CS5340
Uncertainty Modeling in AI

Lecture 11:
Planning and Inference

"How to Act in an Uncertain World"

Asst. Prof. Harold Soh

AY 2023/24

Semester 2

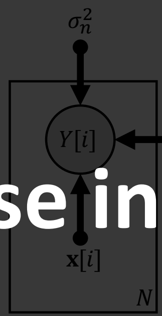
Course Schedule (Tentative)

Week	Date	Lecture Topic	Tutorial
1	16 Jan	Introduction to Uncertainty Modeling + Probability Basics	~~Introduction~~
2	23 Jan	Simple Probabilistic Models	Introduction and Probability Basics
3	30 Jan	Bayesian networks (Directed graphical models)	More Basic Probability
4	6 Feb	Markov random Fields (Undirected graphical models)	DGM modelling and d-separation
5	13 Feb	Variable elimination and belief propagation	MRF ~~+ Sum/Max Product~~
6	20 Feb	Factor graphs	**Quiz 1**
-	-	**RECESS WEEK**	
7	5 Mar	Mixture Models and Expectation Maximization (EM)	Linear Gaussian Models
8	12 Mar	Hidden Markov Models (HMM)	Probabilistic PCA
9	19 Mar	Monte-Carlo Inference (Sampling)	Linear Gaussian Dynamical Systems
10	26 Mar	Variational Inference	MCMC + Langevin Dynamics
11	2 Apr	Inference and Decision-Making (optional)	Diffusion Models + Sequential VAEs
12	9 Apr	Gaussian Processes (optional)	**Quiz 2**
13	16 Apr	Closing Lecture	Project Presentations

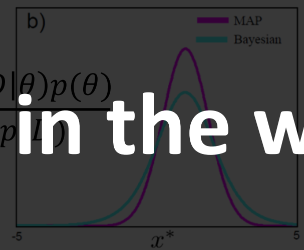
CS5340 in a nutshell

CS5340 is about how to "**represent**" and "**reason**" with uncertainty in a computer.

Can I use inference for acting in the world?



$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$



Representation: The *language* is probability and probabilistic graphical models (PGM).

The language is used to model problems.

Reasoning: We use learning and inference algorithms to answer questions.

e.g., Belief-propagation/sum-product, MCMC, and variational Bayes

Disclaimer: An introduction

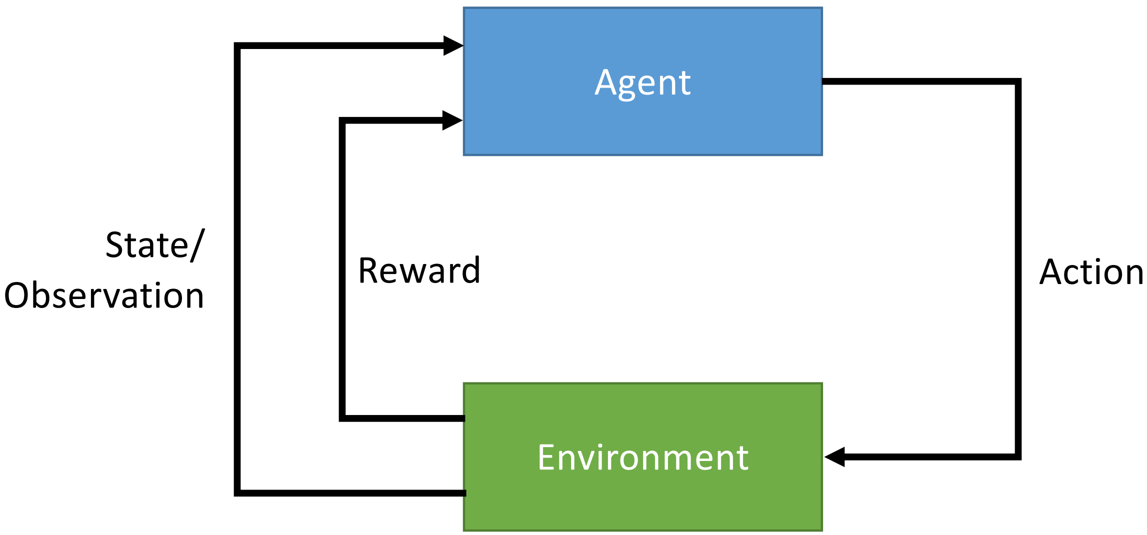
- **Condenses** part of a robotics / reinforcement learning course
- **Focus:** Key Ideas and Techniques
- We will **not** be able to:
  - Delve into theory
  - All variants of the shown methods
- **Selected Sources:**
  - Probabilistic Robotics, Chapters 14 and 15
  - Reinforcement Learning: An Introduction, Chapters 3 and 4
  - UC Berkeley CS287 Lecture 5, and CS294-112 Lecture 10
    - https://people.eecs.berkeley.edu/~pabbeel/cs287-fa19/
    - http://rail.eecs.berkeley.edu/deeprlcourse-fa18/
  - “Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review”, Sergei Levine, 2018, https://arxiv.org/abs/1805.00909
    - UC Berkeley CS285 Lecture 14 and 15

NUS National University of Singapore | School of Computing

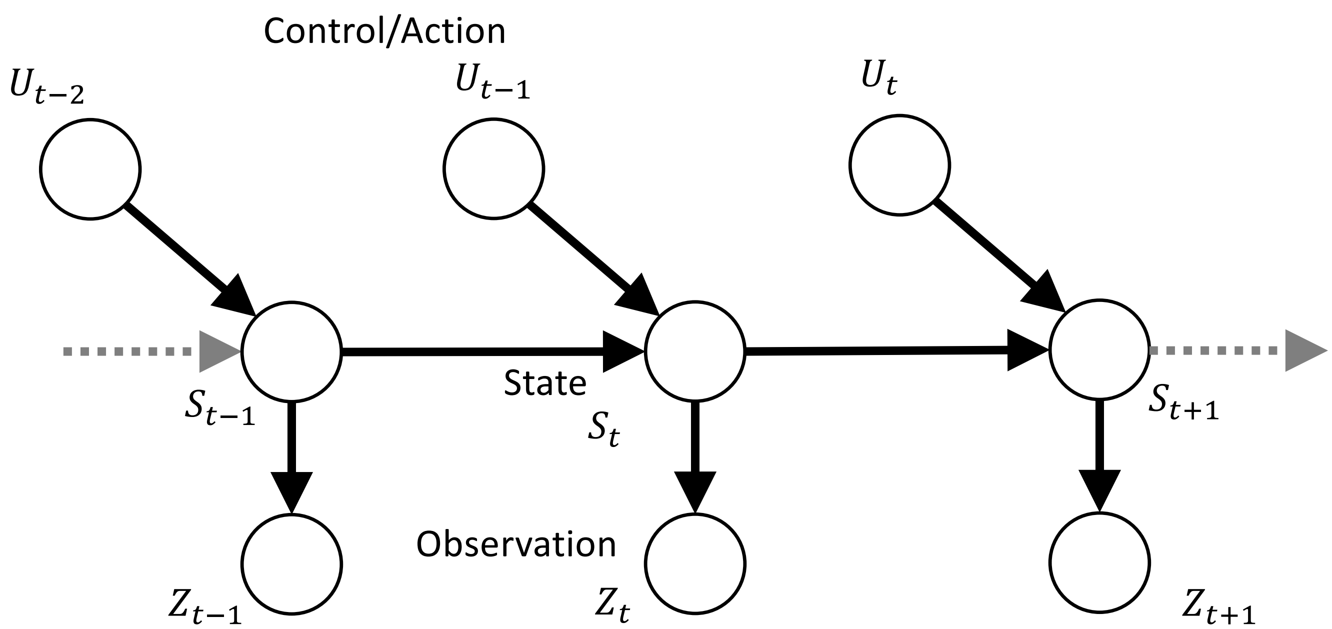
Markov Decision Processes

Probabilistic Model and Problem Setup

Our setup



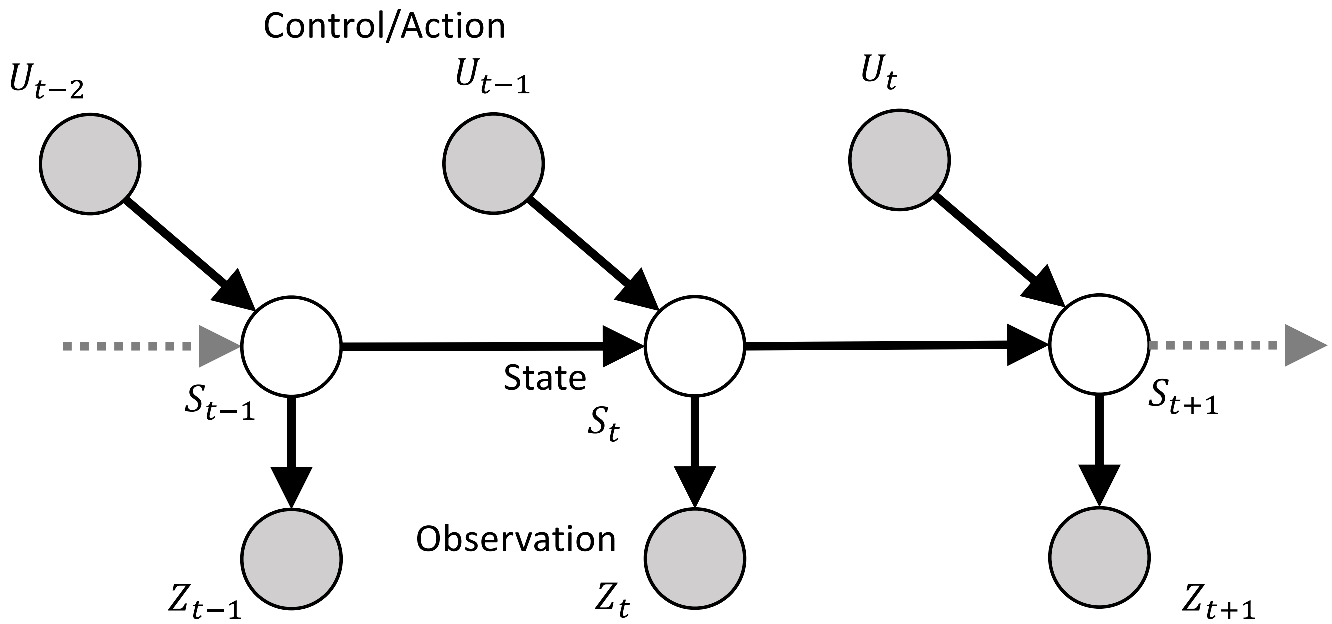
Model of the environment



$$S_{t+1} \perp S_{t-1} \mid S_t$$

"Markov Assumption"

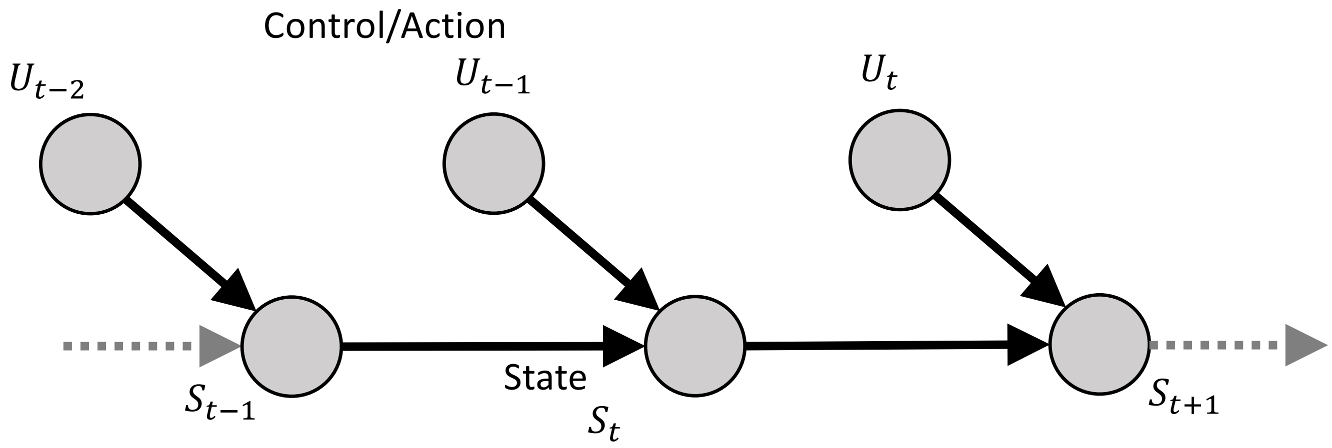
Model of the environment



$$S_{t+1} \perp S_{t-1} \mid S_t$$

"Markov Assumption"

Model of the environment

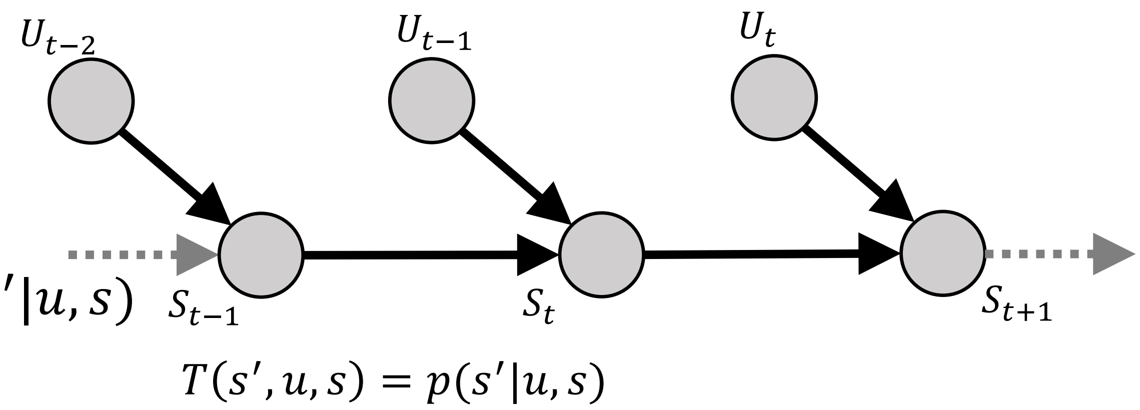


"Fully Observable"

$$S_{t+1} \perp S_{t-1} \mid S_t$$

"Markov Assumption"

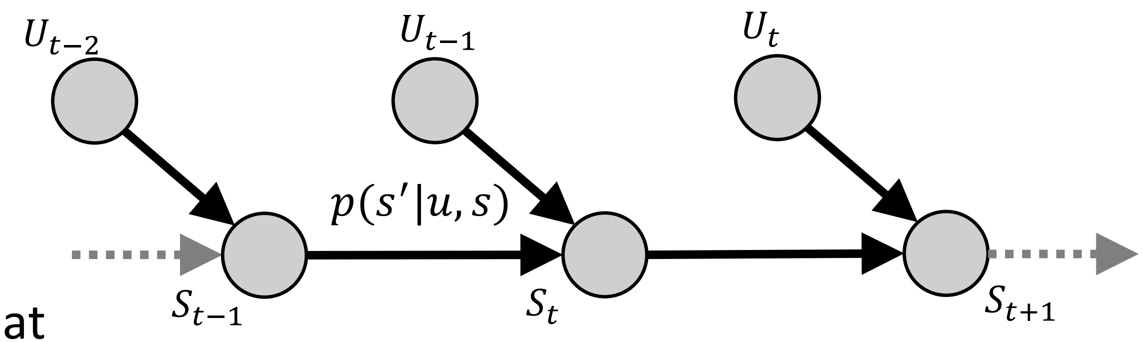
Markov Decision process (MDP)



- A tuple (S, U, T, R, γ)
  - States S
  - Actions/Controls U
  - Transitions $T(s', u, s) = p(s'|u, s)$
  - Reward $R(s, u)$
  - Discount γ where $0 \leq \gamma \leq 1$
- Actions are generated by *policy* $u = \pi(s)$
- Can also have reward functions $R(s)$ or $R(s', u, s)$
  - **Exercise:** draw the reward node in the graph above.

Problem Definition

- **Goal:** Given MDP (S, U, T, R, γ)



- Find the optimal policy $\pi^*(s)$ that maximizes the expected discounted sum of rewards:

$$\arg\max_{\pi} \mathbb{E}_{\tau}\left[\sum_{t=0}^{T-1} \gamma^t R\big(s_t, u = \pi(s_t)\big)\right]$$

$H = T - 1$ is the "*planning horizon*", τ is a r.v. representing trajectories of length H

Solving MDPs

Value and Policy Iteration

Solution via dynamic programming

Key idea: Recursively compute the utility of actions/controls

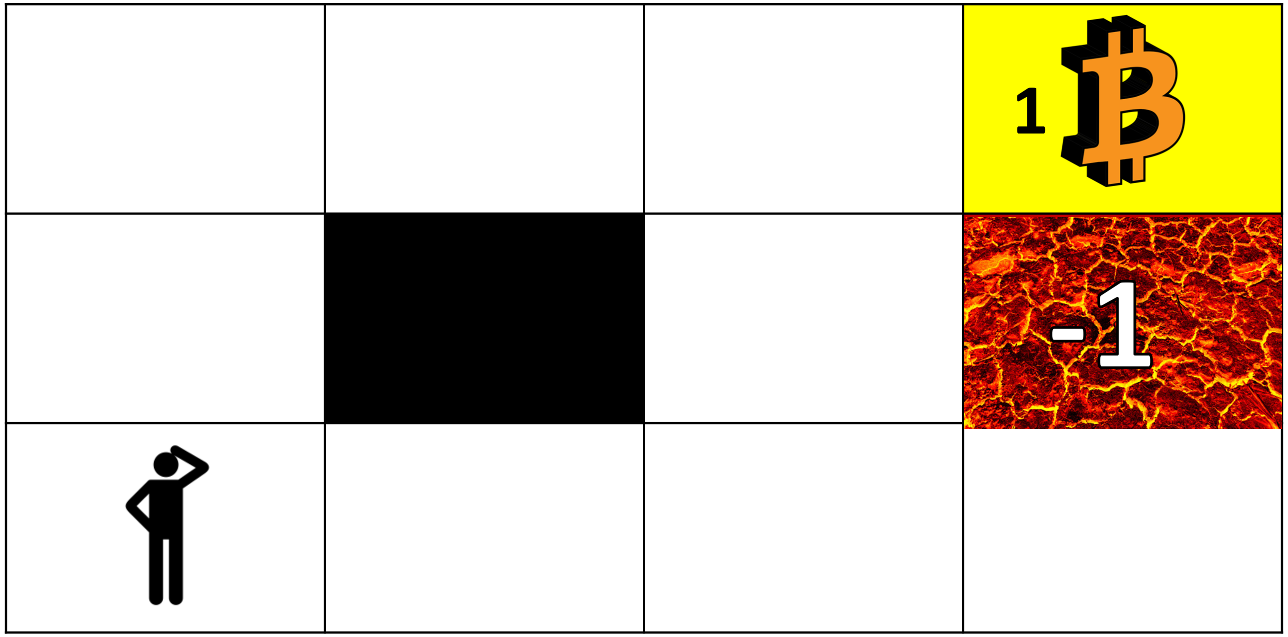
To begin, assume:

- finite state space
- finite action space
- finite planning horizon.

- Two methods:
- Value Iteration
- Policy iteration

Grid World Example

Discount factor of 0.9
Can move UP, DOWN, LEFT or RIGHT
Action success probability of 0.8.

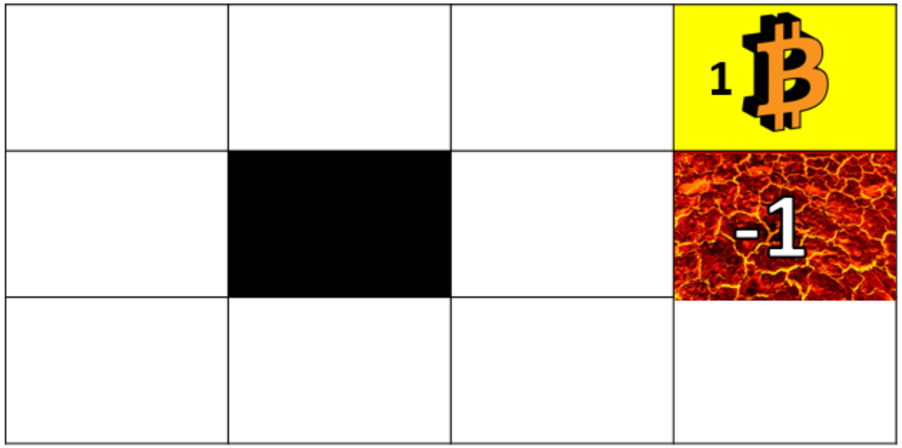


$R(s, a)$

Value function

- Value function $V_k^{\pi}(s)$: how "good" it is for an agent following policy π to be in state s given k steps remaining:

$$V_k^{\pi}(s) = \mathbb{E}_{\tau|s}\left[\sum_{t=0}^{k-1} \gamma^t R\big(s_t, u = \pi(s_t)\big)\right] \text{ for all } s \in S$$



Example:

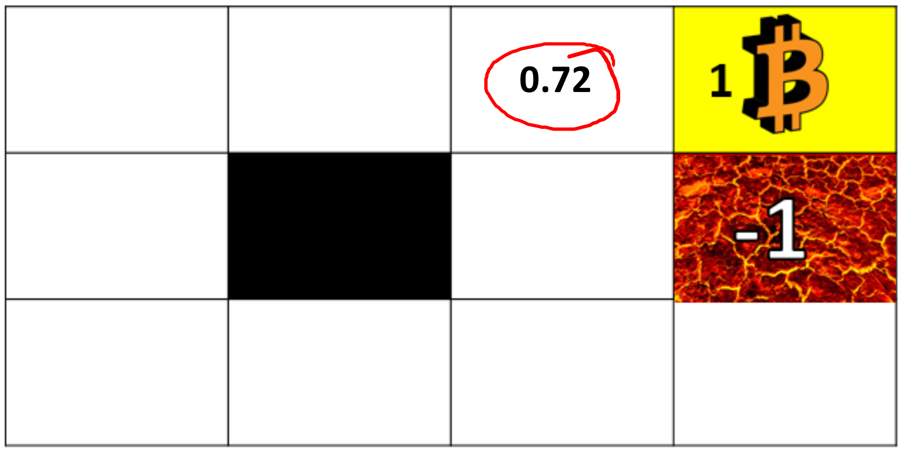
If $k = 1$, 1 time step left, $V_1^{\pi}(s) = R(s, \pi(s))$

If $k = 2$, 2 time steps left, $V_2^{\pi}(s)$?

Value function

- Value function $V_k^\pi(s)$: how "good" it is for an agent following policy π to be in state s given k steps remaining:

$$V_k^\pi(s) = \mathbb{E}_{\tau|s}\left[\sum_{t=0}^{k-1} \gamma^t R\big(s_t, u = \pi(s_t)\big)\right] \text{ for all } s \in S$$



Example:
If $k = 1$, 1 time step left, $V_1^\pi(s) = R(s, \pi(s))$

If $k = 2$, 2 time steps left,

$$V_2^\pi(s) = \underbrace{R\big(s, \pi(s)\big)}_{0} + \underbrace{\gamma}_{0.9} \sum_{s'} \underbrace{p(s'|s, \pi(s))}_{0.8}\underbrace{R(s', \pi(s))}_{1}$$

$= 0.72$

Value function

- Value function $V_k^\pi(s)$: how "good" it is for an agent following policy π to be in state s given k steps remaining:

$$V_k^\pi(s) = \mathbb{E}_{\tau|s}\left[\sum_{t=0}^{k-1} \gamma^t R\big(s_t, u = \pi(s_t)\big)\right] \text{ for all } s \in S$$

- Let's rewrite:

$$\begin{aligned} V_k^\pi(s) &= \mathbb{E}_{\tau|s}\left[\sum_{t=0}^{k-1} \gamma^t R\big(s_t, \pi(s_t)\big)\right] \\ &= R\big(s, \pi(s)\big) + \sum_{s'} p(s'|s, \pi(s))\, \gamma \mathbb{E}_{\tau|s'}\left[\sum_{l=0}^{k-2} \gamma^l R(s_l, \pi(s_l))\right] \\ &= \underbrace{R\big(s, \pi(s)\big)}_{\textbf{Immediate Reward}} + \underbrace{\gamma \sum_{s'} p(s'|s, \pi(s)) V_{k-1}^\pi(s')}_{\textbf{Future Rewards}} \end{aligned}$$

Value function

- Value function $V_k^\pi(s)$: how "good" it is for an agent following policy π to be in state s given k steps remaining:

$$V_k^\pi(s) = \mathbb{E}_{\tau|s}\left[\sum_{t=0}^{k-1} \gamma^t R\big(s_t, u = \pi(s_t)\big)\right] \text{ for all } s \in S$$

- Let's rewrite:

$$\boxed{V_k^\pi(s) = R\big(s, \pi(s)\big) + \gamma \sum_{s'} p(s'|s, \pi(s)) V_{k-1}^\pi(s')}$$

"Bellman update or Bellman backup"

Optimal value function

- Value function under a policy π

$$V_k^{\pi}(s) = R\big(s, \pi(s)\big) + \gamma \sum_{s'} p(s'|s, \pi(s)) V_{k-1}^{\pi}(s')$$

- Optimal Value function:

$$\boxed{V_k^{*}(s) = \max_{\mathrm{u}}\ R(s, u) + \gamma \sum_{s'} p(s'|s, u) V_{k-1}^{*}(s')}$$

- Optimal Policy:

$$\boxed{\pi_k^{*}(s) = \operatorname*{argmax}_{\mathrm{u}}\ R(s, u) + \gamma \sum_{s'} p(s'|s, u) V_{k-1}^{*}(s')}$$

Value iteration (finite horizon)

$$V_k^*(s) = \max_{\mathrm{u}} \; R(s,u) + \gamma \sum_{s'} p(s'|s,u) V_{k-1}^*(s')$$

Initialize $V_0^*(s) = 0$ for all $s \in S$

For $k = 0, \ldots, T-1$

For all $s \in S$

$$V_{k+1}^*(s) = \max_{\mathrm{u}} \; R(s,u) + \gamma \textstyle\sum_{s'} p(s'|s,u) V_k^*(s')$$

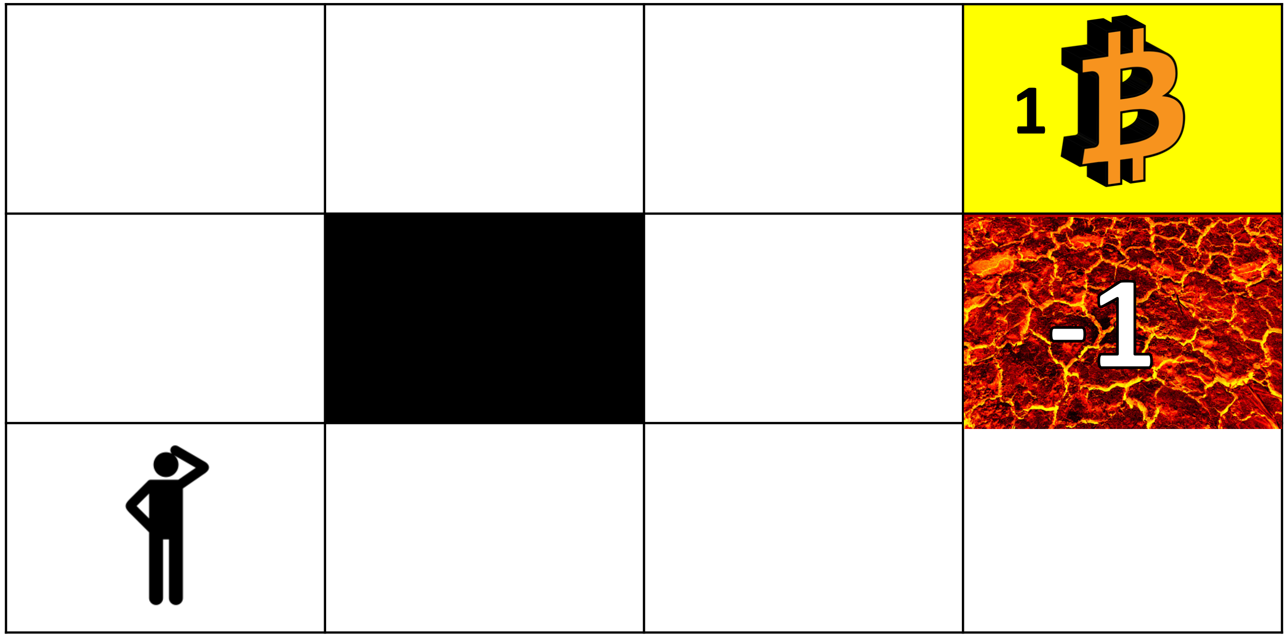
$$\pi_{k+1}^*(s) = \operatorname*{argmax}_{\mathrm{u}} \; R(s,u) + \gamma \textstyle\sum_{s'} p(s'|s,u) V_k^*(s')$$

$O(|S|^2|A|)$

$O(T|S|^2|A|)$

Jupyter notebook

Discount factor of 0.9
Can move UP, DOWN, LEFT or RIGHT
Action success probability of 0.8.



Value iteration (infinite horizon)

- If the planning horizon $H = \infty$,
- The following equation holds for the optimal value function:

$$\boxed{V^*(s) = \max_{\mathrm{u}} R(s,u) + \gamma \sum_{s'} p(s'|s,u) V^*(s')}$$

"Bellman optimality equation"

- Only a *single* value function (compare against finite horizon case) for all time steps.

Value iteration (infinite horizon)

$$V^*(s) = \max_{\mathrm{u}} R(s,u) + \gamma \sum_{s'} p(s'|s,u)V^*(s')$$

Initialize $V^*(s) = 0$ for all $s \in S$

While not converged

For all $s \in S$

$$V^*(s) = \max_{\mathrm{u}} R(s,u) + \gamma \sum_{s'} p(s'|s,u)V^*(s')$$

Compute policy:

$$\pi^*(s) = \operatorname*{argmax}_{\mathrm{u}} R(s,u) + \gamma \sum_{s'} p(s'|s,u)V^*(s')$$

Value iteration converges to V^* for the discounted infinite horizon problem.

Solution via dynamic programming

Key idea: Recursively compute the utility of actions/controls

To begin, assume:

- finite state space
- finite action space
- finite planning horizon.

- Two methods:
- Value Iteration
- Policy iteration

Policy evaluation

- Recall the value of a policy in the finite horizon case:

$$V_k^{\pi}(s) = R\big(s, \pi(s)\big) + \gamma \sum_{s'} p(s'|s, \pi(s)) V_{k-1}^{\pi}(s')$$

- The infinite horizon case:

$$V^{\pi}(s) = R\big(s, \pi(s)\big) + \gamma \sum_{s'} p(s'|s, \pi(s)) V^{\pi}(s')$$

Policy improvement

- How can we improve a given policy π?
- **Key Idea:** Choose a control u in current time step, and follow π thereafter
- Define:

$$Q^{\pi}(s,u) = R(s,u) + \gamma \sum_{s'} p(s'|s,u)V^{\pi}(s')$$

- **Policy Improvement Theorem:** Given deterministic policies π and π' s.t.

$$\forall s \in S \quad Q^{\pi}\big(s,\pi'(s)\big) \geq V^{\pi}(s)$$

- Then

$$\forall s \in S \;\; V^{\pi'}(s) \;\geq V^{\pi}(s)$$

Policy improvement

- Choose a greedy policy π'

$$\pi'(s) = \operatorname{argmax}_u Q^\pi(s,u)$$
$$= \operatorname{argmax}_u \underbrace{R(s,u) + \gamma \sum_{s'} p(s'|s,u)V^\pi(s')}_{\substack{\text{one-step lookahead} \\ \text{(based on } V^\pi\text{)}}}$$

- **Intuition:** takes action that is best after *one-step lookahead*

Policy iteration

Policy iteration is guaranteed to converge.
At convergence, the policy and value function are both optimal.

Evaluation Improvement

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \ldots \xrightarrow{I} \pi^* \xrightarrow{E} V^*$$

- **Policy Evaluation ($\boldsymbol{E}$)**
  - Iterate until convergence:

$$V_k^{\pi_i}(s) = R\big(s, \pi_i(s)\big) + \gamma \sum_{s'} p(s'|s, \pi_i(s)) V_{k-1}^{\pi_i}(s')$$

- **Policy Improvement ($\boldsymbol{I}$)**
  - Compute:

$$\pi_{i+1}(s) = \mathrm{argmax}_u\, R(s, u) + \gamma \sum_{s'} p(s'|s, u) V^{\pi_i}(s')$$

- Repeat E and I until convergence

Generalized policy iteration

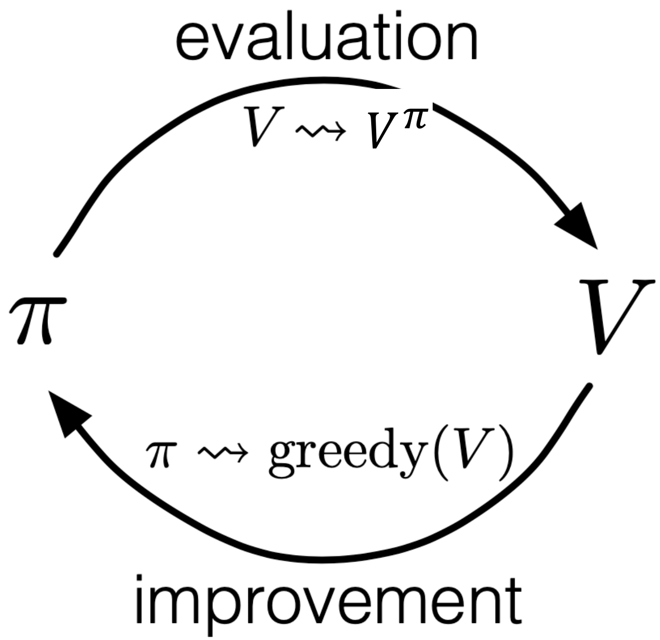


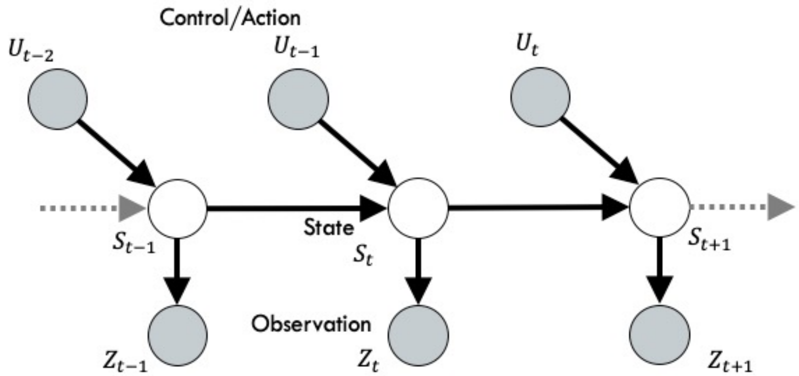
Image from: Introduction to Reinforcement learning, Chapter 4.

To Explore: Extensions

- Large/Continuous State Spaces
- Partial Observability
- Large/Continuous Action Spaces
- Partially-Specified or Mis-specified Models
- Function approximation

Partially-observable MDP (POMDP)

- A tuple $(S, U, T, R, Z, O, \gamma)$
  - States S
  - Actions/Controls U
  - Transitions $T(s', u, s) = p(s'|u, s)$
  - Reward $R(s, u)$
  - Observations Z
  - Observation function $O(z, s) = p(z|s)$
  - Discount γ
- Variants: also can have $O(z, s, u) = p(z|s, u)$

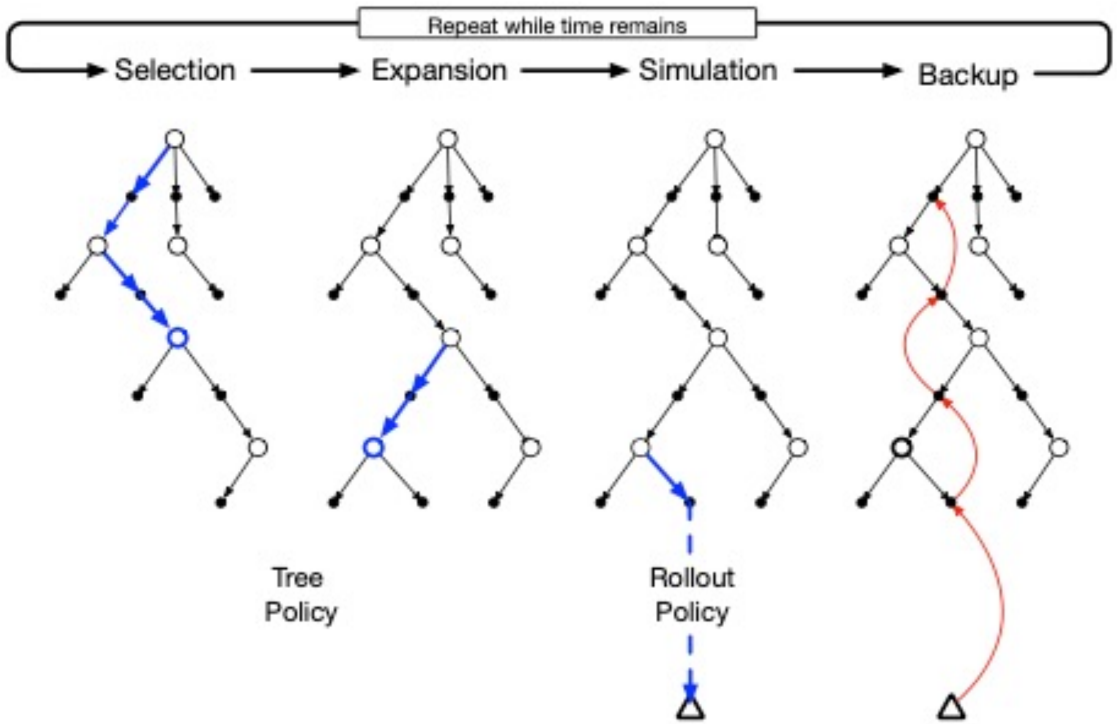


Solving POMDPs: methods

- Exact Methods:
  - Value Iteration
  - Enumeration Algorithm [Monahan, 82]
  - One-pass [Sondik, 71]
  - Witness [Litmann et al., 94]
  - Incremental Pruning [Zhang and Liu, 96]
- Approximate solutions:
  - Point-based Value Iteration (PBVI) [Pinneau, Gordon, Thrun, 2003]
  - SARSOP [Kurniawati, Hsu, & Lee. 2008].
  - Partially Observable Monte-Carlo Planning (POMCP) [Silver & Veness, 2010]
  - DESPOT [Somani, Ye, Hsu, & Lee, 2013].

POMCP

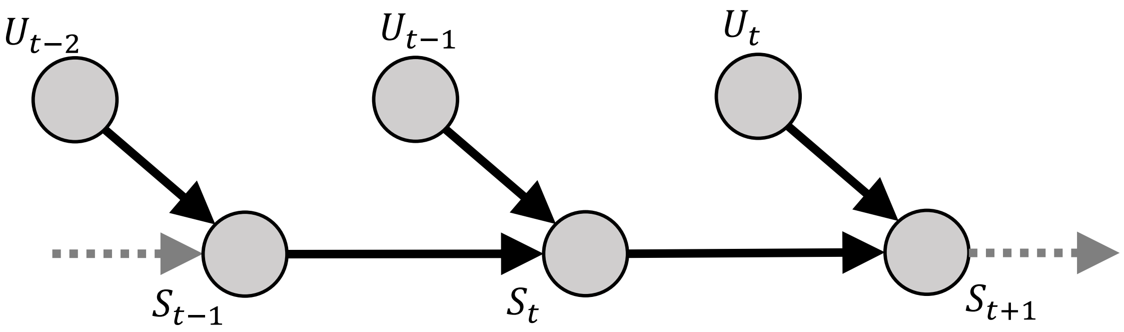
- Based on **Monte-Carlo Tree Search** (MCTS)
- More info in:
- [Browne et al, 2012]
  - https://core.ac.uk/download/pdf/9589938.pdf
- [Silver & Veness, 2010]:
  - https://papers.nips.cc/paper/4031-monte-carlo-planning-in-large-pomdps



Linear Quadratic Regulator

Control for Linear Dynamical Systems with Quadratic Cost

Markov Decision process (MDP)

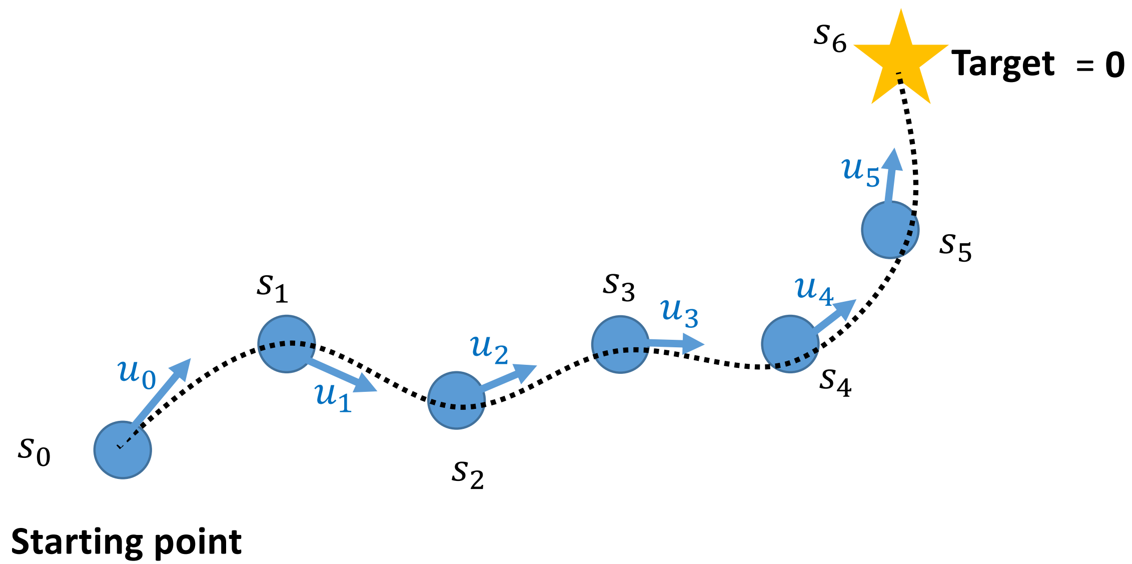


$T(s', u, s) = p(s'|u, s)$

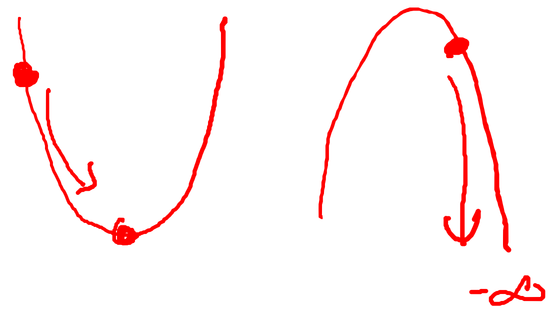
What if $s \in S$ is **continuous**?

The problem

The problem



Continuous state spaces



Additional Assumptions:

Linear System:

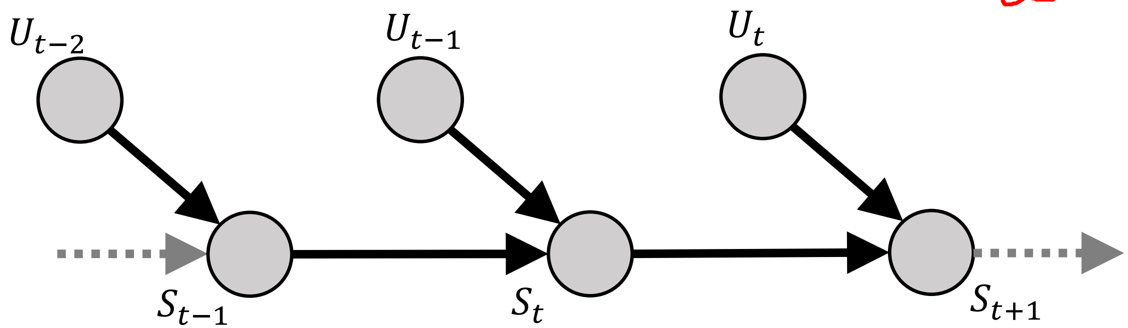
$$s_{t+1} = As_t + Bu_t$$

Quadratic Cost ("negative reward")

$$g(s_t, u_t) = s_t^\top Q s_t + u_t^\top R u_t$$

where:

- $Q \succ 0$ and $R \succ 0$
  - Q and R are symmetric Positive Definite (PD)



$$T(s', u, s) = p(s'|u, s)$$

Reminder: **Positive Definite:** a square matrix $X \in \mathbb{R}^{d\times d}$ is PD iff $\forall\, z \in \mathbb{R}^d$, if $z \neq \mathbf{0}, t$ then $z^\top X z > 0$.

$xz^2 > 0$

Linear quadratic regulator

Additional Assumptions:

Linear System:

$$s_{t+1} = As_t + Bu_t$$

Quadratic Cost ("negative reward")

$$g(s_t, u_t) = s_t^\top Q s_t + u_t^\top R u_t$$

where:

$Q \succ 0$ and $R \succ 0$

- Q and R are symmetric Positive Definite (PD)

- **Key idea:** To obtain policy, apply value iteration to this setting.
- Will see:
  - Assumptions keep things tractable.

Linear Quadratic Regulator

Value iteration (in terms of "cost to go" J):

$$J_{i+1}(s) = \min_{u} \boxed{g(s,u)} + \boxed{\sum_{s'} p(s'|s,u) J_i(s')}$$

For LQR, substitute in our assumptions:

$$J_{i+1}(s) = \min_{u} \boxed{s^\top Q s \quad + u^\top R u} + \boxed{J_i(As \quad + Bu \quad)}$$

Value iteration solution

Initialize $P_0 = 0$

For $i = 1,2,3, \ldots, H$ ("backward pass")

$$K_i = -(R + B^\top P_{i-1} B)^{-1} B^\top P_{i-1} A$$

$$P_i = Q + K_i^\top R K_i + (A + B K_i)^\top P_{i-1} (A + B K_i)$$

Optimal policy for i-step horizon: $\pi_i(s) = K_i s$

Cost-to-go: $J_i(s) = s^\top P_i s$

For $t = 0,1,2,3, \ldots, H-1$ ("forward pass")

$$u_t = \pi_{H-t}(s_t) = K_{H-t} s_t$$

$$s_{t+1} = f(s_t, u_t)$$

Apply value iteration to LQR

$$\boxed{J_{i+1}(s) = \min_u s^\top Q s \quad + u^\top R u \quad + J_i(As + Bu \quad)}$$

Let $J_0(s) = \underline{s^\top P_0 s}$

Let's write down J_1:

$$J_1(s) = \min_u s^\top Q s \quad + u^\top R u \quad + J_0(As \quad + Bu \quad)$$
$$= \min_u s^\top Q s \quad + u^\top R u \quad + (\underline{As \quad + Bu} \quad)^\top \underline{P_0}(\underline{As \quad + Bu} \quad)$$

How can we solve this minimization problem?

Apply value iteration to LQR – (1)

$$J_{i+1}(s) = \min_u s_t^\top Q s_t + u_t^\top R u_t + J_i(As_t + Bu_t)$$

Let $J_0(s) = s^\top P_0 s$

Let's write down J_1:

$$\begin{aligned} J_1(s) &= \min_u s^\top Q s + u^\top R u + J_0(As + Bu) \\ &= \min_u s^\top Q s + u^\top R u + (As + Bu)^\top P_0 (As + Bu) \end{aligned}$$

Set gradient wrt u equal to zero

$$\nabla_u \left[s^\top Q s + u^\top R u + (As + Bu)^\top P_0 (As + Bu) \right] = 0$$

Solving for u:

$$u = \underbrace{-(R + B^\top P_0 B)^{-1} B^\top P_0 A}_{K_1} s$$

Apply value iteration to LQR – (2)

$$J_{i+1}(s) = \min_u s_t^\top Q s_t + u_t^\top R u_t + J_i(As_t + Bu_t)$$

$$J_1(s) = \min_u s^\top Q s \quad + u^\top R u \quad + (As \quad + Bu \quad)^\top P_0 (As \quad + Bu \quad)$$

Substitute $u = K_1 s$ into $J_1(s)$:

$$\boxed{J_1(s) = s^\top P_1 s}$$

$J_0(s) = s^\top P_0 s$

where

$$P_1 = Q + K_1^\top R K_1 + (A + BK_1)^\top P_0 (A + BK_1)$$
$$K_1 = -(R + B^\top P_0 B)^{-1} B^\top P_0 A$$

What can we notice about the form of J_1?

Same quadratic form as $J_0 = s^\top P_0 s$. Can repeat process for $J_2, J_3, \ldots$

Value iteration solution

Initialize $P_0 = 0$

For $i = 1,2,3, \ldots, H$ ("backward pass")

$$K_i = -(R + B^\top P_{i-1} B)^{-1} B^\top P_{i-1} A$$

$$P_i = Q + K_i^\top R K_i + (A + BK_i)^\top P_{i-1}(A + BK_i)$$

Optimal policy for i-step horizon: $\pi_i(s) = K_i s$

Cost-to-go: $J_i(s) = s^\top P_i s$

For $t = 0,1,2,3, \ldots, H-1$ ("forward pass")

$$u_t = \pi_{H-t}(s_t) = K_{H-t} s_t$$

$$s_{t+1} = f(s_t, u_t)$$

Extensions to Basic LQR

- Affine transitions: $s_{t+1} = As_t + Bu_t + c$
- Linear Stochastic dynamics: $s_{t+1} = As_t + Bu_t + \epsilon_t$ where $\epsilon_t \sim p(v)$
  - w_t has zero mean and independent.
  - Linear Quadratic Gaussian (LQG) $\epsilon_t \sim \mathcal{N}(v|0, \sigma_n^2)$
- Observation functions: $y_t = Ds_t + b$
- Linear Time-Varying (LTV) systems: $s_{t+1} = A_t s_t + B_t u_t$
- Non-linear systems with nonlinear cost
- Trajectory following for non-linear systems
- Bounded controls
- etc.

Linear time varying (LTV) systems

Linear Time Varying System:

$$s_{t+1} = A_t s_t + B_t u_t$$
$$g(s_t, u_t) = s_t^\top Q_t s_t + u_t^\top R_t u_t$$

Similar solution:

Initialize $P_0 = 0$

For $i = 1,2,3, \ldots, H$ ("backward pass")

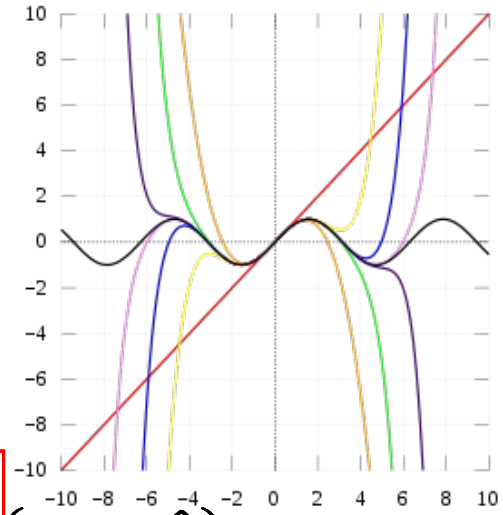
$$K_i = -\left(R_{H-i} + B_{H-i}{}^\top P_{i-1} B_{H-i}\right)^{-1} B_{H-i}^\top P_{i-1} A_{H-i}$$
$$P_i = Q_{H-i} + K_i^\top R_{H-i} K_i + (A_{H-i} + B_{H-i} K_i)^\top P_{i-1} (A_{H-i} + B_{H-i} K_i)$$

Optimal policy for i-step horizon: $\pi(s) = K_i s$

Cost-to-go: $J_i(s) = s^\top P_i s$

Do "forward pass"

Non-Linear Transition Systems



Nonlinear System:

$$s_{t+1} = f(s_t, u_t)$$

Use Taylor expansions to linearize around $\hat{s}$ and $\hat{u}$

$$s_{t+1} \approx f(\hat{s}, \hat{u}) + \underbrace{\boxed{\frac{\partial f}{\partial s}(\hat{s}, \hat{u})}}_{\boldsymbol{A}}(s_t - \hat{s}) + \underbrace{\boxed{\frac{\partial f}{\partial u}(\hat{s}, \hat{u})}}_{\boldsymbol{B}}(u_t - \hat{u})$$

So, $s_{t+1} - \hat{s}_{t+1} \approx A(s_t - \hat{s}) + B(u_t - \hat{u})$

Define $z_t = s_t - \hat{s}$ and $v_t = u_t - \hat{u}$

$$z_{t+1} = Az_t + Bv_t$$
$$g(z_t, u_t) = z_t^\top Q z_t + v_t^\top R v_t$$

Can use standard LQR!
Exercise: Have to transform from v_t to u_t. How?

Image: https://commons.wikimedia.org/wiki/File:Exp_series.gif

Non-linear transition and cost

Nonlinear System:

$$\min_{u_{0:H}} \sum_t g(s_t, u_t) \text{ such that } s_{t+1} = f(s_t, u_t)$$

How can we solve this?

Key Idea:

- Iteratively approximate (f, g) and run LQR to solve for optimal policy
- Approximate dynamics f using 1st order Taylor expansion (linear)
- Approximate cost g using 2nd order Taylor expansion (quadratic)

Iterative LQR: overview

Iterative LQR (iLQR)

Initialize policy π^0

For $i = 0,1,2, \ldots$

Execute π^i with $f\left(s_t^i, u_t^i\right)$ to get sequence $\left(s_t^i, u_t^i\right)_{0:H} = (s_0^i, u_0^i), (s_1^i, u_1^i), \ldots, (s_H^i, u_H^i)$

Approximate f with 1st order Taylor expansion around $\left(s_t^i, u_t^i\right)_{0:H}$

Approximate g with 2nd order Taylor expansion $\left(s_t^i, u_t^i\right)_{0:H}$

Compute LQR using approximate system to get π^{i+1}

LQR/iLQR: more information

- [Li and Todorov, 2004] https://homes.cs.washington.edu/~todorov/papers/LiICINCO04.pdf
- Peter Abeel's CS287 Lecture 5: https://people.eecs.berkeley.edu/~pabbeel/cs287-fa19/
- Florian Shkurti's Lecture 2: http://www.cs.toronto.edu/~florian/courses/imitation_learning/lectures/Lecture2.pdf
- Katerina Fragkiadaki iLQR slides: https://katefvision.github.io/katefSlides/RECITATIONtrajectoryoptimization_katef.pdf
- iLQR tutorial (Stanford): http://roboticexplorationlab.org/papers/iLQR_Tutorial.pdf

To explore: Embed-to-Control (E2C)

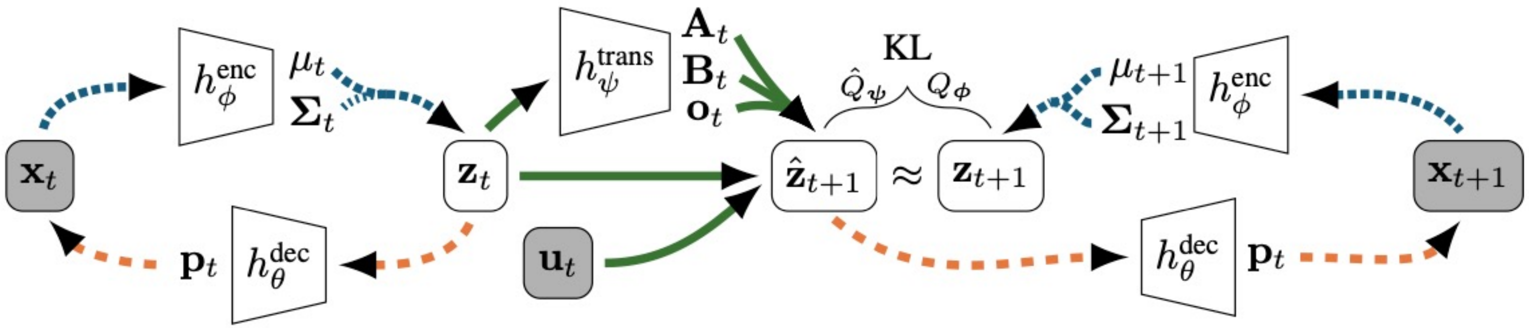


Figure 1: The information flow in the E2C model. From left to right, we encode and decode an image $\mathbf{x}_t$ with the networks $h_{\boldsymbol{\phi}}^{\text{enc}}$ and $h_{\boldsymbol{\theta}}^{\text{dec}}$, where we use the latent code $\mathbf{z}_t$ for the transition step. The $h_{\boldsymbol{\psi}}^{\text{trans}}$ network computes the local matrices $\mathbf{A}_t, \mathbf{B}_t, \mathbf{o}_t$ with which we can predict $\hat{\mathbf{z}}_{t+1}$ from $\mathbf{z}_t$ and $\mathbf{u}_t$. Similarity to the encoding $\mathbf{z}_{t+1}$ is enforced by a KL divergence on their distributions and reconstruction is again performed by $h_{\boldsymbol{\theta}}^{\text{dec}}$.

Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, Martin Riedmiller
https://arxiv.org/abs/1506.07365

Inference and Control

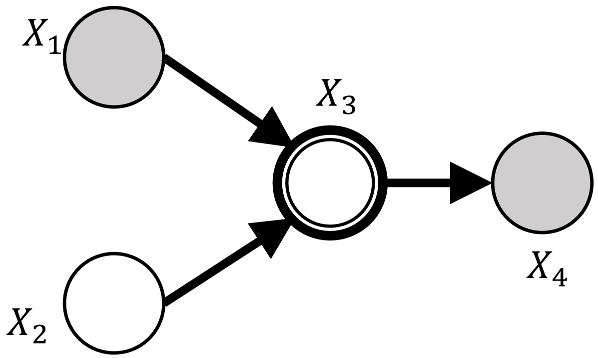
Bridging Probabilistic Inference and Control – A First Step

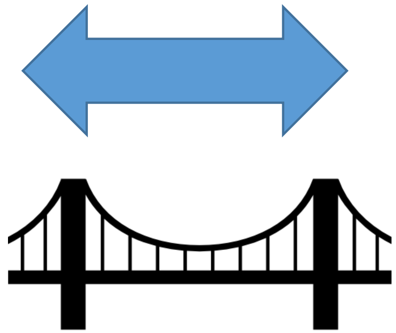
Take Note:

- Assumptions (for simplicity of exposition):
  - Finite-time horizon
  - Discount factor $\gamma = 1$

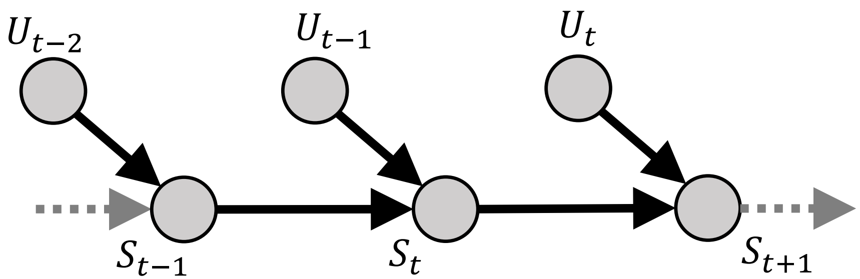
Inference v.s. planning/control

Inference





Planning/Control



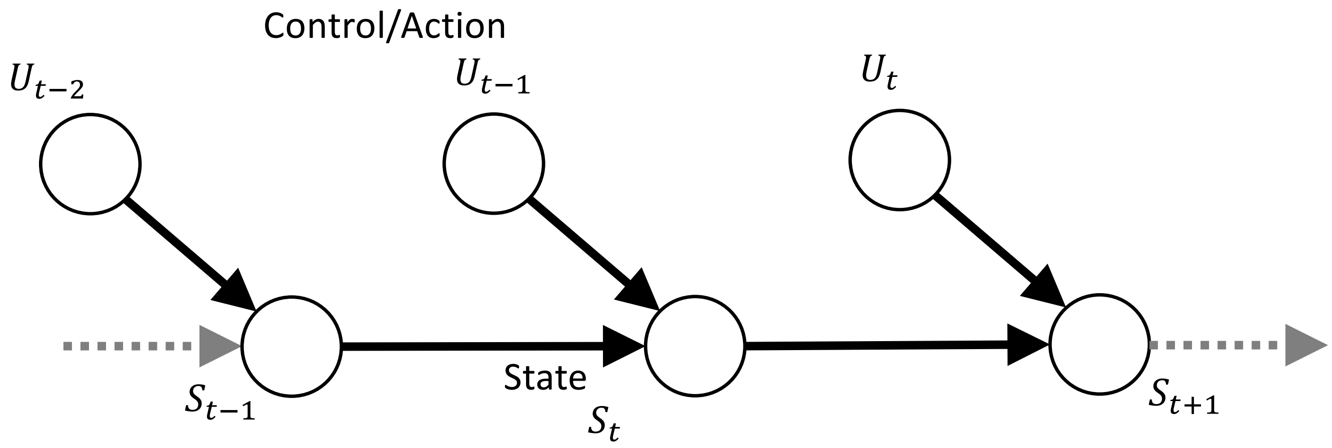
Our goal is to calculate $p(X_F|X_E)$ for arbitrary subsets E and F.

Our goal is to find the optimal policy $\pi^*(s)$

High Level Structure

- A Probabilistic Graphical Model (PGM) for Control
  - Performing inference to obtain a policy
    - Backward Messages
  - "Structured" variational inference to obtain a constrained policy

Model of the environment



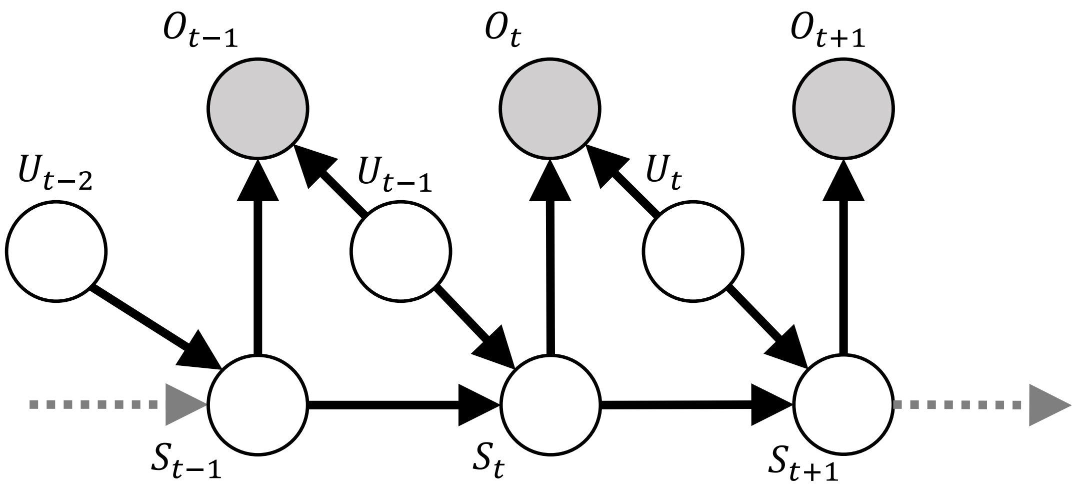
"Fully Observable"

$$S_{t+1} \perp S_{t-1} \mid S_t$$

"Markov Assumption"

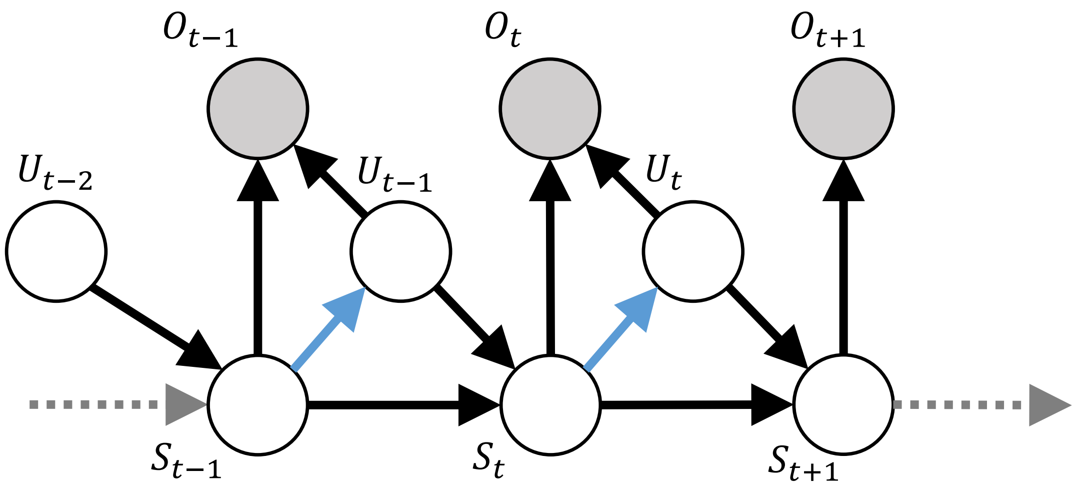
Optimality variables

$p(O_t = 1 | s_t, u_t) = \exp(r(s_t, u_t))$ where $r(s_t, u_t) \leq 0$



Optimality variables

$$p(O_t = 1|s_t, u_t) = \exp(r(s_t, u_t)) \text{ where } r(s_t, u_t) \leq 0$$



$p(u_t|s_t)$ "Prior Policy"; assume Uniform.

Unnormalized Posterior over trajectories

Condition on $O_t = 1$ for all $t \in \{1, \ldots, T\}$

$$p(\tau|O_{1:T}) \propto p(\tau, O_{1:T}) = p(s_1) \prod_{t=1}^{T} p(O_t = 1|s_t, u_t) p(s_{t+1}|s_t, u_t)$$

Since $p(O_t = 1|s_t, u_t) = \exp(r(s_t, u_t))$

$$p(\tau|O_{1:T}) \propto p(s_1) \prod_{t=1}^{T} \exp(r(s_t, u_t))\, p(s_{t+1}|s_t, u_t)$$

Grouping,

$$p(\tau|O_{1:T}) \propto \left[p(s_1) \prod_{t=1}^{T} p(s_{t+1}|s_t, u_t) \right] \exp\left(\sum_{t=1}^{T} r(s_t, u_t) \right)$$

Unnormalized Posterior over trajectories

$$p(\tau | O_{1:T}) \propto \underbrace{\left[p(s_1) \prod_{t=1}^{T} p(s_{t+1} | s_t, u_t) \right]}_{\text{Probability of trajectory according to } \textit{dynamics}} \underbrace{\exp\left(\sum_{t=1}^{T} r(s_t, u_t) \right)}_{\text{Exponential of Total Reward along trajectory}}$$

Question: Consider the case of *deterministic* dynamics.
What does the equation above reduce to?

Unnormalized Posterior over trajectories

$$p(\tau|o_{1:T}) \propto \underbrace{I[p(\tau) \neq 0]}_{\text{Constant for all feasible } \tau} \underbrace{\exp\left(\sum_{t=1}^{T} r(s_t, u_t)\right)}_{\text{Exponential of Total Reward along trajectory}}$$

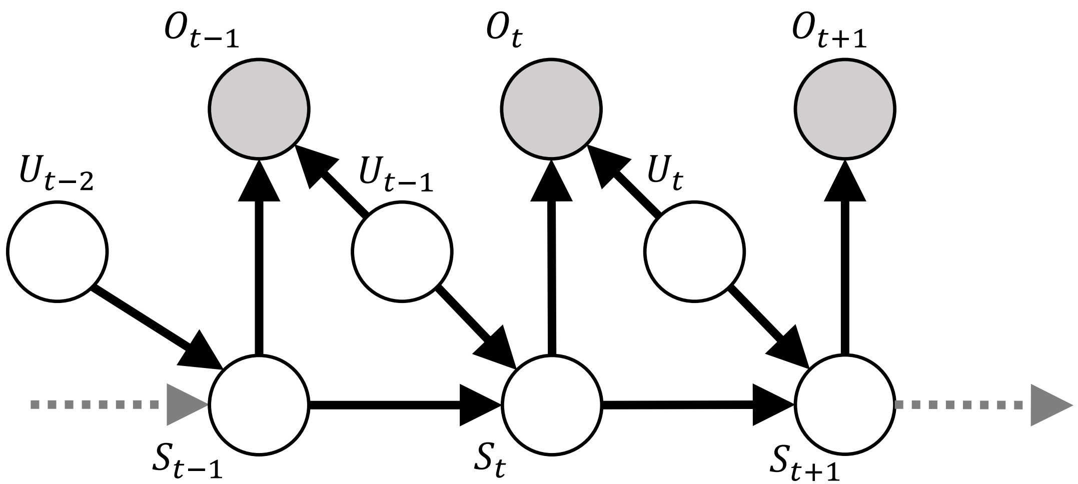
Special case of deterministic dynamics. For planning, we set $p(s_1) = \delta(s_1)$ and perform MAP inference.

Obtaining the policy as inference

- **Show:** Obtain optimal policy $p(u_t|s_t, O_{t:T})$ using inference
  - Note: drop explicit notation $O_{t:T} = 1$
- Apply Sum-Product / Belief-Propagation algorithm

Sum-product algorithm

- **Goal:** Compute $p(u_t|s_t, O_{t:T})$
- In HMMs, remember the Forward-Backward Algorithm



Backward messages

Goal: Compute $p(u_t|s_t, O_{t:T})$

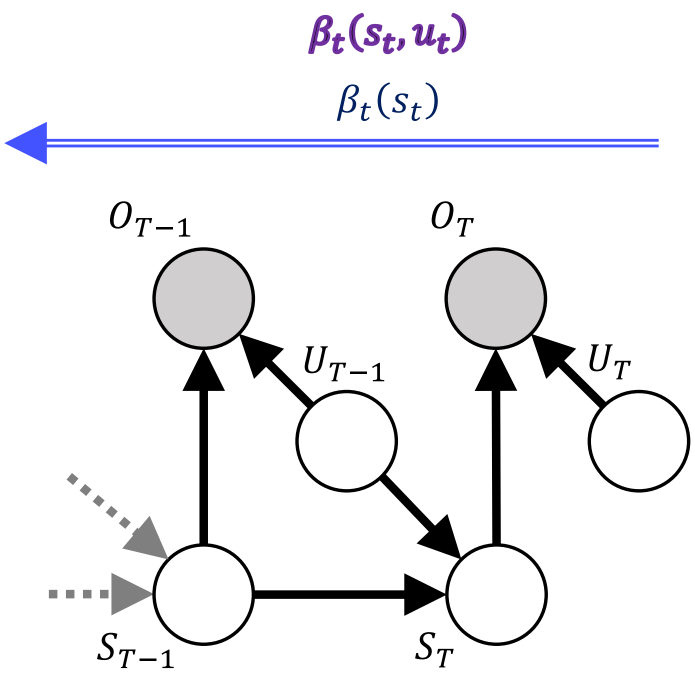
Define the backward message:

$$\boxed{\beta_t(s_t, u_t) = p(O_{t:T}|s_t, u_t)}$$

Then,

$$\beta_t(s_t) = p(O_{t:T}|s_t) = \sum_{u_t} p(O_{t:T}|s_t, u_t)p(u_t|s_t)$$

$$= \sum_{u_t} \beta_t(s_t, u_t)\underbrace{p(u_t|s_t)}_{\substack{\text{Action prior}(\textbf{\textit{not}}\ \textit{policy})\\ \textbf{\textit{Assume Uniform}}}}$$



Backward messages

$$\beta_t(s_t, u_t) = p(O_{t:T} | s_t, u_t)$$
$$\beta_t(s_t) = p(O_{t:T} | s_t)$$

Goal: Compute $p(u_t | s_t, O_{t:T})$

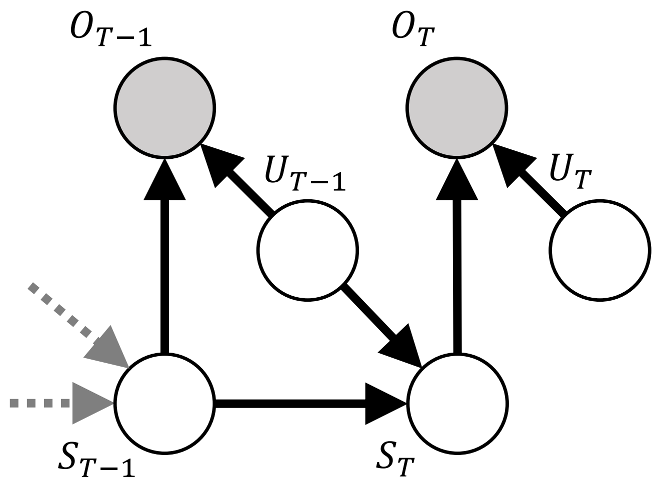
$$p(u_t | s_t, O_{t:T}) = \frac{p(s_t, u_t | O_{t:T})}{p(s_t | O_{t:T})}$$ (conditional probability)

$$= \frac{p(O_{t:T} | s_t, u_t) p(u_t | s_t) p(s_t)}{p(O_{t:T} | s_t) p(s_t)}$$ (Bayes Rule)

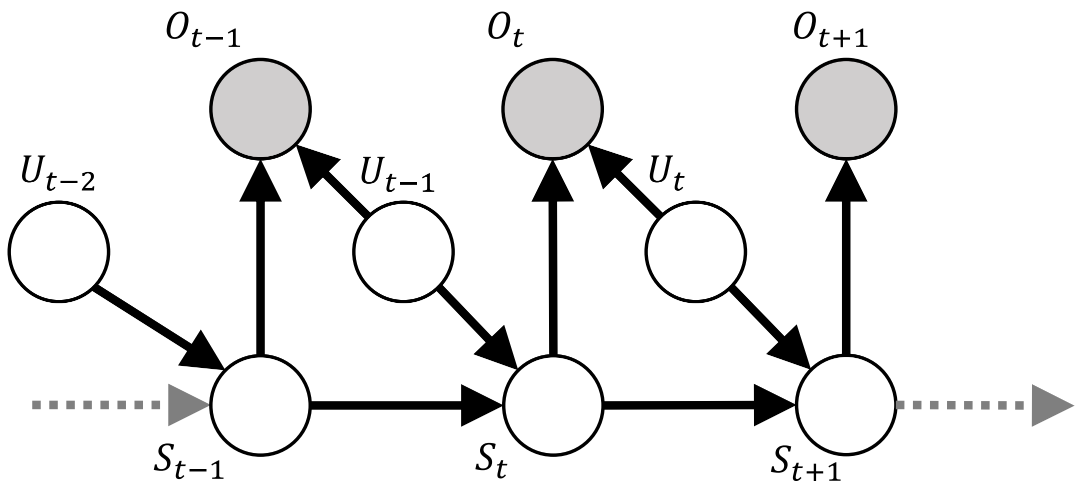
$$= \frac{p(O_{t:T} | s_t, u_t) \cancel{p(s_t)}}{p(O_{t:T} | s_t)\ |U|\ \cancel{p(s_t)}}$$ (since $p(u_t | s_t) = \frac{1}{|\mathrm{U}|}$)

$$\propto \frac{p(O_{t:T} | s_t, u_t)}{p(O_{t:T} | s_t)}$$

$$= \frac{\beta_t(s_t, u_t)}{\beta_t(s_t)}$$



Backward messages



- Can be computed recursively:

$\beta_t(s_t, u_t) = p(O_{t:T}|s_t, u_t)$

$= \sum_{s_{t+1},u_{t+1}} p(O_{t:T}, s_{t+1}, u_{t+1}|s_t, u_t)$ (introduce s_{t+1}, u_{t+1} and marginalize)

$= \sum_{s_{t+1},u_{t+1}} p(O_{t:T}|s_{t+1}, u_{t+1}, s_t, u_t)p(s_{t+1}, u_{t+1}|s_t, u_t)$ (chain rule)

$= \sum_{s_{t+1},u_{t+1}} p(O_t, O_{t+1:T}|s_{t+1}, u_{t+1}, s_t, u_t)p(s_{t+1}, u_{t+1}|s_t, u_t)$ (split $O_{t:T}$)

$= \sum_{s_{t+1},u_{t+1}} p(O_{t+1:T}|s_{t+1}, u_{t+1}, s_t, u_t, O_t)p(O_t|s_{t+1}, u_{t+1}, s_t, u_t)p(s_{t+1}, u_{t+1}|s_t, u_t)$ (chain rule)

$= \sum_{s_{t+1},u_{t+1}} p(O_{t+1:T}|s_{t+1}, u_{t+1})\, p(O_t|s_t, u_t)p(s_{t+1}, u_{t+1}|s_t, u_t)$ (conditional independence)

$= \sum_{s_{t+1},u_{t+1}} \beta_{t+1}(s_{t+1}, u_{t+1})p(O_t|s_t, u_t)p(u_{t+1}|s_{t+1})p(s_{t+1}|s_t, u_t)$

$= p(O_t|s_t, u_t)\mathbb{E}_{s_{t+1}\sim p(s_{t+1}|s_t,u_t)}[\beta_{t+1}(s_{t+1})]$

Factor Tree Sum-Product Algorithm

$\text{SUM-PRODUCT}(\mathcal{T}, E)$ // main steps of **Sum-Product algorithm**

1. $\text{EVIDENCE}(E)$
   $f = \text{CHOOSEROOT}(\mathcal{V})$
2. **for** $s \in \mathcal{N}(f)$
   $\mu\text{-COLLECT}(f, s)$
3. **for** $s \in \mathcal{N}(f)$
   $\nu\text{-DISTRIBUTE}(f, s)$
4. **for** $i \in \mathcal{V}$
   $\text{COMPUTEMARGINAL}(i)$

3. $\nu\text{-DISTRIBUTE}(i, s)$ // distribute messages from root to leaves
   $\nu\text{-SENDMESSAGE}(i, s)$
   for $j \in \mathcal{N}(s) \backslash i$
   $\mu\text{-DISTRIBUTE}(s, j)$

   $\mu\text{-DISTRIBUTE}(s, i)$
   $\mu\text{-SENDMESSAGE}(s, i)$
   for $t \in \mathcal{N}(i) \backslash s$
   $\nu\text{-DISTRIBUTE}(i, t)$

Message from variable node X_i to the factor node f_s:

$\nu\text{-SENDMESSAGE}(i, s)$

$$\nu_{is}(x_i) = \prod_{t \in \mathcal{N}(i) \backslash s} \mu_{ti}(x_i)$$

Message from factor node f_s to the variable node X_i:

$\mu\text{-SENDMESSAGE}(s, i)$

$$\mu_{si}(x_i) = \sum_{x_{\mathcal{N}(s) \backslash i}} \left(f_s(x_{\mathcal{N}(s)}) \prod_{j \in \mathcal{N}(s) \backslash i} \nu_{js}(x_j) \right)$$

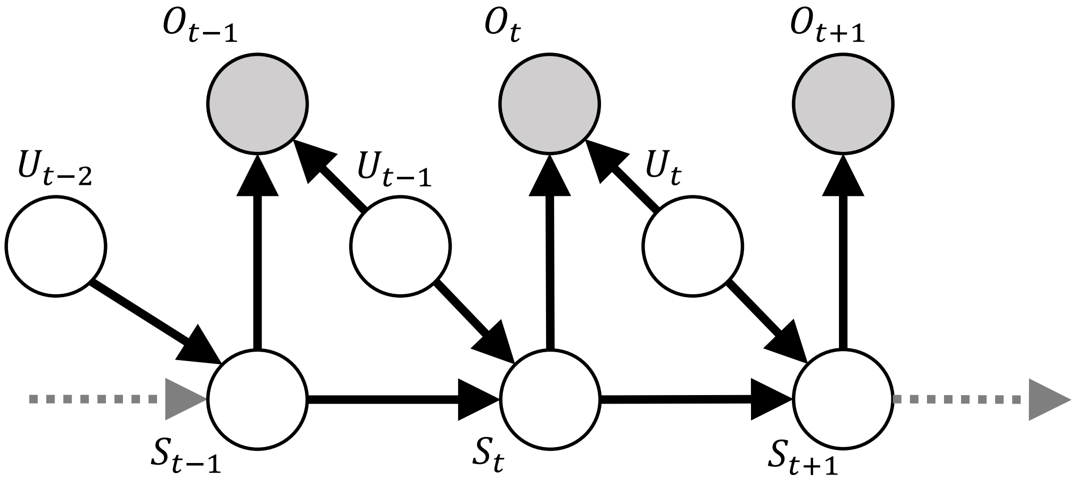
4. $\text{COMPUTEMARGINAL}(i)$ // compute **marginal probability**
   $p(x_i) \propto \nu_{is}(x_i) \mu_{si}(x_i)$

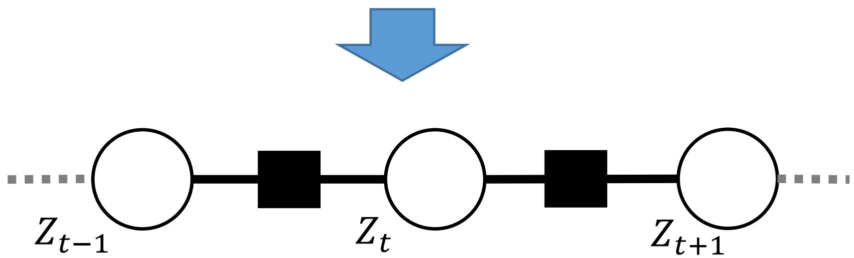
Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Easier Derivation

Backward Message according to factor graph

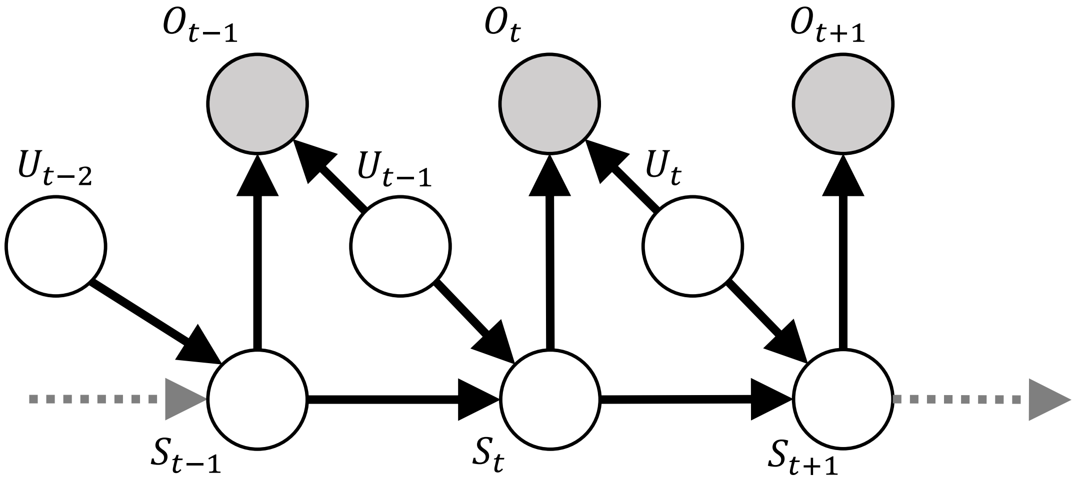
$$\beta_t(z_t)$$
$$= \sum_{z_{t+1}} f(z_{t+1}, z_t)\beta_{t+1}(z_{t+1})$$
$$= \sum_{z_{t+1}} p(O_t|s_t, u_t)p(s_{t+1}, u_{t+1}|s_t, u_t)\beta_{t+1}(z_{t+1})$$
$$= p(O_t|s_t, u_t)\sum_{s_{t+1},u_{t+1}} p(s_{t+1}, u_{t+1}|s_t, u_t)\beta_{t+1}(z_{t+1})$$
$$= p(O_t|s_t, u_t)\sum_{s_{t+1},u_{t+1}} p(u_{t+1}|s_{t+1})p(s_{t+1}|s_t, u_t)\beta_{t+1}(z_{t+1})$$
$$= p(O_t|s_t, u_t)\mathbb{E}_{s_{t+1}\sim p(s_{t+1}|s_t,u_t)}[\beta_{t+1}(s_{t+1})]$$





$$z_t = (s_t, u_t)$$
$$f(z_t, z_{t+1}) = p(s_{t+1}, u_{t+1}|s_t, u_t)p(O_t|s_t, u_t)$$

Backward pass



for $t = T-1, T-2, \ldots, 1$

$$\beta_t(s_t, u_t) = p(O_t|s_t, u_t)\mathbb{E}_{s_{t+1}\sim p(s_{t+1}|s_t,u_t)}[\beta_{t+1}(s_{t+1})]$$

$$\beta_t(s_t) = \mathbb{E}_{u_t\sim p(u_t|s_t)}[\beta_t(s_t, u_t)]$$

Exercise: What is $\beta_T(s_T)$?

Obtaining the policy as inference

- **Show:** Obtain optimal policy $p(u_t|s_t, O_{t:T})$ using inference
  - Note: drop explicit notation $O_{t:T} = 1$
- **Summary:**

for $t = T, T-1, \ldots, 1$

Compute messages $\beta_t(s_t, u_t)$ and $\beta_t(s_t)$

Compute $p(u_t|s_t, O_{t:T}) \propto \frac{\beta_t(s_t,u_t)}{\beta_t(s_t)}$

But what is the intuition?

Value and Q-function

$$V^{\pi}(s) = R\big(s, \pi(s)\big) + \gamma \sum_{s'} p(s'|s, \pi(s)) V^{\pi}(s')$$

$$Q^{\pi}(s, u) = R(s, u) + \gamma \sum_{s'} p(s'|s, u) V^{\pi}(s')$$

"Bellman update or Bellman backup"

Value iteration (finite horizon)

$$V_k^*(s) = \max_{\mathrm{u}} \; R(s,u) + \gamma \sum_{s'} p(s'|s,u) V_{k-1}^*(s')$$

Initialize $V_0^*(s) = 0$ for all $s \in S$

For $k = 0, \ldots, T-1$

For all $s \in S$

$$V_{k+1}^*(s) = \max_{\mathrm{u}} \; R(s,u) + \gamma \sum_{s'} p(s'|s,u) V_k^*(s')$$

$$\pi_{k+1}^*(s) = \operatorname*{argmax}_{\mathrm{u}} \; R(s,u) + \gamma \sum_{s'} p(s'|s,u) V_k^*(s')$$

Intuition

Recall: $\beta_t(s_t) = \mathbb{E}_{p(u_t|s_t)}[\beta_t(s_t, u_t)]$

Define "Soft" Value and Q functions:

$$\hat{V}(s_t) = \log \beta_t(s_t)$$
$$\hat{Q}(s_t, u_t) = \log \beta_t(s_t, u_t)$$

Consider:

$$\hat{V}(s_t) = \log \beta_t(s_t) = \log \mathbb{E}_{p(u_t|s_t)}[\beta_t(s_t, u_t)]$$
$$= \log \mathbb{E}_{p(u_t|s_t)}\left[\exp \hat{Q}(s_t, u_t)\right]$$

($\log \sum \exp(x)$ Operates like a "soft" maximization)

When $\hat{Q}(s_t, u_t)$ is large, then

$$\hat{V}(s_t) = \log \mathbb{E}_{p(u_t|s_t)}[\exp \hat{Q}(s_t, u_t)] \approx \max_{u_t} \hat{Q}(s_t, u_t) + c$$

Intuition: Relationship to Bellman Backup

Recall:

$$\beta_t(s_t, u_t) = \sum_{s_{t+1}, u_{t+1}} \beta_{t+1}(s_{t+1}, u_{t+1}) p(O_t | s_t, u_t) p(u_{t+1} | s_{t+1}) p(s_{t+1} | s_t, u_t)$$

$$= \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) p(O_t | s_t, u_t) p(s_{t+1} | s_t, u_t)$$

If we consider **deterministic dynamics**:

$$\hat{Q}(s_t, u_t) \quad = \log \beta_t(s_t, u_t) = \log \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) p(O_t | s_t, u_t) p(s_{t+1} | s_t, u_t)$$

$= \log \beta_{t+1}(s_{t+1}) p(O_t | s_t, u_t)$ (why?)

$$= \log \beta_{t+1} + \log p(O_t | s_t, u_t)$$

$$= \hat{V}(s_{t+1}) + r(s_t, u_t)$$

$$= r(s_t, u_t) + \hat{V}(s_{t+1})$$

Intuition: Relationship to Bellman Backup

Recall:

$$\beta_t(s_{t+1}, u_{t+1})$$

$$= \sum_{s_{t+1}, u_{t+1}} \beta_{t+1}(s_{t+1}, u_{t+1}) p(O_t | s_t, u_t) p(u_{t+1} | s_{t+1}) p(s_{t+1} | s_t, u_t)$$

$$= \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) p(O_t | s_t, u_t) p(s_{t+1} | s_t, u_t)$$

If we consider **deterministic dynamics**:

$$\hat{Q}(s_t, u_t) = r(s_t, u_t) + \hat{V}(s_{t+1})$$

Compare to Bellman Backup

$$Q^\pi(s, u) = r(s, u) + \gamma \sum_{s'} p(s' | s, u) V^\pi(s')$$

$$= r(s, u) + V^\pi(s') \quad \text{(for } \gamma = 1 \text{ and det. dynamics)}$$

Intuition: Relationship to Bellman Backup

If we consider **stochastic dynamics**:

$$\hat{Q}(s_t, u_t) \;\; = \log \beta_t(s_t, u_t) =$$

$$\log \textstyle\sum_{s_{t+1}} \beta_{t+1}(s_{t+1}\,)p(O_t|s_t, u_t)p(s_{t+1}|s_t, u_t)$$

$$= \log p(O_t|s_t, u_t) + \log \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}\,)p(s_{t+1}|s_t, u_t)$$

$$= r(s_t, u_t) + \log \underbrace{\mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, u_t)}\big[\exp \hat{V}(s_{t+1})\big]}_{\text{"Risk seeking behavior"}}$$

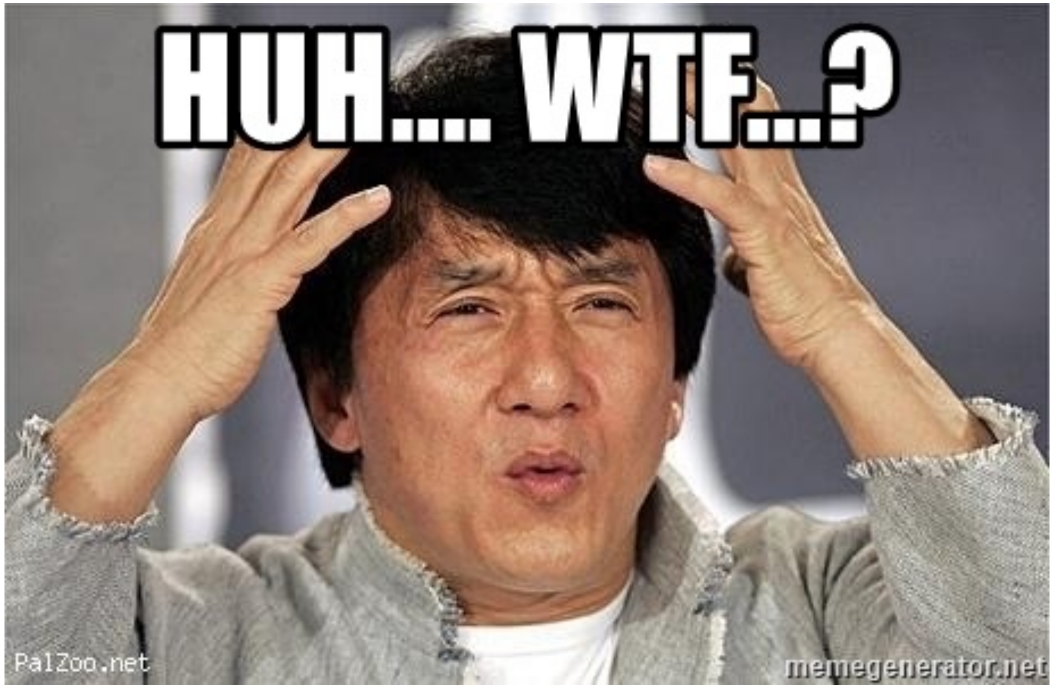
(not good!)

Compare to Bellman Backup:

$$Q^{\pi}(s, u) = r(s, u) + \gamma \sum_{s'} p(s'|s, u)V^{\pi}(s')$$

$$= r(s, u) + \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, u_t)}[V^{\pi}(s_{t+1})]$$

Wait.. What happened?



Backward messages

$$\beta_t(s_t, u_t) = p(O_{t:T}|s_t, u_t)$$
$$\beta_t(s_t) = p(O_{t:T}|s_t)$$

Goal: Compute $p(u_t|s_t, O_{t:T})$

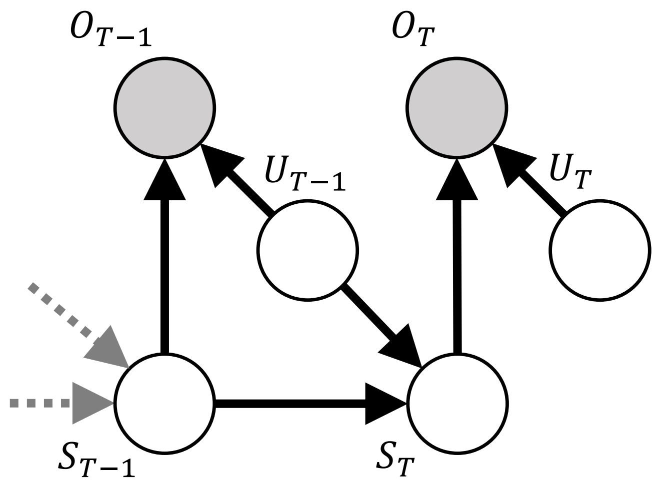
$$p(u_t|s_t, O_{t:T}) = \frac{\boxed{p(s_t, u_t|O_{t:T})}}{p(s_t|O_{t:T})}$$ (conditional probability)

$$= \frac{p(O_{t:T}|s_t, u_t)p(u_t|s_t)p(s_t)}{p(O_{t:T}|s_t)p(s_t)}$$ (Bayes Rule)

$$= \frac{p(O_{t:T}|s_t, u_t)\cancel{p(s_t)}}{p(O_{t:T}|s_t)\,|U|\,\cancel{p(s_t)}}$$ (since $p(u_t|s_t) = \frac{1}{|U|}$)

$$\propto \frac{p(O_{t:T}|s_t, u_t)}{p(O_{t:T}|s_t)}$$

$$= \frac{\beta_t(s_t, u_t)}{\beta_t(s_t)}$$



Wait.. What happened?

- The inference problem involves $p(s_{1:T}, u_{1:T} | O_{1:T})$
  - *"Given you obtained high reward, what was the probability of states and actions?"*
- We obtained:
- The **policy** $p(u_t | s_t, O_{1:T})$
  - *"Given you obtained high reward, what was your action probability?"*
- The **state distributions** $p(s_{t+1} | s_t, u_t, O_{1:T})$
  - *"Given you obtained high reward, what was your transition probability?"*
  - **Problem:** $p(s_{t+1} | s_t, u_t, O_{1:T}) \neq p(s_{t+1} | s_t, u_t)$

Example

- Numbers drawn randomly from 1 to 100 with replacement.
- *What is the probability of 7 given the first number drawn was 12?*
  - *1/100*
- ***Given that I know you won the lottery**, what is the probability of 7 given that the first number was 12?*
  - *1/2*

Winning Lottery Numbers:

- 42, 32, 43
- 12, 7, 6
- 12, 3, 5

Wait.. What happened?

- The **policy** $p(u_t|s_t, O_{1:T})$
  - *"Given you obtained high reward, what was your action probability?"*
- The **state distributions** $p(s_{t+1}|s_t, u_t, O_{1:T})$
  - *"Given you obtained high reward, what was your transition probability?"*
  - **Problem:** $p(s_{t+1}|s_t, u_t, O_{1:T}) \neq p(s_{t+1}|s_t, a_t)$
- What we *actually* want:

*"Given you have obtained high reward **<u>and your transition probability did not change</u>**, what was your action probability?"*

Inference and Control

Structured Variational Inference

High-Level Structure

- A Probabilistic Graphical Model (PGM) for Control
  - Performing inference to obtain a policy
    - Backward Messages
  - “Structured” variational inference to obtain a constrained policy

Fix using approximate inference

Approximate trajectory distribution, $p(\tau\ )$ where $\tau = (s_{1:T}, u_{1:T})$

$$p(\tau\ ) = \left[p(s_1) \prod_{t=1}^{T} p(s_{t+1}|s_t, u_t) \right] \exp\left(\sum_{t=1}^{T} r(s_t, u_t) \right)$$

with the distribution

$$q(\tau) = q(s_1) \prod_{t=1}^{T} q(s_{t+1}|s_t, u_t) \underbrace{q(u_t|s_t)}_{\text{This is our policy!}}$$

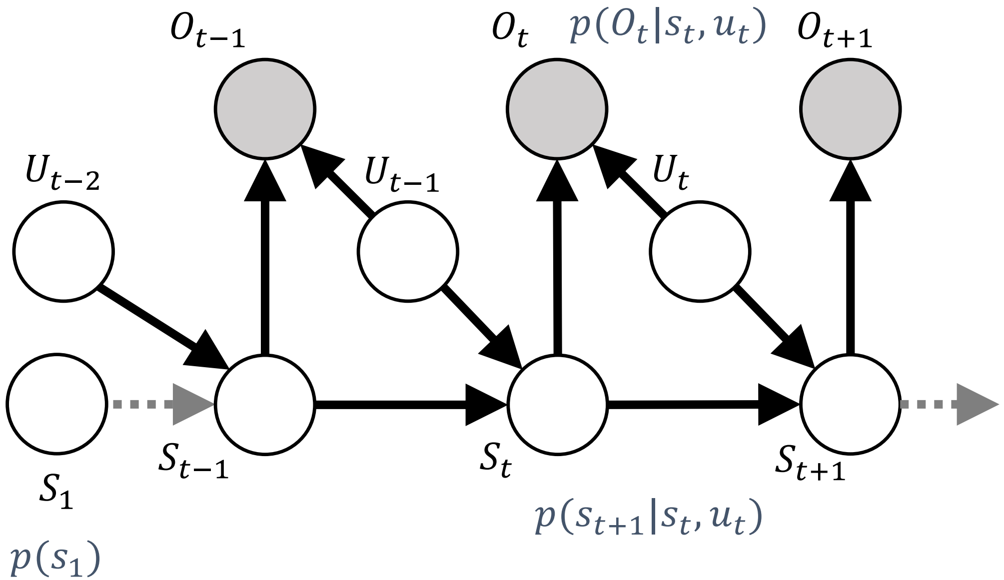
Don't want the agent to "control" the dynamics so, fix:

$$q(s_1) = p(s_1)$$
$$q(s_{t+1}|s_t, u_t) = p(s_{t+1}|s_t, u_t)$$

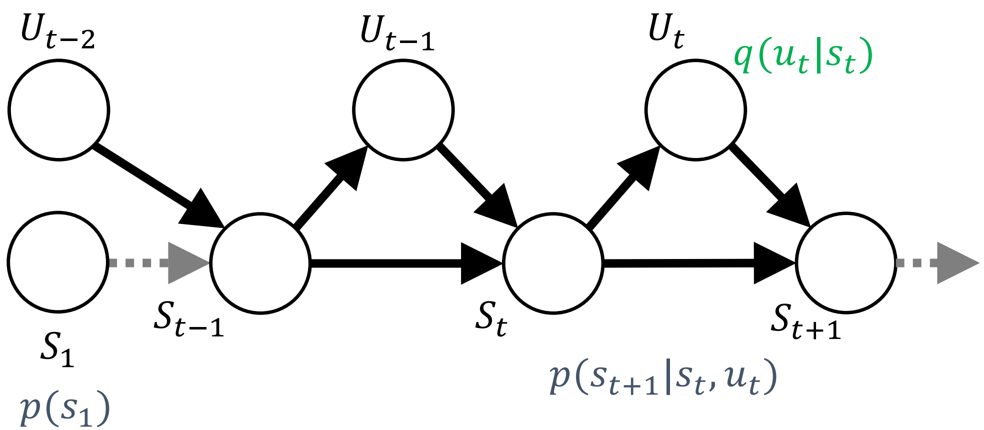
P and Q in pictures

$$q(\tau) = p(s_1) \prod_{t=1}^{T} p(s_{t+1}|s_t, u_t) q(u_t|s_t)$$

$$\boldsymbol{p(s_{1:T}, u_{1:T}|O_{1:T})}$$



$$\boldsymbol{q(s_{1:T}, u_{1:T})}$$



Variational lower-bound (ELBO)

$$\log p(O_{1:T}) = \log \sum_{s_{1:T}} \sum_{u_{1:T}} p(O_{1:T}, s_{1:T}, u_{1:T})$$
$$= \log \sum_{s_{1:T}} \sum_{u_{1:T}} p(O_{1:T}, s_{1:T}, u_{1:T}) \frac{q(s_{1:T}, u_{1:T})}{q(s_{1:T}, u_{1:T})}$$
$$= \log \mathbb{E}_{q(s_{1:T}, u_{1:T})} \left[\frac{p(O_{1:T}, s_{1:T}, u_{1:T})}{q(s_{1:T}, u_{1:T})} \right]$$
$$\geq \mathbb{E}_{q(s_{1:T}, u_{1:T})} \left[\log \frac{p(O_{1:T}, s_{1:T}, u_{1:T})}{q(s_{1:T}, u_{1:T})} \right]$$
$$= \mathbb{E}_{q(s_{1:T}, u_{1:T})} [\log p(O_{1:T}, s_{1:T}, u_{1:T})] - \mathbb{E}_{q(s_{1:T}, u_{1:T})} [\log q(s_{1:T}, u_{1:T})]$$

Variational lower-bound (ELBO)

$$\mathcal{L}(q) = \mathbb{E}_{q(s_{1:T}, u_{1:T})}\left[\log \frac{p(O_{1:T}, s_{1:T}, u_{1:T})}{q(s_{1:T}, u_{1:T})}\right]$$

Variational lower-bound (ELBO)

$$\mathcal{L}(q) = \mathbb{E}_{q(s_{1:T},u_{1:T})}\left[\log \frac{p(O_{1:T}, s_{1:T}, u_{1:T})}{q(s_{1:T}, u_{1:T})}\right]$$

Substituting p and q,

$$\mathcal{L}(q) = \mathbb{E}_{q(s_{1:T},u_{1:T})}\left[\log \frac{[\cancel{p(s_1) \prod_{t=1}^{T} p(s_{t+1}|s_t, u_t)}] \exp(\sum_{t=1}^{T} r(s_t, u_t))}{\cancel{p(s_1) \prod_{t=1}^{T} p(s_{t+1}|s_t, u_t)} q_\theta(u_t|s_t)}\right]$$

So,

$$\mathcal{L}(q) = \mathbb{E}_{q(s_{1:T},u_{1:T})}\left[\sum_{t=1}^{T} r(s_t, u_t) - \log q(u_t|s_t)\right]$$

Variational lower-bound (ELBO)

$$\mathcal{L}(q) = \mathbb{E}_{q(s_{1:T},u_{1:T})}[\sum_{t=1}^{T} r(s_t, u_t) - \log q(u_t|s_t)]$$

$$= \sum_t \mathbb{E}_{q(s_t,u_t)}[\, r(s_t, u_t)] - \mathbb{E}_{q(s_t)q(u_t|s_t)}[\log q(u_t\, |s_t)]$$

$$= \sum_t \mathbb{E}_{q(s_t,u_t)}[\, r(s_t, u_t)] + \mathbb{E}_{q(s_t)}[\mathbb{H}[\, q(u_t\, |s_t)]]$$

Variational lower-bound (ELBO)

$$\max_{q_\theta} \mathcal{L}(q) = \max_{q_\theta} \sum_t \mathbb{E}_{q(s_t,u_t)}[\, r(s_t, u_t)] + \underbrace{\mathbb{E}_{q(s_t)}[\mathbb{H}[\, q(u_t \,|s_t)]]}_{\text{Action Entropy}}$$

- Maximizing the above yields the correct policy
  - Doesn't have "risk-seeking" behavior
  - Can apply other structural constraints
    - "Maximum Entropy" RL

the update messages

Using the Variational approach, the backward messages are:

$$\hat{V}_t(s_t) = \log \sum \exp \hat{Q}_t(s_t, u_t)$$

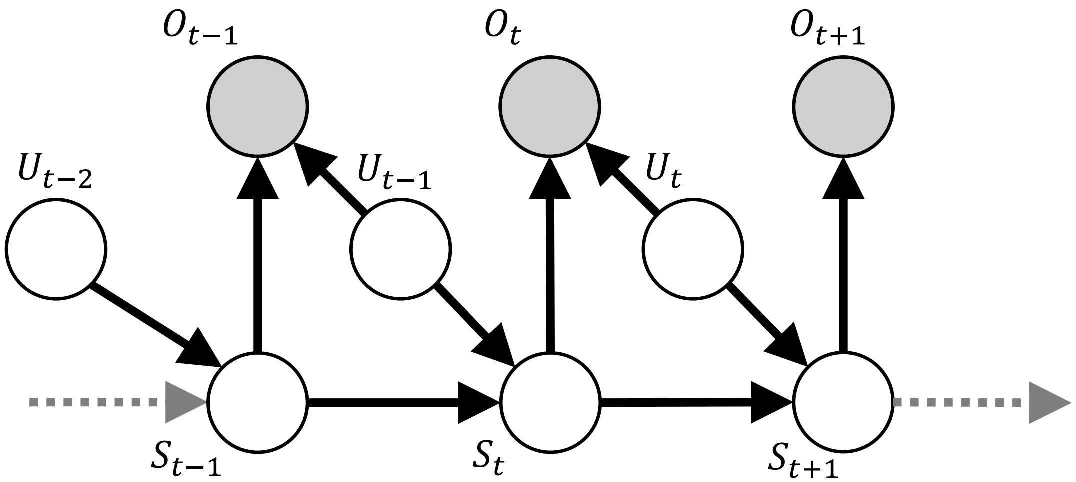
$$\hat{Q}_t(s_t, u_t) = r(s_t, u_t) + \mathbb{E}\left[\hat{V}_{t+1}(s_{t+1})\right]$$

$$q(u_t|s_t) = \exp(\hat{Q}_t(s_t, u_t) - \hat{V}_t(s_t))$$

Compare with:

- $\hat{Q}(s_t, u_t) = r(s_t, u_t) + \log \mathbb{E}_{s_{t+1}\sim p(s_{t+1}|s_t, u_t)}\left[\exp \hat{V}(s_{t+1})\right]$ (standard inference, risk-seeking)
- $Q^{\pi}(s, u) = r(s, u) + \mathbb{E}_{s_{t+1}\sim p(s_{t+1}|s_t, u_t)}[V^{\pi}(s_{t+1})]$

Summary



- A PGM for *control* via *inference*
- A *variational* solution for inference

Learning rewards from an expert

- *"If we use, to achieve our purposes, a mechanical agency with whose operation we cannot interfere effectively . . . we had better be quite sure that the purpose put into the machine is the purpose which we really desire."*
  - \- Norbert Wiener, 1960

Learning rewards from an expert

Democratizing robot programming

Image credit:
https://www.popsci.com/scitech/article/2008-04/why-grandma-may-get-coolest-robot-block/

Image Credit: https://ai.googleblog.com/2016/10/how-robots-can-acquire-new-skills-from.html

Inverse reinforcement learning

- **Reinforcement learning**
- **Given:**
  - MDP (S, U, T, r, γ)
- **Goal:**
  - Obtain/Learn $\pi^*(s)$

***Inverse* Reinforcement learning**

Given:

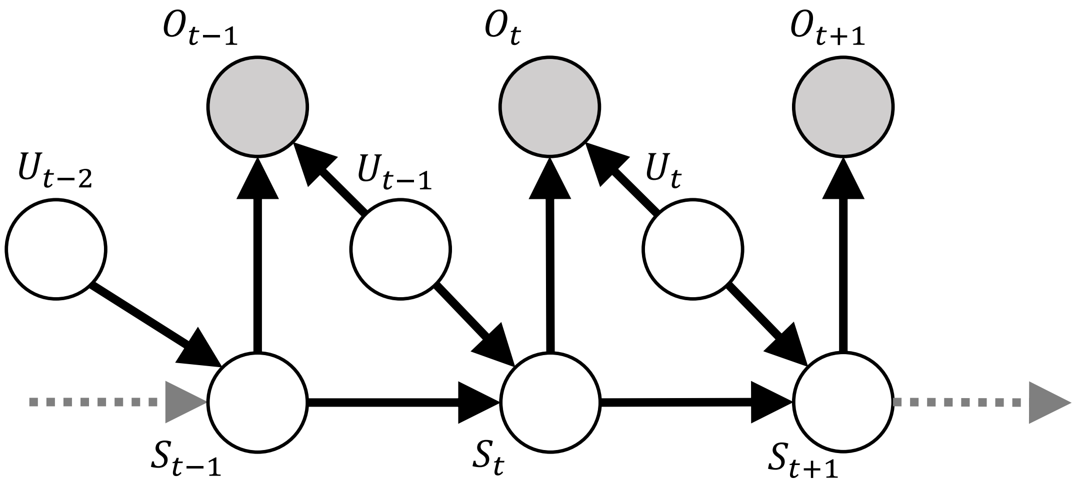
- MDP\r (S, U, T, γ)
- Dataset of trajectories $D = \{\tau_i\}$ sampled from $\pi^*(\tau)$

Goal:

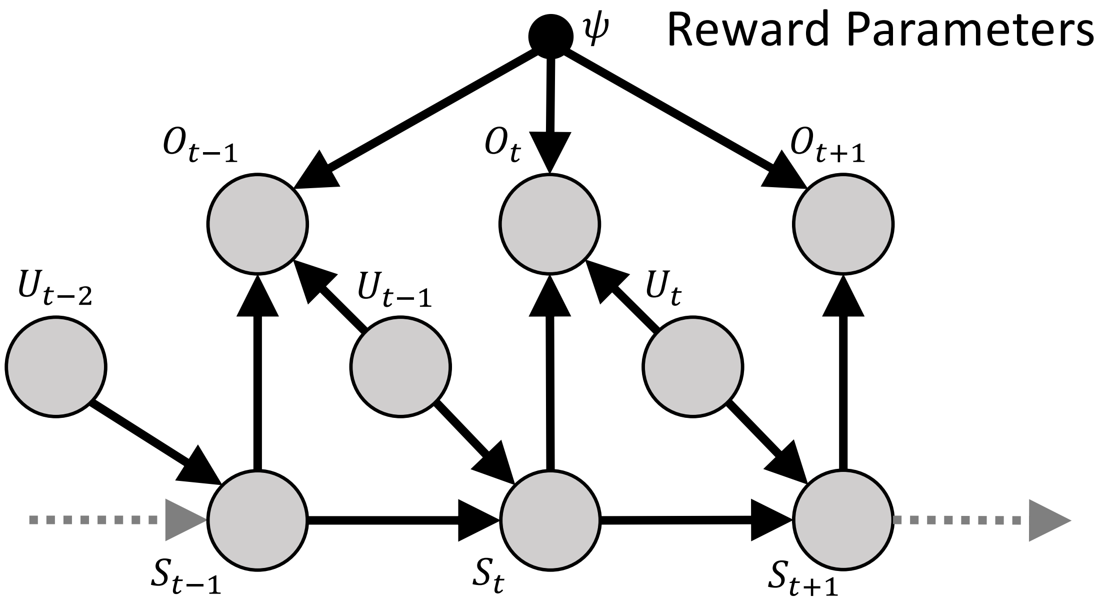
- Obtain/Learn $r_\psi(s, u)$

Variants: Sometimes, no transitions T

Can we use our graphical model?



Can we use our graphical model?



Goal: Learn ψ via maximum likelihood estimation (MLE)

Maximum likelihood estimation

Goal: $\psi^* = \arg\max_{\psi} \log p(D|\psi) = \arg\max_{\psi} \sum_i r_\psi(\tau_i) - \log Z_\psi$

We need to maximize: $\frac{1}{\mathrm{N}} \sum_i r_\psi(\tau_i) - \log Z_\psi$

How to maximize?

$$\nabla_\psi L = \frac{1}{N} \sum_i \nabla_\psi r_\psi(\tau_i) - \log \nabla_\psi Z_\psi$$

$$= \boxed{\mathbb{E}_{\tau \sim \pi^*}\left[\nabla_\psi r_\psi(\tau)\right]} - \boxed{\mathbb{E}_{\tau \sim p(\tau | O_{1:T}, \psi)}[\nabla_\psi r_\psi(\tau)]}$$

Estimate with Expert Trajectories

Compute using soft optimal policy under current reward