CS 690: Human-Centric Machine Learning

Prof. Scott Niekum

Inverse reinforcement learning

How to learn from human data?

- **Behavioral Cloning**
  - Quadratic regret in worst case; bad performance out of expert distribution
  - Can’t learn from additional data collected by the agent
- **DAgger / TAMER / COACH**
  - Provides data/feedback on-policy
  - Still can’t learn from additional data collected by the agent
- **Inverse reinforcement learning**
  - Infers reward function from demonstrations so that RL can be used

Inverse reinforcement learning

MDP: (S, A, T, γ, D, R)

- S — states
- A — actions
- T — transition dynamics
- γ — discount rate
- D — start state distribution
- R — reward function

Policy: $\pi(s, a) \rightarrow [0, 1]$

Value function: $V^\pi(s_0) = \sum_{t=0}^{\infty} \gamma^t R(s_t)$

What if we have an **MDP/R**?

Inverse reinforcement learning

1. Collect user demonstration $(s_0, a_0), (s_1, a_1), \ldots, (s_n, a_n)$ and assume it is sampled from the expert's policy, π^E

2. Explain expert demos by finding R^* such that:

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

$$E_{s_0 \sim D}[V^{\pi^E}(s_0)] \quad \geq \quad E_{s_0 \sim D}[V^{\pi}(s_0)] \quad \forall \pi$$

How can search be made tractable?

Abbeel, Pieter, and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning." *Proceedings of the twenty-first international conference on Machine learning*. 2004.

Linear reward functions

Define R^* as a linear combination of features:

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

Then,

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

Thus, the expected value of a policy can be expressed as a weighted sum of the expected features $\mu(\pi)$

A simplified optimization problem

Originally - Explain expert demos by finding R^* such that:

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

Use expected features:

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

Restated - find w^* such that:

$$w^* \mu(\pi^E) \quad \geq \quad w^* \mu(\pi) \quad \forall \pi$$

Iterative reward search

Goal: Find w^* such that: $w^* \mu(\pi^E) \geq w^* \mu(\pi) \ \forall \pi$

1. Initialize π_0 to any policy

Iterate for i = 1, 2, … :

2. Find w^* s.t. expert maximally outperforms all previously examined policies $\pi_{0 \ldots i-1}$:

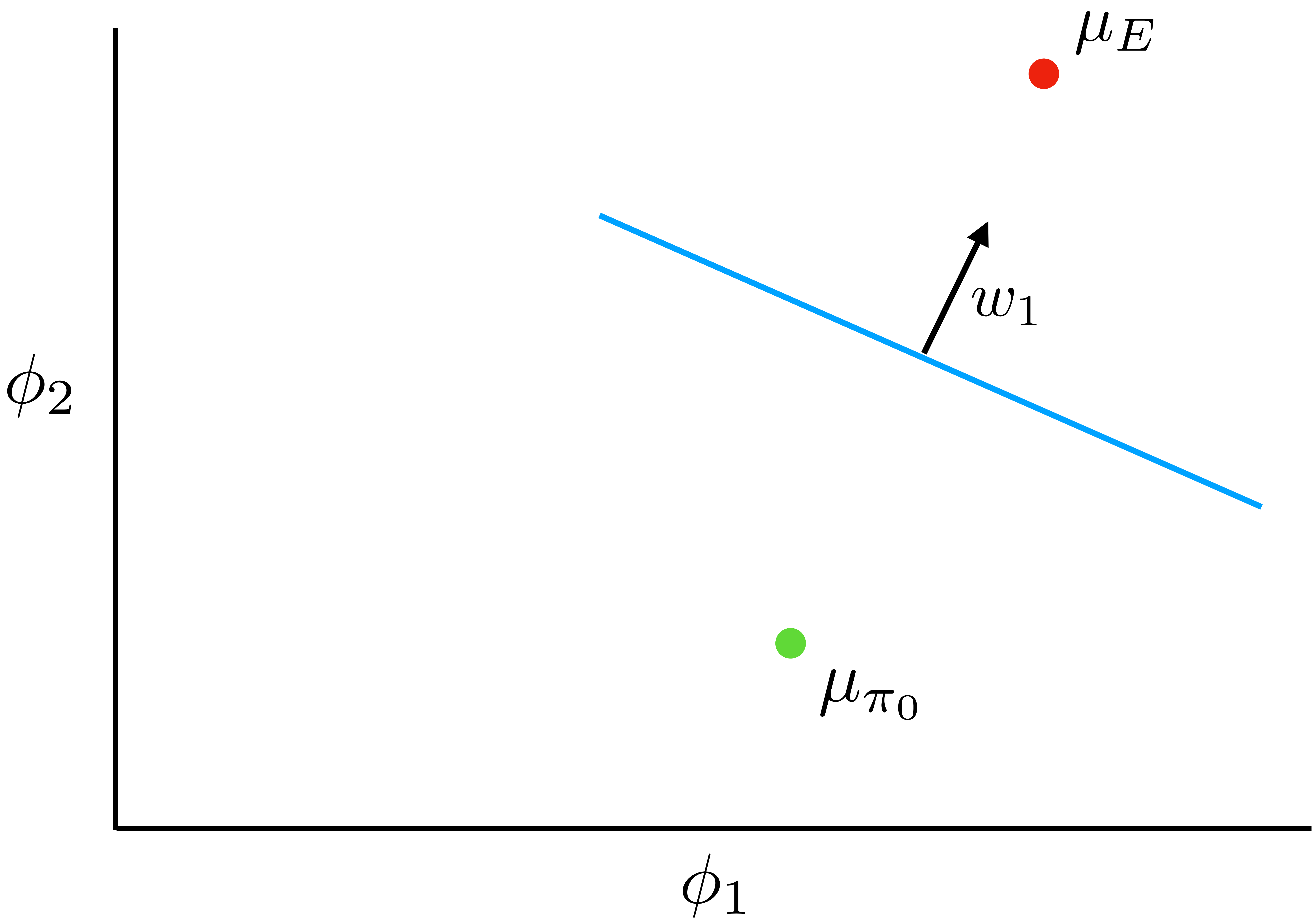
$$\max_{\epsilon, w^* : \|w^*\|_2 \leq 1} \epsilon \quad \text{s.t.} \quad w^* \mu(\pi^E) \geq w^* \mu(\pi_j) + \epsilon$$

SVM solver

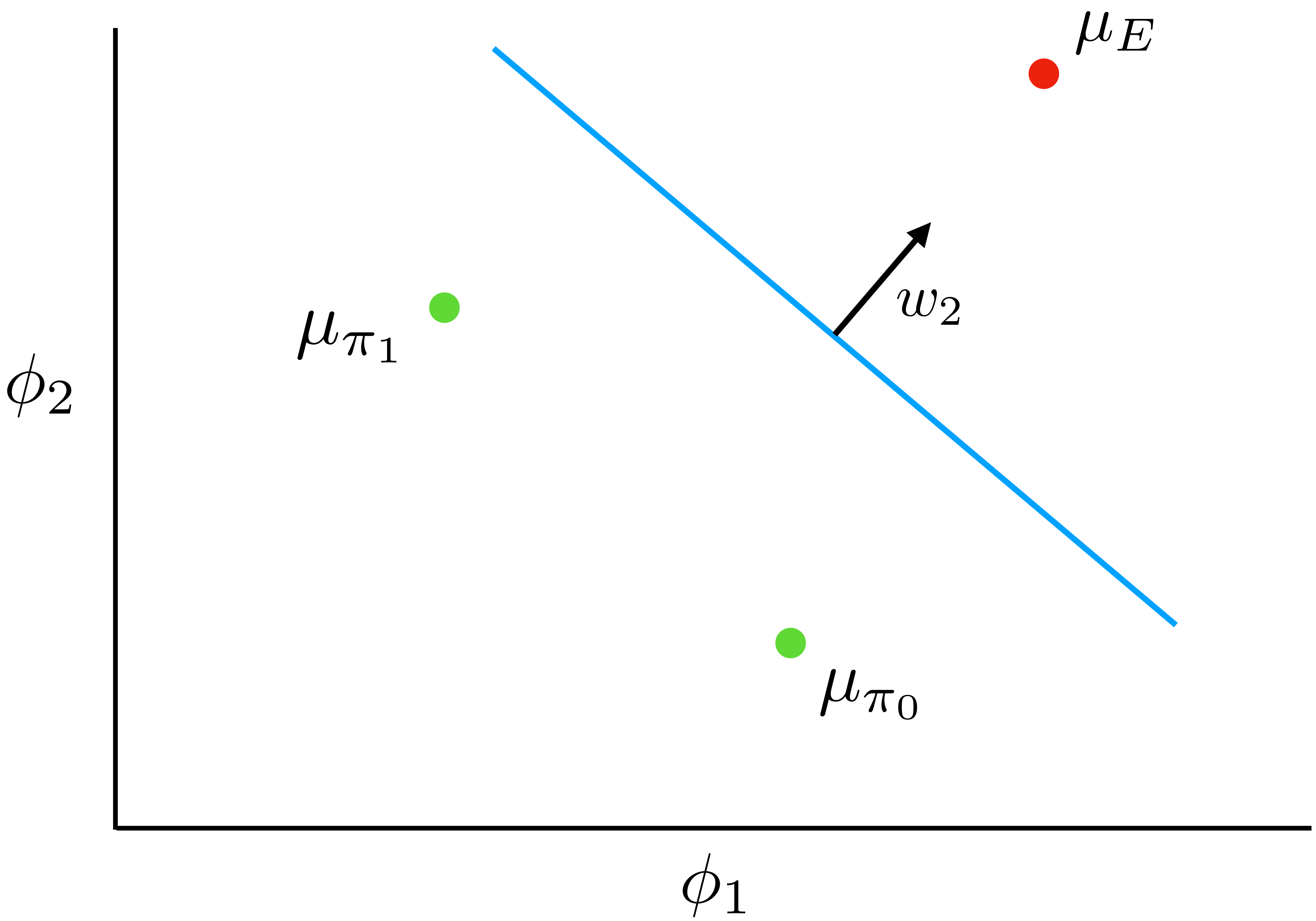
3. Use RL to calc. optimal policy π_i associated with w^*

4. Stop if $\epsilon \leq$ threshold

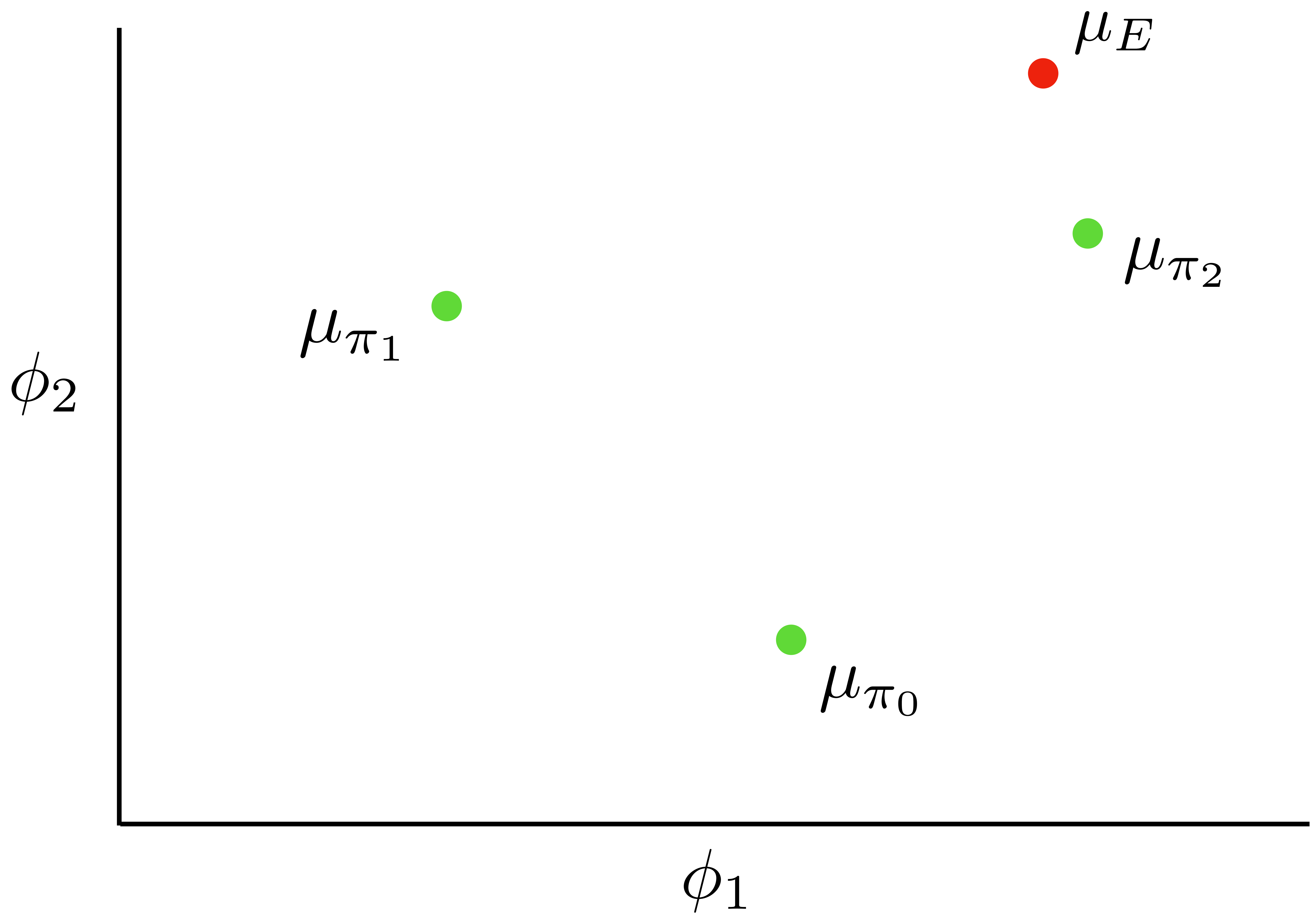
A (rough) illustration



A (rough) illustration



A (rough) illustration



Naive IRL challenges

- RL in the inner loop
- Where do linear features come from?
- Underspecified inference problem: infinite reward functions that explain behavior equally well. Which one to choose?
- Policies are underspecified too: many policies lead to the same expected features counts. Which one to choose?
- What if demonstrated behavior was actually suboptimal?

Suboptimality and policy mixtures

- If a demonstrator acts optimally, then there trivially is some reward function for which **at least one** optimal policy exists that matches the demonstrator's expected feature counts exactly

- But if the demonstrations are sometimes suboptimal, then there may be no single reward function with this property (aside from degenerate ones e.g. all zeros, under which everything is optimal)

- This can be thought of as the demonstrator sometimes acting optimally under a different reward function, so a mixture of reward functions (and their corresponding optimal policies) would be needed to match the feature counts

- Instead, we will now consider policies that are not strictly optimal, but produce trajectories in proportion to their return:

$$P(\zeta_i|\theta) = \frac{1}{Z(\theta)} e^{\theta^\top \mathbf{f}_{\zeta_i}}$$

Ziebart, Brian D., et al. "Maximum entropy inverse reinforcement learning." *AAAI*. Vol. 8. 2008.

Principle of maximum entropy

- **Definition:** the probability distribution which best represents the current state of knowledge about a system is the one with largest entropy, subject to your constraints.

- **Intuitively:** Don't overcommit in ways that aren't supported by the data — e.g. don't prefer one trajectory over another if they have the same return.

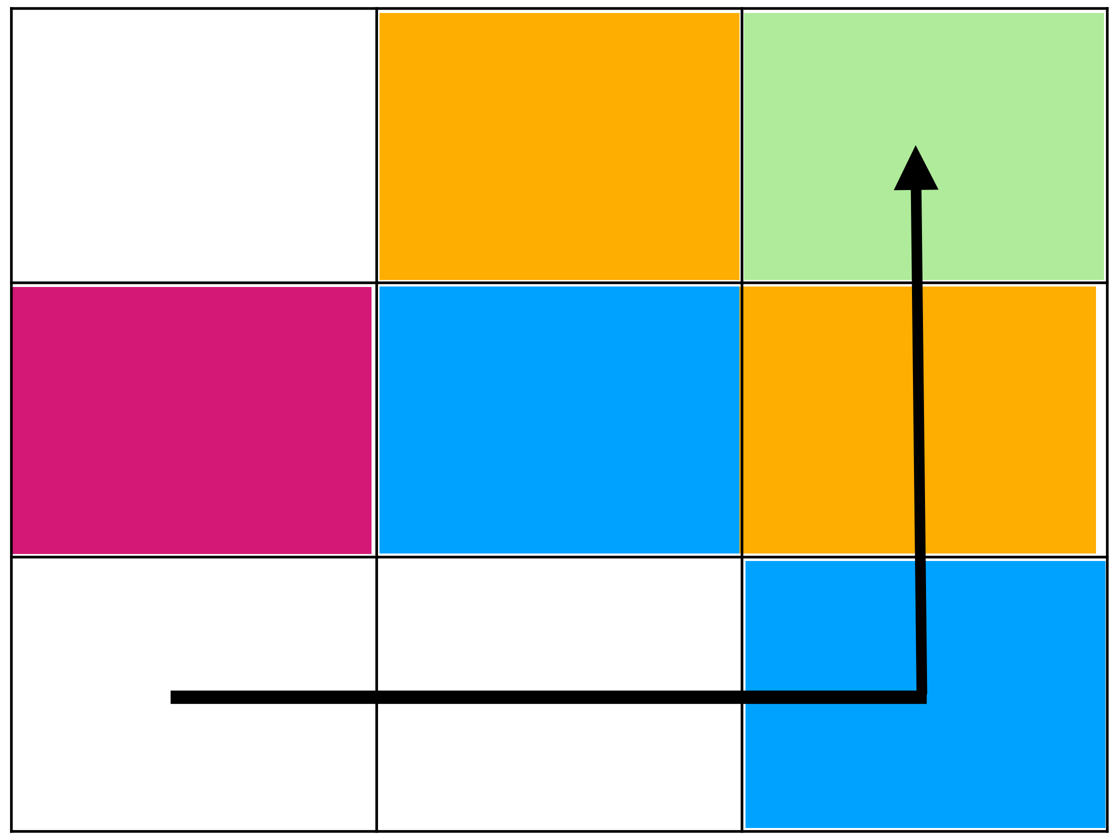
- **Practical consequence for IRL:** Tells us how to tiebreak between reward functions that explain the data equally well.

- **How?:** Find a reward function that matches expert feature counts under a specific trajectory distribution:

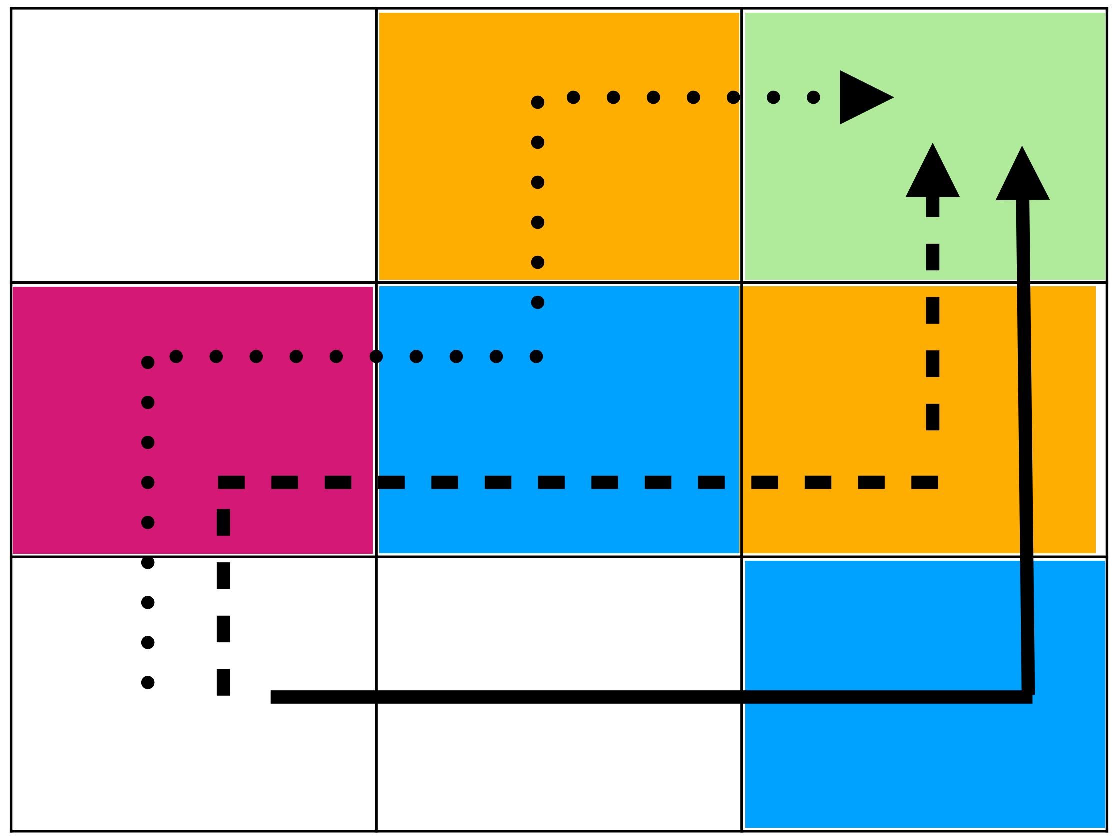
$$P(\zeta_i|\theta) = \frac{1}{Z(\theta)} e^{\theta^\top \mathbf{f}_{\zeta_i}}$$

- **Why this distribution?:** If you have specific expected feature counts **f** that you wish to match, it is known that the maximum entropy trajectory distribution that matches **f** is of the above form for some θ

Principle of maximum entropy



Vs.

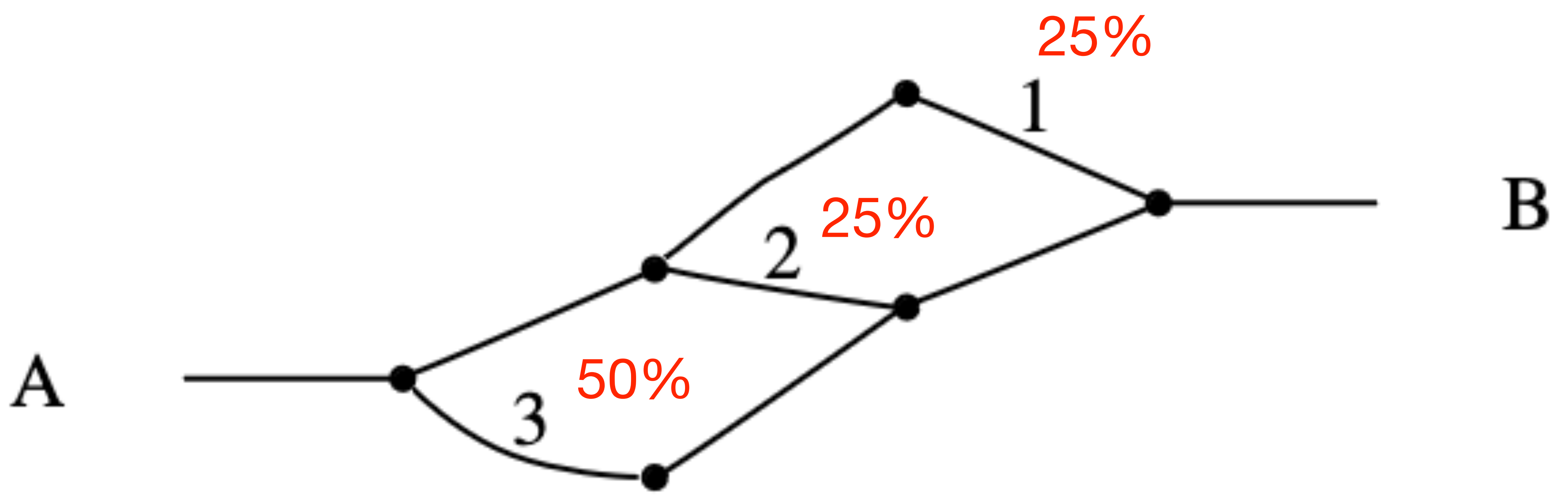


Trajectory vs. action-based reasoning

$$P(\text{action } a|\theta, T) \propto \sum_{\zeta: a \in \zeta_{t=0}} P(\zeta|\theta, T)$$

Vs.

$$P(\text{action } a|s_i, \theta) \propto e^{Q^*(s_i, a)}$$



Paths 1, 2, and 3 have equal return, so all should be p=1/3 under MaxEnt

Trajectory probabilities

Deterministic: $$P(\zeta_i|\theta) = \frac{1}{Z(\theta)} e^{\theta^\top \mathbf{f}_{\zeta_i}}$$

Stochastic: $$P(\zeta|\theta, T) \approx \frac{e^{\theta^\top \mathbf{f}_\zeta}}{Z(\theta, T)} \prod_{s_{t+1}, a_t, s_{t \in \zeta}} P_T(s_{t+1}|a_t, s_t)$$

Learning a reward function

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, L(\theta) = \underset{\theta}{\operatorname{argmax}} \sum_{\text{examples}} \log P(\tilde{\zeta}|\theta, T)$$

$$\nabla L(\theta) = \tilde{\mathbf{f}} - \sum_{\zeta} P(\zeta|\theta, T)\mathbf{f}_{\zeta} = \tilde{\mathbf{f}} - \sum_{s_i} \boxed{D_{s_i}}\mathbf{f}_{s_i}$$

How to compute?

Calculating state visitation frequencies

Algorithm 1 Expected Edge Frequency Calculation

Backward pass

1. Set $Z_{s_{\text{terminal}}} = 1$
2. Recursively compute for N iterations

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

Total unnormalized prob of all trajs that start at s_i and take action a_j
Total (weighted) exp return of all trajs that start at s_i and take action a_j

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

Total unnormalized prob of all trajs that start at s_i
Total (weighted) exp return of all trajs that start at s_i

Local action probability computation

3. $P(a_{i,j}|s_i) = \frac{Z_{a_{i,j}}}{Z_{s_i}}$ MaxEnt policy under reward function theta

Forward pass

4. Set $D_{s_i,t} = P(s_i = s_{\text{initial}})$ Typo! Should be $D_{s_i,1}$
5. Recursively compute for $t = 1$ to N

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

Prob of being in each state at each timestep t
Typo! Should be: $D_{s_i,t+1} = \sum_{a_{k,j}} \sum_k D_{s_k,t} P(a_{k,j}|s_k) P(s_i|a_{k,j}, s_k)$

Summing frequencies

6. $D_{s_i} = \sum_t D_{s_i,t}$ State visitation frequencies summed over all timesteps

Learning a reward function

$$\theta^* = \operatorname*{argmax}_\theta L(\theta) = \operatorname*{argmax}_\theta \sum_{\text{examples}} \log P(\tilde{\zeta}|\theta, T)$$

$$\nabla L(\theta) = \tilde{\mathbf{f}} - \sum_{\zeta} P(\zeta|\theta, T)\mathbf{f}_\zeta = \tilde{\mathbf{f}} - \sum_{s_i} \boxed{D_{s_i}} \mathbf{f}_{s_i}$$

How to compute?

Applications

[Ng et al, 2004]

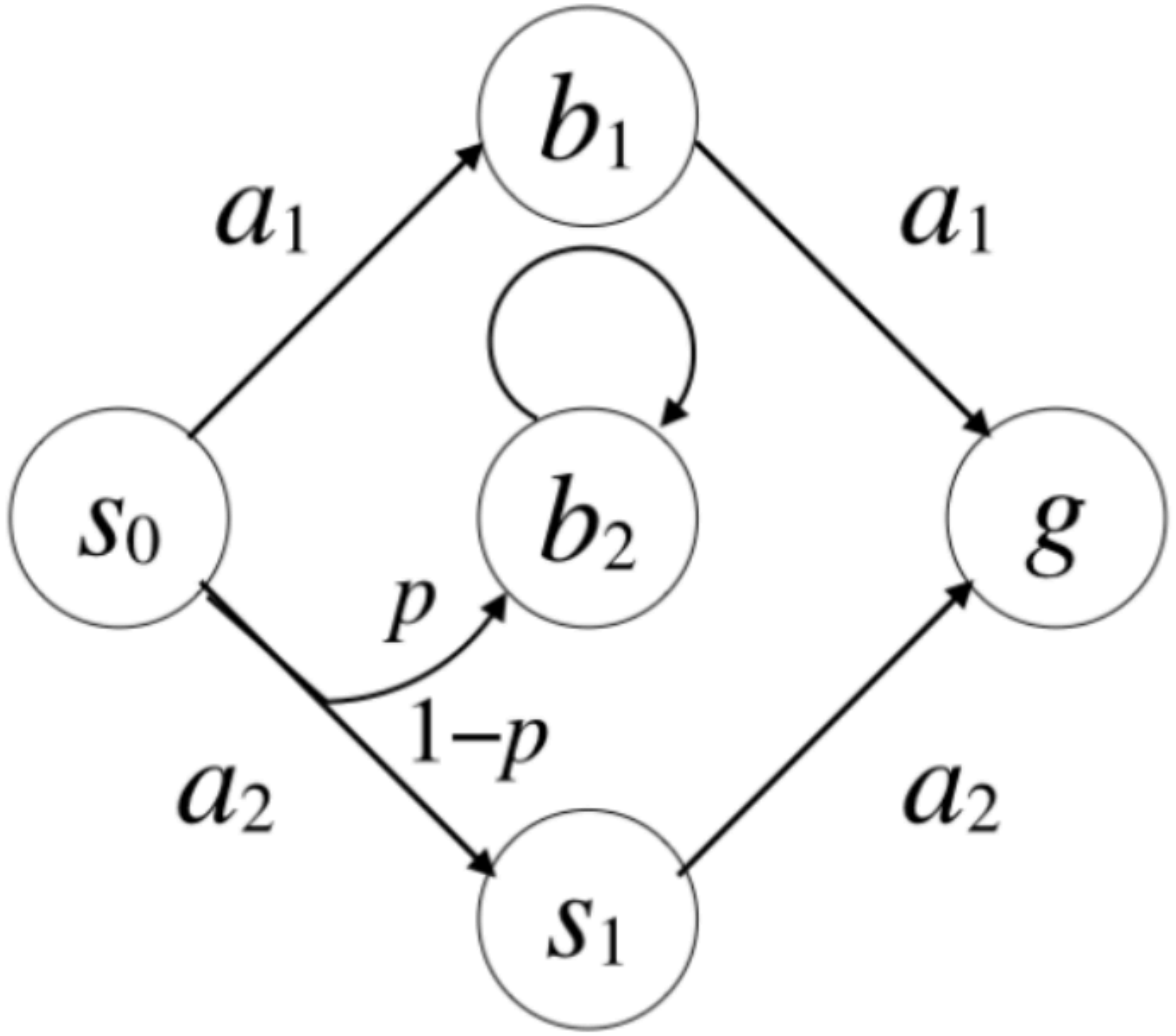
Applications

[Ng et al, 2004]

Applications

[Kolter, Abbeel & Ng, 2008]

Aside: Is reward enough?



States b1 and b2 are bad: reward of -r

Desired: maximize the probability of reaching g without hitting a bad state

Always better to take action a2 — seems trivial to specify

Assume gamma = 0.8, assign a value of r to meet specification

p = 0.1 possible, but p = 0.3 impossible!

Littman, Michael L., et al. "Environment-independent task specifications via GLTL." *arXiv preprint arXiv:1704.04341* (2017).