

# Inverse Optimal Control and Inverse Reinforcement Learning

D.A. Forsyth, UIUC

# Idea: Benefit from Expert

- We've done:
  - behavior cloning - take expert traces, use to train prediction of action
  - DAGGER - obtain further expert labels to avoid problems with correlation
- Now:
  - Rather than learn policy, learn reward function
    - Why:
      - reward function could be “smaller” than policy
    - Use:
      - apply greedy strategy to reward function

# Optimal control

- Choose control to ensure that trajectory gets best reward

$$\min_{x(\cdot), u(\cdot), T} \int_0^T \Phi(x(t), u(t)) dt \quad \leftarrow \text{Cost function}$$

State

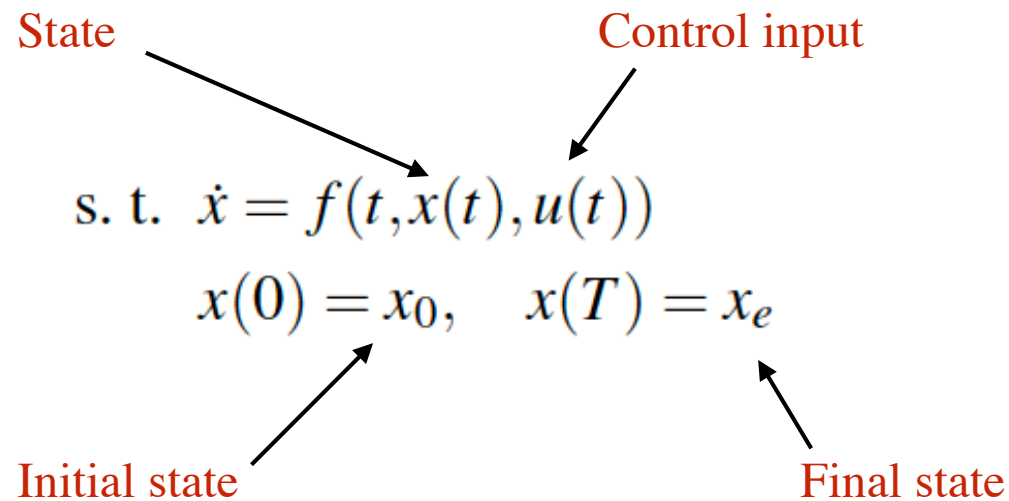
Control input

s. t.  $\dot{x} = f(t, x(t), u(t))$

$x(0) = x_0, \quad x(T) = x_e$

Initial state

Final state



# Optimal control

- Choose control to ensure that trajectory gets best reward
  - Well understood problem
  - Many cases, many solution processes
    - Easy
      - discrete time,  $f$  - linear,  $\Phi$  - quadratic = dynamic programming
      - Works for continuous time, too, but...
    - Many very much harder cases

# Optimal control as DP

- Dynamics:  $\mathbf{x}_{t+1} = \mathcal{A}_t \mathbf{x}_t + \mathcal{B}_t \mathbf{u}_t$

$$\text{Cost} = \sum_t C(\mathbf{x}_t, \mathbf{u}_t, t)$$

$$C(\mathbf{x}_t, \mathbf{u}_t, t) = \mathbf{x}_t^T \mathcal{M}_t \mathbf{x}_t + \mathbf{v}_t^T \mathbf{x}_t + \mathbf{u}_t^T \mathcal{P}_t \mathbf{u}_t + \mathbf{q}_t^T \mathbf{u}_t$$

$$\mathcal{P}_N = 0$$

$$\mathbf{q}_N = 0$$

# Optimal control as DP

- Assume we are at:  $\mathbf{x}_{N-1}$

- Cost is:

$$\mathbf{x}_N^T \mathcal{M}_N \mathbf{x}_N + \mathbf{v}_N^T \mathbf{x}_N + \mathbf{x}_{N-1}^T \mathcal{M}_{N-1} \mathbf{x}_{N-1} + \mathbf{v}_{N-1}^T \mathbf{x}_{N-1} + \mathbf{u}_{N-1}^T \mathcal{P}_{N-1} \mathbf{u}_{N-1} + \mathbf{q}_{N-1}^T \mathbf{u}_{N-1}$$

- BUT we'd choose the best control
  - AND  $\mathbf{x}_N$  is a function of  $\mathbf{u}_{N-1}$

# Optimal control as DP

- Assume we are at:  $\mathbf{x}_{N-1}$

- Cost is:

$$\mathbf{x}_N^T \mathcal{M}_N \mathbf{x}_N + \mathbf{v}_N^T \mathbf{x}_N + \mathbf{x}_{N-1}^T \mathcal{M}_{N-1} \mathbf{x}_{N-1} + \mathbf{v}_{N-1}^T \mathbf{x}_{N-1} + \mathbf{u}_{N-1}^T \mathcal{P}_{N-1} \mathbf{u}_{N-1} + \mathbf{q}_{N-1}^T \mathbf{u}_{N-1}$$

- BUT we'd choose the best control
  - AND  $\mathbf{x}_N$  is a function of  $\mathbf{u}_{N-1}$
- Substitute

$$\mathbf{x}_N = \mathcal{A}_{N-1} \mathbf{x}_{N-1} + \mathcal{B}_{N-1} \mathbf{u}_{N-1}$$

# Optimal control as DP

- Substituting, taking gradient, etc. gives

$$(2\mathcal{P}_{N-1} + 2\mathcal{B}_N^T \mathcal{M}_N \mathcal{B}_N) \mathbf{u}_{N-1} + \mathbf{q}_N + 2\mathcal{B}_{N-1}^T \mathcal{M}_N \mathcal{A}_{N-1} \mathbf{x}_{N-1} + \mathcal{B}_{N-1}^T \mathbf{v}_{N-1} = \mathbf{0}$$

Check this!

- Which we can solve for  $\mathbf{u}_{N-1}$  as a function of  $\mathbf{x}_{N-1}$ 
  - which is linear
- Now we have a quadratic cost function for  $\mathbf{x}_{N-1}$ 
  - choose best  $\mathbf{u}_{N-1}$  and substitute
- and we can use induction!



# Inverse Optimal Control

- Given a trajectory  $(x_0^*, x_1^*, x_2^*, \dots, x_T^*)$ 
  - at known time intervals
- Find the Phi that caused that trajectory
  - assuming that  $f$  is known and that there is no noise
- Crucial step:
  - assume a representation of Phi

$$\int \Phi(x(t), u(t), t) dt = \sum_i \alpha_i \int \phi_i(x(t), u(t), t) dt$$

Unknown Known

# Inverse Optimal Control

- Obtain  $\alpha$  and  $u(t)$  by solving:

$$\min_{\alpha} \sum_{j=1}^m \|z^*(t_j; \alpha) - z_M(t_j)\|^2$$

where  $z^*(t; \alpha)$  is the solution of

$$\min_{x, u, T} \int_0^T \left[ \sum_{i=1}^n \alpha_i \phi_i(x(t), u(t)) \right] dt$$

$$\text{s. t. } \dot{x} = f(t, x(t), u(t))$$

$$x(0) = x_0, \quad x(T) = x_e$$

# NASTY!

$$\min_{\alpha} \sum_{j=1}^m \|z^*(t_j; \alpha) - z_M(t_j)\|^2$$

Cost(alpha)

where  $z^*(t; \alpha)$  is the solution of

$$\min_{x, u, T} \int_0^T \left[ \sum_{i=1}^n \alpha_i \phi_i(x(t), u(t)) \right] dt$$

Optimal control  
for fixed alpha

$$\text{s. t. } \dot{x} = f(t, x(t), u(t))$$

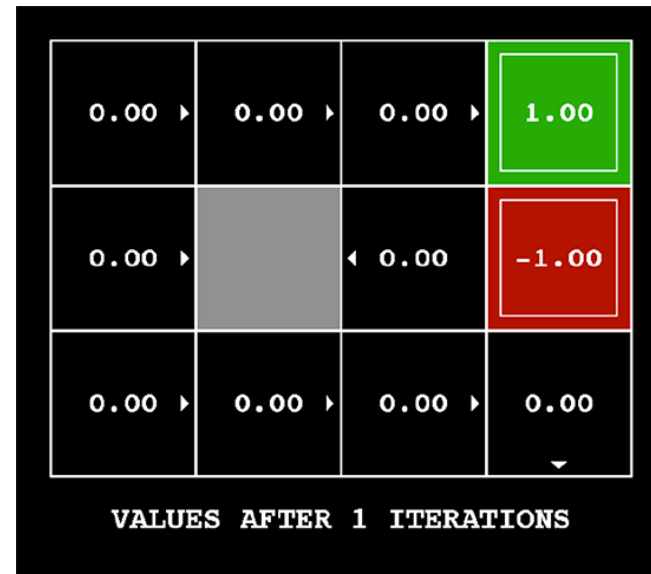
$$x(0) = x_0, \quad x(T) = x_e$$

But do-able - see:

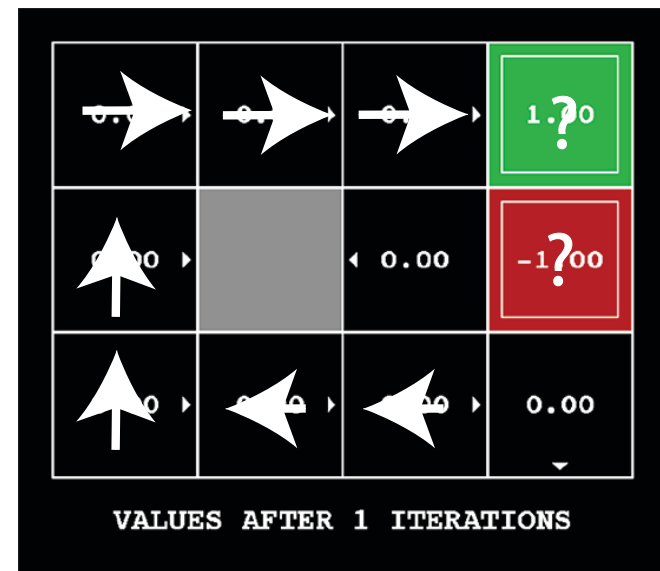
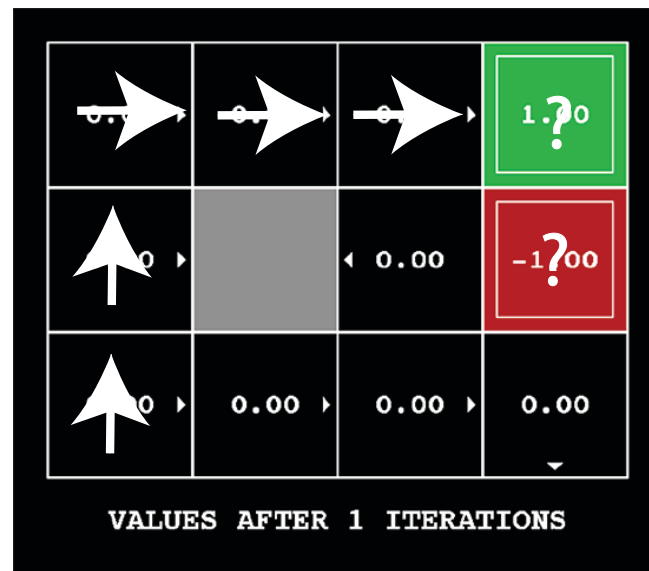
Mombauer, Laumond, Truong

# Example case: learning to search a map

- Recall gridworld ->



- We see:



0.00 ▶	0.00 ▶	0.00 ▶	1.00 ?
0.00 ▶		◀ 0.00	-1.00 ?
0.00 ▶	0.00 ▶	0.00 ▶	0.00 ▼

VALUES AFTER 1 ITERATIONS

0.00 ▶	0.00 ▶	0.00 ▶	1.00 ?
0.00 ▶		◀ 0.00	-1.00 ?
0.00 ▶	0.00 ▶	0.00 ▶	0.00 ▼

VALUES AFTER 1 ITERATIONS

# Learning to search

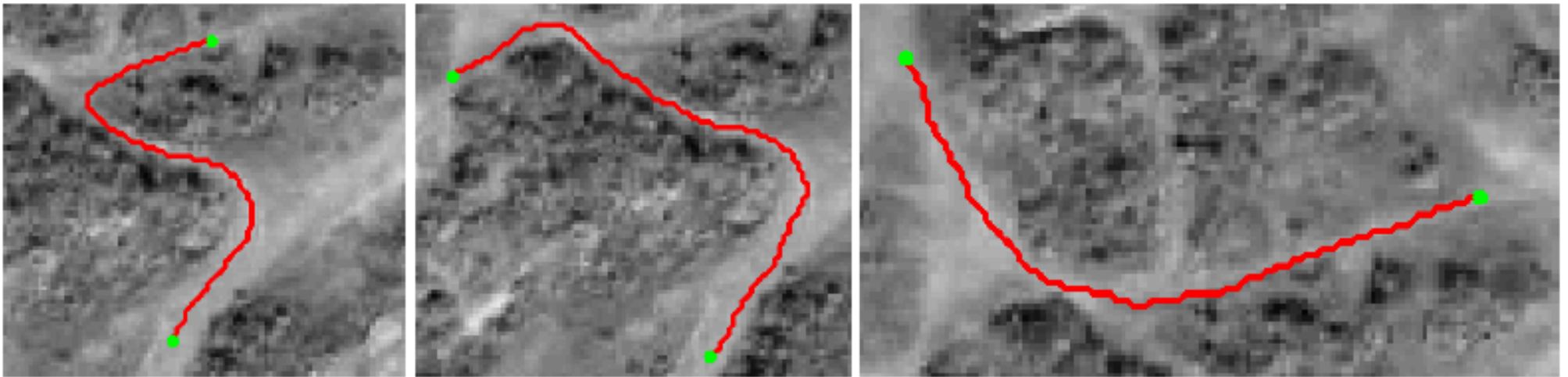
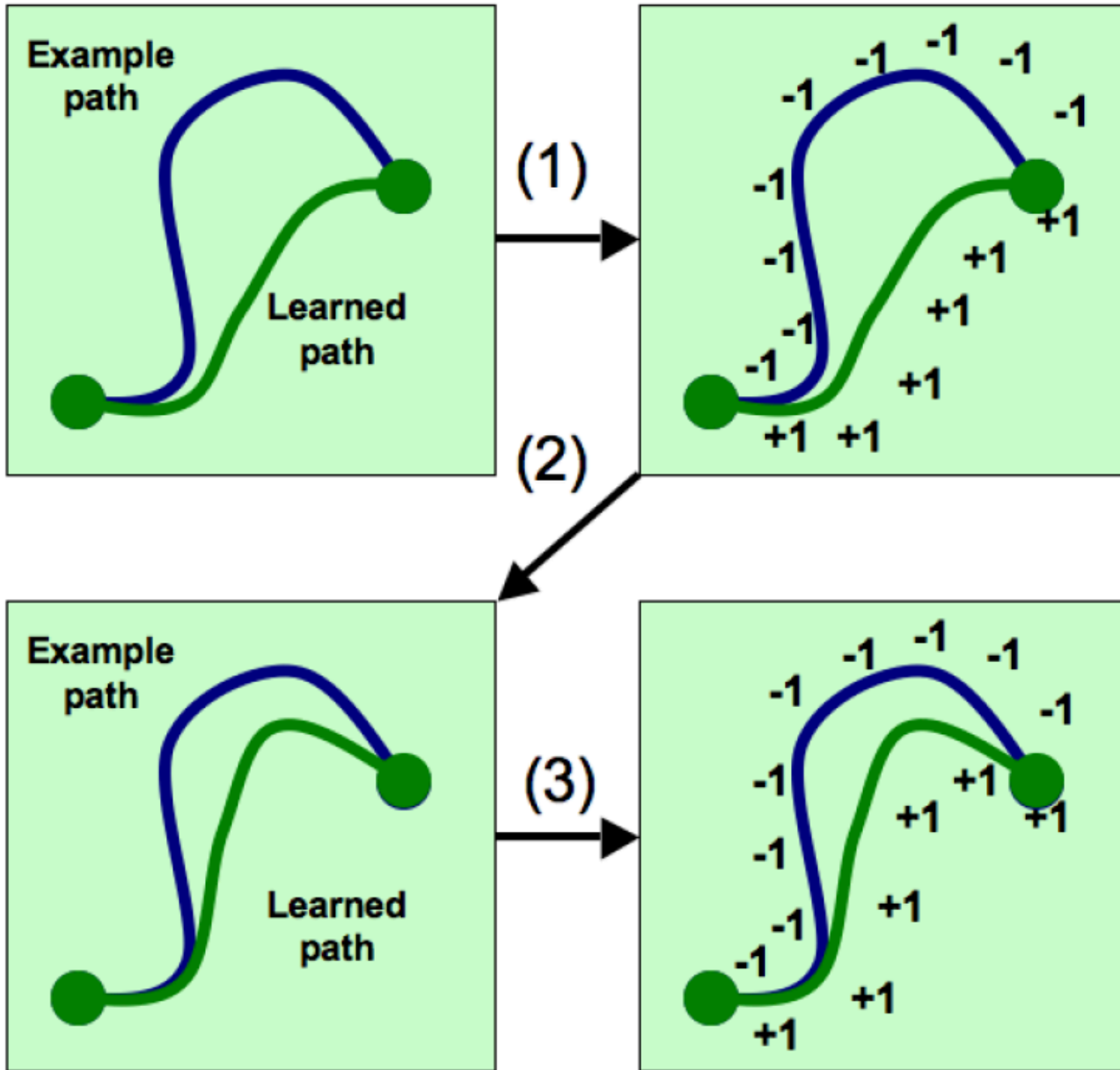


Figure 2: This figure demonstrates the flavor of the training data considered here in the context of imitation learning. In this case, a human expert specified by hand examples of the path (red) that a mobile robot should take between pairs of end points (green) through overhead satellite images. These example paths demonstrate the form of training data used for the outdoor navigational planning experiment discussed in section 6.





# Inverse Reinforcement Learning

- We now observe an agent acting in an environment

- see trace or traces

$$(s_0, a_0, s_1, a_1, \dots)$$

- Q: Why does it do that? = What is the reward function?

- Model

feature functions, known or chosen

$$R(s) = \sum_i w_i \phi_i(s) = \mathbf{w}^T \phi(s)$$

Reward for state

unknown weights

# Assume

- We observe an agent acting optimally
  - we'll fix this
- The true reward function can be represented by model
  - we'll fix this, too

# We have

Total reward for acting  
using  $\pi$

$$\downarrow \\ G(\pi) =$$

=

=

Expected value of all rewards collected  
by applying  $\pi$

$$\downarrow \\ \mathbf{E} \left[ \sum_t \gamma^t R(s_t) \mid \pi \right]$$

$$\sum_i w_i \mathbf{E} \left[ \sum_t \gamma^t \phi_i(s_t) \mid \pi \right]$$

$$\sum_i w_i \mu_i^\pi$$

↑  
We could measure these;  
act using  $\pi$ , and see  
what states we see

# Formulate

$$G(\pi^*) \geq G(\pi) \quad \forall \pi \longleftarrow \text{The expert's policy yields a better reward than any other}$$

$$\mathbf{w}^T \mu^{\pi^*} \geq \mathbf{w}^T \mu^{\pi} \quad \forall \pi \longleftarrow \text{which constrains the coefficients}$$

# Not good enough...

$G(\pi^*) \geq G(\pi) \quad \forall \pi$  ——— The expert's policy yields a better reward than any other

$\mathbf{w}^T \mu^{\pi^*} \geq \mathbf{w}^T \mu^{\pi} \quad \forall \pi$  ——— which constrains the coefficients

- Notice:
  - $\mathbf{w}=0$  is a solution (eww!)
- Resolve this:
  - require that expert's policy is better than any other
    - by some amount that depends on the difference!

# This yields

$$\min_{\mathbf{w}} \mathbf{w}^T \mathbf{w}$$

such that

$$\mathbf{w}^T \mu^{\pi^*} \geq \mathbf{w}^T \mu^{\pi} + d(\pi^*, \pi)$$

For example, number of states in which  $\pi^*$  was observed and also  $\pi$  and  $\pi^*$  disagree

- Notice
  - each policy yields a vector  $\mu$  and so a linear constraint on  $w$
  - but the feasible set created by the inequalities might be empty!
- Solution - slack variables

# Final problem - NASTY (ish)

$$\min_{\mathbf{w}} \mathbf{w}^T \mathbf{w} + C \sum_i \xi_i$$

such that

$$\mathbf{w}^T \mu^{\pi^*} \geq \mathbf{w}^T \mu^{\pi_i} + d(\pi^*, \pi_i) - \xi_i$$

$$\xi_i \geq 0$$



Nasty, because there are a lot of constraints  
(one per policy)

BUT quite good algorithms exist

Slack variables

# Constraint generation

Initialize  $\Pi^{(i)} = \{\}$  for all  $i$  and then iterate

- Solve

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

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

- For current value of  $w$ , find the most violated constraint for all  $i$  by solving:

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

= find the optimal policy for the current estimate of the reward function (+ loss augmentation  $m$ )

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

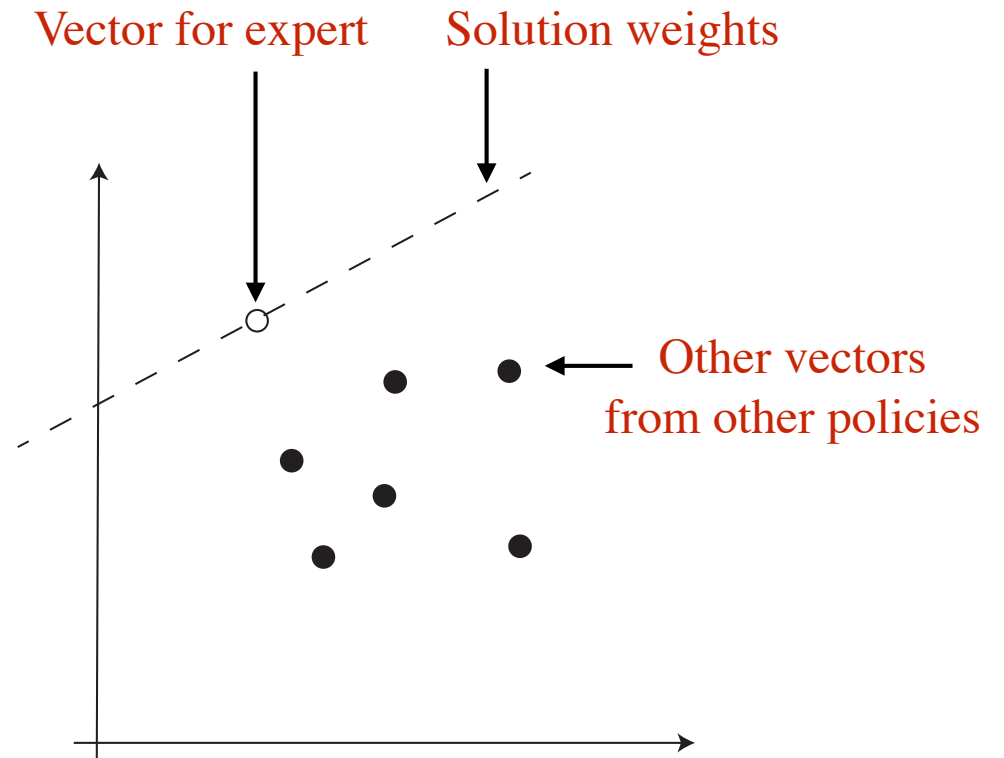
From Abbeel slides

- If no constraint violations were found, we are done.

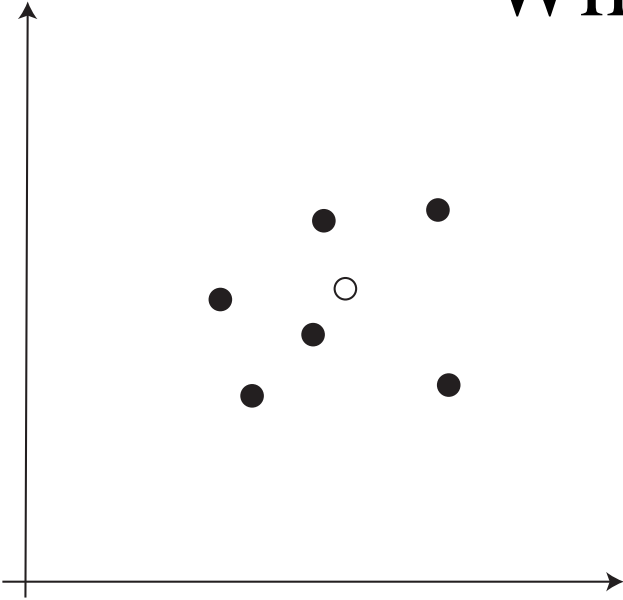


# Visualize...

- Each policy gives a vector
  - pretend these are 2D vectors



# What could go wrong?



- This could happen because
  - the expert wasn't optimal
  - the representation of rewards isn't good enough
- Options
  - find several policies and mix
  - use better feature functions

# Alternative: feature matching

$$\mathbf{w}^T \mu^{\pi^*} \geq \mathbf{w}^T \mu^{\pi} \quad \forall \pi \quad \text{—— which constrains the coefficients}$$

- Notice that if we can get

$$\|\mu^{\pi} - \mu^{\pi^*}\|_1 \leq \epsilon \quad \text{ie a policy vector very close to experts}$$

- then for any  $\mathbf{w}$  such that

$$\|\mathbf{w}\|_{\infty} \leq 1 \quad \text{ie reasonably sized weight vector}$$

- we have

$$|\mathbf{w}^T \mu^{\pi} - \mathbf{w}^T \mu^{\pi^*}| \leq \epsilon \quad \text{reward is close to experts}$$

# Apprenticeship learning [Abbeel & Ng, 2004]

- Assume  $R_w(s) = w^\top \phi(s)$  for a feature map  $\phi : S \rightarrow \mathbb{R}^n$ .
- Initialize: pick some controller  $\pi_0$ .
- Iterate for  $i = 1, 2, \dots$  :

- **“Guess” the reward function:**

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

$$\begin{aligned} & \max_{\gamma, w: \|w\|_2 \leq 1} \gamma \\ & \text{s.t. } w^\top \mu(\pi^*) \geq w^\top \mu(\pi) + \gamma \quad \forall \pi \in \{\pi_0, \pi_1, \dots, \pi_{i-1}\} \end{aligned}$$

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

From Abbeel slides