Carnegie Mellon
School of Computer Science

Deep Reinforcement Learning and Control

Maximum Entropy Inverse RL, Adversarial imitation learning

Katerina Fragkiadaki

Reinforcement Learning

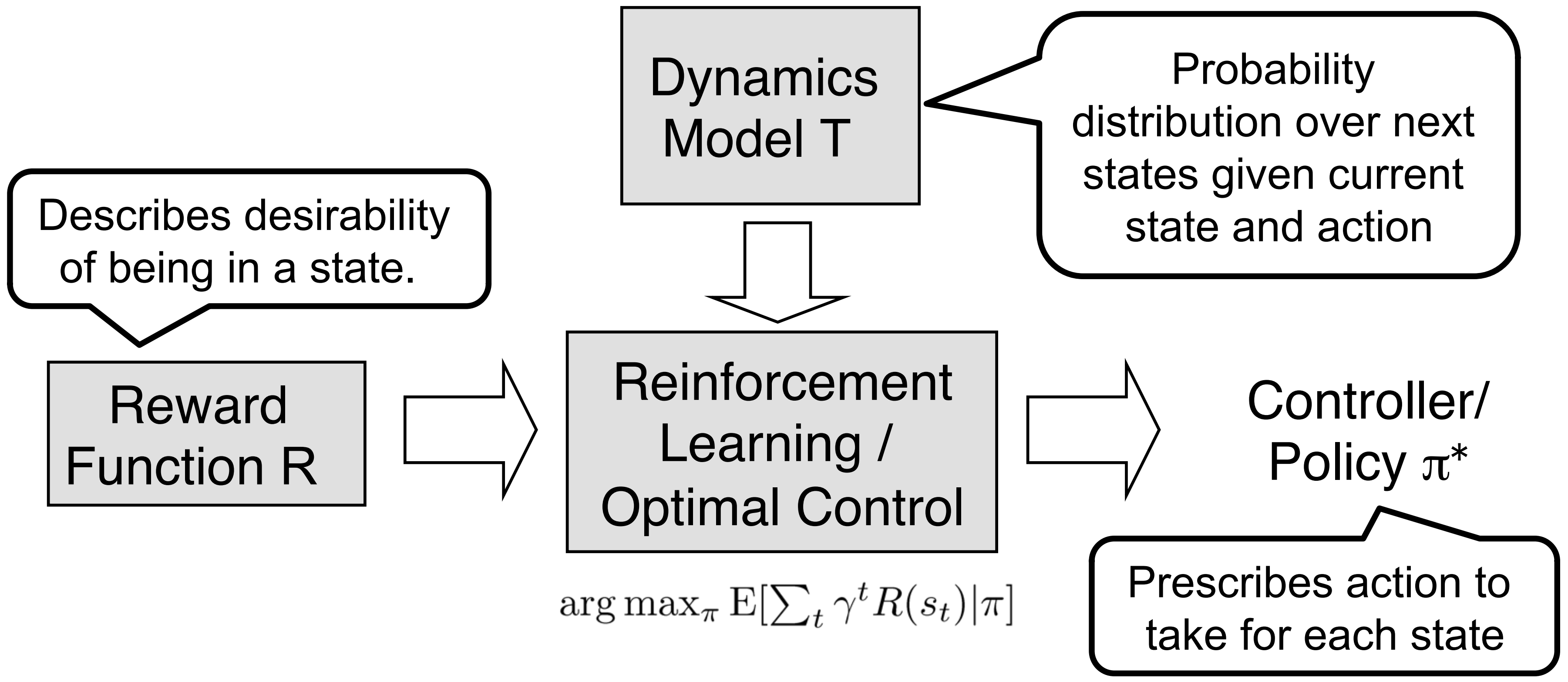

Dynamics Model T
Probability distribution over next states given current state and action
Describes desirability of being in a state.
Reward Function R
Reinforcement Learning / Optimal Control
$\arg\max_\pi \mathrm{E}[\sum_t \gamma^t R(s_t)|\pi]$
Controller/ Policy π*
Prescribes action to take for each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

Diagram: Pieter Abbeel

Inverse Reinforcement Learning

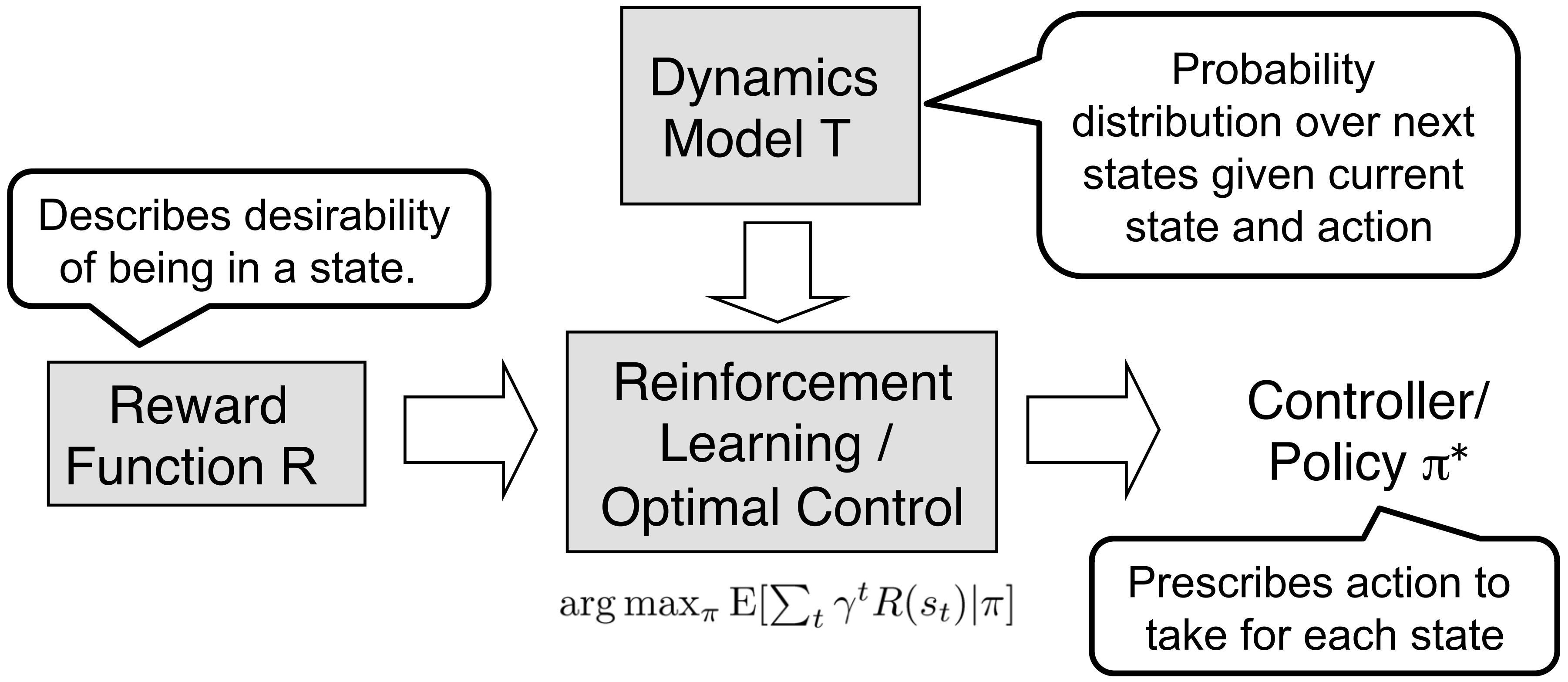

Dynamics Model T
Probability distribution over next states given current state and action
Describes desirability of being in a state.
Reward Function R
Reinforcement Learning / Optimal Control
$\arg\max_\pi \mathrm{E}[\sum_t \gamma^t R(s_t)|\pi]$
Controller/ Policy π^*
Prescribes action to take for each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

Diagram: Pieter Abbeel

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

Inverse Reinforcement Learning

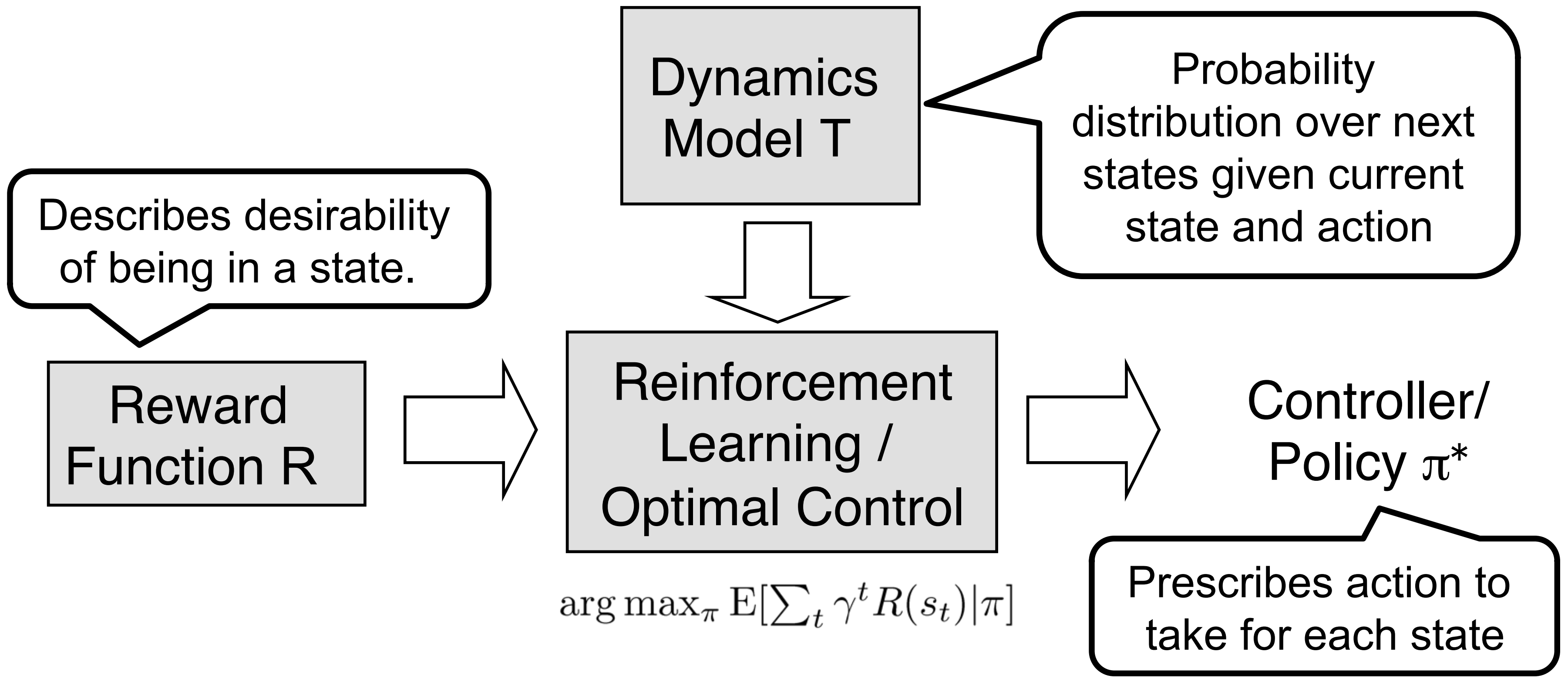
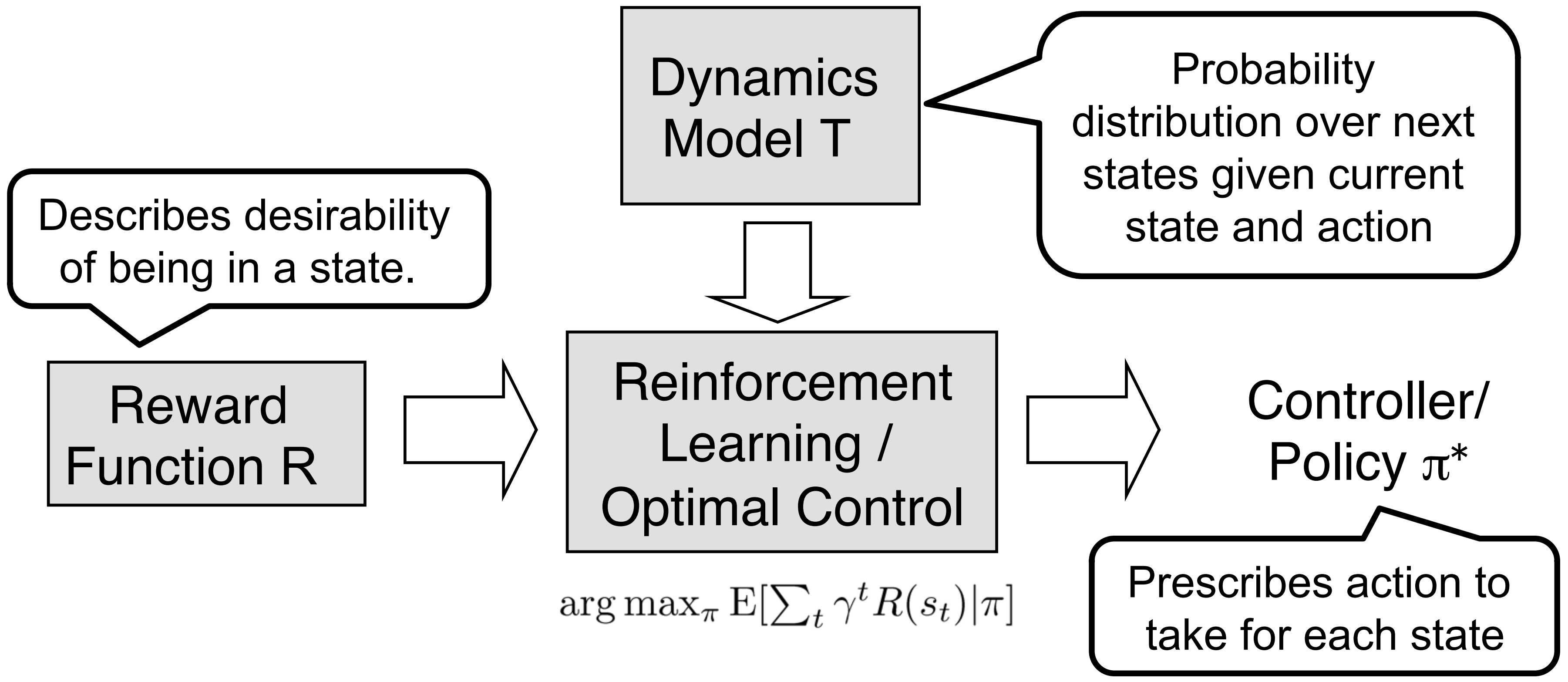

Dynamics Model T
Probability distribution over next states given current state and action
Describes desirability of being in a state.
Reward Function R
Reinforcement Learning / Optimal Control
$\arg\max_\pi \mathrm{E}[\sum_t \gamma^t R(s_t)|\pi]$
Controller/ Policy π^*
Prescribes action to take for each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

Diagram: Pieter Abbeel

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.

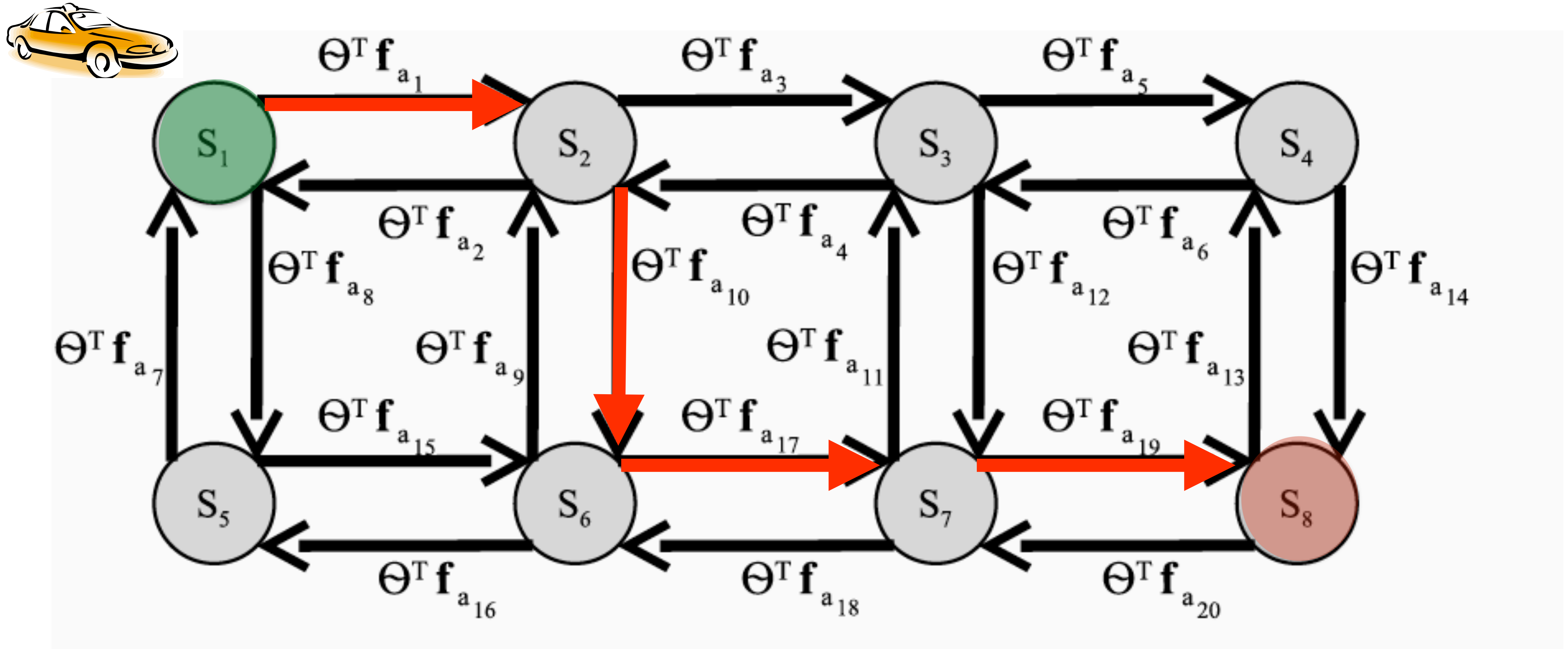
Inverse Reinforcement Learning


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
$\arg\max_\pi \mathrm{E}[\sum_t \gamma^t R(s_t)|\pi]$
Controller/ Policy π^*
Prescribes action to take for each state

Mathematically imitation boils down to a distribution matching problem: the learner needs to come up with a reward/policy whose resulting state, action trajectory distribution matches the expert trajectory distribution.

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?

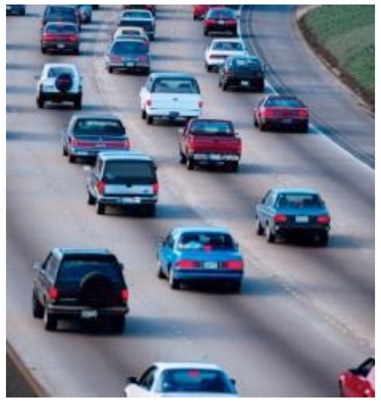


- Assumption: cost is independent of the goal state, so it only depends on road features, e.g., traffic width tolls etc.

State features

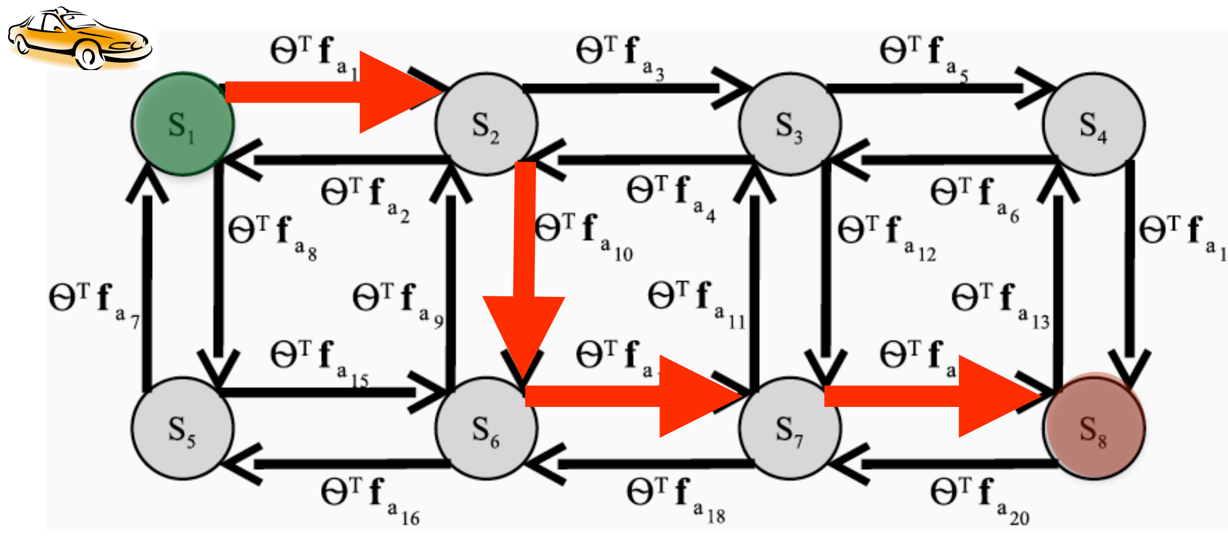
Features f can be:

# Bridges crossed



# Miles of interstate

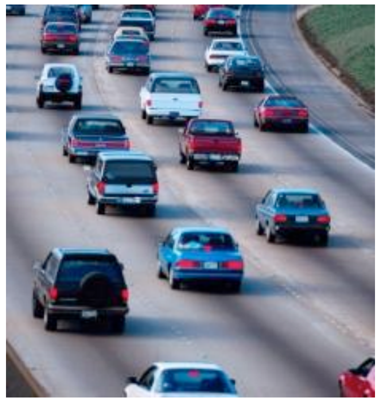
# Stoplights



A good guess: Match expected features

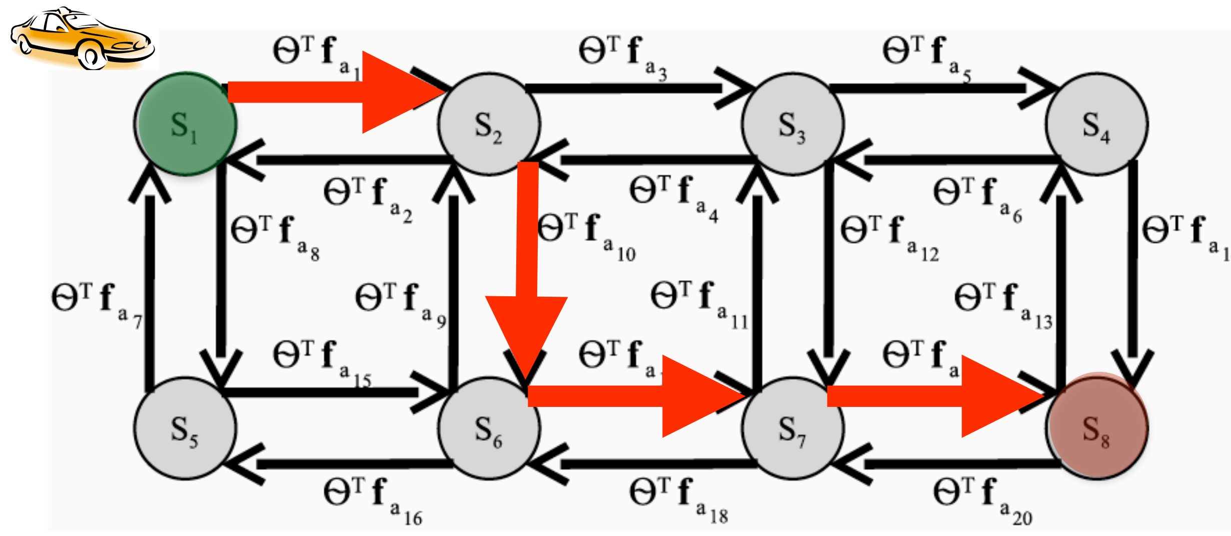
Features f can be:

# Bridges crossed



# Miles of interstate

# Stoplights



Feature matching:

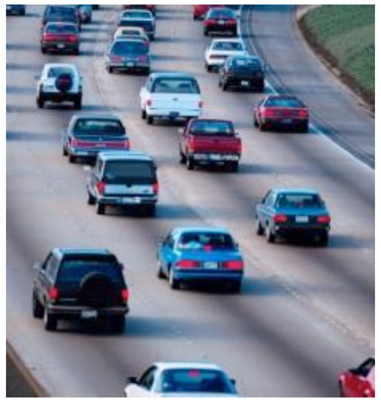
$$\sum_{\text{Path}\,\tau_i} P(\tau_i) f_{\tau_i} = \tilde{\text{f}}$$

"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."

A good guess: Match expected features

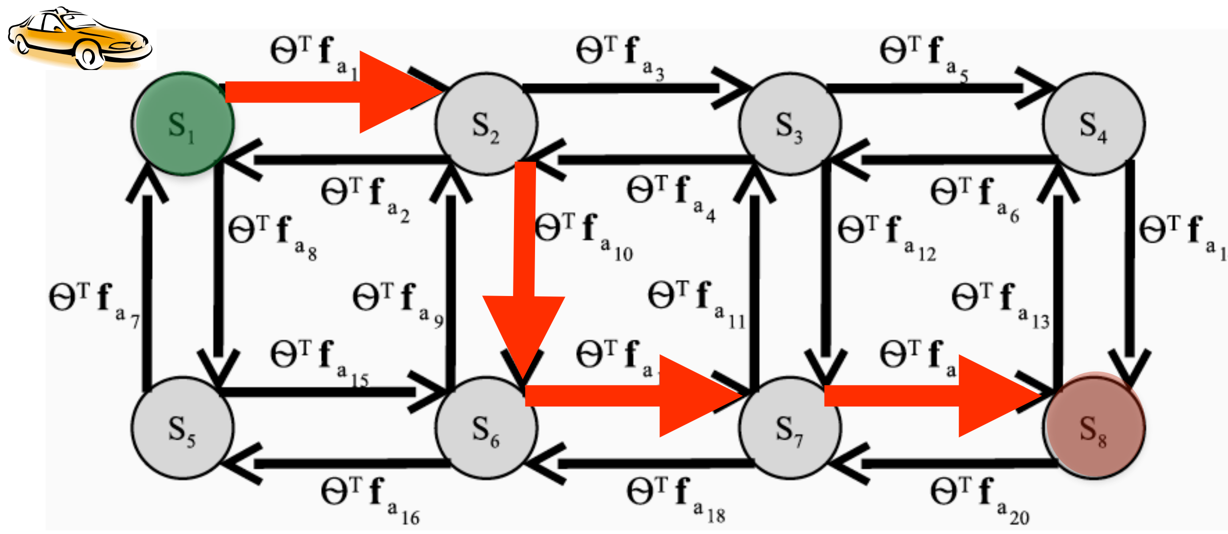
Features f can be:

# Bridges crossed



# Miles of interstate

# Stoplights



Feature matching:

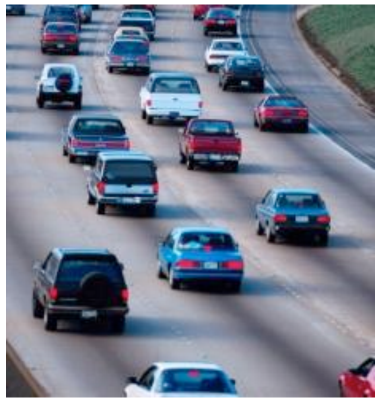
$$\sum_{\text{Path}\,\tau_i} P(\tau_i) f_{\tau_i} = \tilde{\mathrm{f}}$$

Demonstrated feature counts

A good guess: Match expected features

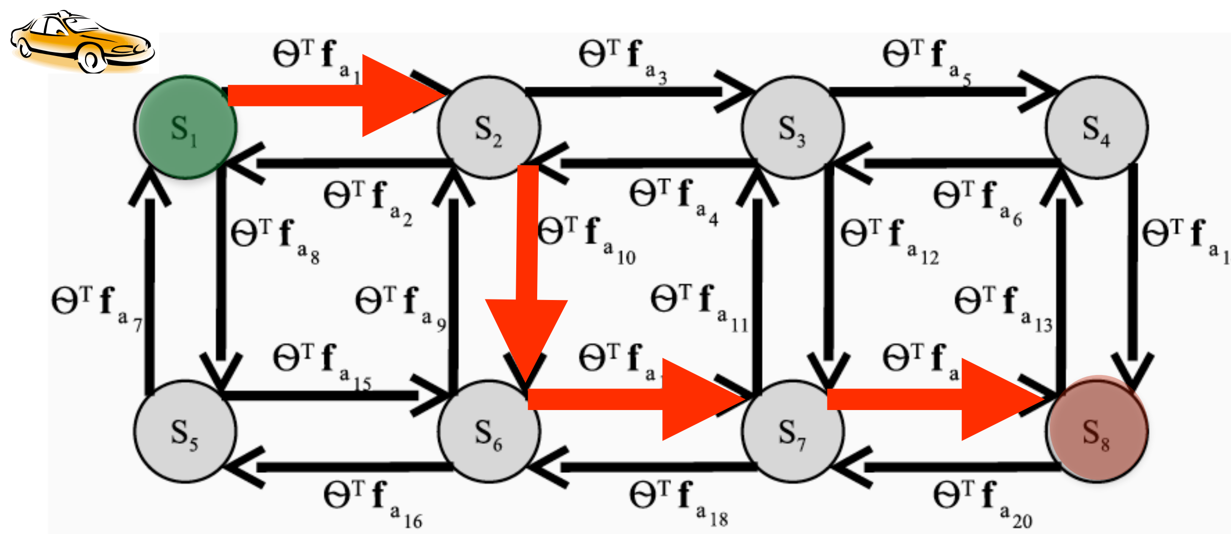
Features f can be:

# Bridges crossed



# Miles of interstate

# Stoplights



Feature matching:

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

Demonstrated feature counts

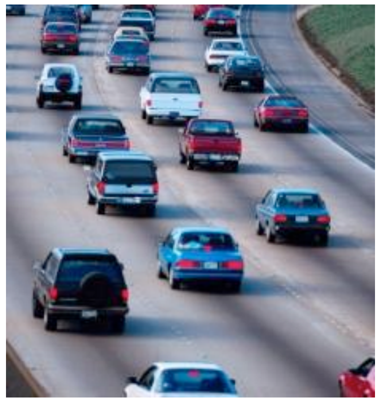
a policy induces a distribution over trajectories

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

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:

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

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

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i)$$

The probability distribution which best represents the current state of knowledge is the one with largest entropy, in the context of precisely stated prior data (such as a proposition that expresses testable information).

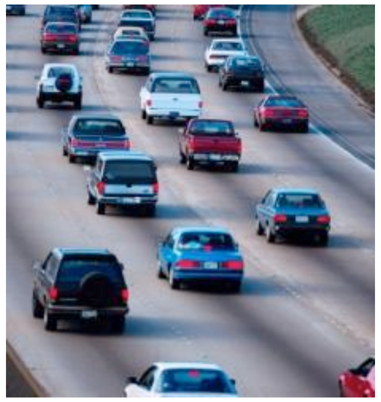
Another way of stating this: Take precisely stated prior data or testable information about a probability distribution function. Consider the set of all trial probability distributions that would encode the prior data. The distribution with maximal information entropy is the best choice.

- Maximizing entropy minimizes the amount of prior information built into the distribution
- Many physical systems tend to move towards maximal entropy configurations over time

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!

$$\max_P - \sum_\tau P(\tau) \log P(\tau)$$

Feature matching constraint:

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

Demonstrated feature counts

a policy induces a distribution over trajectories

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

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):

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

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

From features to costs

Cost of a trajectory τ (linear):

$$c_\theta(\tau) = \theta^T \mathbf{f}_\tau = \sum_{s \in \tau} \theta^T \mathbf{f}_s$$

Constraint: Match the cost of expert trajectories in expectation:

$$\int p(\tau) c_\theta(\tau) d\tau = \frac{1}{|D|} \sum_{\tau^* \in D_\tau} c_\theta(\tau^*)$$

Maximum Entropy

$$\min. \quad -H(p(\tau))$$

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

From maximum entropy to exponential family

Maximum Entropy
$$\min. \quad -H(p(\tau))$$
$$\text{s.t.} \quad \int p(\tau)c_\theta(\tau)d\tau = \tilde{c}, \int p(\tau)d\tau = 1$$

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

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

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

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

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

From maximum entropy to exponential family

- 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

Maximum Likelihood

$$\max_{\theta} . \log \prod_{\tau^* \in D} p(\tau^*) \iff \max_{\theta} . \sum_{\tau^* \in D} \log p(\tau^*)$$

$$\max_{\theta} . \sum_{\tau^* \in D} \log \frac{e^{-c_\theta(\tau^*)}}{Z}$$

$$\max_{\theta} . \sum_{\tau^* \in D} -c_\theta(\tau^*) - \sum_{\tau^*} \log(\sum_{\tau} e^{-c_\theta(\tau)})$$

$$\max_{\theta} . \sum_{\tau^* \in D} -c_\theta(\tau^*) - \log(\sum_{\tau} e^{-c_\theta(\tau)})|D|$$

$$\min_{\theta} . \sum_{\tau^* \in D} c_\theta(\tau^*) + |D| \log(\sum_{\tau} e^{-c_\theta(\tau)}) \rightarrow J(\theta)$$

$$\nabla_\theta J(\theta) = \sum_{\tau^* \in D} \frac{dc_\theta(\tau^*)}{d_\theta} + |D| \frac{1}{\sum_\tau e^{-c_\theta(\tau)}} \sum_{\tau} \left(e^{-c_\theta(\tau)} \left(-\frac{dc_\theta(\tau)}{d\theta} \right) \right)$$

$$= \sum_{\tau^* \in D} \frac{dc_\theta(\tau^*)}{d_\theta} - |D| \sum_{\tau} p(\tau|\theta) \frac{dc_\theta(\tau)}{d\theta}$$

From trajectories to states

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

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

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

$$\nabla_\theta J(\theta) = \sum_{s \in \tau^* \in D} \frac{dc_\theta(s)}{d\theta} - |D| \sum_s p(s|\theta, \tau) \frac{dc_\theta(s)}{d\theta}$$

Successful imitation boils down to learning a policy that matches the state visitation distribution (or state/action visitation distribution)

$$\sum_{s,a} p(s, a|\theta, \tau) \frac{dc_\theta(s, a)}{d\theta}$$

State densities

In the tabular case and for known dynamics we can compute them with dynamic programming, assuming we have obtained the policy:

$$\mu_1(s) = p(s_s)$$

Time indexed state densities

$$\text{for } t = 1, \ldots, T$$

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

$$p(s|\theta, \mathcal{T}) = \sum_t \mu_t(s)$$

$$\nabla_\theta J(\theta) = \sum_{s_t \in \tau^* \in D} \frac{dc_\theta(s)}{d\theta} - |D| \sum_s p(s|\theta, \mathcal{T}) \frac{dc_\theta(s)}{d\theta}$$

For linear costs: $c_\theta(s) = \theta^T f_s$

$$\nabla_\theta J(\theta) = \sum_{s \in \tau^*} f_s + |D| \sum_s p(s|\theta, \mathscr{T}) f_s$$

Maximum entropy Inverse RL

Known dynamics, linear costs

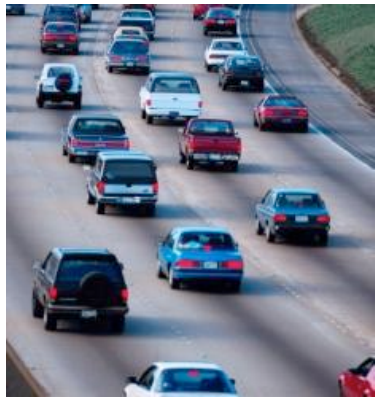
0. Initialize θ, gather demonstrations $\mathcal{D}$
1. Solve for optimal policy $\pi(a|s)$ w.r.t. c_θ with value iteration
2. Solve for state visitation frequencies $p(s \mid \theta, T)$
3. Compute gradient $\nabla_\theta \mathcal{L} = \frac{1}{M} \sum_{\tau_d \in \mathcal{D}} \mathbf{f}_{\tau_d} -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

Maximum entropy Inverse RL

Demonstrated Behavior

Bridges crossed: **3**

Miles of interstate: **20.7**



Stoplights: **10**

Model Behavior (Expectation)

Bridges crossed: **?**

Miles of interstate: **?**

Cost Weight: 5.0

Stoplights: **?**

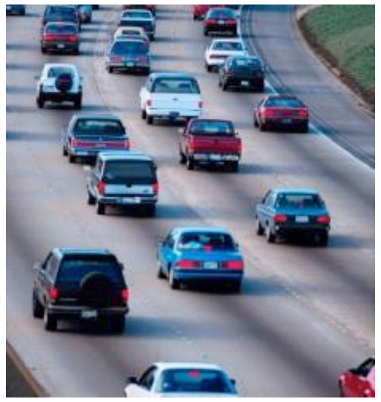
Cost Weight: 3.0

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

-4.5

Stoplights : **7.4**

-2.6

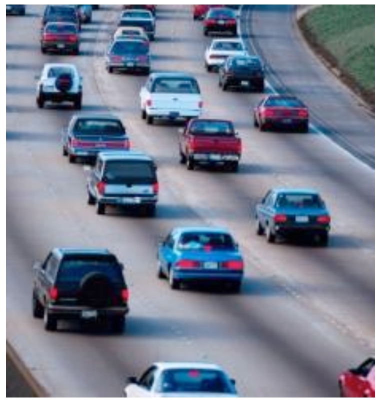
Cost Weight: 3.0

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

Stoplights: **7.4**

1.1

Cost Weight:

Limitations of MaxEntIRL

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

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

MaxEnt IRL 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 IRL 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:

$$c_\theta(x_t, u_t) = \theta^\top \mathbf{f}(x_t, u_t)$$

MaxEnt IRL 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:

$$c_\theta(x_t, u_t) = \theta^\top \mathbf{f}(x_t, u_t)$$

In the form of a loss function:

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

Approximating Z with Importance Sampling

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

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

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

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

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

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

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

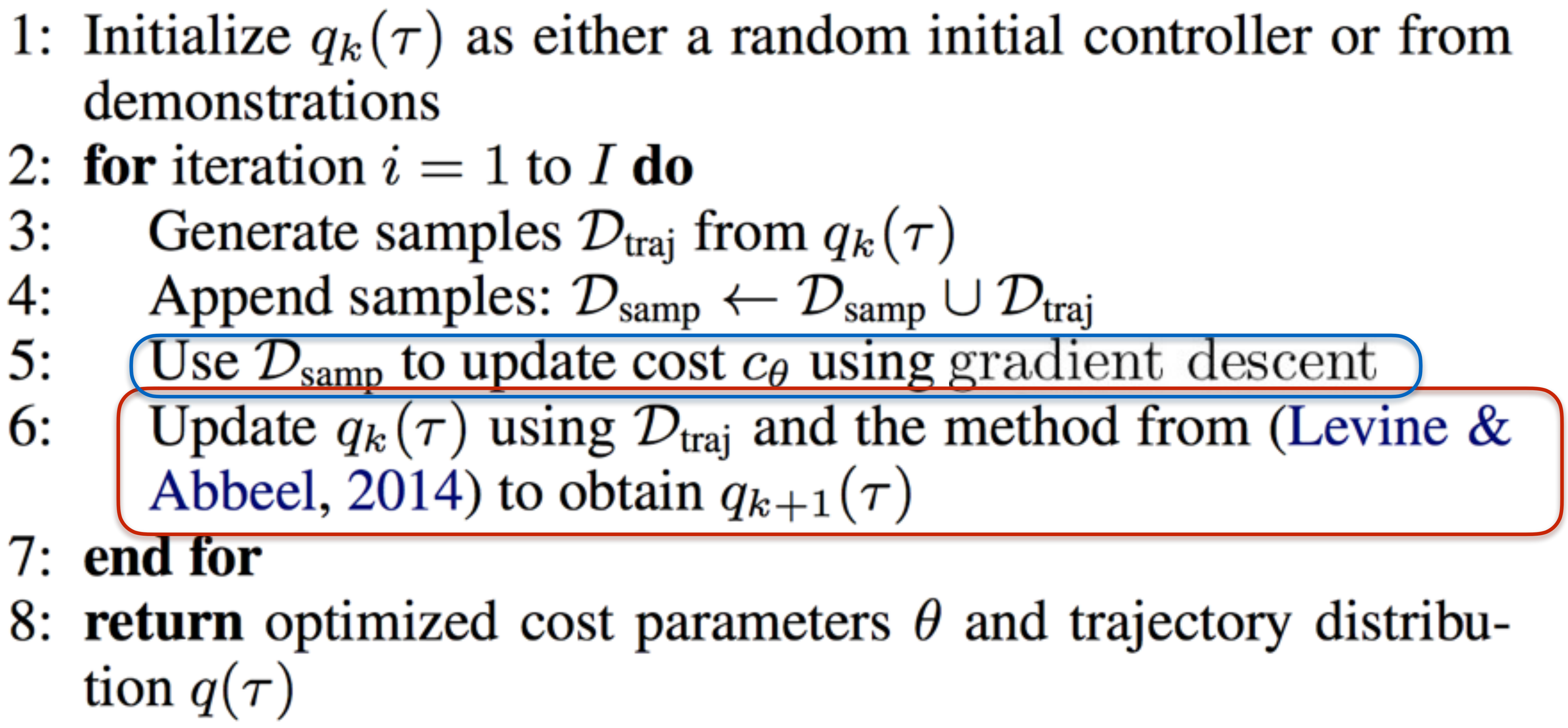
$$\frac{d\mathcal{L}_{\text{IOC}}}{d\theta} = \frac{1}{N} \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} \frac{dc_\theta}{d\theta}(\tau_i) - \frac{1}{Z} \sum_{\tau_j \in \mathcal{D}_{\text{samp}}} w_j \frac{dc_\theta}{d\theta}(\tau_j)$$

Adapting the sampling distribution q

What should be the 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 IRL with policy optimization, then sample trajectories according to the policy -> better trajectories (have much higher likelihood) guided by your current estimate of the cost

MaxEntIRL with Adaptive Importance Sampling



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

This can be any method that given rewards computes a policy (the forward RL problem)

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

Diagram from Chelsea Finn

MaxEntIRL with Adaptive Importance Sampling

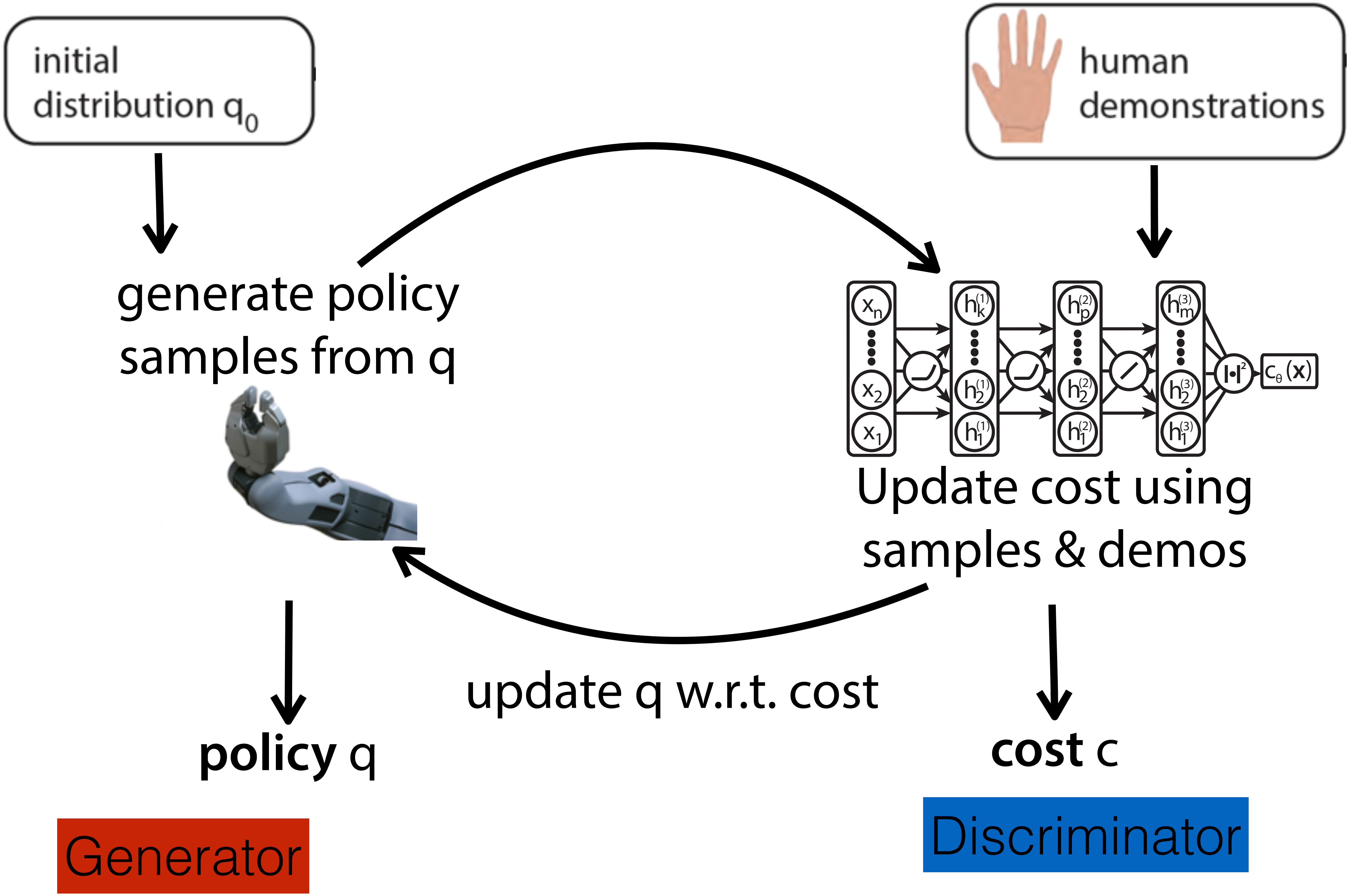
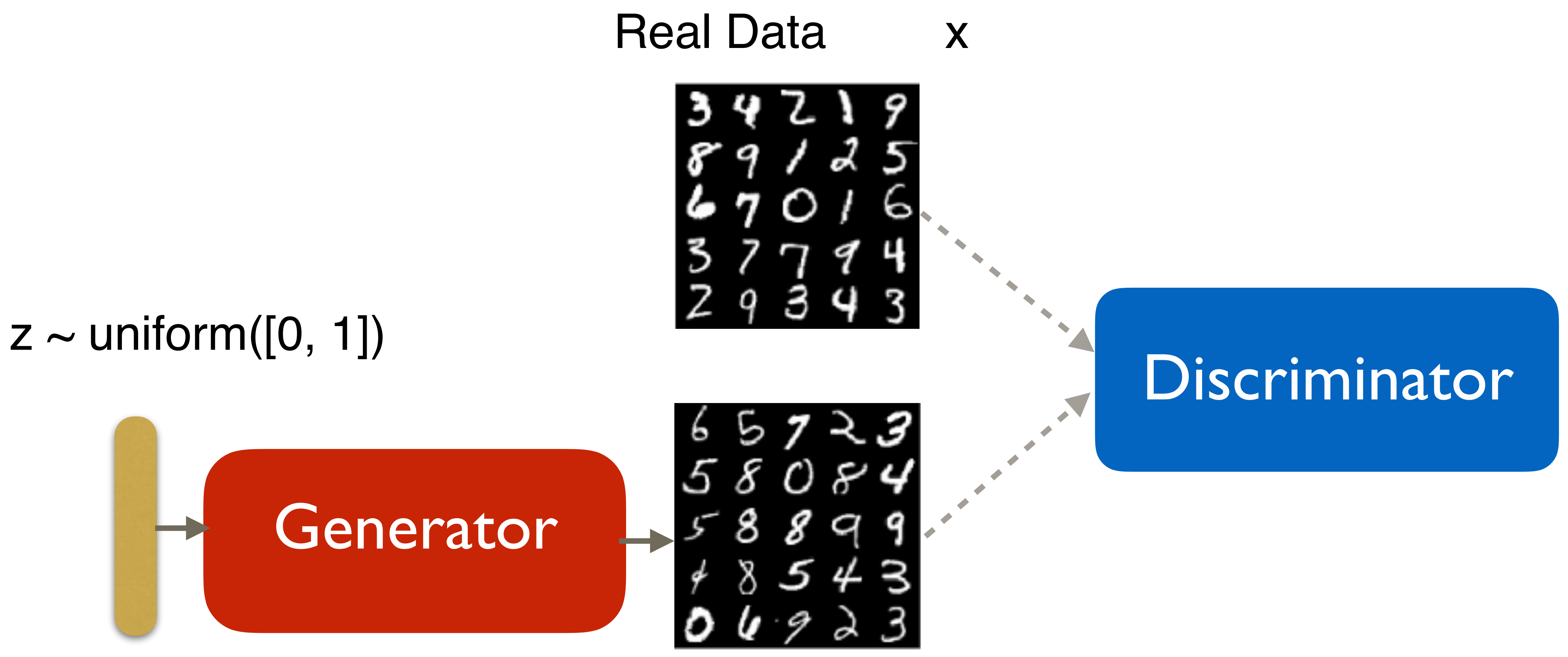


Diagram from Chelsea Finn

Generative Adversarial Networks

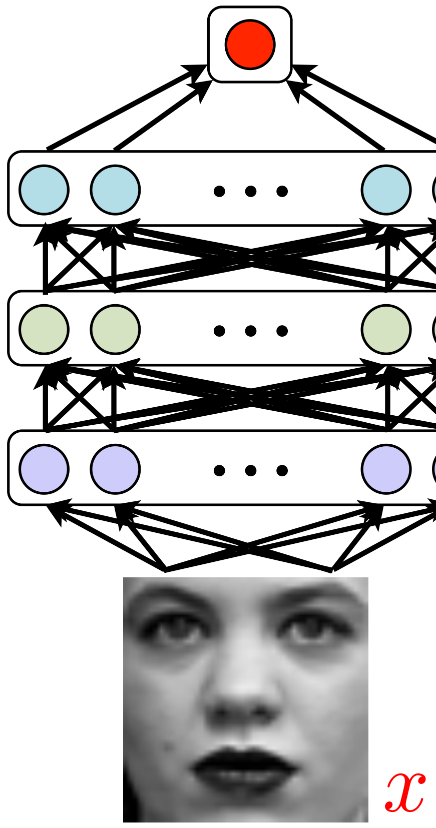
D(x): the probability that x came from the data rather than the generator

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



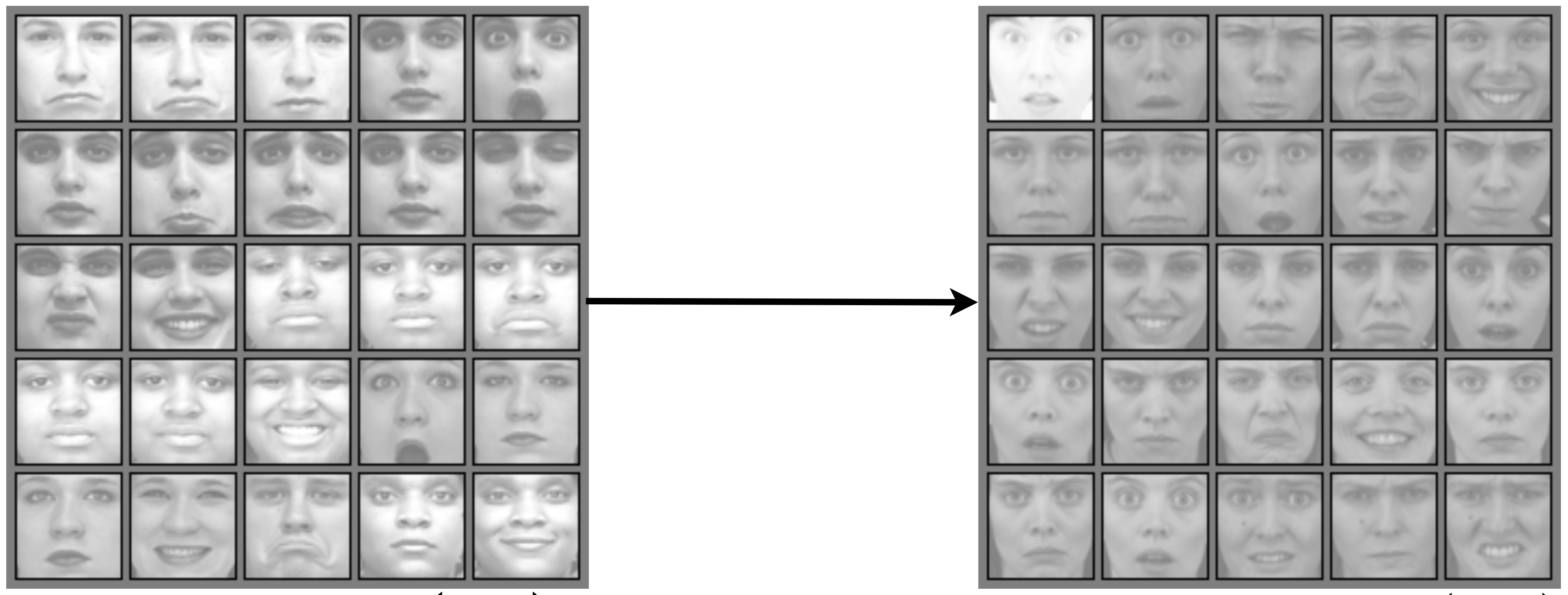
Discriminative Deep Learning

Recipe for success



Generative Modeling

- 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 $\mathbf{p}_{\text{model}} \approx \mathbf{p}_{\text{data}}$



$x \sim p_{\text{data}}(x)$ $x \sim p_{\text{model}}(x)$

Why generative models?

- Conditional generative models
    - Speech synthesis: Text->Speech
    - Machine Translation: French->English
        - French: Si mon tonton tond ton tonton, ton tonton sera tondu.
        - English: If my uncle shaves your uncle, your uncle will be shaved
    - Image->Image segmentation
- Environment simulator
    - Reinforcement learning
    - Planning
- Leverage unlabeled data

Maximum Likelihood: the dominant approach

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

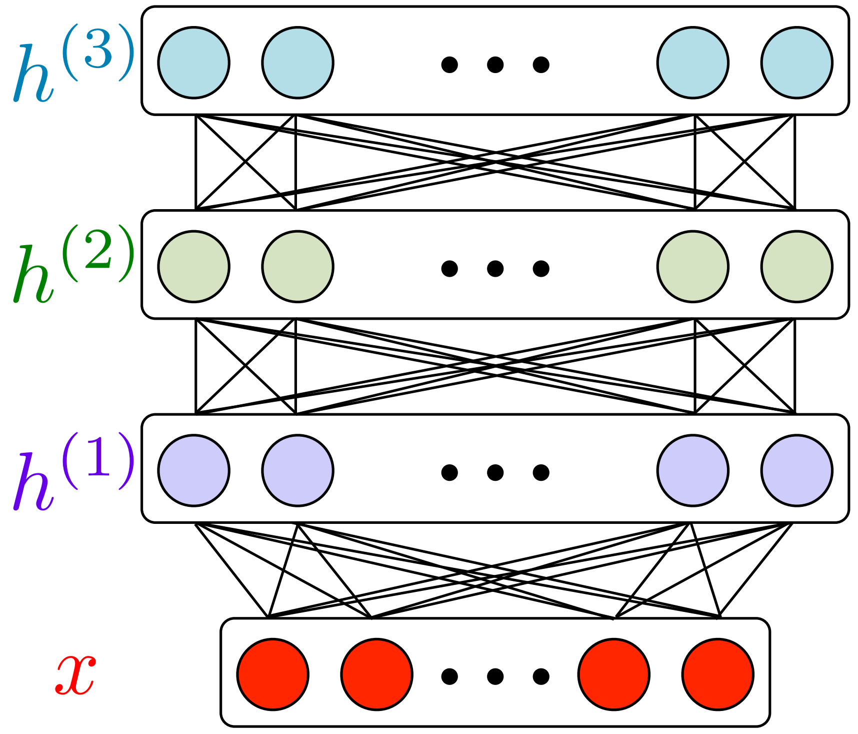
Undirected Graphical Models

- State-of-the-art general purpose undirected graphical model: **Deep Boltzmann machines**
- Several "hidden layers" h

$$p(h, x) = \frac{1}{Z}\tilde{p}(h, x)$$

$$\tilde{p}(h, x) = \exp(-E(h, x))$$

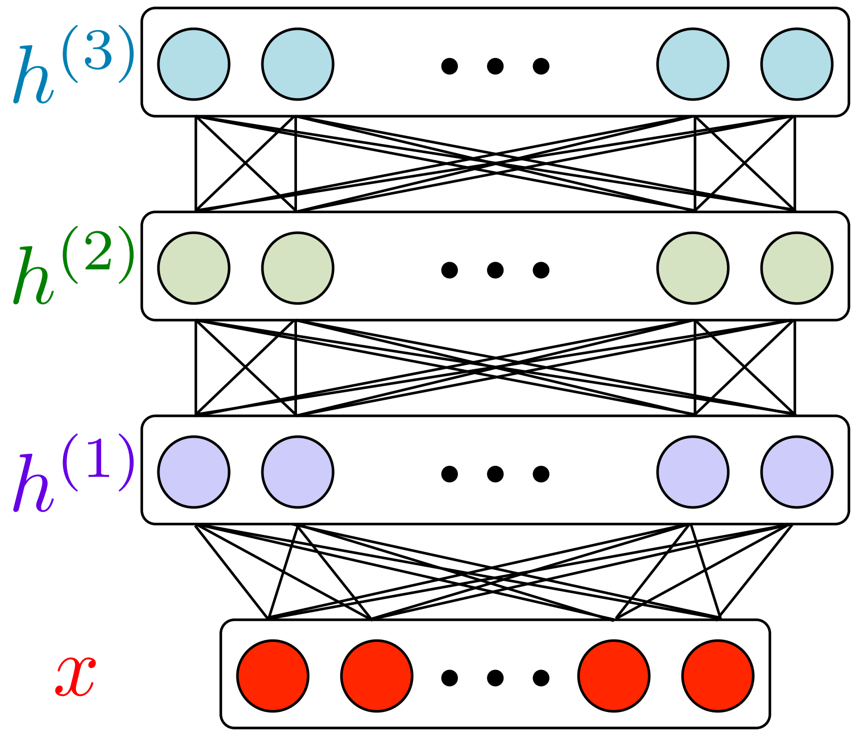
$$Z = \sum_{h,x} \tilde{p}(h, x)$$



Undirected Graphical Models: Disadvantage

- ML Learning requires that we draw samples

$$\frac{d}{d\theta_i} \log p(x) = \frac{d}{d\theta_i} \left[\log \sum_h \tilde{p}(h, x) - \log Z(\theta) \right]$$

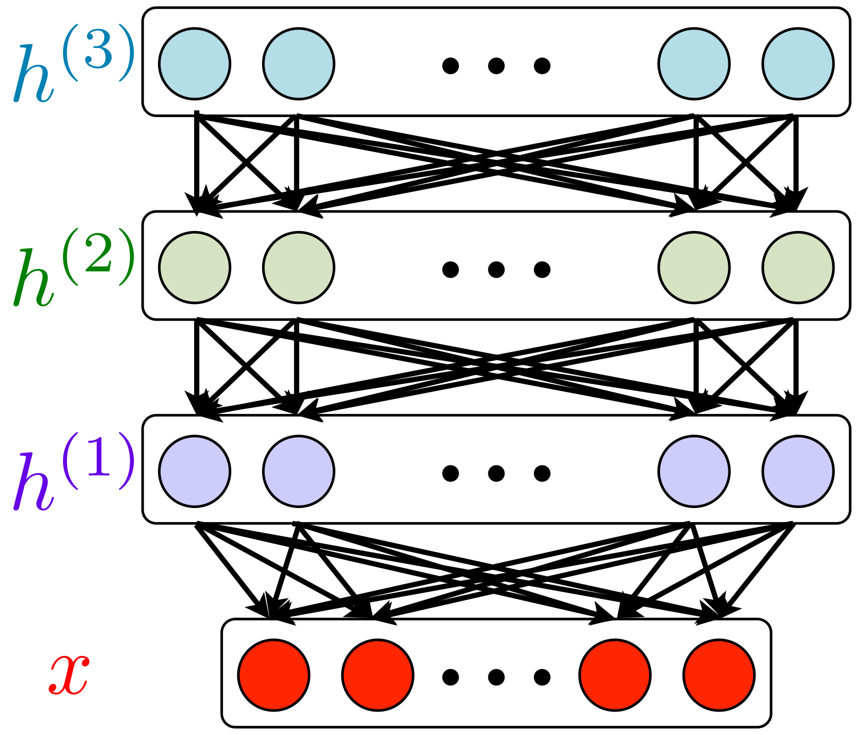


Directed graphical models

$$p(x, h) = p(x \mid h^{(1)})p(h^{(1)} \mid h^{(2)}) \dots p(h^{(L-1)} \mid h^{(L)})p(h^{(L)})$$

$$\frac{d}{d\theta_i} \log p(x) = \frac{1}{p(x)} \frac{d}{d\theta_i} p(x)$$

$$p(x) = \sum_h p(x \mid h)p(h)$$



- Two problems:
  1. Summation over exponentially many states in h
  2. Posterior inference, i.e. calculating $\mathbf{p}(\mathbf{h}|\mathbf{x})$, is intractable

Directed graphical models: new approaches

The Variational Autoencoder model:

- Kingma and Welling, *Auto-Encoding Variational Bayes*, International Conference on Learning Representations (ICLR) 2014
- Rezende, Mohamed and Wierstra, *Stochastic back-propagation and variational inference in deep latent Gaussian models*. ArXiv.

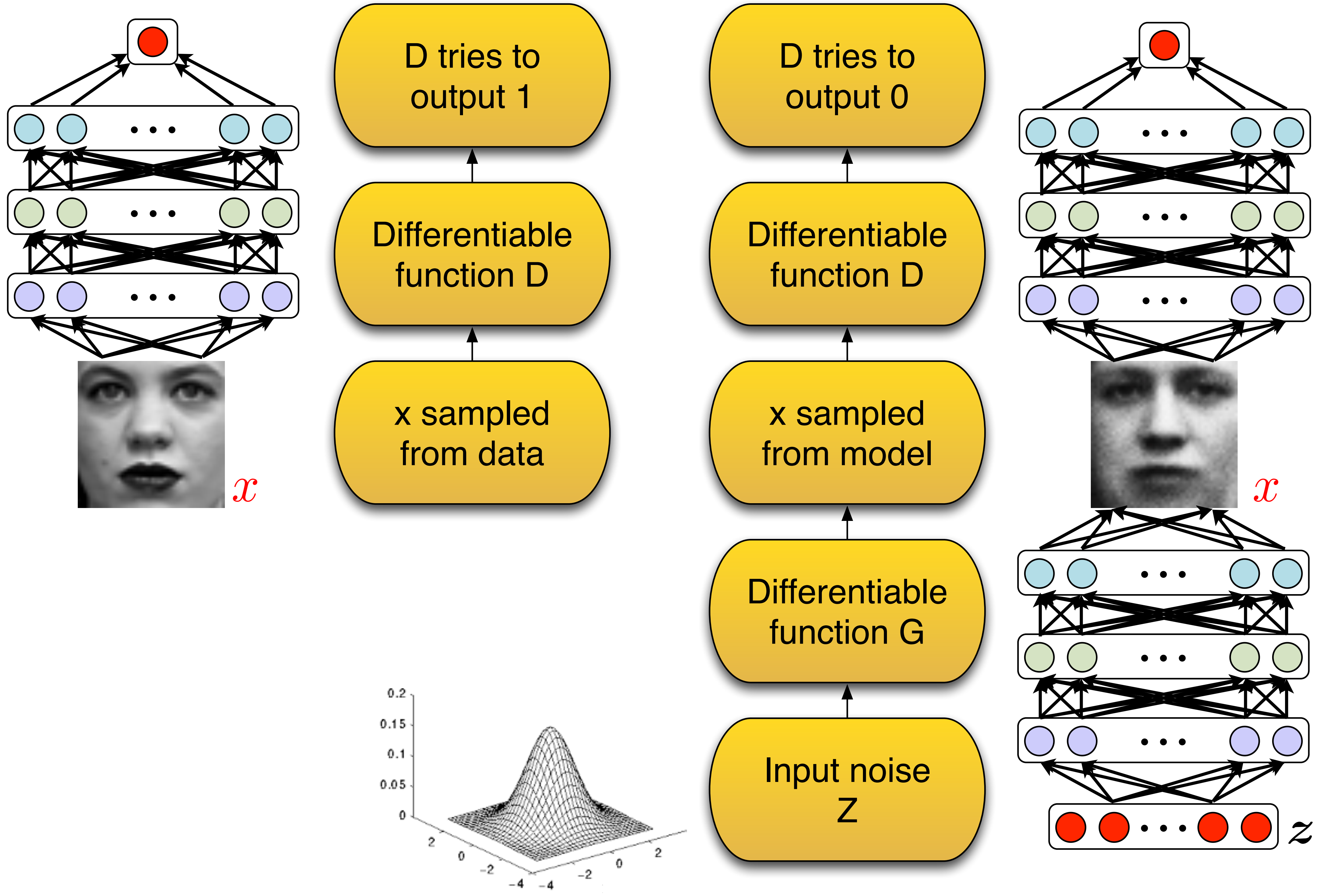
Use a reparametrization that allows them to train very efficiently with gradient backpropagation

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 had for D to distinguish from data

Generative Adversarial Networks



Zero-sum game

- Minimax objective function:

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

- In practice, to estimate G we use:

$$\max_{G} \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log D(G(\boldsymbol{z}))]$$

Why? Stronger gradient for G when D is very good.

Adapted from Ian Goodfellow

Learning process

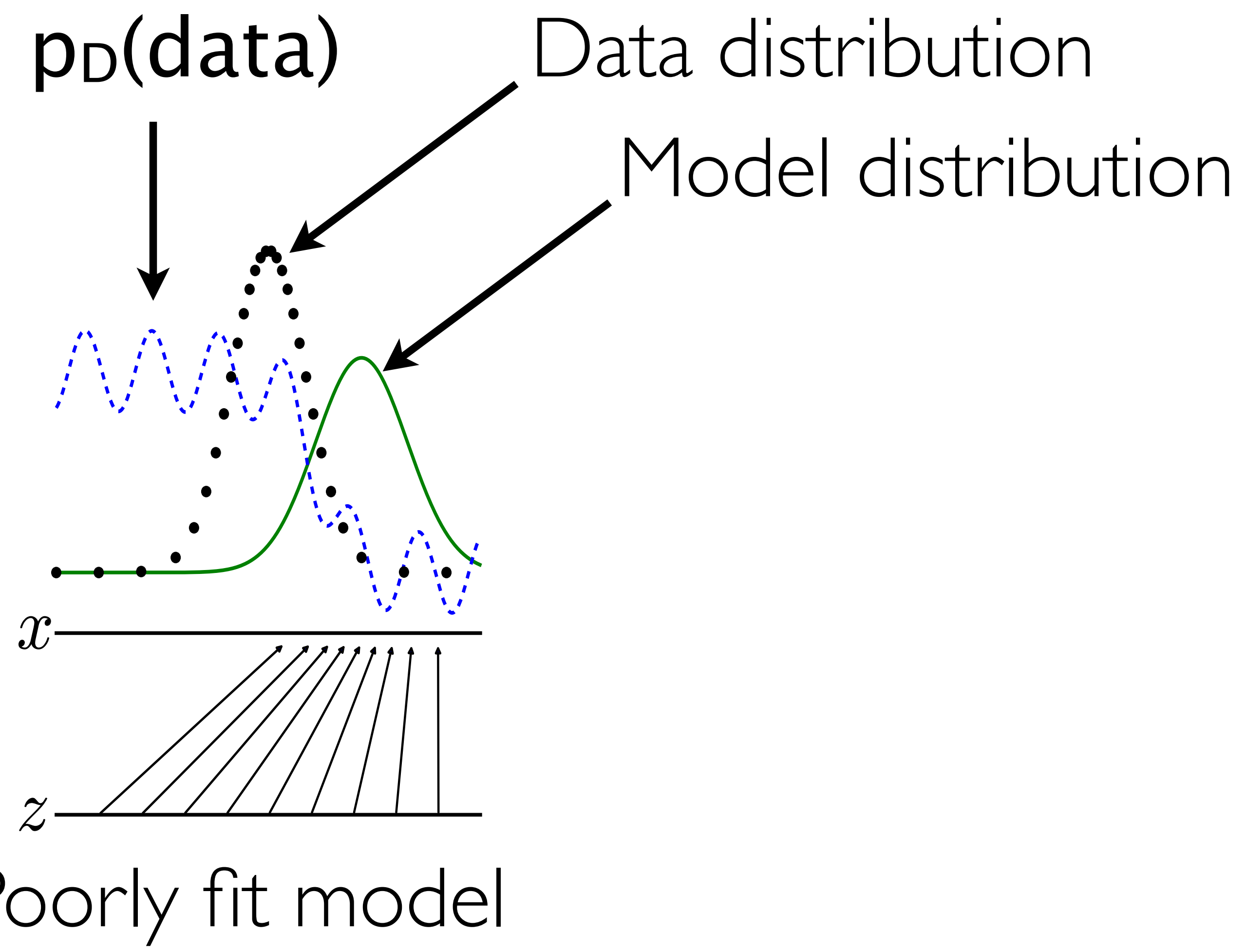
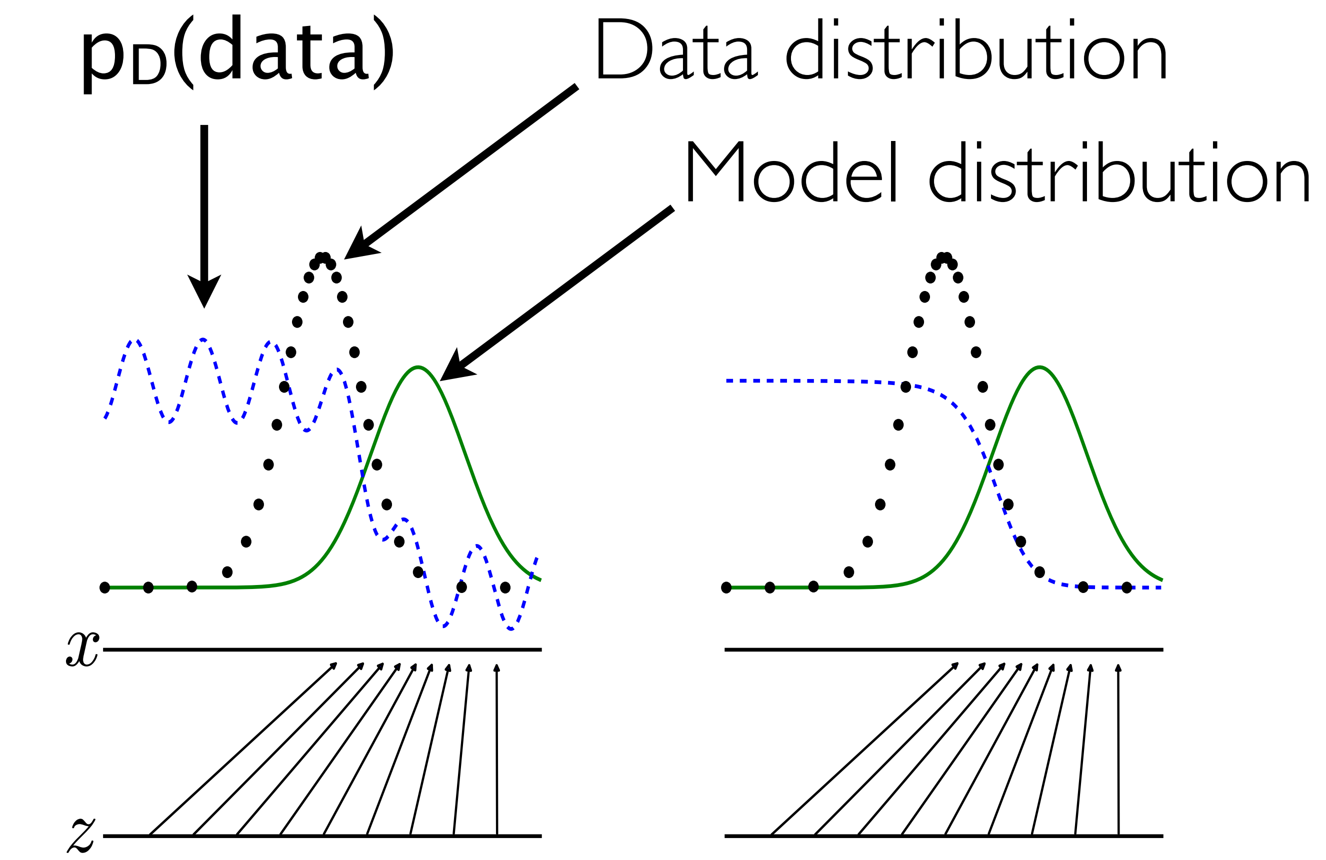


Diagram from Ian Goodfellow

Learning process



Poorly fit model   After updating D

Diagram from Ian Goodfellow

Learning process

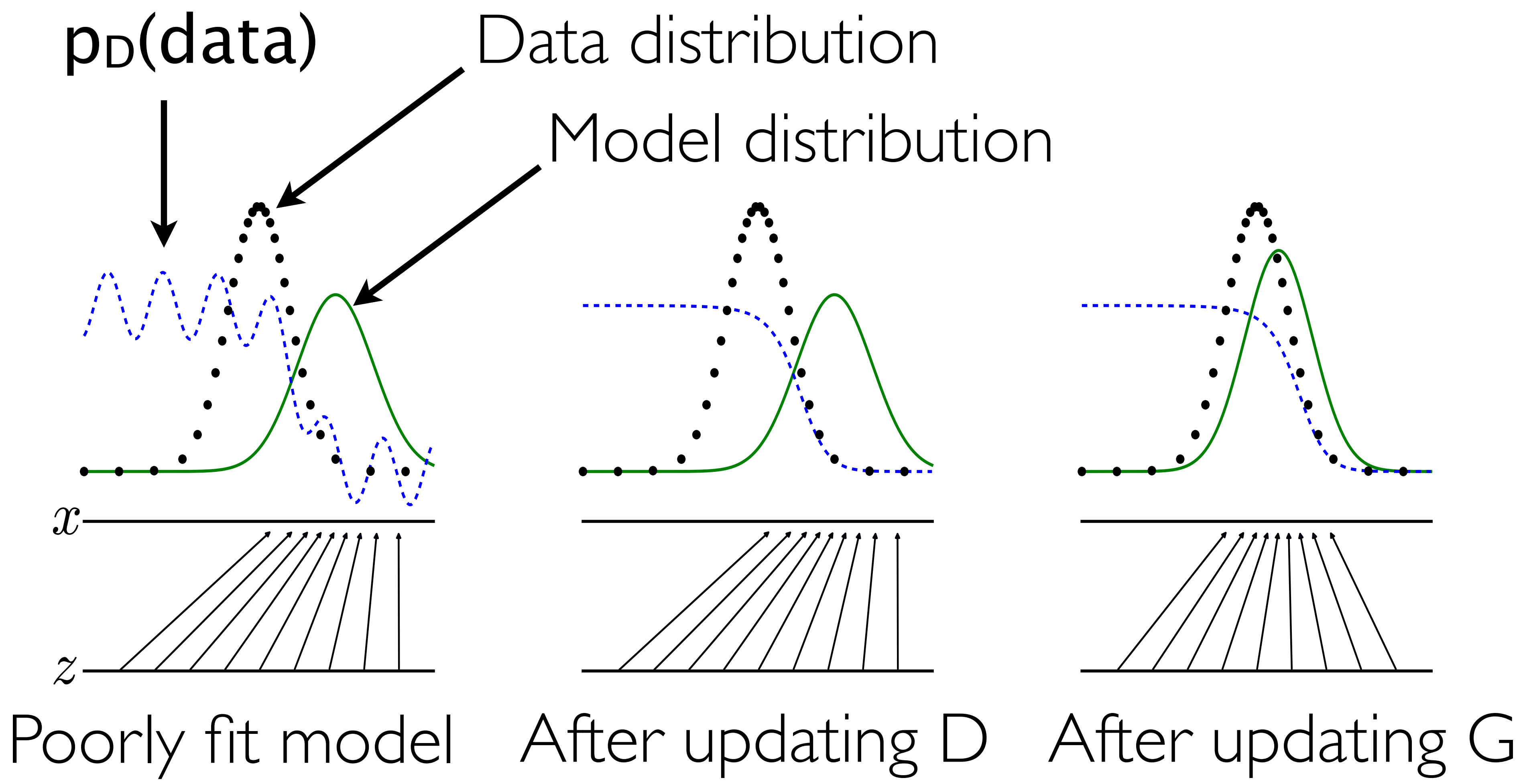


Diagram from Ian Goodfellow

Learning process

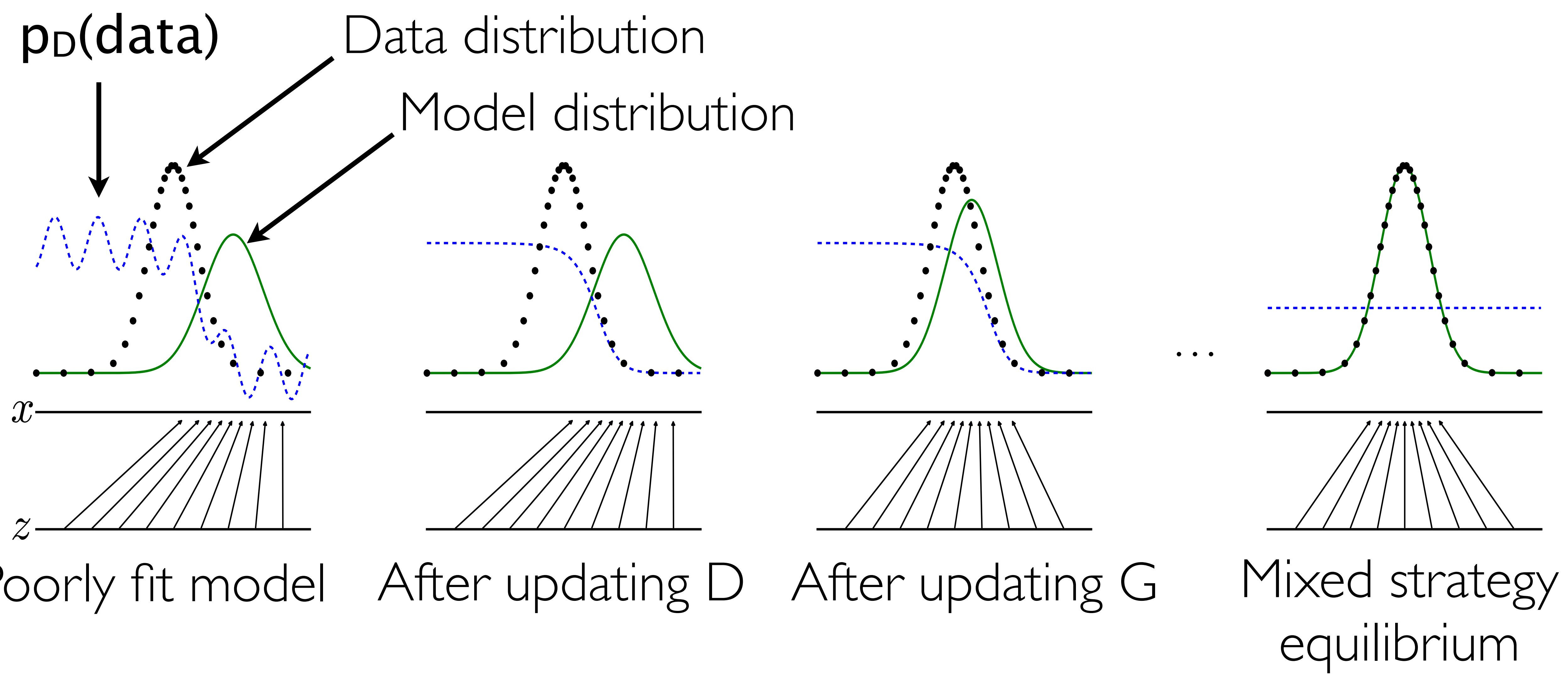
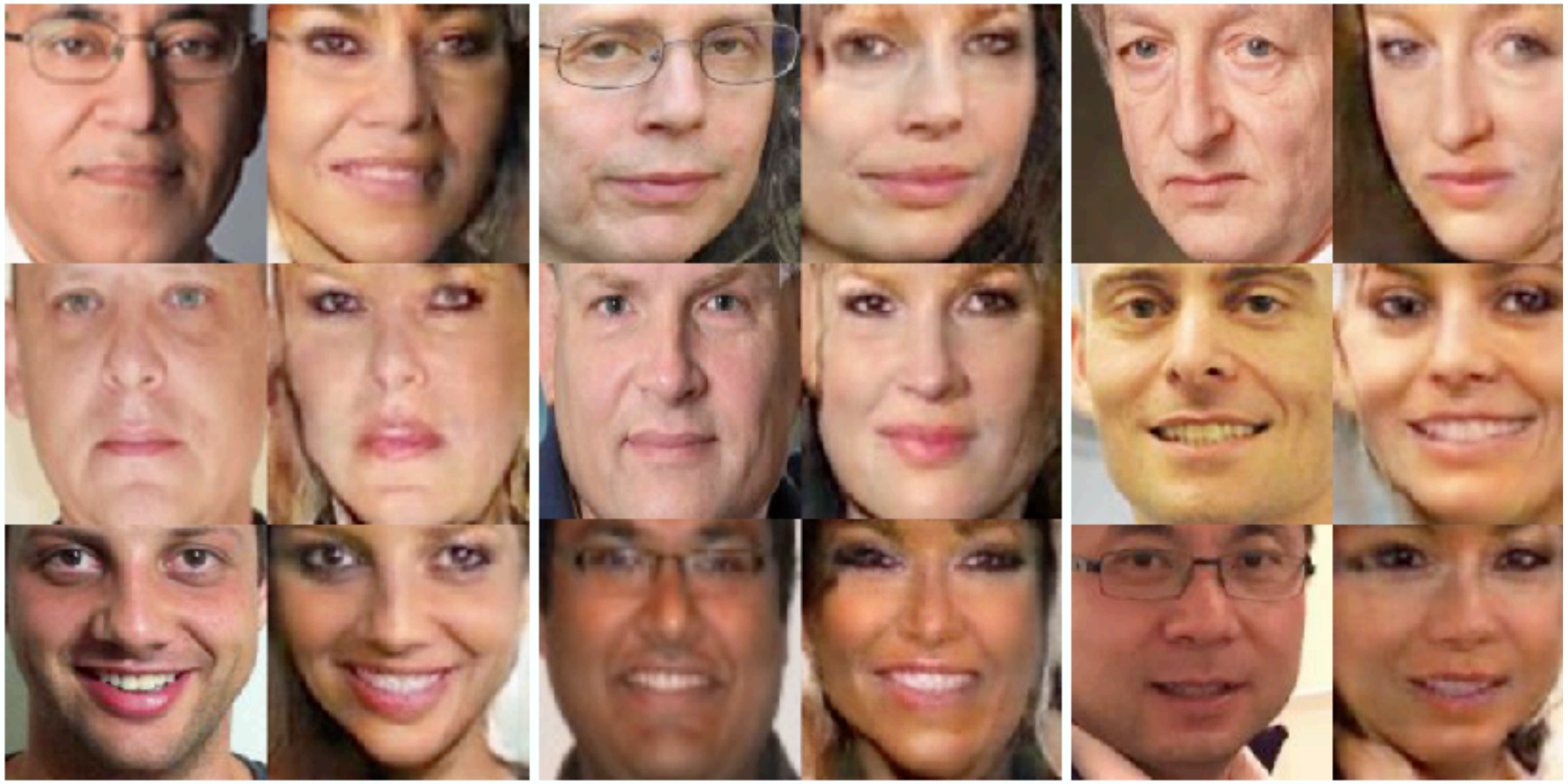


Diagram from Ian Goodfellow

GANs are both fun and useful

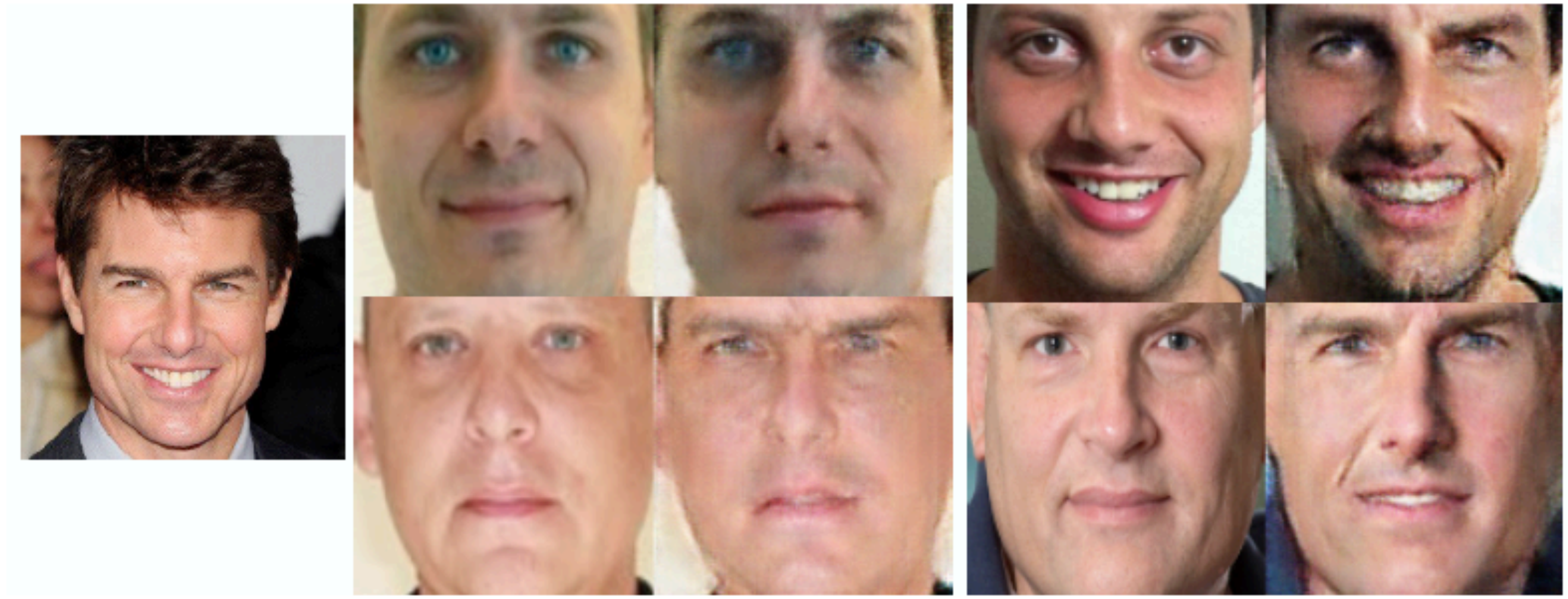
male -> female



Adversarial Inverse Graphics Networks, Tung et al. 2017

GANs are both fun and useful

anybody -> Tom Cruise



Adversarial Inverse Graphics Networks, Tung et al. 2017

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

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

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: logD(s)

Generative Adversarial Imitation Learning

Jonathan Ho
Stanford University
hoj@cs.stanford.edu

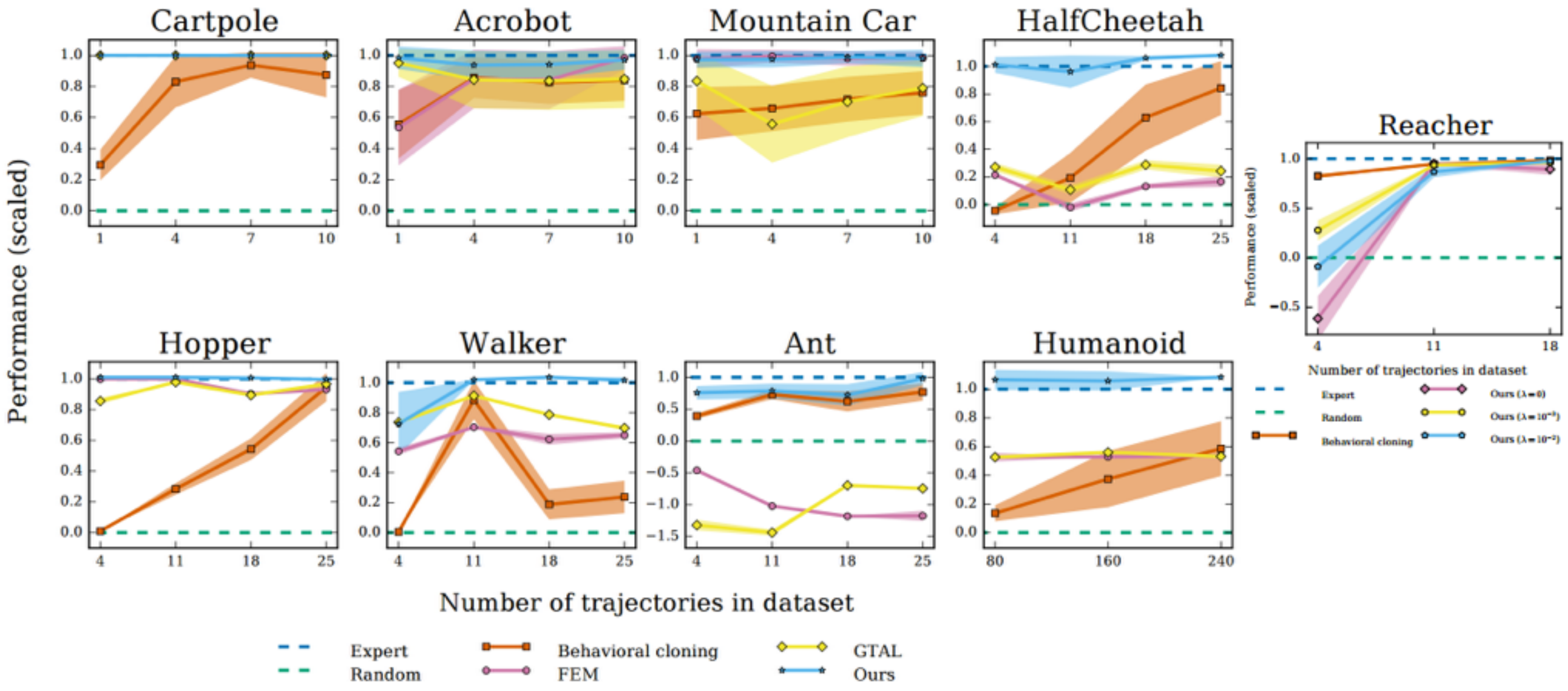
Stefano Ermon
Stanford University
ermon@cs.stanford.edu

NIPS 2016

Algorithm 1 Generative adversarial imitation learning

```
1: Input: Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters $\theta_0, w_0$
2: for $i = 0, 1, 2, \ldots$ 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
           $\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))]$   (17)
5:     Take a policy step from $\theta_i$ to $\theta_{i+1}$, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$.
       Specifically, take a KL-constrained natural gradient step with
           $\hat{\mathbb{E}}_{\tau_i}[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)] - \lambda \nabla_\theta H(\pi_\theta)$,
           where $Q(\bar{s}, \bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}]$   (18)
6: end for
```

Generative Adversarial Imitation learning


Cartpole
Acrobot
Mountain Car
HalfCheetah
Reacher
Hopper
Walker
Ant
Humanoid
Performance (scaled)
Number of trajectories in dataset
Expert
Random
Behavioral cloning
FEM
GTAL
Ours
Ours (λ=0)
Ours (λ=10⁻³)
Ours (λ=10⁻²)