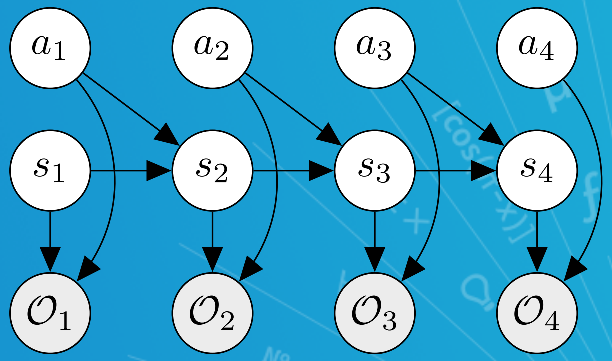
Probabilistic Graphical Models

Reinforcement Learning & Control Through Inference in GM (part 2)



Maruan Al-Shedivat

Lecture 20, April 1, 2020

Reading: see class homepage

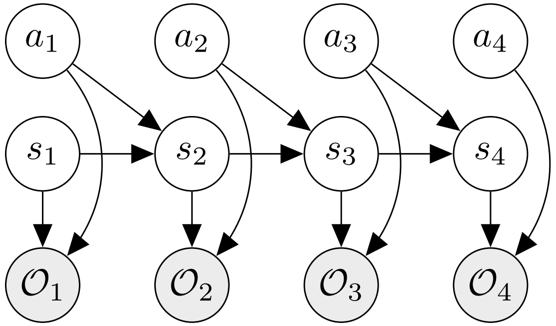
A note on materials used in this module

- Sutton & Barto. Reinforcement Learning: An Introduction. 2nd edition.
- David Silver's UCL course on reinforcement learning.
- Materials from UC Berkeley's Deep RL course.
- Sergey Levine's tutorial on RL and control as inference.
- Brian Ziebart's PhD thesis (maximum causal entropy models).

Plan

Part 1: Intro to RL and Control as Inference Framework

- ❑ Intro to Reinforcement Learning (RL)
- ❑ RL and Control as Inference: The GM framework
- ❑ Connections to variational inference



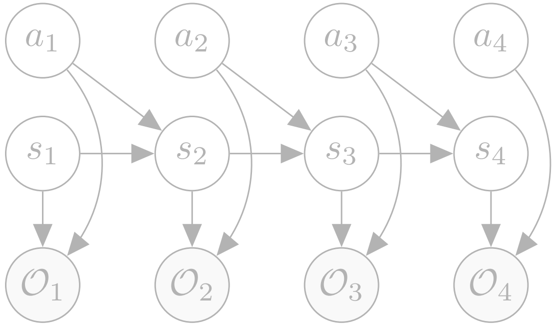
Part 2: Max-entropy RL Algorithms

- ❑ Recap and an inferential approach to RL
- ❑ Classical Q-learning and policy gradient methods
- ❑ Soft Q-learning and soft policy gradients

Algorithm 1 Soft Actor-Critic

Initialize parameter vectors ψ, $\bar{\psi}$, θ, ϕ.
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$
  end for
end for

Plan



Part 1: Intro to RL and Control as Inference Framework

- Intro to Reinforcement Learning (RL)
- RL and Control as Inference: The GM framework
- Connections to variational inference

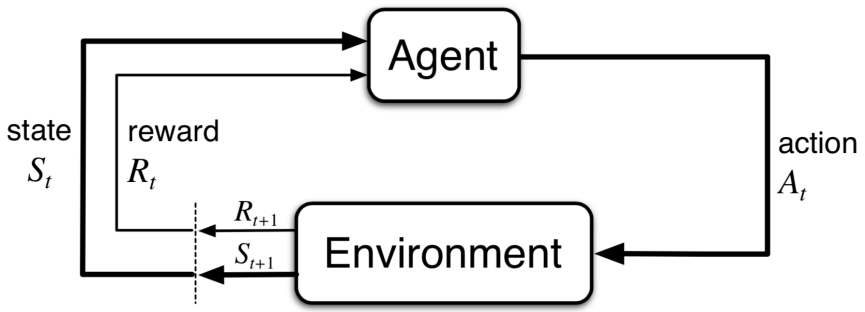
Part 2: Max-entropy RL Algorithms

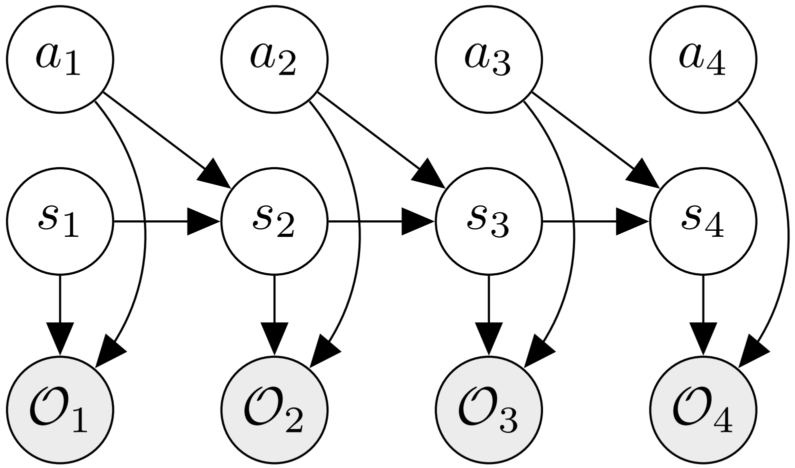
- Recap and an inferential approach to RL
- Classical Q-learning and policy gradient methods
- Soft Q-learning and soft policy gradients

Algorithm 1 Soft Actor-Critic

Initialize parameter vectors $\psi, \bar{\psi}, \theta, \phi$.
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$
  end for
end for

Recap: Control as Inference

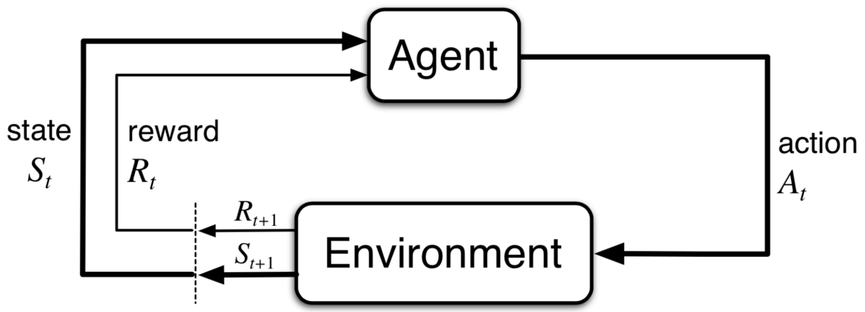




Initial state	$s_0 \sim p_0(s)$
Transition	$s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$
Policy	$a_t \sim \pi(a_t \mid s_t)$
Reward	$r_t = r(s_t, a_t)$

Initial state	$s_0 \sim p_0(s)$
Transition	$s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$
Policy	$a_t \sim \pi(a_t \mid s_t)$
Reward	$r_t = r(s_t, a_t)$
Optimality	$p(\mathcal{O}_t = 1 \mid s_t, a_t) = \exp(r(s_t, a_t))$

Recap: Control as Inference



Initial state	$s_0 \sim p_0(s)$
Transition	$s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$
Policy	$a_t \sim \pi(a_t \mid s_t)$
Reward	$r_t = r(s_t, a_t)$

In the classical RL setup, we have:

$$V_\pi(s) := \mathbb{E}_\pi \left[\sum_{k=0}^{T} \gamma^k r_{t+k+1} \mid s_t = s \right]$$

$$Q_\pi(s, a) := \mathbb{E}_\pi \left[\sum_{k=0}^{T} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right]$$

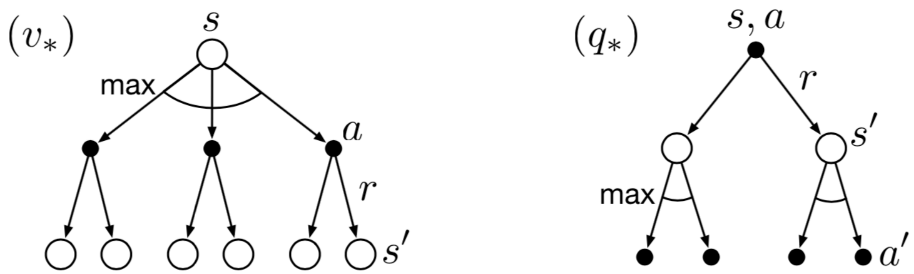
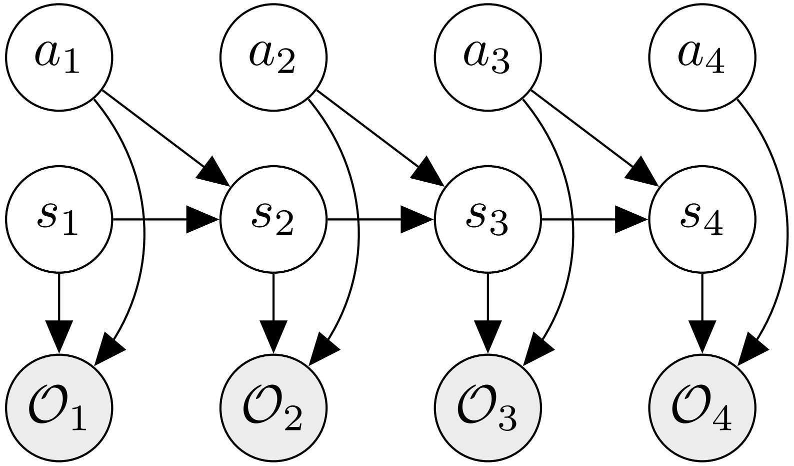


Figure 3.4: Backup diagrams for v_* and q_*

$$\pi_\star(a \mid s) = \delta\left(a = \arg\max_a Q_\star(s, a)\right)$$

Recap: Control as Inference



Initial state	$s_0 \sim p_0(s)$
Transition	$s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$
Policy	$a_t \sim \pi(a_t \mid s_t)$
Reward	$r_t = r(s_t, a_t)$
Optimality	$p(\mathcal{O}_t = 1 \mid s_t, a_t) = \exp(r(s_t, a_t))$

Running inference in this GM allows us to compute:

$$p(\tau \mid \mathcal{O}_{1:T}) \propto \left[p(s_1) \prod_{t=1}^{T-1} p(s_{t+1} \mid s_t, a_t)\right] \times \exp\left(\sum_t r_t\right)$$

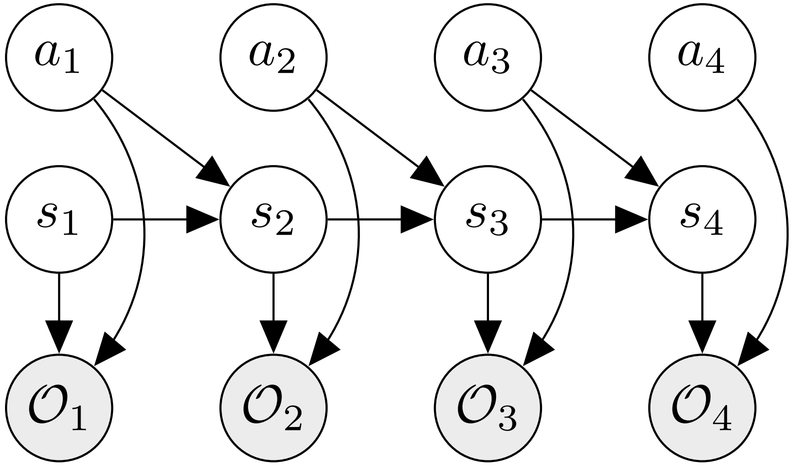
$$\text{let } V_t(\mathbf{s}_t) = \log \beta_t(\mathbf{s}_t)$$

$$\text{let } Q_t(\mathbf{s}_t, \mathbf{a}_t) = \log \underbrace{\beta_t(\mathbf{s}_t, \mathbf{a}_t)}_{p(\mathcal{O}_{t:T} \mid s_t, a_t)}$$

$$V(\mathbf{s}_t) = \underbrace{\log \int}_{\text{softmax}} \exp(Q(\mathbf{s}_t, \mathbf{a}_t) + \log p(\mathbf{a}_t|\mathbf{s}_t))\mathbf{a}_t$$

$$p(a_t \mid s_t, \mathcal{O}_{1:T}) = \exp(\underbrace{Q_t(s_t, a_t) - V_t(s_t)}_{A_t(s_t, a_t)})$$

Recap: Control as Inference



Initial state	$s_0 \sim p_0(s)$
Transition	$s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$
Policy	$a_t \sim \pi(a_t \mid s_t)$
Reward	$r_t = r(s_t, a_t)$
Optimality	$p(\mathcal{O}_t = 1 \mid s_t, a_t) = \exp(r(s_t, a_t))$

Which objective does inference optimize?

policy-induced $\quad p(\tau \mid \mathcal{O}_{1:T})$

$$-D_{\mathrm{KL}}\left(\hat{p}(\tau) \| p(\tau)\right) =$$

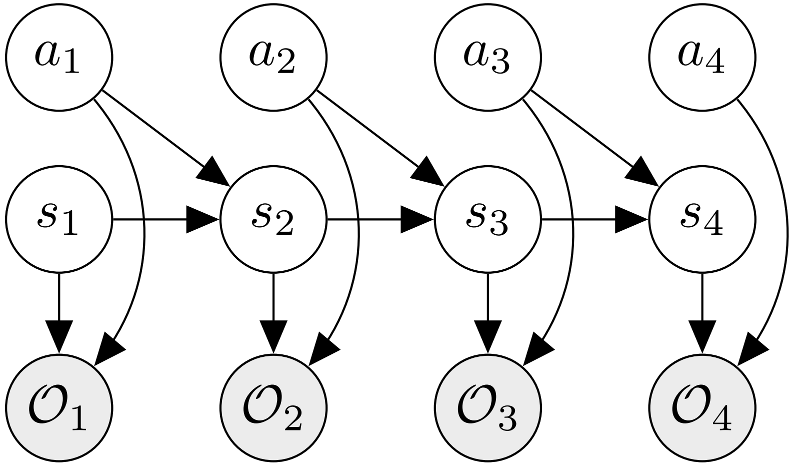
classical RL objective

$$\sum_{t=1}^{T} \mathbb{E}_{(s_t, a_t) \sim \hat{p}(s_t, a_t)}[r(s_t, a_t)] + \mathbb{E}_{(s_t) \sim \hat{p}(s_t)}[\mathcal{H}(\pi(a_t \mid s_t))]$$

entropy reg.

- For deterministic dynamics, get it directly
- For stochastic dynamics, obtain it from the ELBO on the evidence

A unifying perspective on Imitation, RL, and Planning



One of the key advantages of PGM is the unifying approach to learning base on **likelihood maximization.**

Start with the complete likelihood:

$$\log P(s_{1:T}, a_{1:T}, \mathcal{O}_{1:T})$$

Imitation (Behavioral Cloning)

$$\log P(a_{1:T} \mid s_{1:T})$$

Imitation (Inverse RL) & Planning

$$\log P(s_{1:T}, a_{1:T} \mid \mathcal{O}_{1:T})$$

RL

$$\log P(\mathcal{O}_{1:T})$$

A unifying perspective on Imitation, RL, and Planning

Imitation (Behavioral Cloning):

- Given: $\tau = (s_{1:T},\ a_{1:T})$

$$\arg\max_{\pi} \log P(a_{1:T} \mid s_{1:T})$$

Imitation (Inverse RL):

- Given: $\tau = (s_{1:T},\ a_{1:T}), \mathcal{O}_{1:T}$

$$\arg\max_{r_{1:T}} \log P(s_{1:T}, a_{1:T} \mid \mathcal{O}_{1:T})$$

Planning:

- Given: $\tau = (s_{1:T},\ a_{1:T}), \mathcal{O}_{1:T}$

$$\arg\max_{\tau} \log P(s_{1:T}, a_{1:T} \mid \mathcal{O}_{1:T})$$

RL:

- Given: $\mathcal{O}_{1:T}$

$$\arg\max_{\pi} \log P(\mathcal{O}_{1:T}) \geq \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim\hat{p}(s_t,a_t)}\left[r(s_t,a_t)\right] + \mathbb{E}_{s_t\sim\hat{p}(s_t)}\left[\mathcal{H}(\pi(a_t \mid s_t))\right]$$

PETUUM

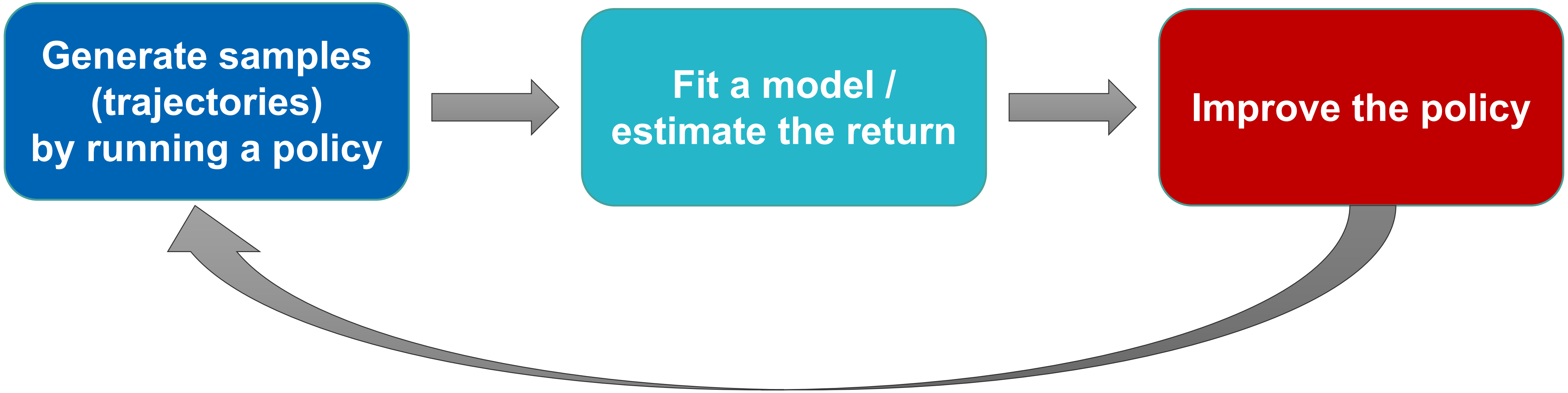
Carnegie Mellon University
School of Computer Science

Policy Gradient Methods

[Barto & Sutton's textbook; Sergey Levine's lectures]

How do we learn in RL?

$$\theta^\star = \arg\max_\theta \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} r(s_t, a_t) \right]$$



How do we learn in RL?

Video from Schulman et al. (2016). High-dimensional continuous control using generalized advantage estimation.

Types of RL algorithms

$$\theta^\star = \arg\max_\theta \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T} r(s_t, a_t)\right]$$

- **Policy gradients:** directly optimize the above stochastic objective
- **Value-based:** estimate V-function or Q-function of the optimal policy (no explicit policy; the policy is derived from the value function)
- **Actor-critic:** estimate V-/Q-function under the current policy and use it to improve the policy
- **Model-based methods:** estimate the transition model $p(s_{t+1}|s_t, a_t)$ and…
  - Use it for planning (plug-in the model into the objective, optimize it w.r.t. a sequence of actions, pick the first action in the best sequence)
  - Use it to improve the policy (e.g., MCTS distillation in AlphaGo)

Types of RL algorithms

$$\theta^\star = \arg\max_\theta \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} r(s_t, a_t) \right]$$

- **Policy gradients:** directly optimize the above stochastic objective
- **Value-based:** estimate V-function or Q-function of the optimal policy (no explicit policy; the policy is derived from the value function)
- **Actor-critic:** estimate V-/Q-function under the current policy and use it to improve the policy
- **Model-based methods:** estimate the transition model $p(s_{t+1}|s_t, a_t)$ and…
  - Use it for planning (plug-in the model into the objective, optimize it w.r.t. a sequence of actions, pick the first action in the best sequence)
  - Use it to improve the policy (e.g., MCTS distillation in AlphaGo)

Policy gradients

$$\underbrace{\theta^\star = \arg\max_\theta \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T} r(s_t,a_t)\right]}_{J(\theta)}$$

$$J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T} r(s_t,a_t)\right] \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} r(s_{i,t},a_{i,t})$$

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[r(\tau)\right] = \int r(\tau)\nabla_\theta p_\theta(\tau)d\tau$$

How to generate a trajectory?

$s_1 = \text{env.init}()$

for $t=1$ to T:

  $a_t \sim \pi_\theta(a \mid s_t)$

  $s_{t+1}, r_{t+1} = \text{env.step}(a_t)$

Policy gradients

$$\theta^\star = \arg\max_\theta \underbrace{\mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T} r(s_t,a_t)\right]}_{J(\theta)}$$

$$\nabla_\theta p_\theta(\tau) = \frac{p_\theta(\tau)}{p_\theta(\tau)} \nabla_\theta p_\theta(\tau)$$

$$= p_\theta(\tau)\nabla_\theta \log p_\theta(\tau)$$

$$J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T} r(s_t,a_t)\right] \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} r(s_{i,t},a_{i,t})$$

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau\sim p_\theta(\tau)}[r(\tau)] = \int r(\tau)\nabla_\theta p_\theta(\tau)d\tau = \int r(\tau)p_\theta(\tau)\nabla_\theta \log p_\theta(\tau)d\tau$$

Policy gradients

$$\theta^\star = \arg\max_\theta \underbrace{\mathbb{E}_{\tau \sim p_\theta(\tau)}\left[\sum_{t=1}^{T} r(s_t, a_t)\right]}_{J(\theta)}$$

$$\nabla_\theta p_\theta(\tau) = \frac{p_\theta(\tau)}{p_\theta(\tau)} \nabla_\theta p_\theta(\tau)$$

$$= p_\theta(\tau) \nabla_\theta \log p_\theta(\tau)$$

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[\sum_{t=1}^{T} r(s_t, a_t)\right] \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} r(s_{i,t}, a_{i,t})$$

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[r(\tau)\right] = \int r(\tau) \nabla_\theta p_\theta(\tau) d\tau = \int r(\tau) p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau$$

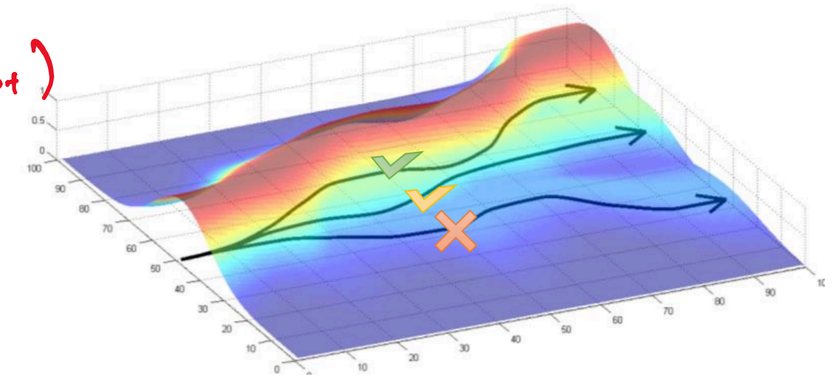
$$= \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[r(\tau) \nabla_\theta \log p_\theta(\tau)\right]$$

For more on how to compute derivatives of stochastic objectives see:
Schulman et al. (2015) Gradient Estimation Using Stochastic Computation Graphs.
Foerster, Farquhar*, A.* et al. (2018) DiCE: The Infinitely Differentiable Monte-Carlo Estimator.
Mohamed*, Rosca*, Figurnov*, Mnih* (2019) Monte Carlo Gradient Estimation in Machine Learning.

Figure adapted from Sergey Levin's slides.

Policy gradients

$p(s_1)\prod_t p(s_{t+1}|s_t,a_t)\pi_\theta(a_t|s_t)$



$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[r(\tau)\nabla_\theta \log p_\theta(\tau)\right]$$

$$\nabla_\theta \log p_\theta(\tau) = \nabla_\theta \left[\cancel{\log p(s_1)} + \sum_{t=1}^{T} \cancel{\log p(s_{t+1} \mid s_t, a_t)} + \log \pi_\theta(a_t \mid s_t)\right]$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right)\left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right]$$

$$\nabla_\theta J_{\mathrm{ML}}(\theta) = \mathbb{E}_{\tau\sim p_{\mathrm{expert}}(\tau)}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

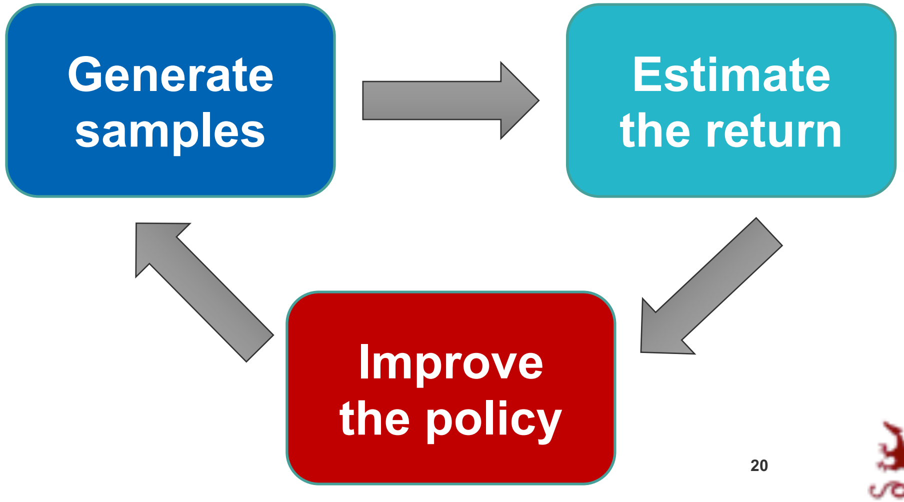
Policy gradients

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right)\left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right]$$

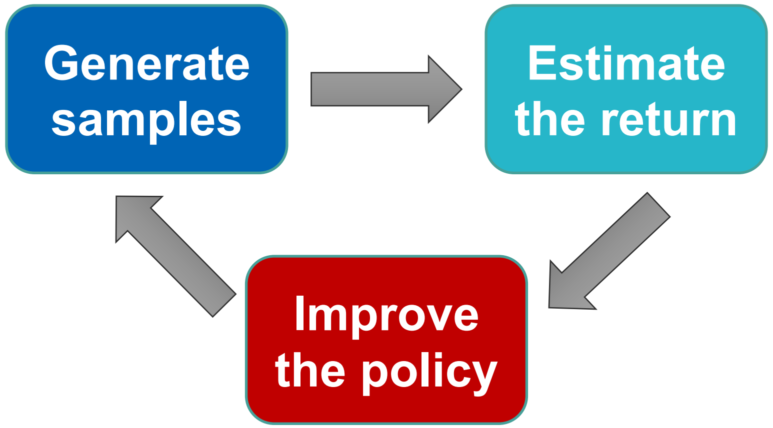
$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t})\right)\right]$$

REINFORCE algorithm:

1. sample $\{\tau_i\}_{i=1}^{N}$ under $\pi_\theta(a_t \mid s_t)$



Policy gradients: Summary



REINFORCE algorithm:

1. sample $\{\tau_i\}_{i=1}^{N}$ under $\pi_\theta(a_t \mid s_t)$
2. $\hat{J}(\theta) = \sum_i \left(\sum_t \log \pi_\theta(a_{i,t} \mid s_{i,t})\right)\left(\sum_t r(s_{i,t}, a_{i,t})\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta \hat{J}(\theta)$

- Represent a policy with a parametric function (e.g., neural net) and learn it by optimizing the REINFORCE objective
- Relationship between PG objective and MLE objective: rewards reweight the samples
- REINFORCE gradients are often extremely high-variance →
  make use of *action causality* + value estimators to reduce the variance
  (look up Sutton & Barto’s textbook, Ch. 13 or check out a deep RL course)

PETUUM

Carnegie Mellon University School of Computer Science

Q-learning

[Barto & Sutton’s textbook; Sergey Levine’s lectures]

Can we get rid of the dependence on the policy?

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t}) \right) \left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \right]$$

- Recall that if we have access to an optimal $Q_\star(s, a)$, it gives us right away a corresponding optimal greedy (deterministic) policy:

$$\pi_\star(a \mid s) = \delta\left(a = \arg\max_a Q_\star(s, a) \right)$$

- If we don't have access to an optimal $Q_\star(s, a)$, we can still try:

$$a_t = \arg\max_a \left[Q_\pi(a, s_t) \right]$$

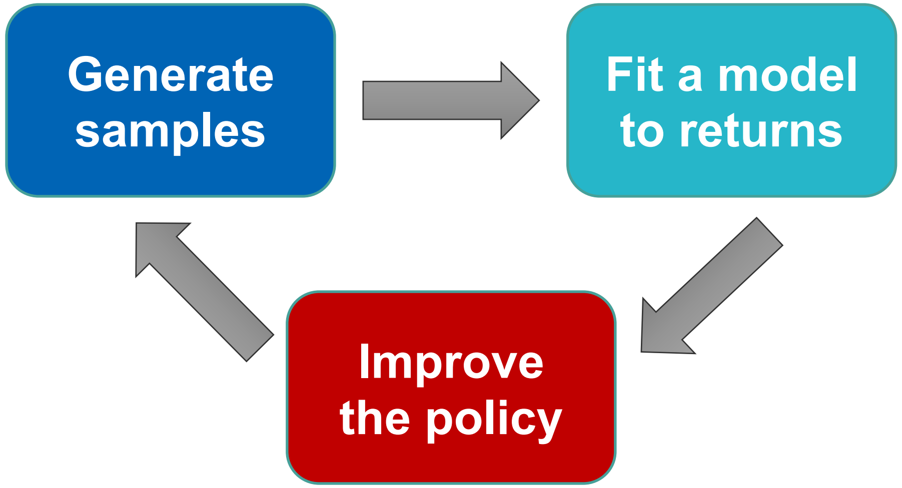
Policy iteration

- Can we learn a Q-function through interaction with the environment without a policy?

$$\pi'(a_t \mid s_t) = \delta\left(a_t = \arg\max_a \left[Q_\pi(a, s_t)\right]\right) \geq \pi$$

Policy iteration:

1. evaluate $Q_\pi(s, a)$
2. update $\pi \leftarrow \pi'$



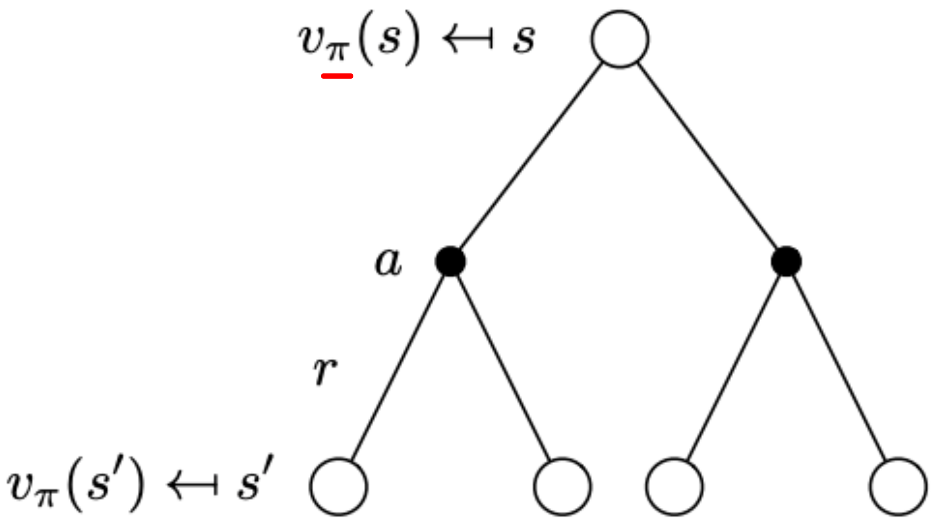
Fit $Q_\pi(s, a)$

$\pi \leftarrow \pi'$

Policy iteration via dynamic programming

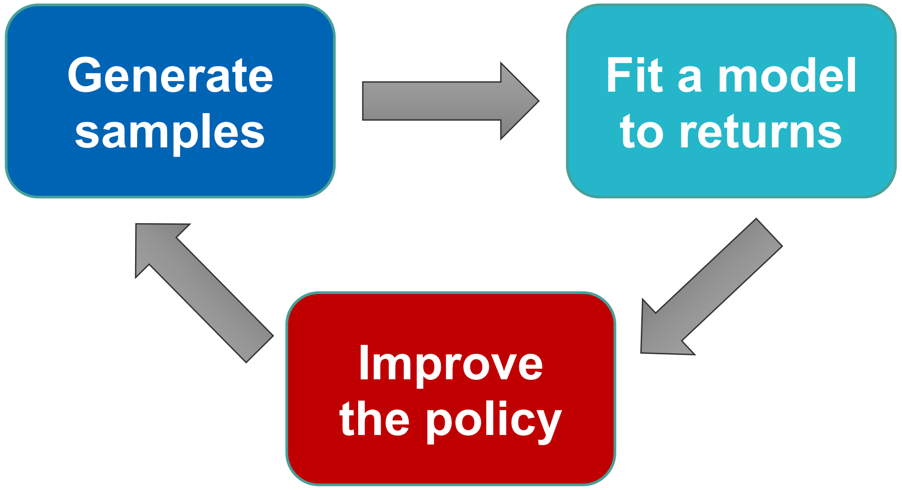
Policy iteration:

1. evaluate $Q_\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim p(s'|s,a)} [V_\pi(s')]$
2. update $\pi \leftarrow \pi'$



Policy iteration via dynamic programming

$$Q_\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim p(s'|s,a)}\left[\underline{V_\pi(s')}\right]$$



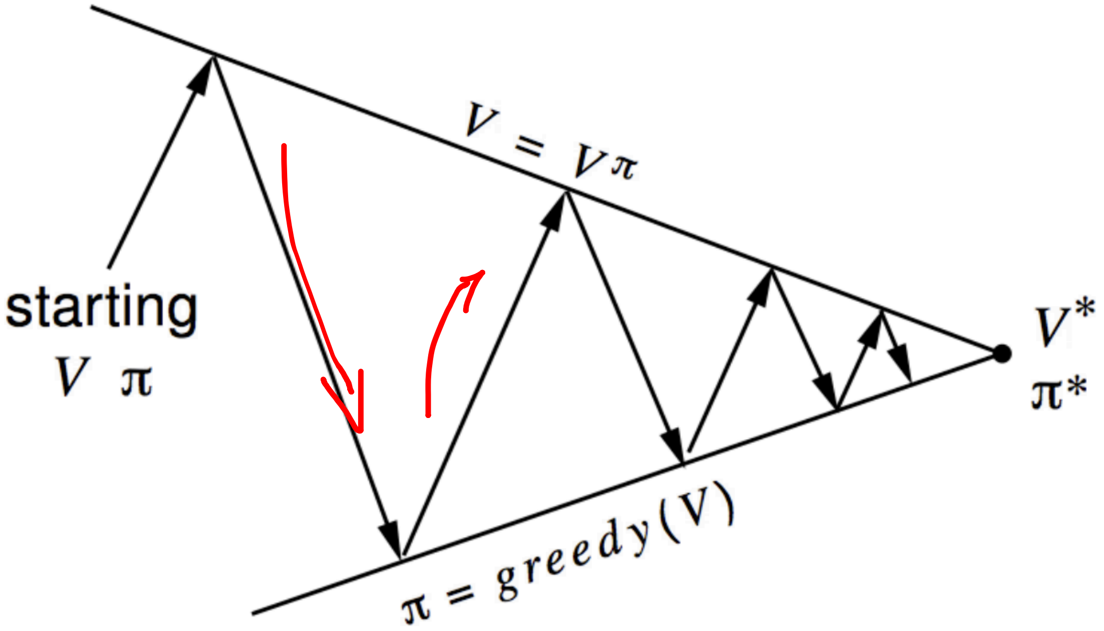
Policy iteration:

1. evaluate $Q_\pi(s,a)$
2. update $\pi \leftarrow \pi'$

Policy evaluation:

$$V_\pi(s) \leftarrow r(s,\pi(s)) + \gamma \mathbb{E}_{s' \sim p(s'|s,\pi(s))}\left[V_\pi(s')\right]$$

Policy iteration



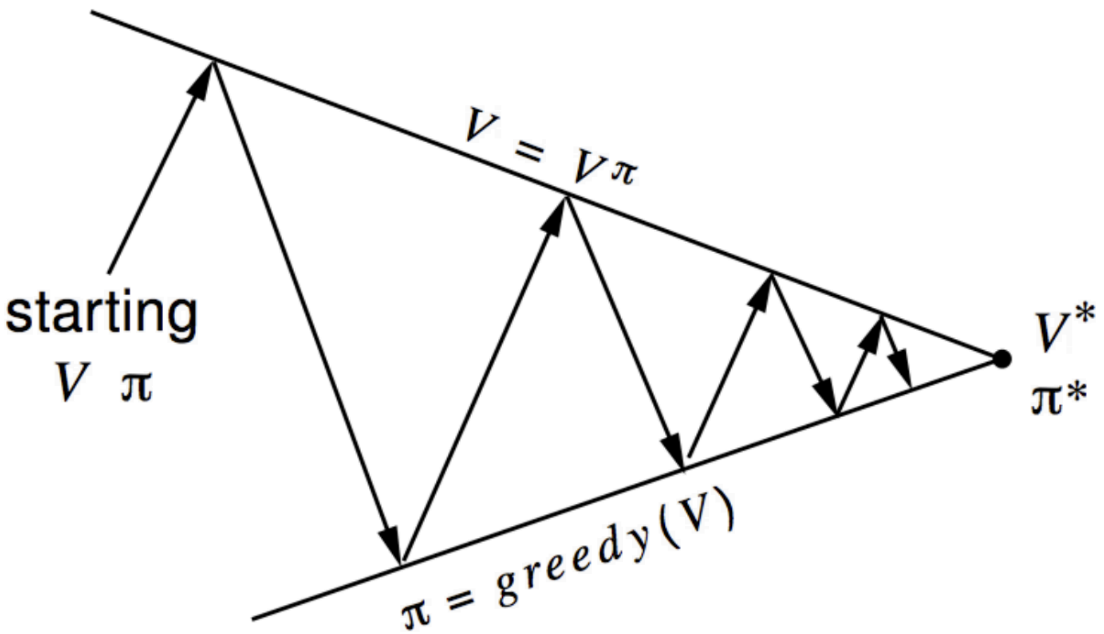
Policy evaluation Estimate v_π

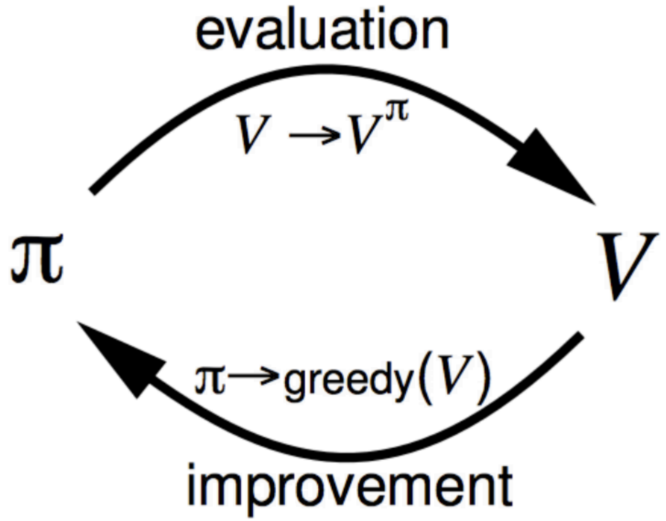
Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$

Greedy policy improvement

Policy iteration

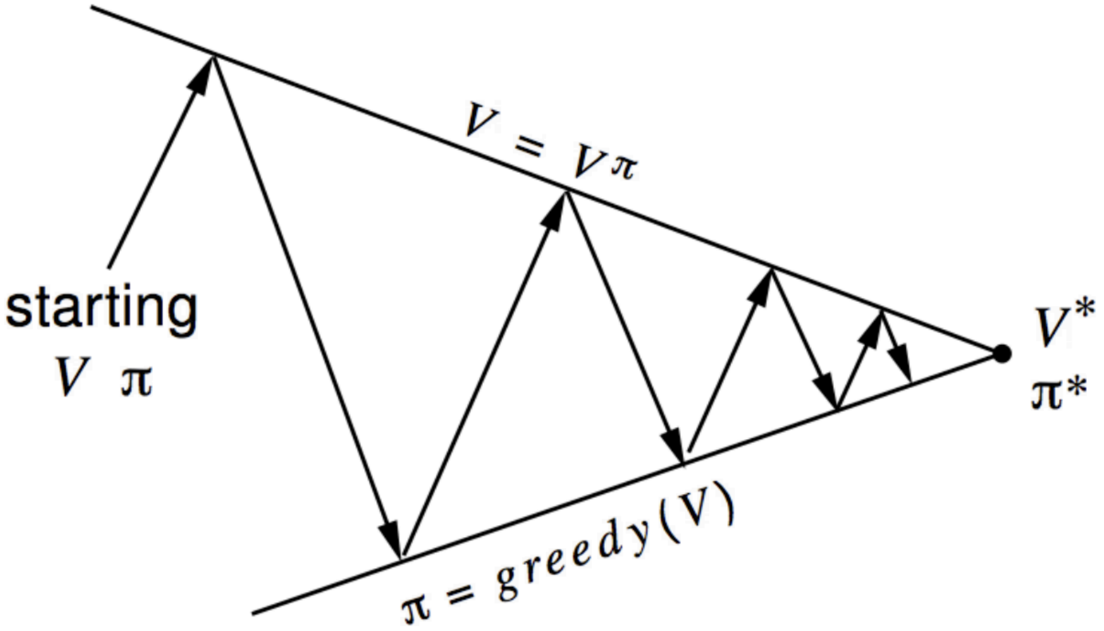


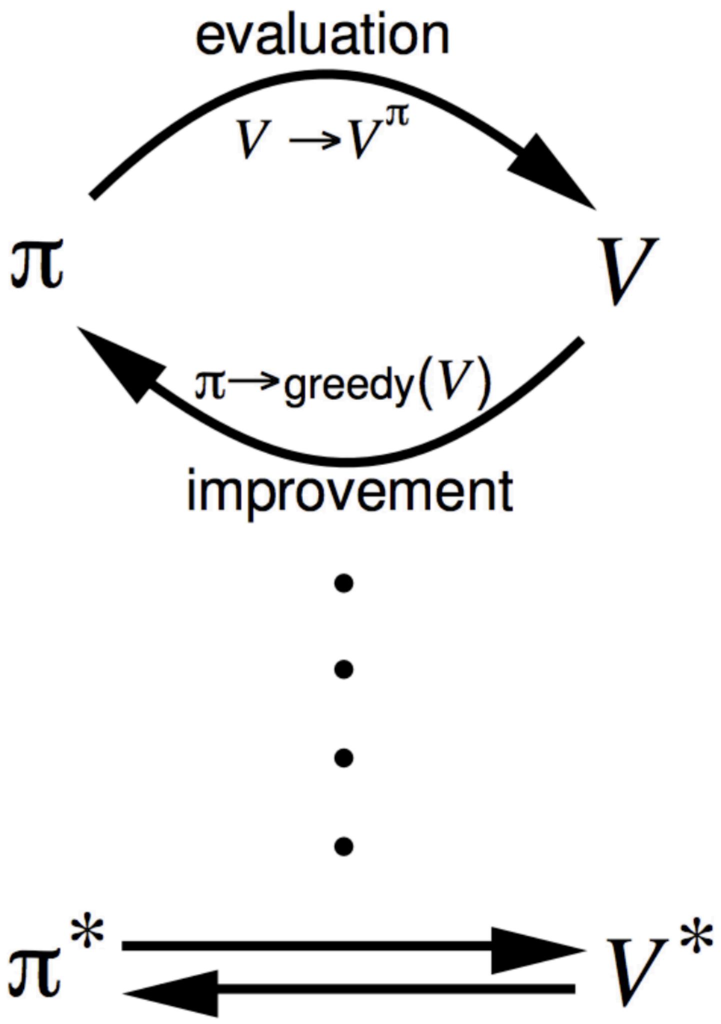


Policy evaluation Estimate v_π
Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$
Greedy policy improvement

Policy iteration

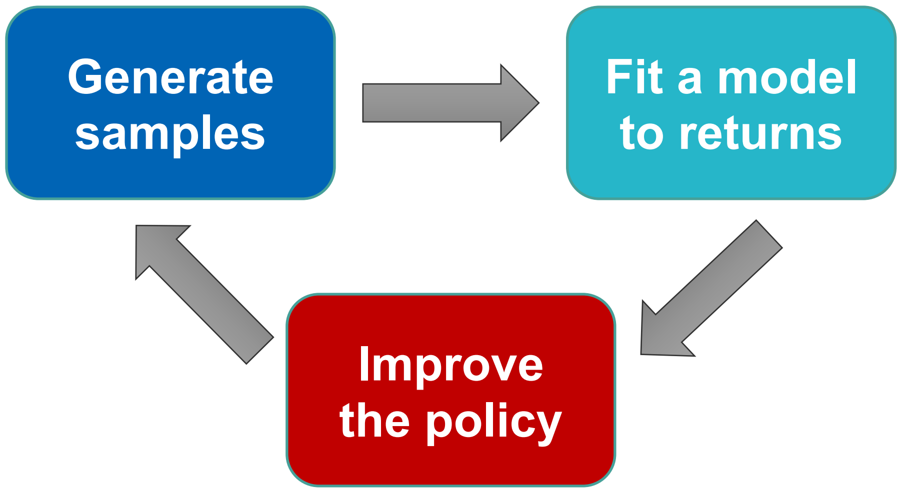




Policy evaluation Estimate v_π
Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$
Greedy policy improvement

Value iteration



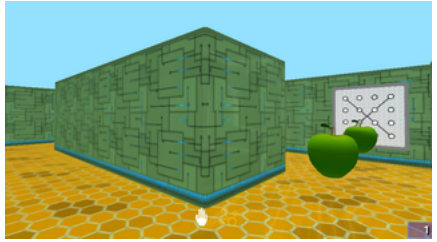
- Can we get rid of the policy?

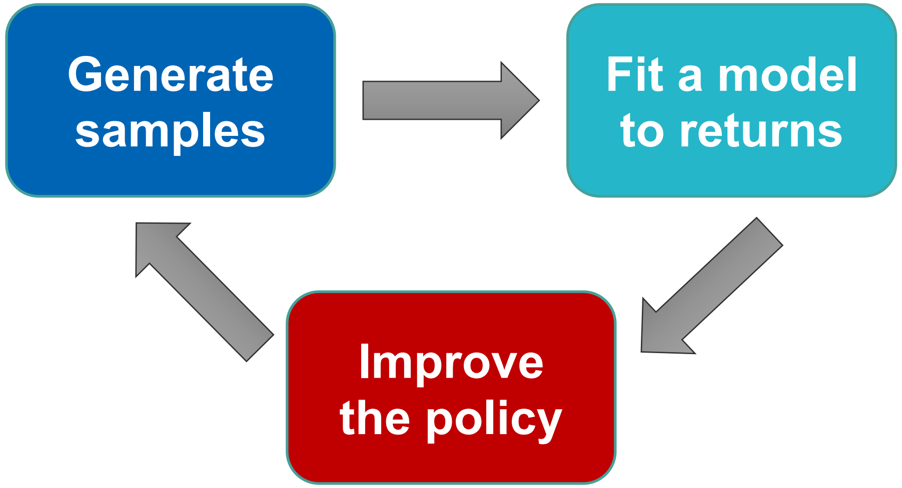
$$\pi'(a_t \mid s_t) = \delta\left(a_t = \arg\max_a \left[Q_\pi(a, s_t)\right]\right)$$

Value iteration:

1. set $Q(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{s' \sim p(s'|s,a)}[V(s')]$
2. set $V(s) \leftarrow \max_a Q(s,a)$ $\leftarrow V^*, Q^*$

Fitted Q-iteration





- If the state space is high-dimensional, let's represent $Q_\phi(s,a)$ with a parametric function instead of a tabular representation.

Fitted Q-iteration:

1. set $y_i \leftarrow r(s_i, a_i) + \gamma \overbrace{\mathbb{E}_{s' \sim p(s'|s,a)}[V_\phi(s'_i)]}^{\approx \max_a Q_\phi(s', a)}$
2. set $\phi \leftarrow \arg\min_\phi \sum_i \|Q_\phi(s_i, a_i) - y_i\|^2$

Fitted Q-iteration

- Here's our policy-independent algorithm:

1. collect a dataset $\{(s_i, a_i, s_i', r_i)\}$ under some policy

Fitted Q-iteration

- Here's our policy-independent algorithm:

1. collect a dataset $\{(s_i, a_i, s'_i, r_i)\}$ under some policy

$\times K$

2. set $y_i \leftarrow r_i + \gamma \max_a Q_\phi(s'_i, a)$
3. set $\phi \leftarrow \arg\min_\phi \sum_i \|Q_\phi(s_i, a_i) - y_i\|^2$

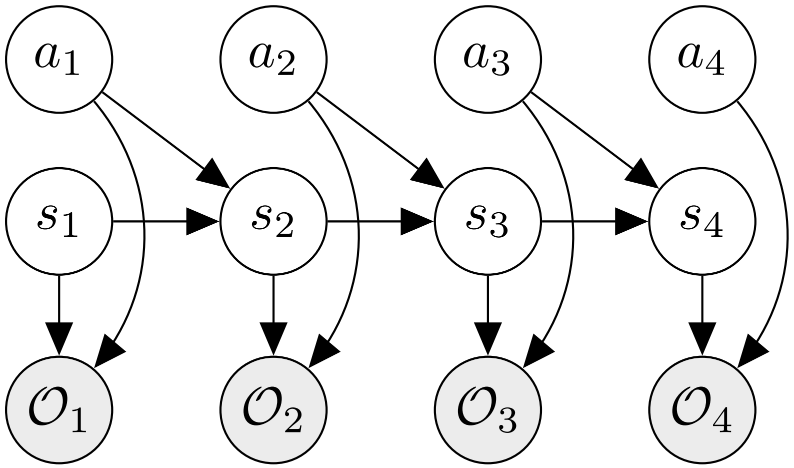
$$\mathcal{E} = \frac{1}{2} E_{(\mathbf{s},\mathbf{a})\sim\beta}\left[Q_\phi(\mathbf{s},\mathbf{a}) - [r(\mathbf{s},\mathbf{a}) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}',\mathbf{a}')]\right]$$

if $\mathcal{E} = 0$, then $\underline{Q_\phi(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a})} + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}',\mathbf{a}')$

this is an *optimal* Q-function, corresponding to optimal policy π'

Soft Policy Gradients and Soft Q-learning

Recap: Control as Inference



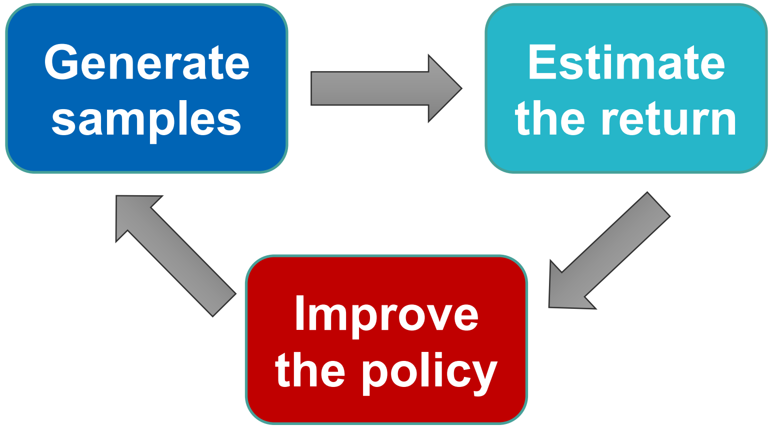
Initial state	$s_0 \sim p_0(s)$
Transition	$s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$
Policy	$a_t \sim \pi(a_t \mid s_t)$
Reward	$r_t = r(s_t, a_t)$
Optimality	$p(\mathcal{O}_t = 1 \mid s_t, a_t) = \exp(r(s_t, a_t))$

Which objective does inference optimize?

$$- D_{\mathrm{KL}}\left(\hat{p}(\tau) \| p(\tau)\right) = \sum_{t=1}^{T} \mathbb{E}_{(s_t, a_t) \sim \hat{p}(s_t, a_t)}[r(s_t, a_t)] + \mathbb{E}_{(s_t) \sim \hat{p}(s_t)}[\mathcal{H}(\pi(a_t \mid s_t))]$$

- For deterministic dynamics, get it directly
- For stochastic dynamics, obtain it from the ELBO on the evidence

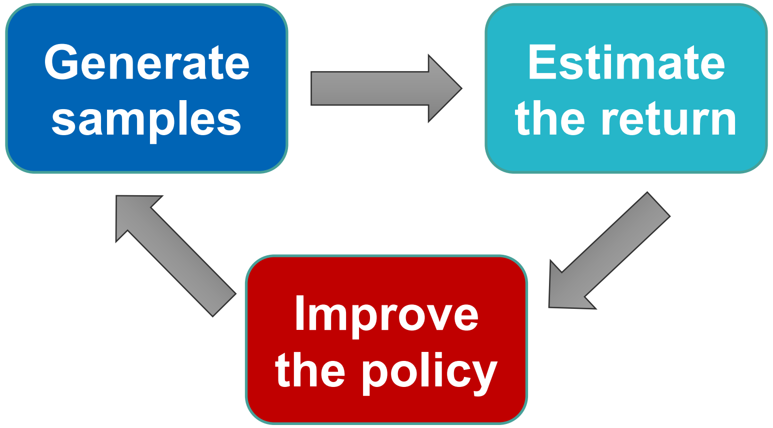
Soft policy gradients



$$\theta^\star = \arg\max_\theta \underbrace{\sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)}[r(s_t,a_t)]}_{J(\theta)}$$

$$\sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)}[r(s_t,a_t)] + \mathbb{E}_{(s_t)\sim \hat{p}(s_t)}[\mathcal{H}(\pi(a_t \mid s_t))] =$$

Soft policy gradients



$$\theta^\star = \arg\max_\theta \underbrace{\sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)}[r(s_t,a_t)]}_{J(\theta)}$$

$$\sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)}[r(s_t,a_t)] + \mathbb{E}_{(s_t)\sim \hat{p}(s_t)}[\mathcal{H}(\pi(a_t \mid s_t))] =$$

$$\nabla_\theta \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)}[r(s_t,a_t) - \log \pi(a_t \mid s_t)]$$

Relationship to Q-learning

optimal policy :

$p(a_t \mid s_t, O_{1:T}) = \exp(Q - \bar{V})$

$Q_\theta(s_t, a_t) - \bar{V}_\theta(s_t)$

$= \log \int \exp Q_\theta$

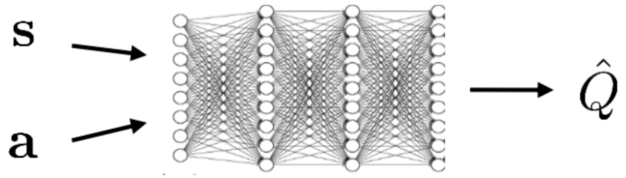
$$\nabla_\theta \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)}[r(s_t,a_t) - \log \underline{\pi(a_t \mid s_t)}]$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \underline{\log \pi_\theta(a_t}\mid s_t)\left(r(s_t,a_t) + \underbrace{\left(\sum_{t'=t+1}^{T} r(s_{t'},a_{t'}) - \log \pi_\theta(a_{t'} \mid s_{t'})\right)}_{\approx \boldsymbol{Q(s_{t+1},a_{t+1})}} - \log \pi_\theta(a_t \mid s_t) - 1\right)$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta Q_\theta(s_t,a_t)\left(\underline{r(s_t,a_t)} + \underbrace{\underline{\operatorname{soft\,max}_{a_{t+1}} Q_\theta(s_{t+1},a_{t+1})}}_{\bar{V}(s_{t+1}) = \log \int \exp Q(s_{t+1},a_{t+1})} - \underline{Q_\theta(s_t,a_t)}\right)$$

Soft Q-learning

Learned (neural network) Q-function: $Q_\theta(\mathbf{s}, \mathbf{a})$



1. collect a dataset $\{(s_i, a_i, s'_i, r_i)\}$ under some policy
2. set $y_i \leftarrow r_i + \gamma \max_a Q_\phi(s'_i, a)$
3. set $\phi \leftarrow \arg\min_\phi \sum_i \|Q_\phi(s_i, a_i) - y_i\|^2$

Q-learning: $\theta \leftarrow \theta + \alpha \nabla_\theta Q_\theta(\mathbf{s}, \mathbf{a})(r(\mathbf{s}, \mathbf{a}) + \gamma V(\mathbf{s}') - Q_\theta(\mathbf{s}, \mathbf{a}))$

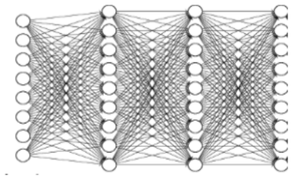
target value: $V(\mathbf{s}') = \max_{\mathbf{a}'} Q_\theta(\mathbf{s}', \mathbf{a}')$

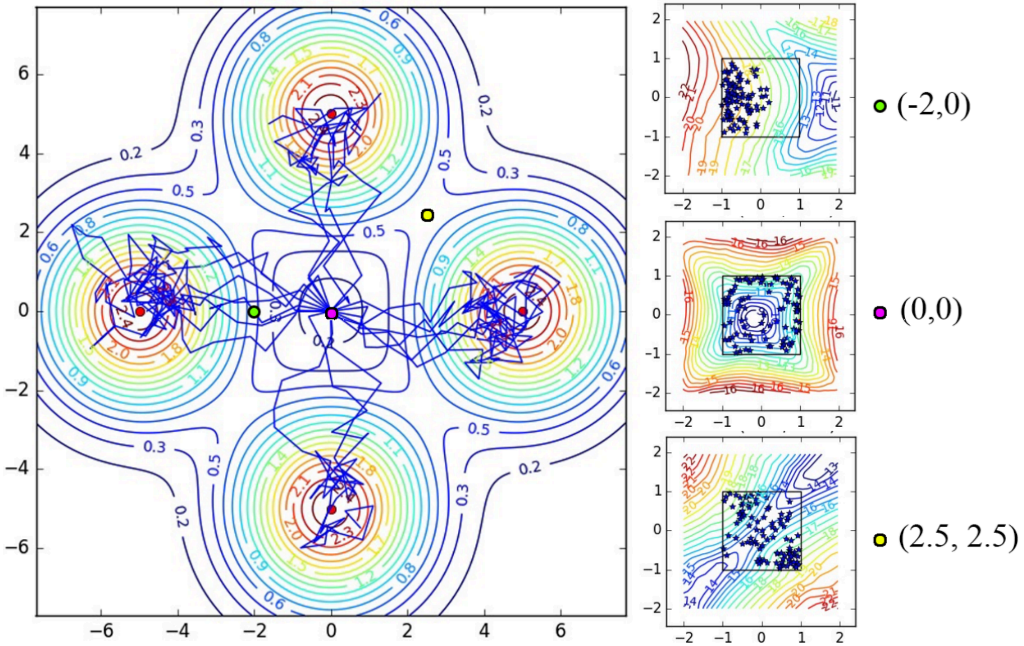
soft Q-learning: $\theta \leftarrow \theta + \alpha \nabla_\theta Q_\theta(\mathbf{s}, \mathbf{a})(r(\mathbf{s}, \mathbf{a}) + \gamma V(\mathbf{s}') - Q_\theta(\mathbf{s}, \mathbf{a}))$

target value: $V(\mathbf{s}') = \text{soft max}_{\mathbf{a}'} Q_\theta(\mathbf{s}', \mathbf{a}') = \log \int \exp(Q_\theta(\mathbf{s}', \mathbf{a}'))d\mathbf{a}'$

Soft Q-learning

Learned (neural network) Q-function: $Q_\theta(\mathbf{s}, \mathbf{a})$





Haarnoja et al. (2017)

Q-learning: $\theta \leftarrow \theta + \alpha \nabla_\theta Q_\theta(\mathbf{s}, \mathbf{a})(r(\mathbf{s}, \mathbf{a}) + \gamma V(\mathbf{s}') - Q_\theta(\mathbf{s}, \mathbf{a}))$

target value: $V(\mathbf{s}') = \max_{\mathbf{a}'} Q_\theta(\mathbf{s}', \mathbf{a}')$

soft Q-learning: $\theta \leftarrow \theta + \alpha \nabla_\theta Q_\theta(\mathbf{s}, \mathbf{a})(r(\mathbf{s}, \mathbf{a}) + \gamma V(\mathbf{s}') - Q_\theta(\mathbf{s}, \mathbf{a}))$

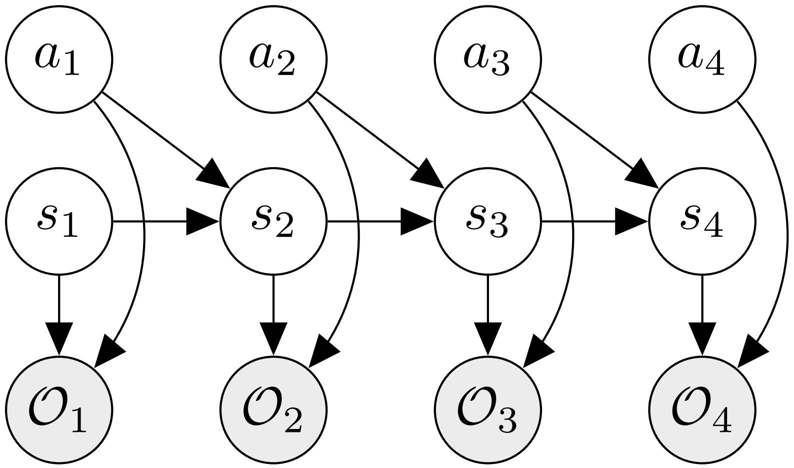
target value: $V(\mathbf{s}') = \text{soft}\max_{\mathbf{a}'} Q_\theta(\mathbf{s}', \mathbf{a}') = \log \int \exp(Q_\theta(\mathbf{s}', \mathbf{a}'))d\mathbf{a}'$

Benefits of soft optimality

- Improves exploration and prevents entropy collapse
- Empirically, policies are easier to finetune for more specific tasks
- Better robustness (due to wider coverage of states)
- Reduces to hard optimality (by increasing the magnitude of the rewards)
- Good model for human behavior

Summary & Takeaways

Takeaways



$$\begin{aligned}
\text{Initial state} \quad & s_0 \sim p_0(s) \\
\text{Transition} \quad & s_{t+1} \sim p(s_{t+1} \mid s_t, a_t) \\
\text{Policy} \quad & a_t \sim \pi(a_t \mid s_t) \\
\text{Reward} \quad & r_t = r(s_t, a_t) \\
\text{Optimality} \quad & p(\mathcal{O}_t = 1 \mid s_t, a_t) = \exp(r(s_t, a_t))
\end{aligned}$$

- **PGM provides a unified perspective** on sequential decision-making problems (RL is just MLE in a corresponding probabilistic model).
- **Recursive optimality relationships in RL have "soft" analogs** that come from message passing in a graphical model.
- **Probabilistic formalism yields new "soft" algorithms that work well in practice.**