Reinforcement Learning

Lecture 13:
Generative Model
Inverse RL

Yishay Mansour, Tel-Aviv University

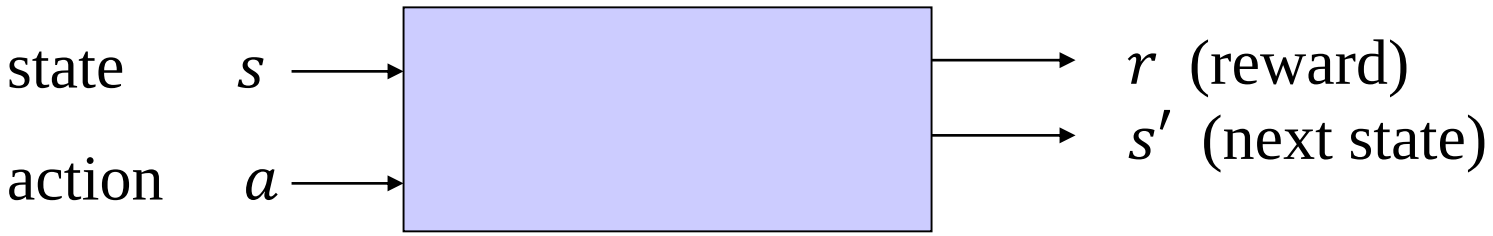
Lecture 13: outline

- Generative Model
  - Definition
  - Policy evaluation
  - Policy class
  - Gradient computation
  - Near optimal policy
- Inverse RL
  - Cloning
  - Small state space
  - Linear function Approximation
- Summary of the course

Generative Model: MDP

- A different representation
  - Exponential size
  - Infinite size
- Implicit representation

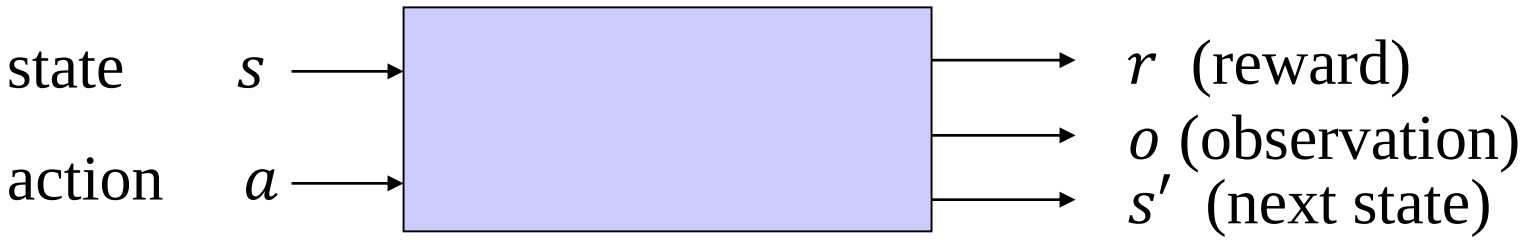
- Model:
  - Inputs: state and action
  - Outputs: reward, next state



Generative Model: POMDP

- A different representation
  - Even for poly-size POMDP
- Implicit representation

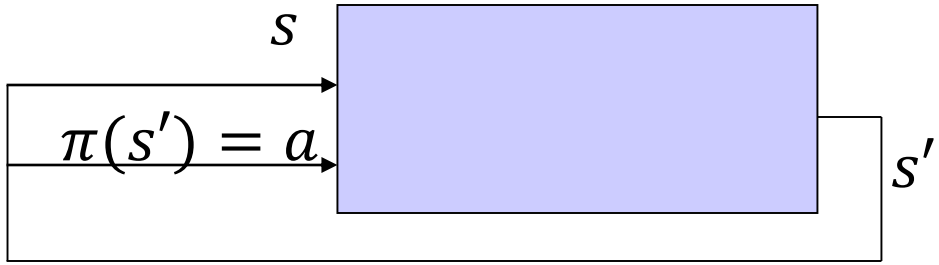
- Model:
  - Inputs: state and action
  - Outputs: reward, observation, next state



Policy Evaluation: MDP

- Truncated Discounted return
- Initialize $s = s_0$
- Time $t \in [0, H]$
  - $a = \pi(s_t)$
  - Call Generative Model
  - Get
    - Reward r_t
    - Next state s_{t+1}

- Single unbiased estimate:
  - $v_i = \sum_{t=0}^{H} \gamma^t r_t$

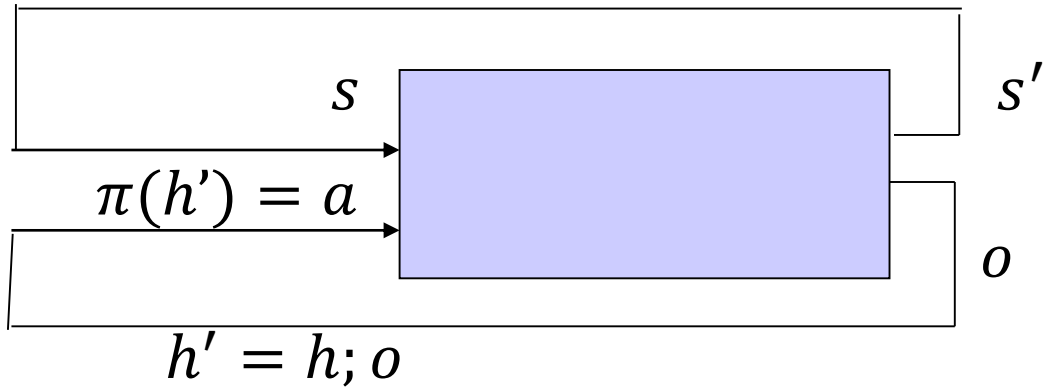


Policy Evaluation: MDP

- Multiple estimates:
  - $m = \frac{V_{max}^2}{\epsilon^2} \log \frac{1}{\delta}$
  - $V_{max} = \frac{R_{max}}{1-\gamma}$
- Estimate:
  - $\hat{V}^{\pi}(s_0) = \frac{1}{m} \sum_{i=1}^{m} v_i$
- Theorem: with prob $1 - \delta$

$$\left| \hat{V}^{\pi}(s_0) - V^{\pi}(s_0) \right| \leq \epsilon$$

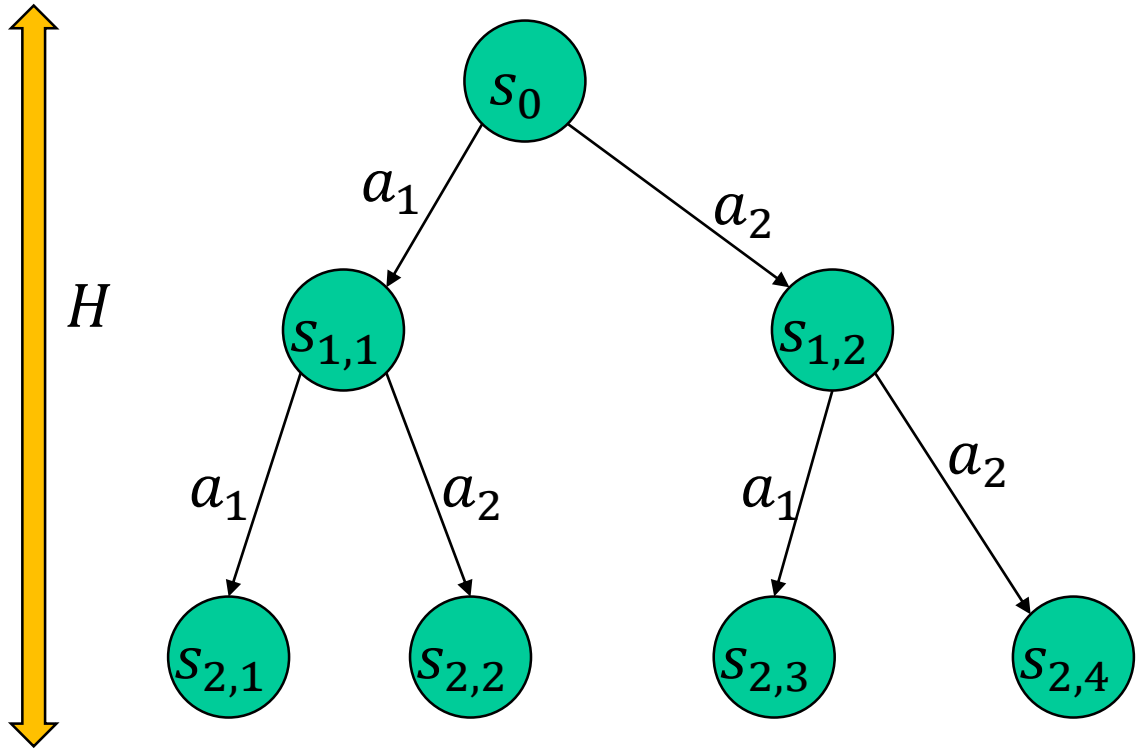
- Same for POMDP



Trajectory tree: Generic unbiased estimator

- Goal
  - Use generator
  - Build data structure
    - Trajectory tree T
- Latter
  - For any policy π
  - Use T to evaluate π
  - $E_T[T(\pi)] = V^{\pi}(s_0)$

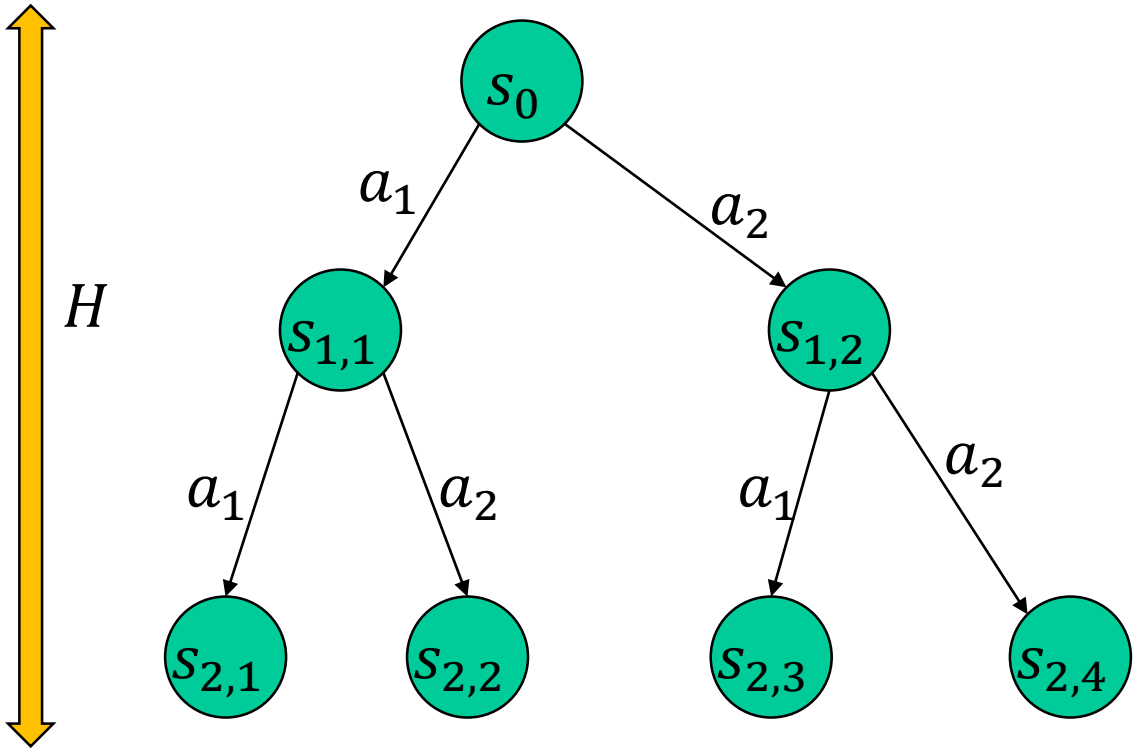
- Trajectory tree:



Trajectory tree: Generic unbiased estimator

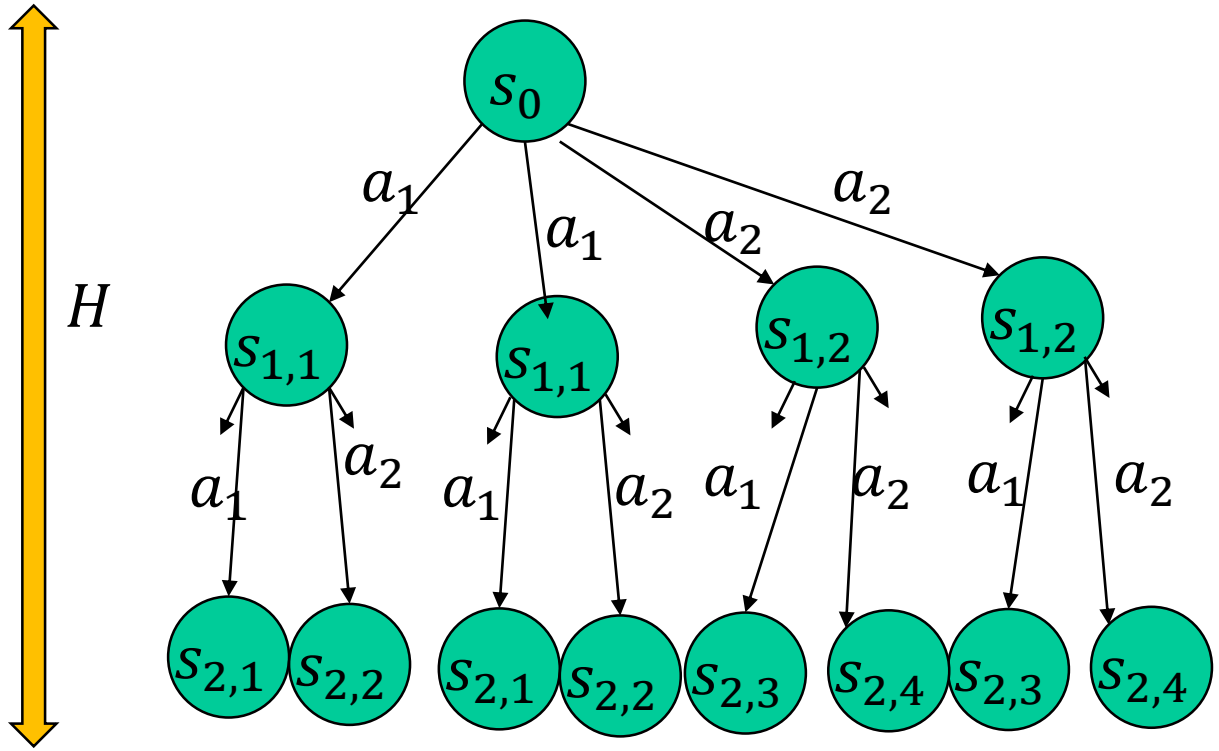
- Basic idea:
  - Tree encodes all trajectory
    - Up to length H
  - Any policy π
    - Can compute $T(\pi)$
  - Clearly
    - $E_T[T(\pi)] = V^{\pi}(s_0)$
- Generates a single sample
  - For any policy π

- Trajectory Tree



Multiple-Trajectory Tree

- Sample each action m times
- Tree
  - Depth H
  - Degreed $m|A|$
- Each policy π
  - m independent trajectories
    - For m Policy Evaluations



Policy Evaluation: Finite policy class

- Finite policy class
  - $\Pi = \{\pi: S \to A\}$
- Build Multi-trajectory Tree
  - Let $m = \frac{V_{max}^2}{\epsilon^2} \log \frac{|\Pi|}{\delta}$
- For each $\pi \in \Pi$
  - $\hat{V}^{\pi}(s_0) = \frac{1}{m} \sum_{i=1}^{m} v_i$
  - with prob $1 - \delta/|\Pi|$
    $\left|\hat{V}^{\pi}(s_0) - V^{\pi}(s_0)\right| \leq \epsilon$

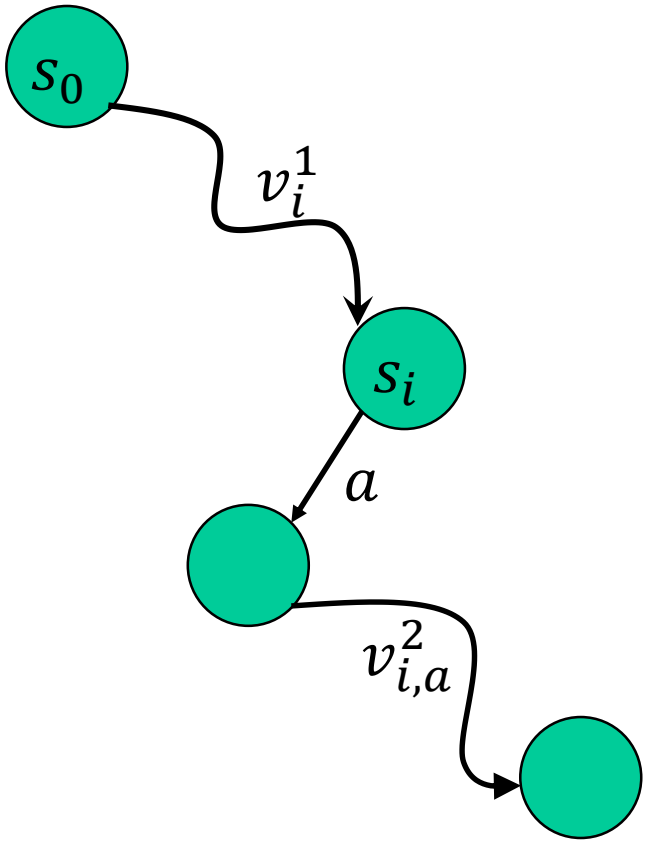
- Theorem:
  - with prob $1 - \delta$
    $\left|\hat{V}^{\hat{\pi}}(s_0) - V^{\pi^*}(s_0)\right| \leq 2\epsilon$
  - where
    - $\hat{\pi} = \arg\max_{\pi \in \Pi} \hat{V}^{\pi}(s_0)$
    - $\pi^* = \arg\max_{\pi \in \Pi} V^{\pi}(s_0)$

Gradient based algorithm

- Assume that policy class Π
  - Smoothly parametrized
  - $\pi(\cdot;\theta_0)$
- We like to compute gradient
  - Fix T
  - Compute $\frac{\partial}{\partial\theta}T(\pi(\cdot;\theta))$
    - Unbiased estimate
    - Efficiently
- For each node $i \in T$
  - Depth d_i and history h_i
    - Value v_i^1 up to it
  - Remaining value v_i^2
    - Using π
- The policy $\pi(\cdot;\theta)$
  - $P[i]$ = prob π reaches node i
  - $\pi(a|i;\theta) = p_{i,a}$
    - prob π selects a
    - Specifies π

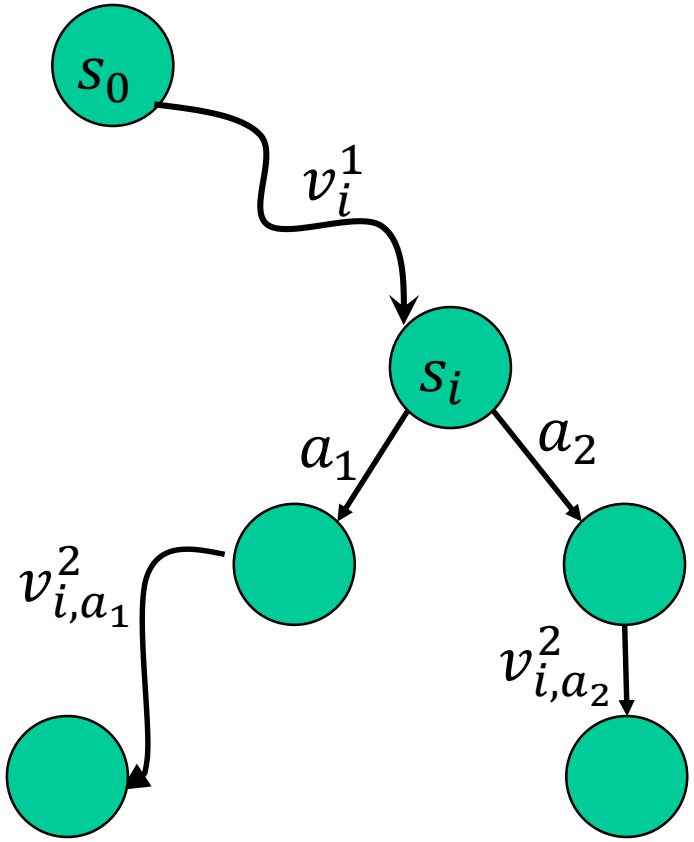
Gradient based algorithm

- Value
  - $E[T(\pi(\cdot;\theta)] = \sum_{t=0}^{H} \gamma^t \, E[r_t|T,\pi]$
  - $\propto \sum_{i\in T, a\in A} P[i](v_i^1 + \gamma^{d_i} E[v_{i,a}^2])\pi(a|i;\theta)$
- Using the chain rule
  - $\frac{\partial}{\partial\theta} T(\pi(\cdot;\theta)) = \sum_{i,a} \frac{\partial T}{\partial p_{i,a}} \frac{\partial p_{i,a}}{\partial \theta}$
  - $\frac{\partial T}{\partial \theta} \propto \sum_{i,a} P[i] \gamma^{d_i} E[v_{i,a}^2] \frac{\partial p_{i,a}}{\partial \theta}$



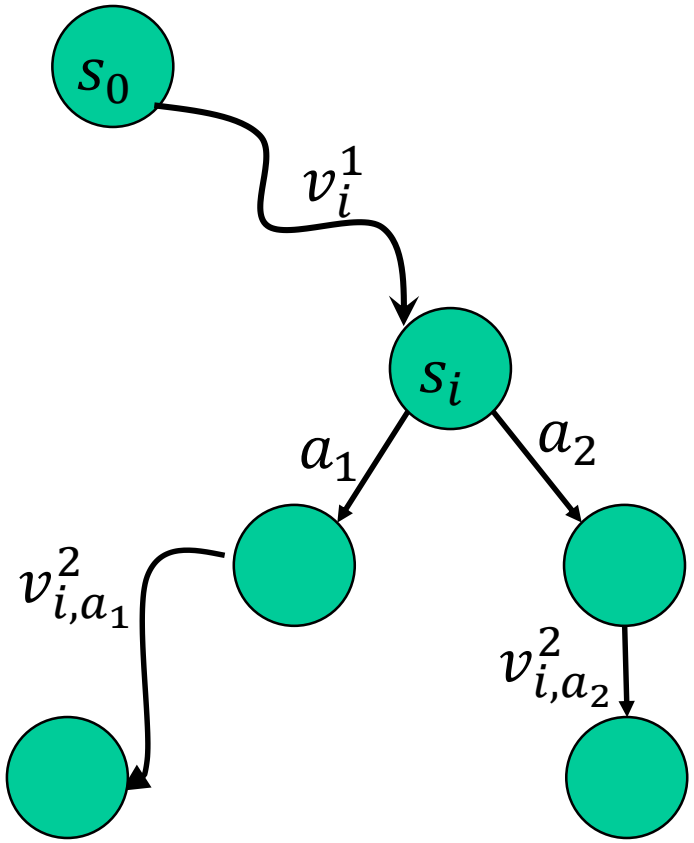
Gradient based algorithm

- Sample
  - Node $i \sim P[i]\gamma^{d_i}$
  - Sample $v_{i,a}^2$
  - Return $v_{i,a}^2 \frac{\partial p_{i,a}}{\partial \theta}$
- Result
  - $E_i\left[v_{i,a}^2 \frac{\partial p_{i,a}}{\partial \theta}\right] \propto \frac{\partial}{\partial \theta} T(\pi(\cdot;\theta))$



Gradient based algorithm

- Implementing the sample
  - Select $d \sim GeomDist(1-\gamma)$
    - $\Pr[d = \ell] \propto \gamma^{\ell}$
  - Follow $\pi(\cdot;\theta)$ to depth d
    - Defines node i
  - For each action a
    - Follow π and Sample $v_{i,a}^2$
  - Return $\sum_a v_{i,a}^2 \frac{\partial p_{i,a}}{\partial \theta}$



Gradient based algorithm

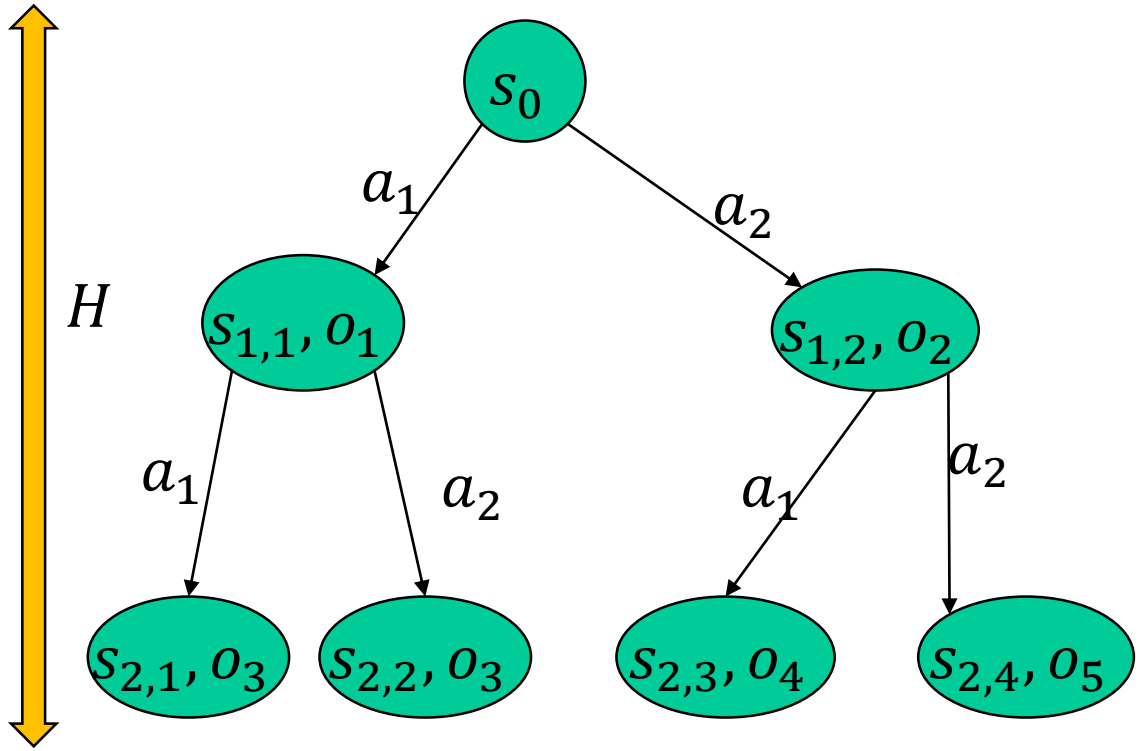
- Output
  - Unbiased gradient estimates
- Can reach a local maximum
  - follow the gradient

- Complexity
  - Do not build the tree
    - Be lazy!
  - Sample the path to node i
  - Do actions $a \in A$
  - For each action, sample continuation
- Calls to generative model
  - $O(H\ |A|)$

From MDP to POMDP

- We have also observations
  - Each node
    - State
    - Observation
- Policy:
  - Depends only on observations
    - The history of observations
- Everything else the same!
  - Policy evaluation
  - Gradient computation

- Trajectory tree:



Planning Model: Near Optimal Policy

- Given
  - Generative model
- Output
  - Near Optimal policy
- How will we define the policy?
  - The generative model is a black-box!

- Output:
  - Algorithm
    - Uses generative model
  - Input: state
  - Output: action
  - Uses the generative model
    - During the computation
  - Implicitly defines a policy

Main result: Near optimal policy

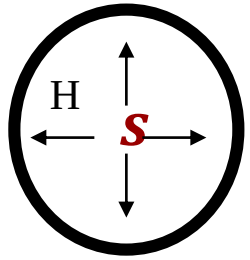
- Theorem:
  - There exists an algorithm that
  - Given a generative model
  - Defines a policy
  - Such that
  - $V^*(s) - V^{\pi}(s) \leq \epsilon$

- Running time of algorithm:
  - Exponential in H
  - Independent from $|S|$
    - Can handle infinite state space!

Near optimal policy: simple case

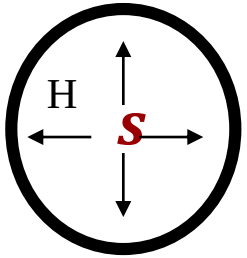
- Bounded support assumption:
  - For any state s
  - Action a
  - Support of $p(\cdot\,|s,a)$ bounded
    - $|\{s' : p(s'|s,a) > 0\}| \leq d$
- Latter:
  - Get rid of this assumption

- Fix state s
  - Consider horizon H
  - How many nodes can we reach?
  - Almost $(d|A|)^H$



Near optimal policy: simple case

- Solution:
  - Consider all nodes at distance at most H
  - Assume reward from other nodes is zero
    - Error at most ϵ
  - Compute $Q^{*,H}(s,a)$ for each a
  - Let a' be the action that maximizes $Q^{*,H}(s,\cdot)$
    - $|Q^*(s,a') - Q^*(s,a^*)| \leq \epsilon$
  - Will show: This implies that the policy is $\frac{\epsilon}{1-\gamma}$ optimal



Near optimal policy: simple case

- Lemma:
- If for every s,
  - $\pi(s)$ gurantees
  - $\left|Q^*\left(s,\pi(s)\right) - Q^*(s,a^*)\right| \leq \epsilon$
  - With prob at least $1-\delta$
- Then for every s,
  - $|V^\pi(s) - V^*(s)| \leq \frac{\epsilon+2\delta V_{max}}{1-\gamma}$

- Proof:
  - $E\left[Q^*\left(s,\pi(s)\right)\right] \geq$
  - $(1-\delta)(Q^*(s,a^*)-\epsilon) - \delta V_{max}$
  - $\geq Q^*(s,a^*) - \epsilon - 2\delta V_{max}$
  - $\geq Q^*(s,a^*) - \beta$
  - Where
    - $\beta = \epsilon + 2\delta V_{max}$

Near optimal policy: simple case

- Claim:
  - If
    - $\left|Q^*(s,a^*) - Q^*\left(s,\pi(s)\right)\right| \leq \beta$
  - Then
    - $V^*(s) - V^{\pi}(s) \leq \frac{\beta}{1-\gamma}$
- Proof:
  - Let π_i run π first i steps
    - Then run π^*
- For π_1
  - $V^*(s) - V^{\pi_1}(s) \leq \beta$
- For π_i
  - $V^{\pi_i}(s) - V^{\pi_{i+1}}(s) \leq \gamma^i \beta$
    - First i steps are identical
- Sum all ineqaulties:
  - $V^*(s) - V^{\pi}(s) \leq \sum_i \gamma^i \beta = \frac{\beta}{1-\gamma}$
- Q.E.D

Near optimal policy: general case

- Goal:
  - Remove the bounded support assumption
- Consider:
  - Next state uniform on S
  - Cannot estimate next state well!
- Can still use trajectory trees
  - Approximate the neighborhood
- Main issue
  - What happens to errors

Near optimal policy: General case

- Basic idea:
  - Given state s
  - For each action $a \in A$
  - Sample m times
    - Using the generative model
      - $s'_{a,i}$
  - Compute recursively
    - For each $s'_{a,i}$ compute its optimal return

- Let
  - $\hat{Q}^*(s,a) = r(s,a) + \frac{\gamma}{m}\sum_i V^*(s'_{a,i})$
- Lemma
  - With prob $1 - e^{-\lambda^2 m/V_{max}^2}$
  - $\left|Q^*(s,a) - \hat{Q}^*(s,a)\right| \leq \lambda$
- Proof:
  - Concentration bound.

Near optimal policy: General case

- Goal
  - Recursively estimate $V^*(s'_{a,i})$
- Problem:
  - We wanted $V^*(s)$
  - We need to compute $V^*(s'_{a,i})$
  - Are we making progress?

- Observation
  - Discounting damps errors
  - We can tolerate more errors!

- We need to solve more problems
  - But can allow higher error

Near Optimal Policy: Algorithm

- $EstimateQ(0, s, a)$:
  - Return 0
- $EstimateQ(n, s, a)$:
  - Sample m times: $s'_{a,i}$
    - Using the generative model
  - Return
    - $r(s, a) + \frac{\gamma}{m} \sum_i EstimateV(n-1, s)$

- $EstimateV(n, s)$:
  - Return $\max_a EstimateQ(n-1, s, a)$

- $Main(s)$:
  - For each $a \in A$ compute
  - $q_a = EstimateQ(H, s, a)$
  - Return $a' = \arg\max_a q_a$

Near Optimal Policy: Analysis

- Sources of errors
  - Finite sample
    - Only m next states
  - Recursive error
    - $EstimateV(n-1, s)$
- Proof idea:
  - Choose m large enough
    - Controls sample error
  - Bound the recursive error
    - Using the discounting
    - Show it decreases!

Near Optimal Policy: Analysis

- Notation
  - $Q^n(s,a) = EstimateQ(n,s,a)$
  - $V^n(s) = EstimateV(n,s)$
- Computation
  - $Q^n(s,a) = r(s,a) + \gamma \frac{1}{m} \sum_i V^{n-1}(s'_{a,i})$
    - $Q^0(s,a) = 0$
  - $V^{n-1}(s) = \max_a Q^{n-1}(s,a)$

- Error:
  - $Q^*(s,a) - Q^n(s,a) \leq \alpha_n$
  - Implies:
  - $V^*(s) - V^n(s) \leq \alpha_n$

- Initially
  - $\alpha_0 = V_{max}$

Near Optimal Policy: Analysis

- Error bound
  - $|Q^*(s,a) - Q^n(s,a)|$
  - $= \gamma |E[V^*(s')] - \frac{1}{m}\sum_i V^{n-1}(s'_{a,i})|$
  - $\leq \gamma \left|E[V^*(s')] - \frac{1}{\mathrm{m}}\sum_i V^*(s'_{a,i})\right| + \gamma \left|\frac{1}{\mathrm{m}}\sum_i V^*(s'_{a,i}) - V^{n-1}(s'_{a,i})\right|$
- $\alpha_n \leq \gamma(\epsilon + \alpha_{n-1})$
- $\alpha_H \leq \sum_{i=1}^{H} \gamma^i \epsilon + \gamma^H V_{max} \leq \frac{\epsilon}{1-\gamma} + \gamma^H V_{max}$

Putting it all together

- Set
  - $\epsilon = \epsilon'(1-\gamma)$
  - $H = \frac{1}{1-\gamma}\log\frac{V_{max}}{\epsilon'}$
- Implies
  - $\frac{\epsilon}{1-\gamma} + \gamma^H V_{max} \leq 2\epsilon'$

- Theorem
  - For any state s, we have
  - $V^*(s) - V^\pi(s) \leq 2\epsilon'$

Lower Bound

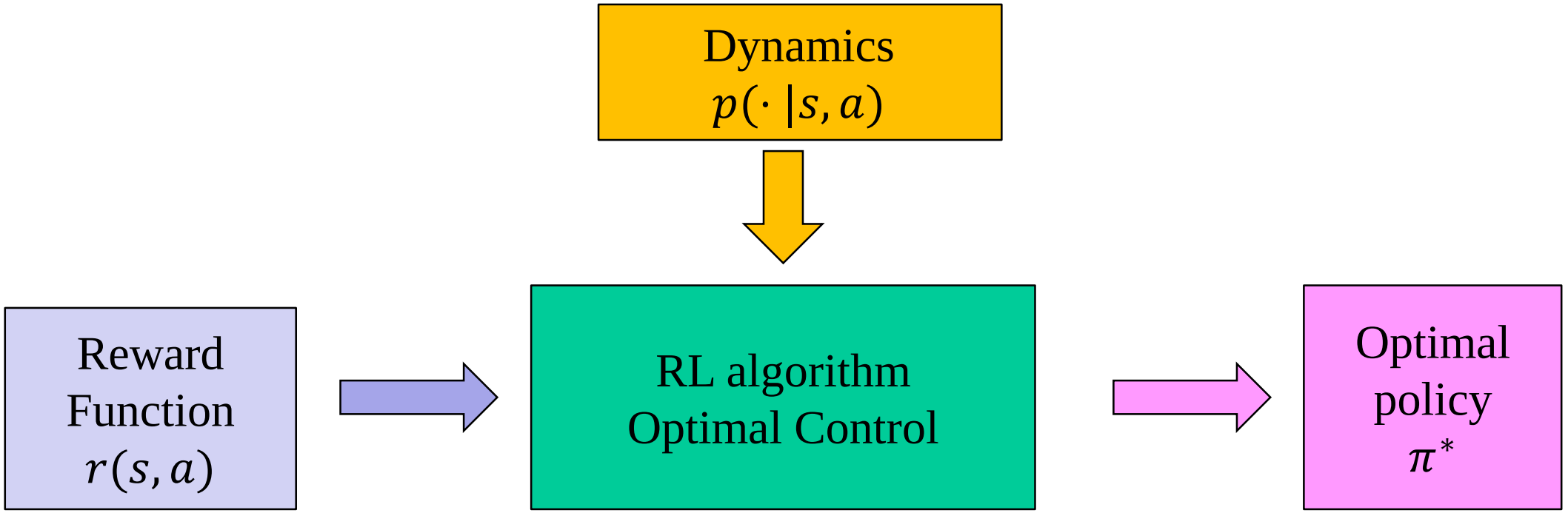
- The exponential dependence on H is inherent.
- MDP:
  - complete $|A|$-degree tree
    - Depth H
  - All rewards zero
  - Except in one leaf
    - Selected uniformly at random
- Claim
  - Any algorithm,
  - Using a generative model
  - Needs at least $\frac{1}{2}|A|^H$ querries
    - In expectation

Lecture 13: outline

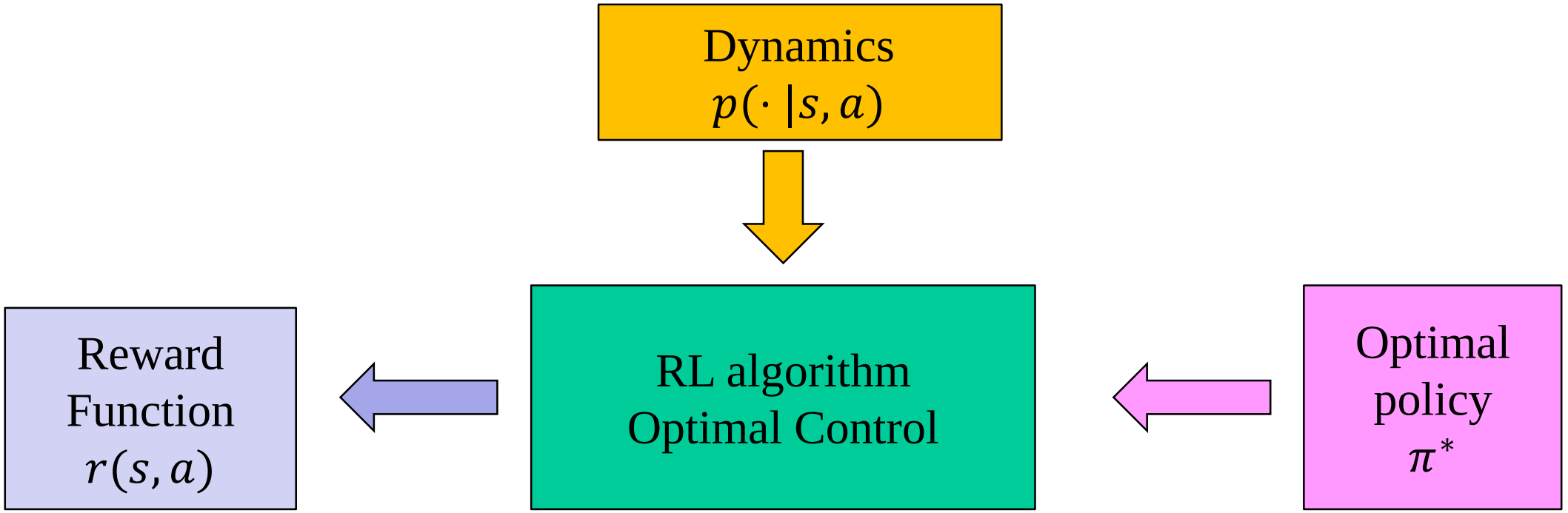
- Generative Model
  - Definition
  - Policy evaluation
  - Policy class
  - Gradient computation
  - Near optimal policy
- Inverse RL
  - Cloning
  - Small state space
  - Linear function Approximation
- Summary of the course

INVERSE REINFORCEMENT LEARNING

High level: Forward view of RL



High level: Inverse RL



IRL: problem setup

- Input:
  - States S, Actions A
  - Dynamics $p(s'|s,a)$
  - No reward function!
  - Optimal policy
    - Explicitly
    - trajectories

- Behavioral cloning
  - Given trajectories
  - Learn optimal policy
- Inverse RL
  - Recover consistent reward R
- Apprentice learning via IRL
  - Learn a good policy using R

Behavioral Cloning

- Formulate as an ML problem
  - Input:
    - Trajectories from π^*
      - (s_t, a_t) where $a_t = \pi^*(s_t)$
    - Rewards unobserved
    - Dynamics unknown
  - Fix a policy class
    - Linear, DNN, SVM …
  - Learn from examples
    - Supervised learning
- Goal: cloning
  - Recover π^* directly
- Problems
  - What happens after an error?
    - New states
  - Generalization (unseen parts)
  - Transfer learning (similar tasks)
- Cloning vs IRL
  - Policy $\hat{\pi}^*$ vs rewards $\hat{R}$

IRL: small state space MDP

- Given:
  - States S, Actions A
  - Dynamics $p(s'|s,a)$
  - Optimal policy π^*
    - Explicitly
- Output
  - Reward function R
    - For which π^* is optimal

- Solution:
  - Characterize all reward R
    - For which π^* is optimal
- Bellman optimality
  - $Q^*\left(s,\pi^*(s)\right) \geq Q^*(s,a)$
    - $\forall s \in S\,, a \in A$

- Can we get more from this?
  - We do not know $Q^*(s,a)$

IRL: Reward Characterization

- Define matrices
  - P^* s.t. $P^*[i,j] = p(s_j|s_i, \pi^*(s_i))$
    - Optimal policy dynamics, $a^* = a_1$
  - P^a s.t. $P^a[i,j] = p(s_j|s_i, a)$
    - Deviation to action a
- Optimal value function
  - $V^* = R + \gamma P^* V^*$
  - $V^* = (I - \gamma P^*)^{-1} R$
- Optimal state-action function
  - $Q^*(s,a) = R + \gamma P^a V^*$
  - $= R + \gamma P^a (I - \gamma P^*)^{-1} R$
- For Bellman optimality
  - $Q^*\big(s, \pi^*(s)\big) - Q^*(s,a) \geq 0$
  - $(P^* - P^a)(I - \gamma P^*)^{-1} R \geq 0$
- Policy π^* is optimal for reward R
  - Iff $(P^* - P^a)(I - \gamma P^*)^{-1} R \geq 0$

IRL: reward characterization

- Challenges
  - Trivial solution
    - $R = 0$
  - Only Trajectories
  - Large state space
  - Noisy teacher
- Other objectives
  - $\sum_{s \in S} Q^*(s, a^*) - \max_{a \neq a^*} Q^*(s, a)$
  - $\sum_{s \in S} \sum_{a \in A} Q^*(s, a^*) - Q^*(s, a)$
- Ambiguity: Max margin
  - $\max_{s \in S} \left\{ Q^*(s, a^*) - \max_{a \neq a^*} Q^*(s, a) \right\}$
  - Simple linear program
    - Rename actions $a^* = a_1$
    - $\max \lambda$
    - $(P^* - P^a)(I - \gamma P^*)^{-1} R \geq \lambda$
      - $\forall a \neq a^*$
    - $-R_{max} \leq R \leq R_{max}$

IRL: Linear function approximation

- State encoding
  - $\phi(s) \in \mathbb{R}^n$
- Reward weights
  - $w \in \mathbb{R}^n$
- Rewards:
  - $r(s) = w^\top \phi(s)$
    - $r(s,a) = r(s)$

- Value function
  - $V^\pi(s_0) = E[\sum_{t=0}^{\infty} \gamma^t w^\top \phi(s_t)|\pi]$
  - $= w^\top E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi]$
  - $= w^\top \mu(\pi)$
    - $\mu(\pi) = \sum_{t=0}^{\infty} \gamma^t E[\phi(s_t)|\pi]$
- IRL:
  - Recover w^* such that
  - ${w^*}^\top \mu(\pi^*) \geq {w^*}^\top \mu(\pi) \quad \forall \pi$

IRL: Linear function approximation

- Approx $\mu(\pi)$ from trajectories
  - $\mu(\pi) \approx \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{T}\gamma^t\,\phi(s_t)$
- Observation
  - If $\left\|\mu(\pi^1) - \mu(\pi^2)\right\|_2 \leq \epsilon$
  - Then $\left|w^\top\mu(\pi^1) - w^\top\mu(\pi^2)\right| \leq \epsilon\|w\|_2$
- Goal
  - Given $\mu(\pi^*)$ find policy π
    - s.t. $\|\mu(\pi^*) - \mu(\pi)\|_2 \leq \epsilon$
    - Good for any rewards!
- Problem
  - Solution $w = 0$
- Again, max margin
    - Or similar variants
  - $\max \lambda$
  - $w^\top\mu(\pi^*) \geq w^\top\mu(\pi) + \lambda \quad \forall\pi$
  - $\|w\|_2^2 \leq 1$
- Weakness
  - Huge number of constraints!

IRL: Linear function approximation

- Constraint generation algorithm:
  - Input μ^*
  - Maintain a set of policies Π
    - Initially $\Pi = \emptyset$
  - Initially set π^0 arbitrarily
    - Estimate $\mu^0 = \mu(\pi^0)$
  - At iteration i
    - $\max_w \min_{j \in \Pi} w^\top (\mu^* - \mu^j)$
      - If at most ϵ terminate
      - Let w^i be the maximizer
    - Solve for optimal policy given w^i
      - Name π^i and add to Π
  - At termination
    - Compute
    - $\min \|\mu^* - \mu\|_2$
      - s.t. $\mu = \sum_i \alpha_i \mu^i$, $\sum_i \alpha_i = 1$, $\alpha_i \geq 0$
    - Output μ

IRL: Linear function approximation

- Implementing the algorithm
- Given finite Π compute

$$\max_{w} \min_{j \in \Pi} w^\top (\mu^* - \mu^j)$$

  - Write a quadratic program:
  - $\max \lambda$
  - $s.t.$
    - $w^\top \mu^* \geq w^\top \mu^j + \lambda \quad \forall j \in \Pi$
    - $w^\top w \leq 1$

- Reduction to SVM:
  - Positive example: μ^*
  - Negative examples: $\mu^j \quad \forall j \in \Pi$
  - Max-margin classifier: w

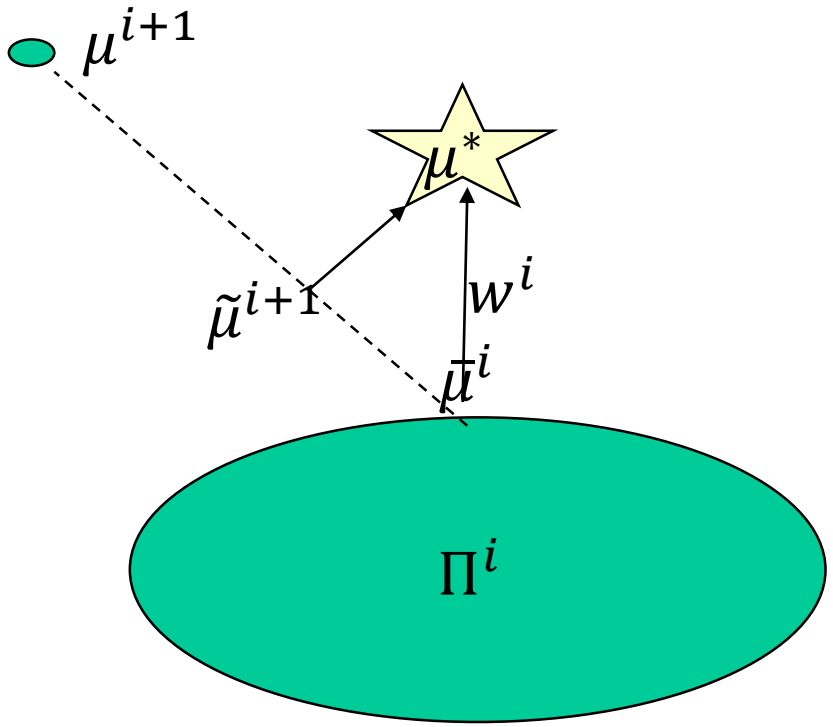
IRL: Linear function approximation

- Implementing the algorithm
- Finding π^i
  - Given w^i
- Build an MDP
  - Rewards: $r^i(s) = w^i \cdot \phi(s)$
- Solve for optimal policy
  - Policy π^i
  - Given rewards r^i

- Upon termination
  - Given $\mu = \sum_i \alpha_i \mu^i$
    - s.t. $\sum_i \alpha_i = 1$, $\alpha_i \geq 0$
      - Support at most $n + 1$
  - Implementing μ
    - Initially, select policy π^i w.p. α_i
    - Run only the selected policy
    - Expected state vector is μ
      - Close to μ^*
  - Note the difference from selecting at each step!

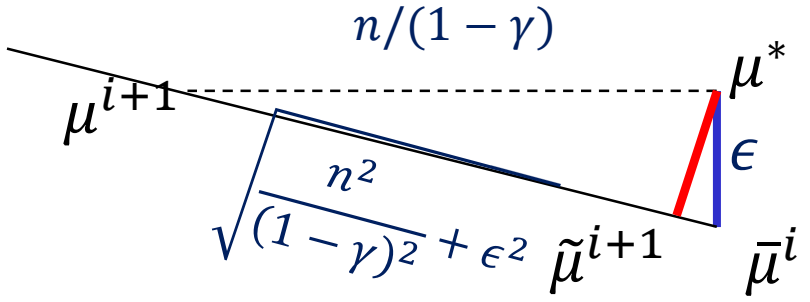
IRL: Linear function approximation

- Geometric view



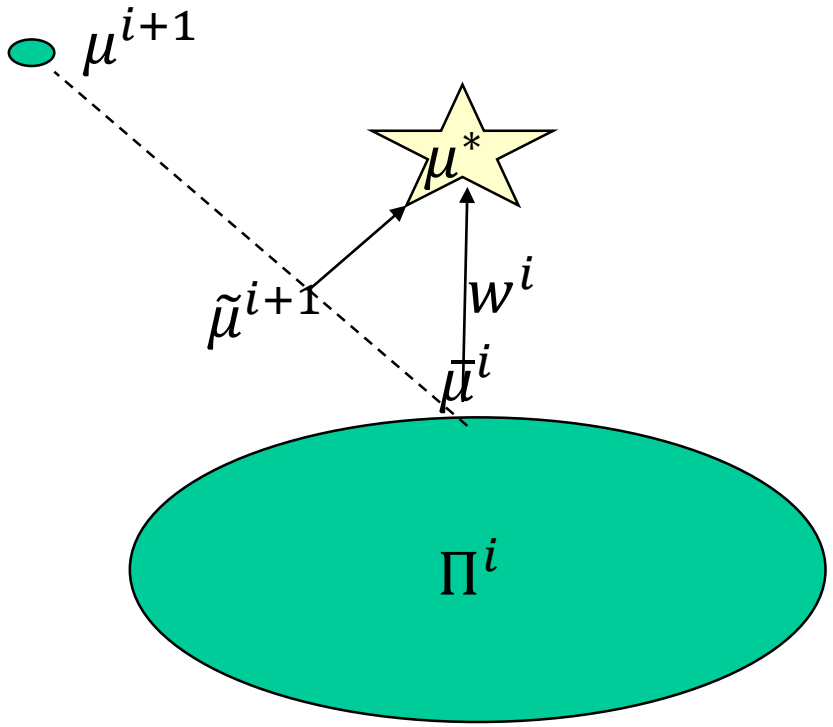
- Lemma (Geometric):
  - $\frac{\|\mu^* - \tilde{\mu}^{i+1}\|_2}{\|\mu^* - \bar{\mu}^i\|_2} \leq \frac{n}{\sqrt{n^2 + (1-\gamma)^2 \|\mu^* - \bar{\mu}^i\|^2}}$

- Proof: worse case:

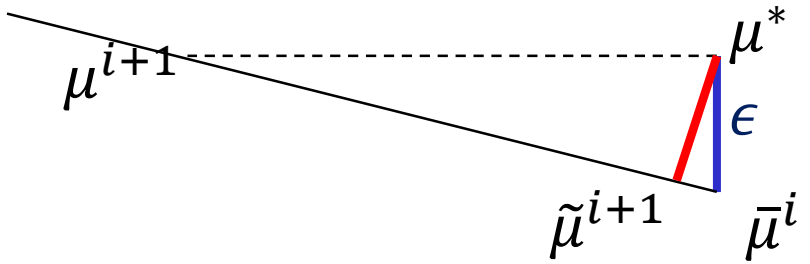


IRL: Linear function approximation

- Geometric view



- Claim:
  - The point $\tilde{\mu}^{i+1}$
  - is between $\bar{\mu}^i$ and μ^{i+1}
  - Guarantees that
    - $\|\mu^* - \tilde{\mu}^{i+1}\|_2 \geq \|\mu^* - \bar{\mu}^{i+1}\|_2$



IRL: Linear function approximation

- Claim: number of iterations at most
  - $O(\frac{n}{(1-\gamma)^2\epsilon^2}\log\frac{n}{(1-\gamma)\epsilon})$
  - Recall: $\|\mu\|_2 \leq \frac{n}{1-\gamma}$

- Sample complexity:
  - $m = O(\frac{n}{\epsilon^2(1-\gamma)^2}\log\frac{n}{\delta})$
  - Guarantees that:
    - $\|\mu - \hat{\mu}\| \leq \epsilon$

- Demonstration videos

http://ai.stanford.edu/~pabbeel//irl/

Lecture 13: outline

- Generative Model
  - Definition
  - Policy evaluation
  - Policy class
  - Gradient computation
  - Near optimal policy
- Inverse RL
  - Cloning
  - Small state space
  - Linear function Approximation
- Summary of the course

COURSE SUMMARY

Course summary

- Planning problems
  - Bellman optimality
  - Value iteration
  - Policy iteration
- Learning: Small state space
  - Model based
  - Model free
    - Value function
- Learning: Large state space
  - Function Approximation
  - Policy gradient
- More topics
  - Bandits
  - POMDP
  - LQR
  - Generative model
  - Inverse RL