. Initialize q_0 either from a random policy or using behavior cloning on expert demonstrations.
2. for iteration $k = 1 \dots I$
 3. Generate samples \mathbf{D}_{traj} from $q_k(\tau)$
 4. Append samples: $\mathbf{D}_{samp} \leftarrow \mathbf{D}_{samp} \cup \mathbf{D}_{traj}$.
 5. Use \mathbf{D}_{samp} to update cost c_θ using gradient descent.
 6. Update $q_k(\tau)$ using any RL method

$$\nabla_{\theta} \mathcal{L}(\theta) = \frac{1}{|\mathbf{D}_{demo}|} \sum_{\tau_i \in \mathbf{D}_{demo}} \frac{dc_{\theta}}{d\theta}(\tau_i) - \log \left(\frac{1}{|\mathbf{D}_{samp}|} \sum_{\tau_j \in \mathbf{D}_{samp}} \frac{e^{-c_{\theta}(\tau_j)}}{q(\tau_j)} \frac{dc_{\theta}}{d\theta}(\tau_j) \right)$$

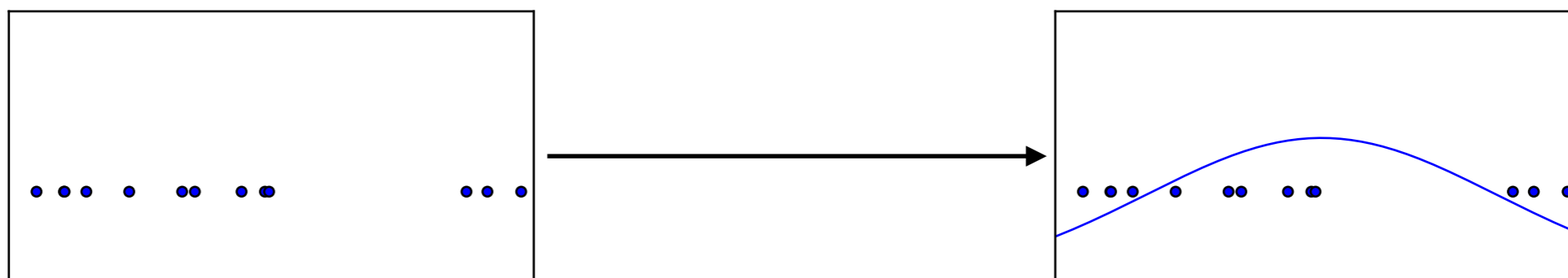
MaxEntIRL with Adaptive Importance Sampling



The discriminator adjusts the cost so that it makes the expert trajectories be better distinguished from the generated ones

Generative models-density estimation

- So far we have been seeking to learn a generative model of trajectories, by computing trajectory densities:
- We were trying to estimate a model that given a trajectory will be able to output the probability of this trajectory: expert trajectories should be highly probable, and non-expert less probable
- This is in general what we do when we maximize likelihood of the data
- The problem is that probabilities need to sum to one

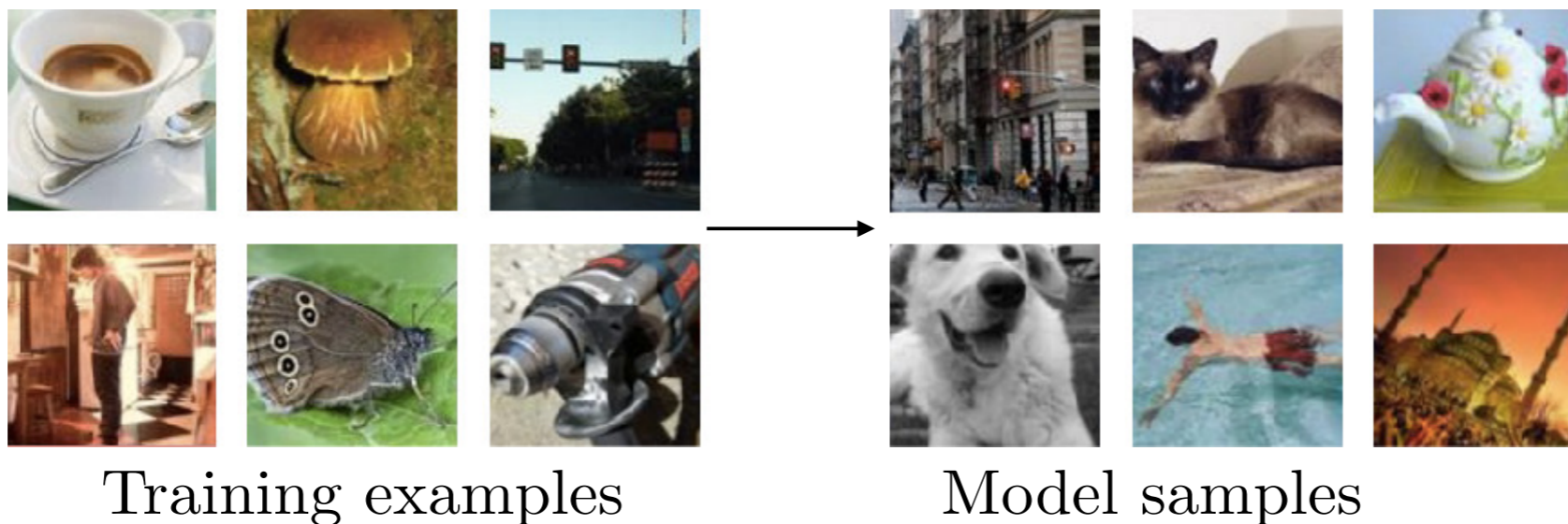


$$p(\tau | \theta) = \frac{e^{-\text{cost}(\tau|\theta)}}{\sum_{\tau'} e^{-\text{cost}(\tau'|\theta)}}$$

$$\theta^* = \max_{\theta} \frac{1}{m} \sum_{i=1}^m \log p(x^{(i)}; \theta)$$

Generative models-sample generation

- Recently, new classes of generative models has been proposed that instead of computing densities, they learn directly a sampler, without necessarily having an explicit density.

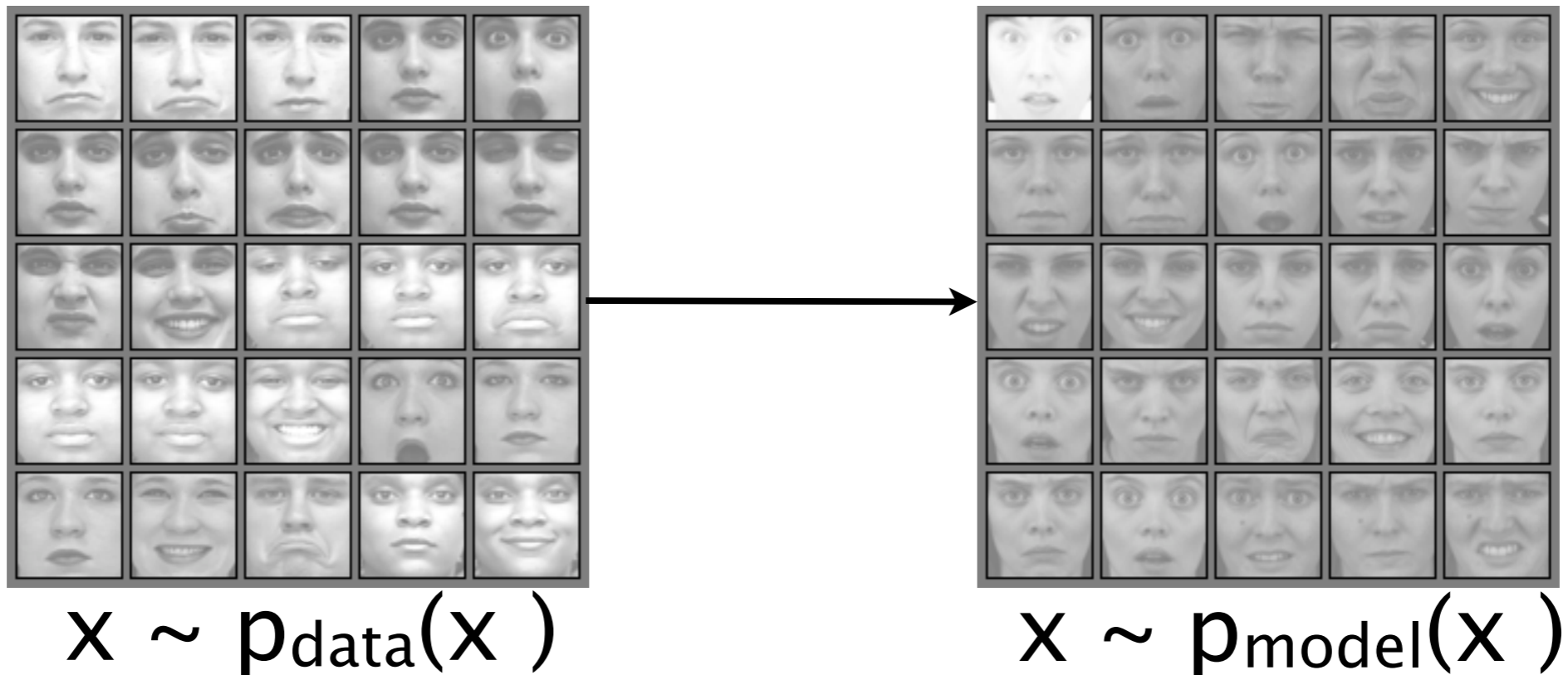


(Goodfellow 2016)

- Have we done this for trajectories?
- Well, we used behavior cloning, but assumed access to a teacher

Generative models-sample generation

- Have training examples $\mathbf{x} \sim \mathbf{p}_{\text{data}}(\mathbf{x})$
- Want a model that can draw samples: $\mathbf{x} \sim \mathbf{p}_{\text{model}}(\mathbf{x})$
- Where $\mathbf{p}_{\text{model}} \approx \mathbf{p}_{\text{data}}$



The sampling can be both conditional and unconditional

male -> female



The sampling can be both conditional and unconditional

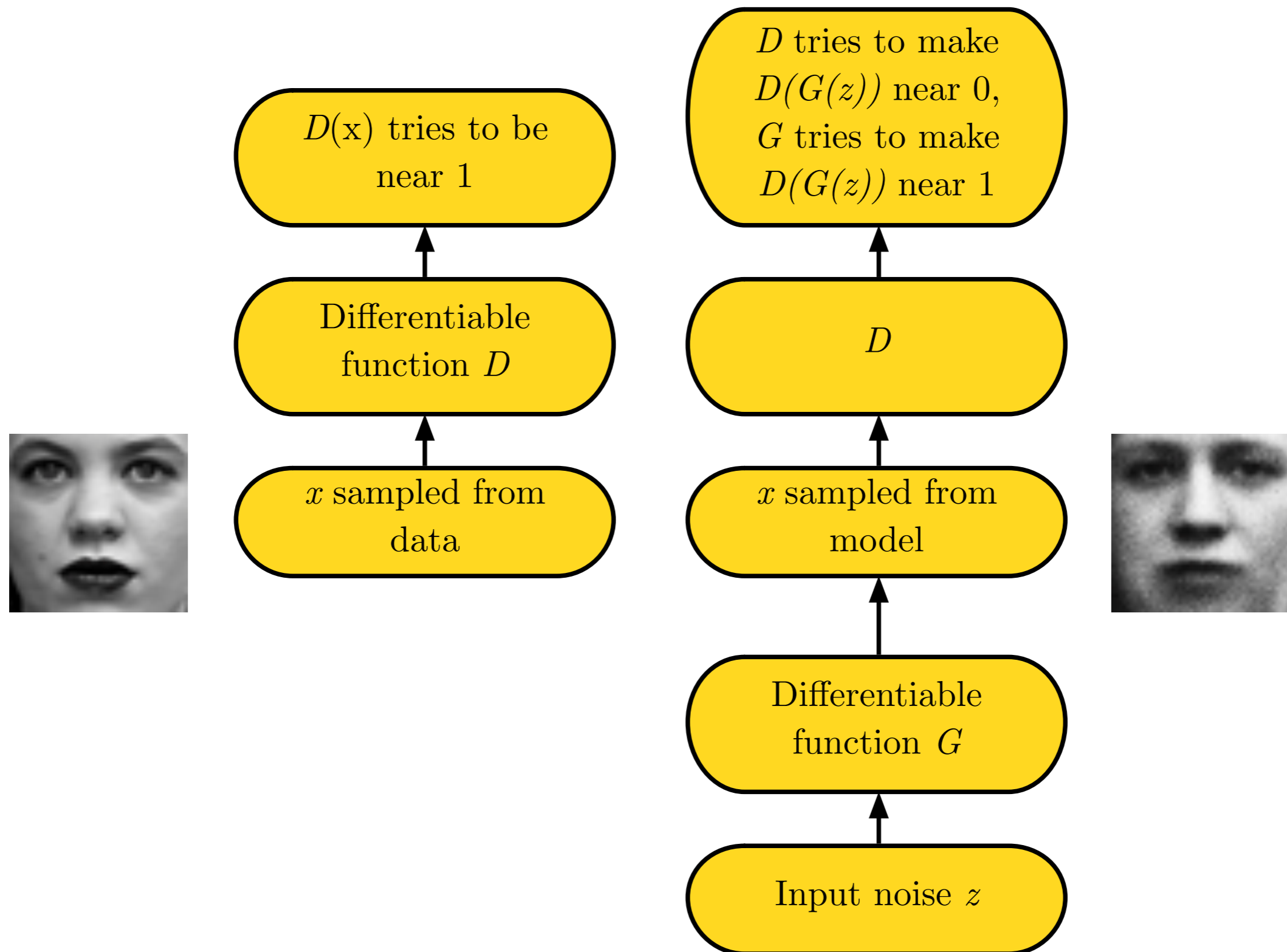
anybody -> Tom Cruise



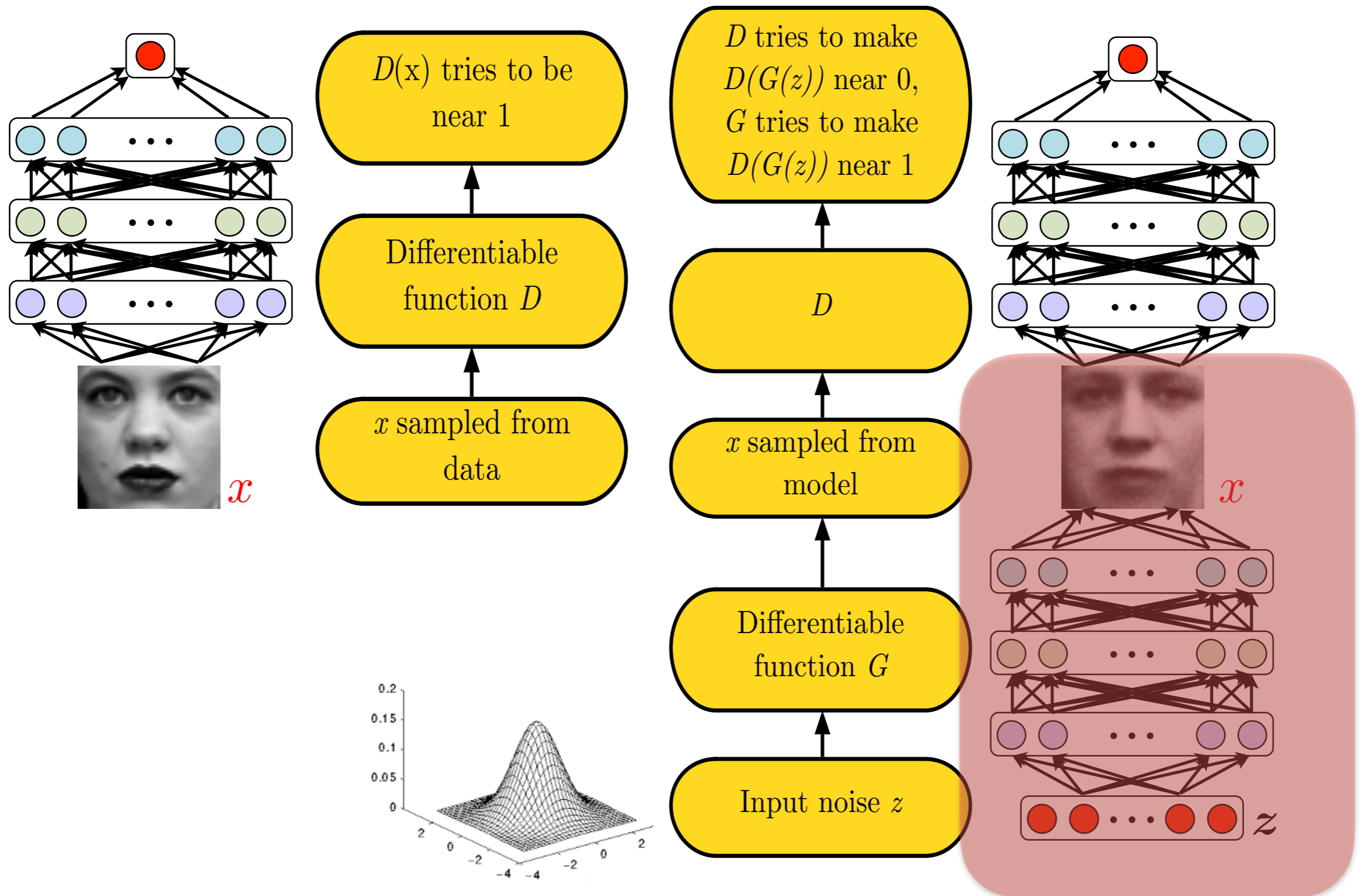
Generative Adversarial Networks

- General strategy: Do not write a formula for $p(x)$, **just learn to sample directly**. No intractable summations!
- A game between two players:
 1. Discriminator D
 2. Generator G
- D tries to discriminate between:
 - A sample from the data distribution
 - And a sample from the generator G
- G tries to “trick” D by generating samples that are hard for D to distinguish from data

Generative Adversarial Networks



Generative Adversarial Networks

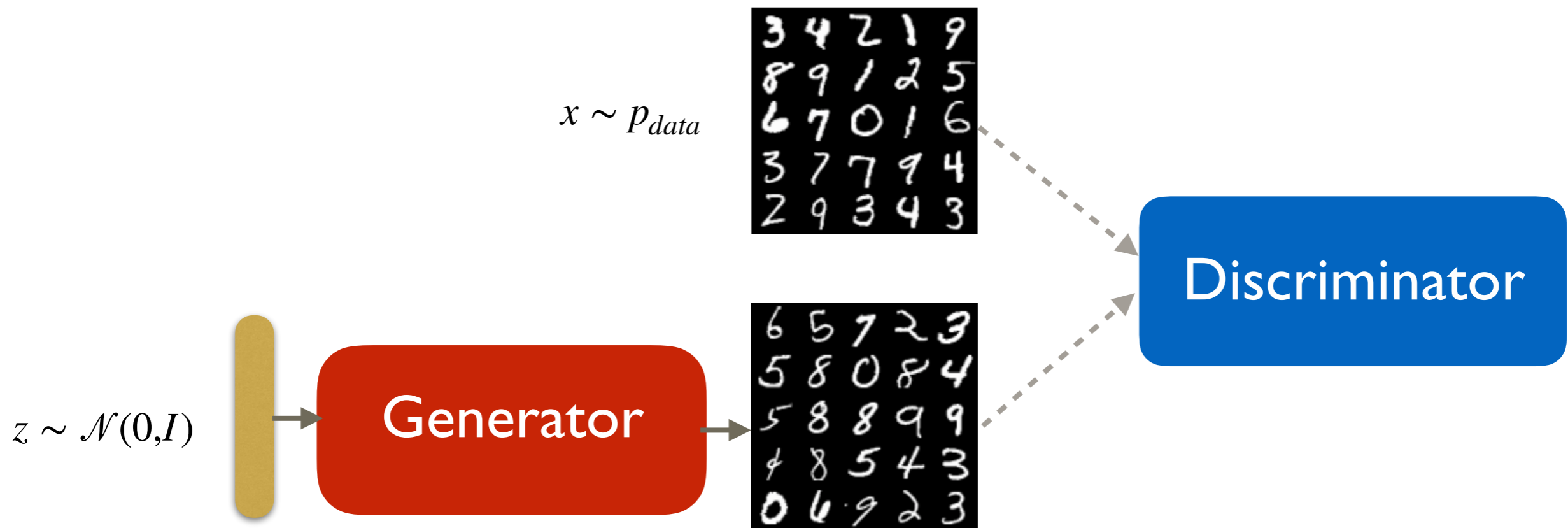


That's our sampler! The rest are only used at training time.

Generative Adversarial Networks

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

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

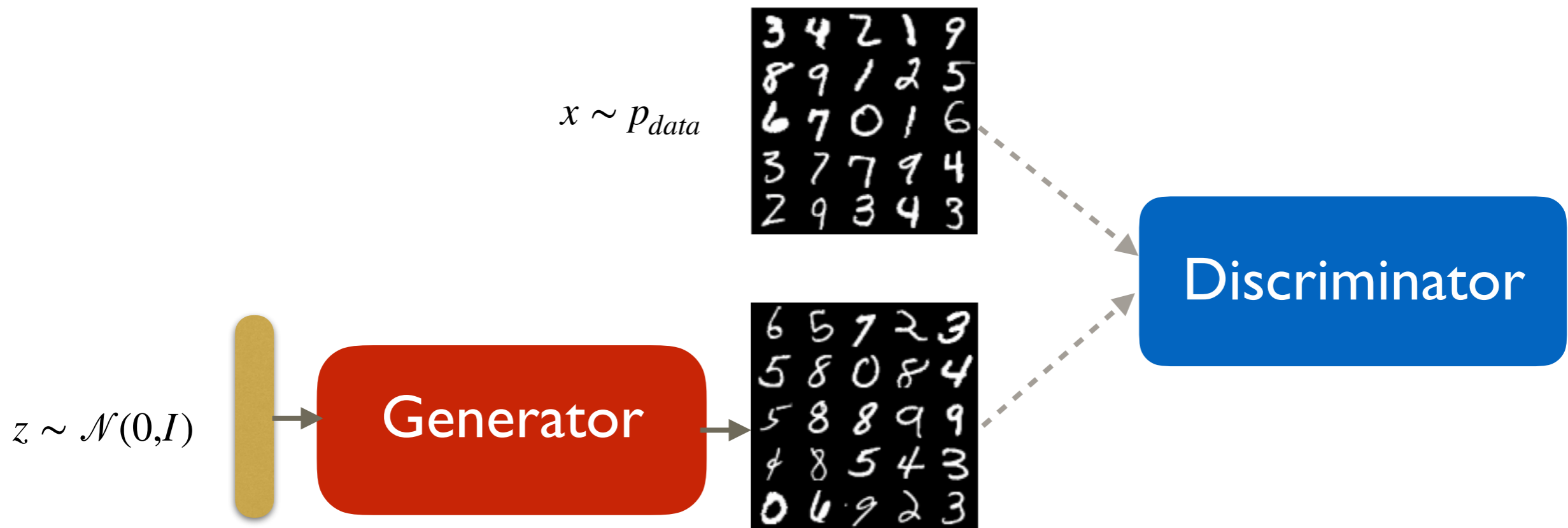


Generative Adversarial Networks-in practise

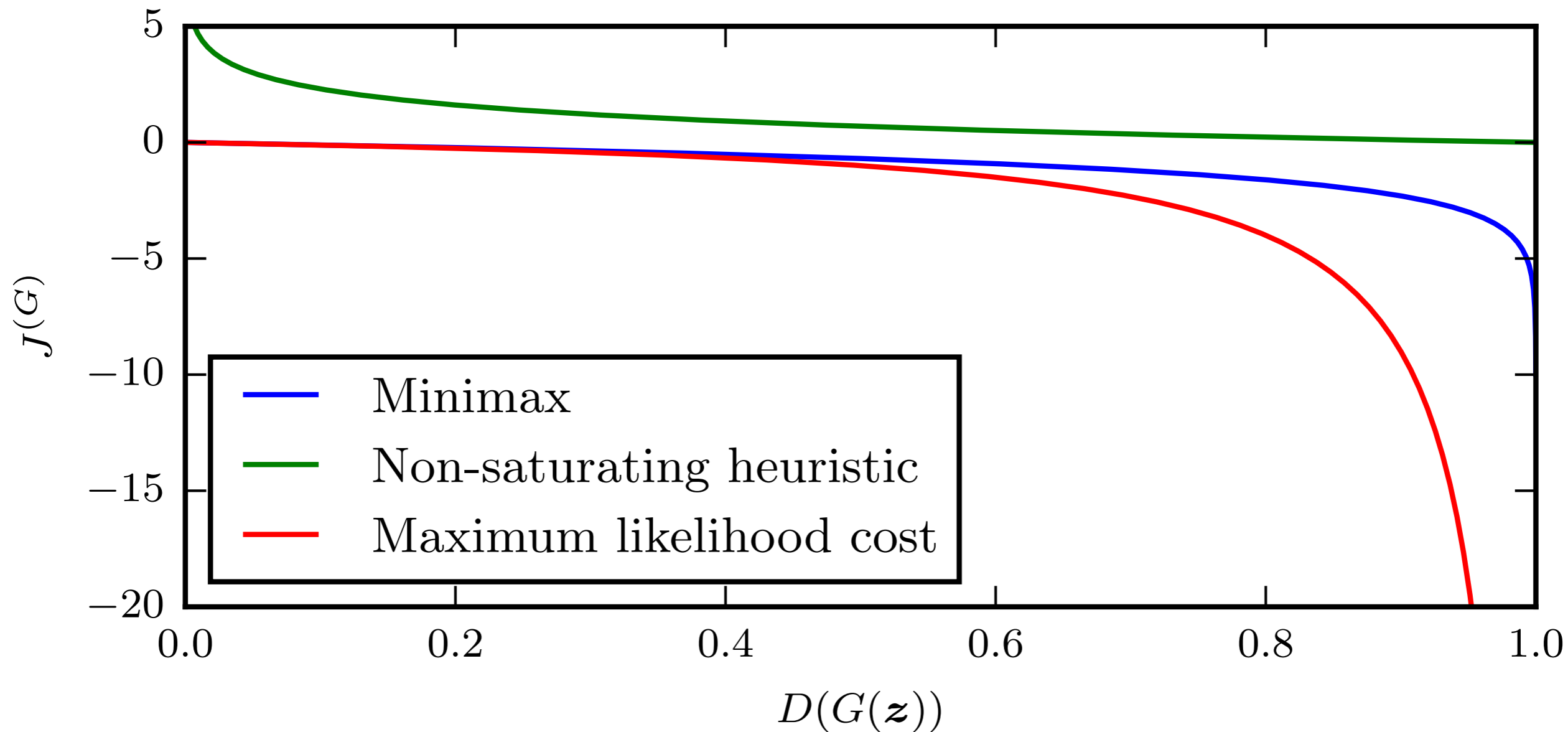
$D(x)$: the probability that x came from the real data rather than the generator

$$\max_D \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

$$\max_G \mathbb{E}_{z \sim p_z(z)} [\log(D(G(z)))]$$



Comparison of generator losses

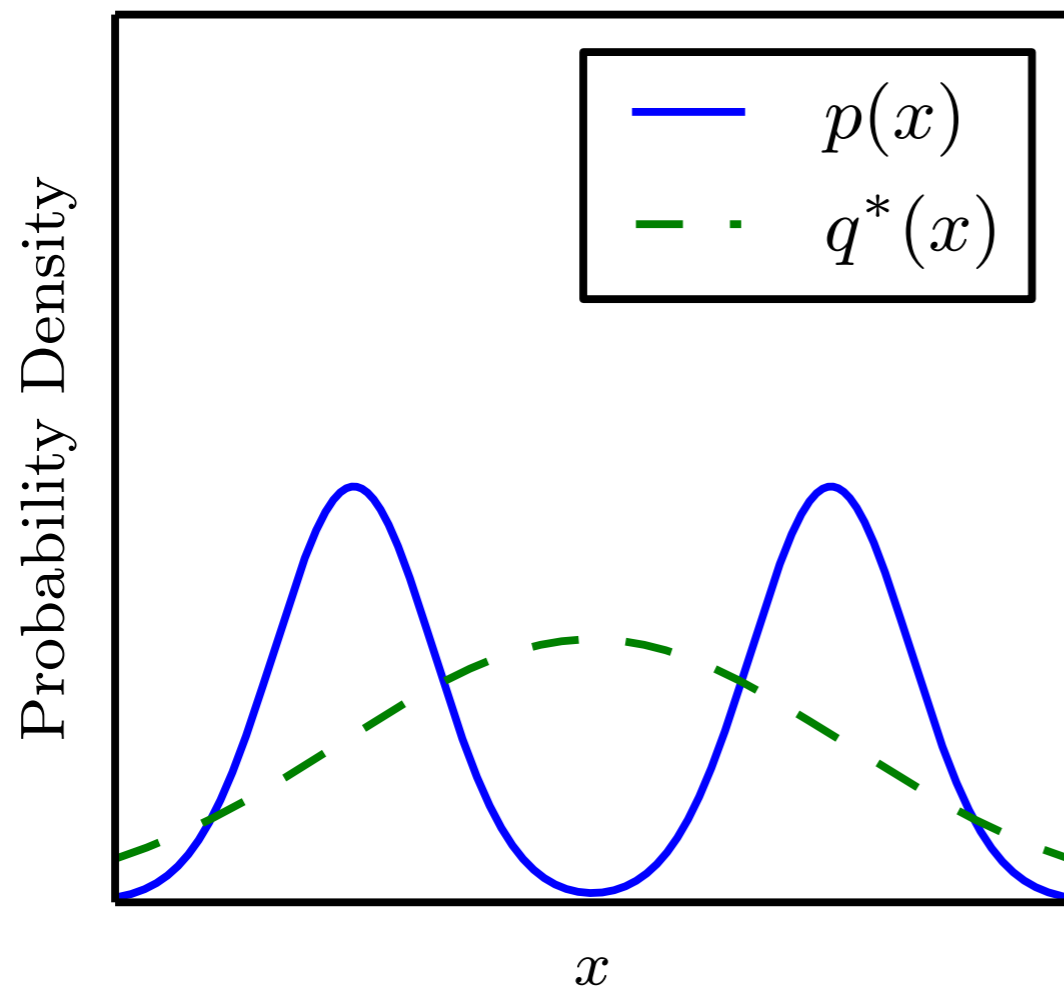


(Goodfellow 2014)

(Goodfellow 2016)

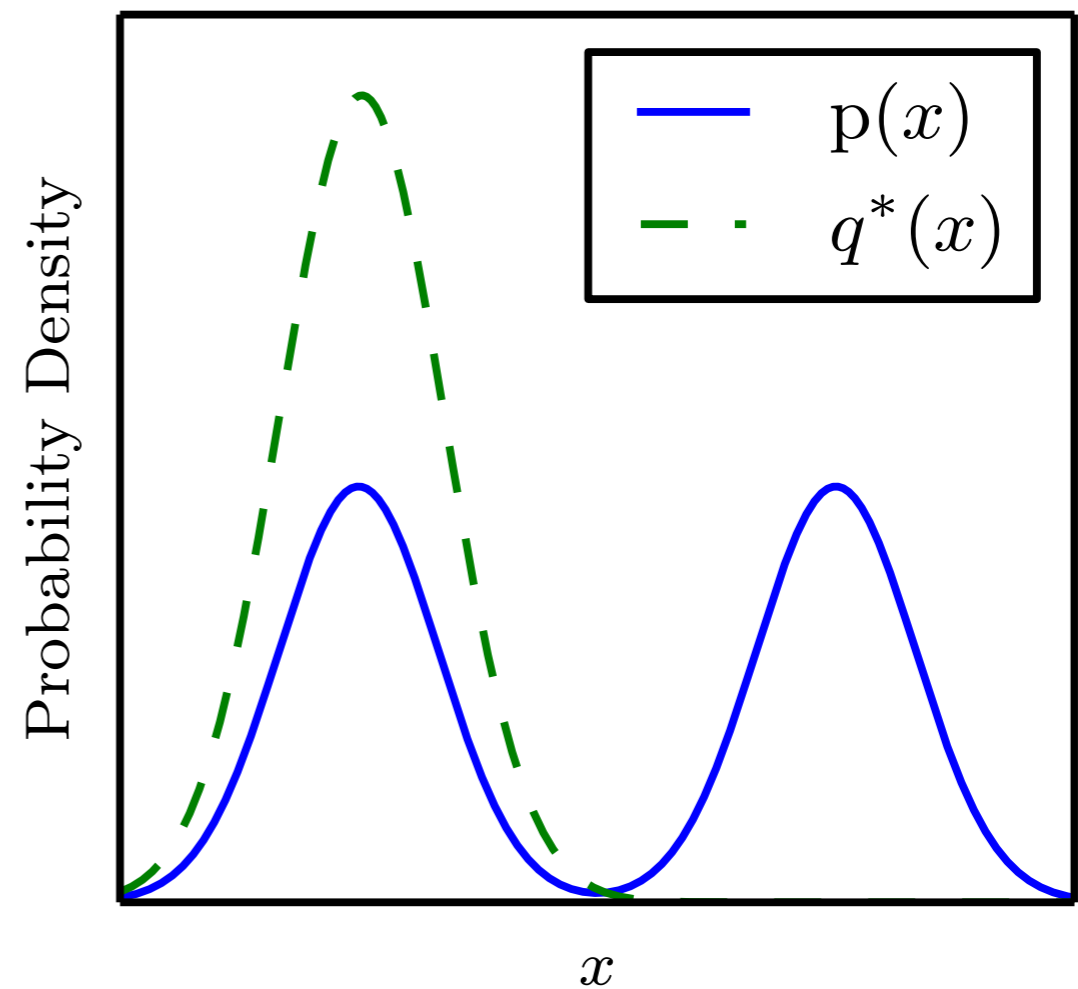
Generative Adversarial Networks

$$q^* = \operatorname{argmin}_q D_{\text{KL}}(p||q)$$



Maximum likelihood

$$q^* = \operatorname{argmin}_q D_{\text{KL}}(q||p)$$



Reverse KL

(Goodfellow et al 2016)

Generative Adversarial Imitation learning

Find a policy π_θ that makes it impossible for a discriminator network to distinguish between trajectory chunks visited by the expert and by the learner's application of π_θ

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

D outputs 1 if state comes from the demo policy

$$\min_{\pi_\theta} \max_D \mathbb{E}_{\pi}^* [\log D(s)] + \mathbb{E}_{\pi_\theta} [\log(1 - D(s))]$$

Reward for the policy optimization is how well I matched the demo trajectory distribution, else, how well I confused the discriminator:

$$\log D(s)$$

Generative Adversarial Imitation Learning

Jonathan Ho
Stanford University
hoj@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

NIPS 2016

Algorithm 1 Generative adversarial imitation learning

- 1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
- 2: **for** $i = 0, 1, 2, \dots$ **do**
- 3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
- 4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i} [\nabla_w \log(D_w(s, a))] + \hat{\mathbb{E}}_{\tau_E} [\nabla_w \log(1 - D_w(s, a))] \quad (17)$$

- 5: Take a policy step from θ_i to θ_{i+1} , using the TRPO rule with cost function $\log(D_{w_{i+1}}(s, a))$. Specifically, take a KL-constrained natural gradient step with

$$\hat{\mathbb{E}}_{\tau_i} [\nabla_{\theta} \log \pi_{\theta}(a|s) Q(s, a)] - \lambda \nabla_{\theta} H(\pi_{\theta}), \quad (18)$$

where $Q(\bar{s}, \bar{a}) = \hat{\mathbb{E}}_{\tau_i} [\log(D_{w_{i+1}}(s, a)) \mid s_0 = \bar{s}, a_0 = \bar{a}]$

- 6: **end for**
-

Generative Adversarial Imitation learning

