CS 4803 / 7643: Deep Learning

Topics:

- Policy Gradients
- Actor Critic

Zsolt Kira

Georgia Tech

Administrative

- PS3/HW3 due Tuesday 03/31
- PS4/HW4 is **optional** and due 04/03
  - There are lots of bonus/Extra credit questions there!

- Sessions with Facebook for project (fill out spreadsheet)

Dhruv Batra & Sameer Dharur	Habitat Embodied Navigation	3/31 (12:30pm EST)
Lucia Specia & Paco Guzman	Transfer learning for machine translation quality estimation	4/1 (11:30am EST)
Matt Muckley & Anuroop Sriram	Attentive Reconstruction/ fastMRI	4/1 (2:00pm EST)
Marc'Aurelio Ranzato	Confident Machine Translation	4/2 (10:30am EST)

Administrative

- How to ask questions during live lecture:
  - Use Q&A window (other students can upvote)
  - Raise hands

Topics we'll cover

- Overview of RL
  - RL vs other forms of learning
  - RL "API"
  - Applications
- Framework: Markov Decision Processes (MDP's)
  - Definitions and notations
  - Policies and Value Functions
  - Solving MDP's
    - Value Iteration (recap)
    - Q-Value Iteration (new)
    - Policy Iteration
- Reinforcement learning
  - Value-based RL (Q-learning, Deep-Q Learning)
  - Policy-based RL (Policy gradients)
  - Actor-Critic

Recap: MDPs

- Markov Decision Processes (MDP):
  - States: $\mathcal{S}$
  - Actions: $\mathcal{A}$
  - Rewards: $\mathcal{R}(s, a, s')$
  - Transition Function: $\mathbb{T}(s, a, s') = p(s'|s, a)$
  - Discount Factor: γ

Value Function

Following policy π that produces sample trajectories s_0, a_0, r_0, s_1, a_1, …

How good is a state?

The **value function** at state s, is the expected cumulative reward from state s (and following the policy thereafter):

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, \pi\right]$$

How good is a state-action pair?

The **Q-value function** at state s and action a, is the expected cumulative reward from taking action a in state s (and following the policy thereafter):

$$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

Optimal Quantities

Given *optimal* policy π^* that produces sample trajectories s_0, a_0, r_0, s_1, a_1, …

How good is a state?

The **optimal value function** at state s, and acting optimally thereafter

$$V^*(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, \pi^*\right]$$

How good is a state-action pair?

The **optimal Q-value function** at state s and action a, is the expected cumulative reward from taking action a in state s and acting optimally thereafter

$$Q^*(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi^*\right]$$

Recap: Optimal Value Function

The **optimal Q-value function** at state s and action a, is the expected cumulative reward from taking action a in state s and acting optimally thereafter

$$Q^*(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi^*\right]$$

Recap: Optimal Value Function

The **optimal Q-value function** at state s and action a, is the expected cumulative reward from taking action a in state s and acting optimally thereafter

$$Q^*(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi^*\right]$$

Optimal policy:

$$\boxed{\pi^*(s) = \arg\max_a Q^*(s,a)}$$

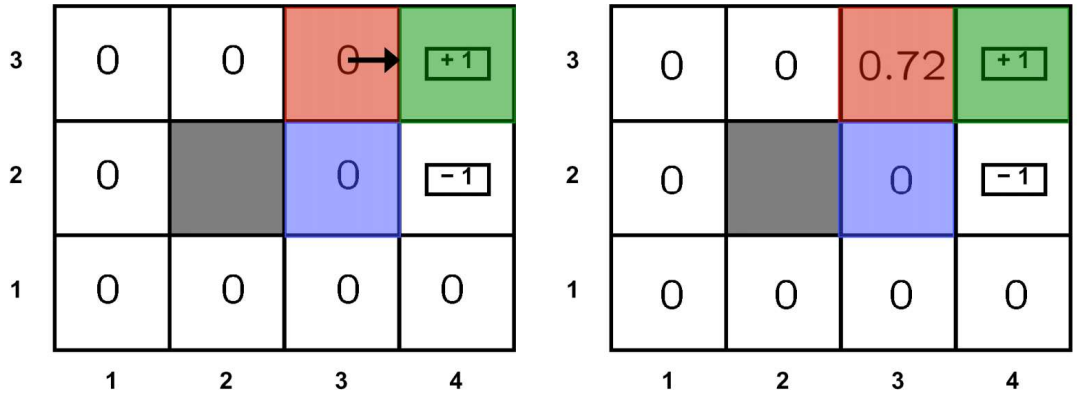
Bellman Optimality Equations

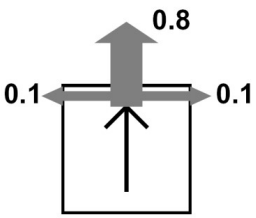
- Relations: $\boxed{V^*(s) = \max_a Q^*(s,a)}$ $\boxed{\pi^*(s) = \arg\max_a Q^*(s,a)}$

- Recursive optimality equations:

$$
\begin{aligned}
Q^*(s,a) &= \mathop{\mathbb{E}}_{s' \sim p(s'|s,a)} \left[r(s,a) + \gamma V^*(s) \right] \\
&= \sum_{s'} p(s'|s,a) \left[r(s,a) + \gamma V^*(s) \right] \\
&= \sum_{s'} p(s'|s,a) \left[r(s,a) + \gamma \max_a Q^*(s',a') \right] \\
V^*(s) &= \max_a \sum_{s'} p(s'|s,a) \left[r(s,a) + \gamma V^*(s') \right]
\end{aligned}
$$

Value Iteration (VI)





$$V^{i+1}(s) \leftarrow \max_a \sum_{s'} p\left(s'|s,a\right)\left[r(s,a) + \gamma V^i\left(s'\right)\right]$$

$$V^2(\langle 3,3\rangle) = \sum_{s'} P\left(s'|\,\text{right}, \langle 3,3\rangle\right)\left[r(\langle 3,3\rangle) + \gamma V^1\left(s'\right)\right]$$

$$= 0.9[0.8 \cdot 1 + 0.1 \cdot 0 + 0.1 \cdot 0]$$

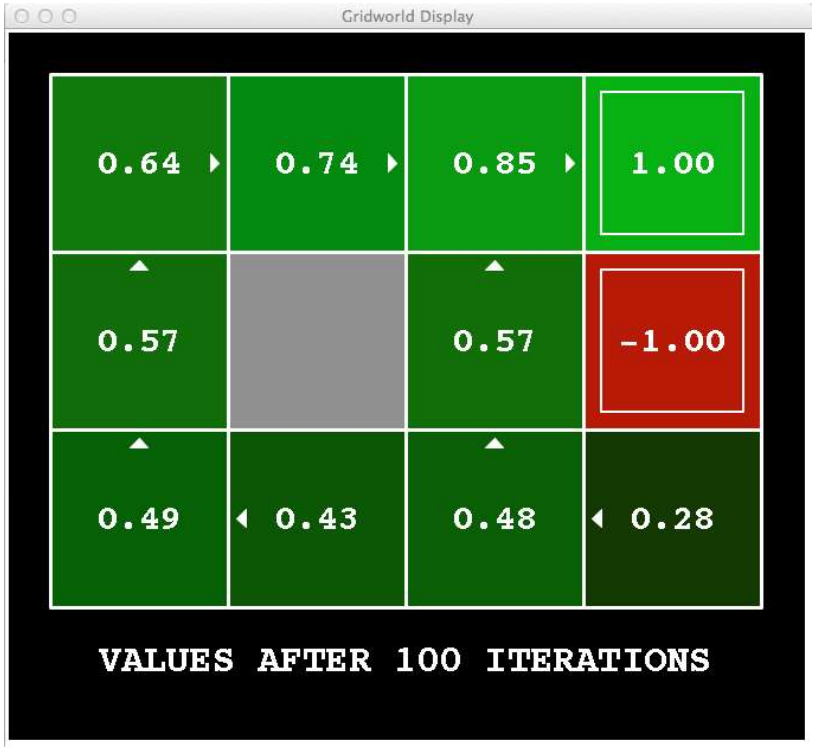
[NOTE: Here we are showing calculations for the action we know is argmax (go right), but in general we have to calculate this for *each* actions and return max]

Slide credit: Pieter Abbeel

3	0	0	0	0
2	0	0	0	0
1	0	0	0	0
	1	2	3	4

$$V^{i+1}(s) \leftarrow \max_a \sum_{s'} p\left(s'|s,a\right)\left[r(s,a) + \gamma V^i\left(s'\right)\right]$$

Snapshot of Demo – Gridworld V Values

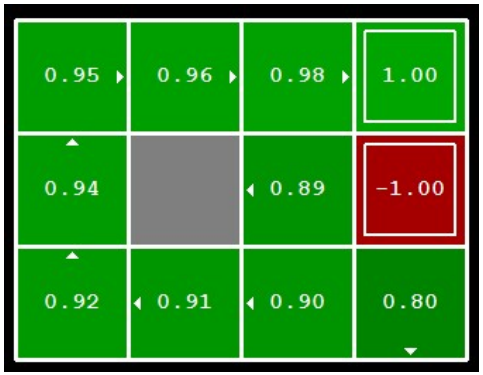


Noise = 0.2
Discount = 0.9
Living reward = 0

Slide Credit: http://ai.berkeley.edu

Computing Actions from Values

- Let's imagine we have the optimal values V*(s)
- How should we act?
  - It's not obvious!
- We need to do a one step calculation

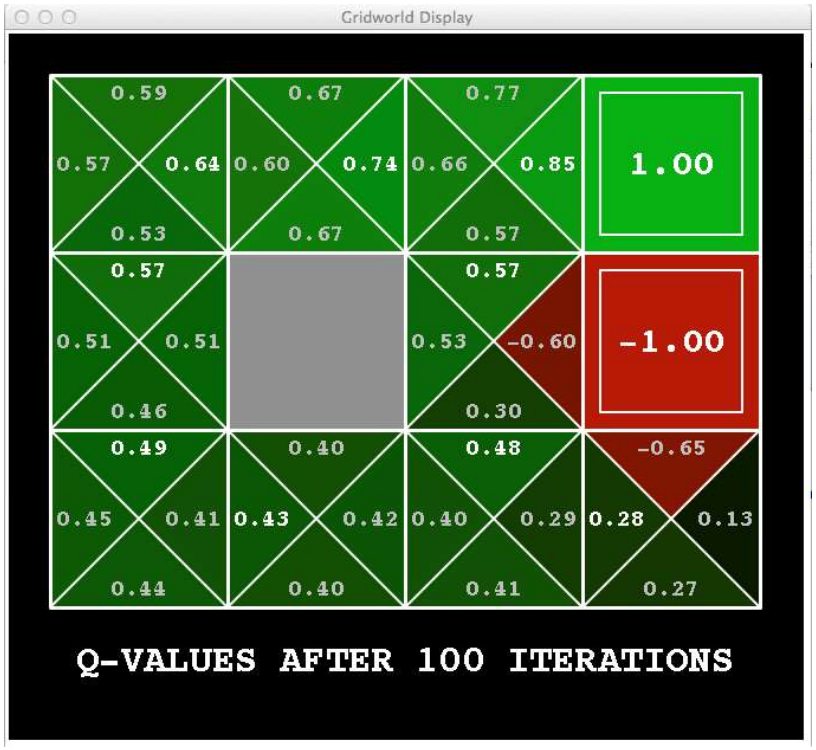


$$\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

- This is called policy extraction, since it gets the policy implied by the values

Slide Credit: http://ai.berkeley.edu

Snapshot of Demo – Gridworld Q Values

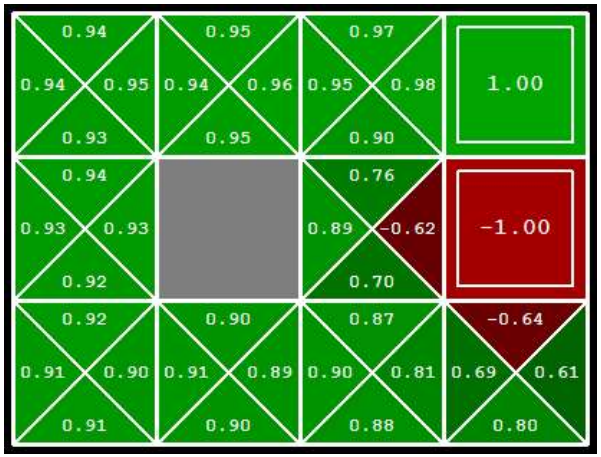


Noise = 0.2
Discount = 0.9
Living reward = 0

Slide Credit: http://ai.berkeley.edu

Computing Actions from Q-Values

- Let's imagine we have the optimal q-values:



- How should we act?
  - Completely trivial to decide!

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

- Important lesson: actions are easier to select from q-values than values!

Slide Credit: http://ai.berkeley.edu

Recap: Learning Based Methods

- Typically, we don't know the environment
  - $\mathbb{T}(s, a, s')$ unknown, how actions affect the environment.
  - $\mathcal{R}(s, a, s')$ unknown, what/when are the good actions?

Recap: Learning Based Methods

- Typically, we don't know the environment
  - $\mathbb{T}(s, a, s')$ unknown, how actions affect the environment.
  - $\mathcal{R}(s, a, s')$ unknown, what/when are the good actions?
- But, we can learn by trial and error.
  - Gather experience (data) by performing actions.

$$\{s, a, s', r\}_{i=1}^{N}$$

  - Approximate unknown quantities from data.

Sample-Based Policy Evaluation?

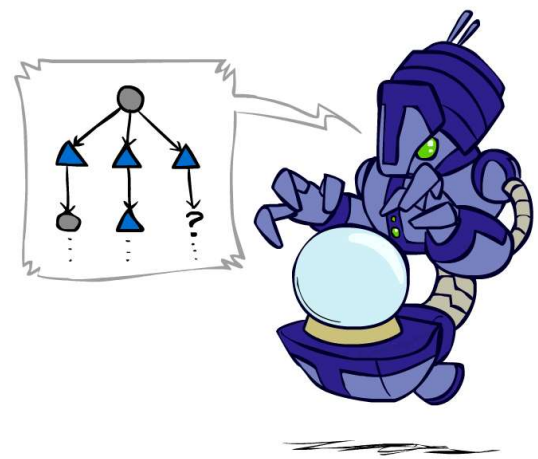
- We want to improve our estimate of V by computing these averages:

$$V^{\pi}_{k+1}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^{\pi}_{k}(s')]$$

- Idea: Take samples of outcomes s' (by doing th

$$sample_1 = R(s, \pi(s), s'_1) + \gamma V^{\pi}_{k}(s'_1)$$
$$sample_2 = R(s, \pi(s), s'_2) + \gamma V^{\pi}_{k}(s'_2)$$
$$\ldots$$
$$sample_n = R(s, \pi(s), s'_n) + \gamma V^{\pi}_{k}(s'_n)$$

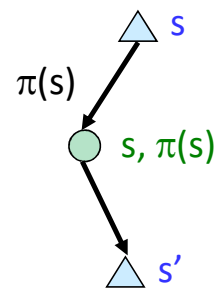
$$V^{\pi}_{k+1}(s) \leftarrow \frac{1}{n} \sum_{i} sample_i$$



What's the difficulty of this algorithm?

Temporal Difference Learning

- Big idea: learn from every experience!
  - Update V(s) each time we experience a transition (s, a, s', r)
  - Likely outcomes s' will contribute updates more often



- Temporal difference learning of values
  - Policy still fixed, still doing evaluation!
  - Move values toward value of whatever successor occurs: running average

Sample of V(s): $sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$

Update to V(s): $V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$

Same update: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

Deep Q-Learning

- Q-Learning with linear function approximators

$$Q(s, a; w, b) = w_a^\top s + b_a$$

  - Has some theoretical guarantees

- Deep Q-Learning: Fit a deep Q-Network

  - Works well in practice

  - Q-Network can take RGB images

$$Q(s, a; \theta)$$

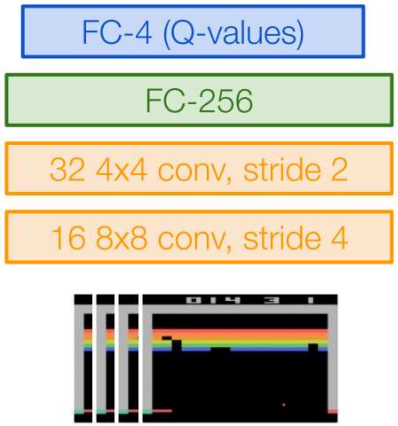


Image Credits: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Recap: Deep Q-Learning

- Collect a dataset $\{(s, a, s', r)_i\}_{i=1}^{N}$
- Loss for a single data point:

$$\text{MSE Loss} := \Big(\underbrace{Q_{new}(s,a)}_{\text{Predicted Q-Value}} - \underbrace{(r + \max_a Q_{old}(s',a))}_{\text{Target Q-Value}}\Big)^2$$

- Act optimally according to the learnt Q function:

$$\pi(s) = \underbrace{\arg\max_{a \in \mathcal{A}} Q(s,a)}_{\text{Pick action with best Q value}}$$

Exploration Problem

- What should π_{gather} be?
  - Greedy? -> Local minimas, no exploration

$$\arg\max_a Q(s, a; \theta)$$

- An exploration strategy:

$$\epsilon\text{-greedy}$$

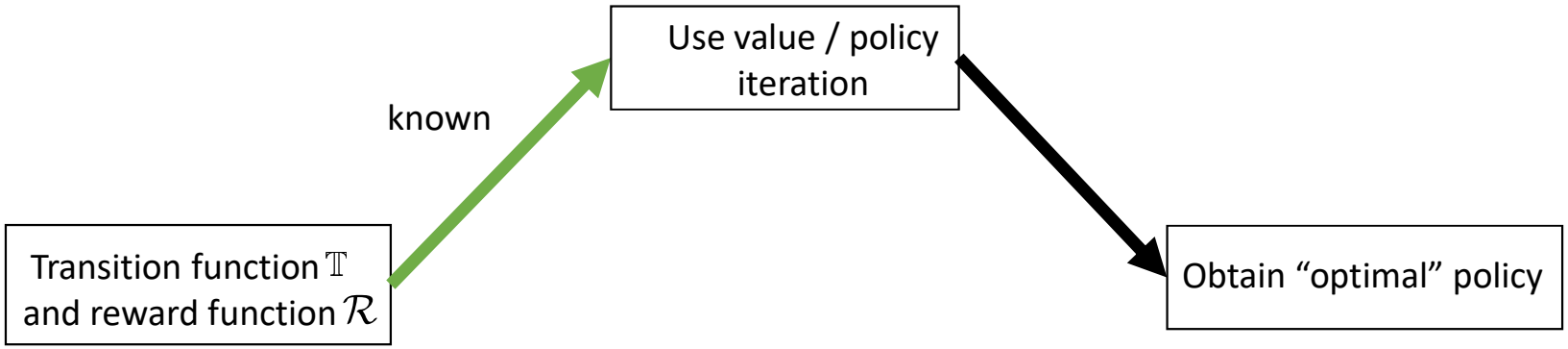
- $$a_t = \begin{cases} \arg\max_a Q(s, a) & \text{with probability } 1 - \epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$$

Experience Replay

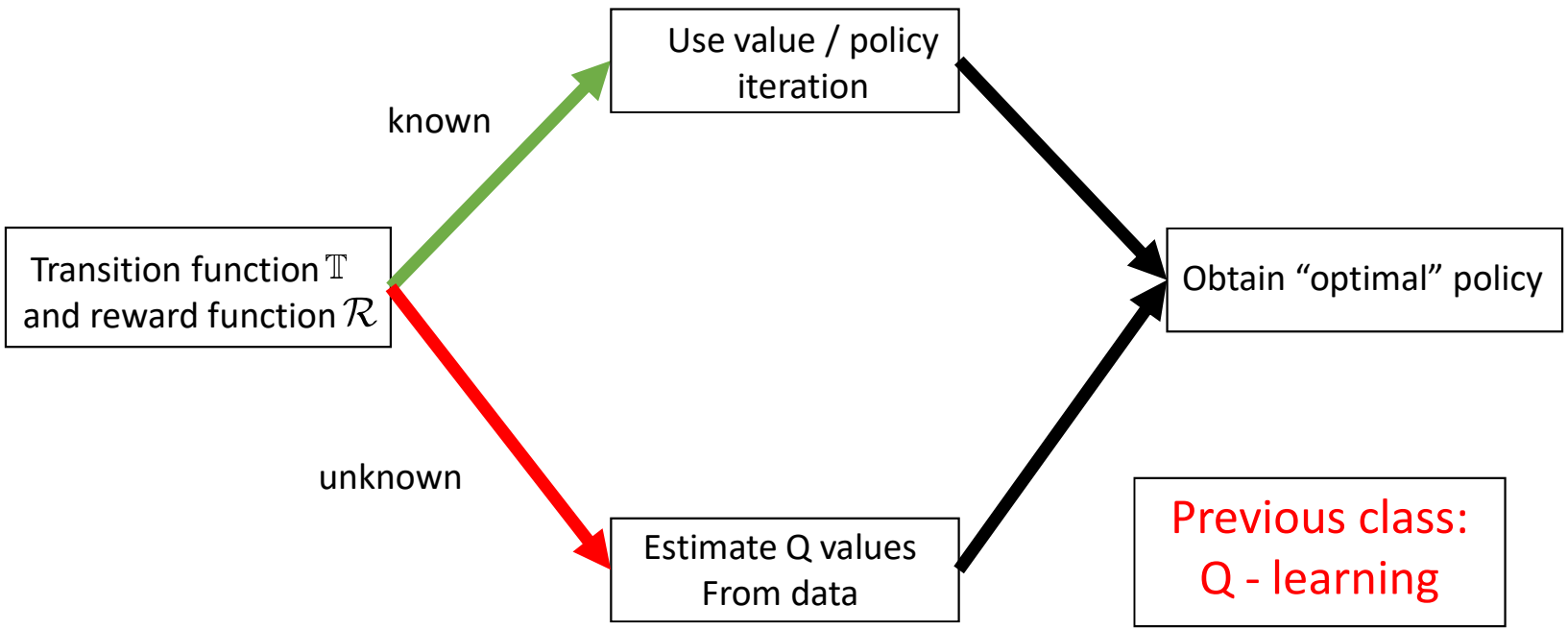
- Address this problem using experience replay
  - A replay buffer stores transitions (s, a, s', r)
  - Continually update replay buffer as game (experience) episodes are played, older samples discarded
  - Train Q-network on random minibatches of transitions from the replay memory, instead of consecutive samples

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

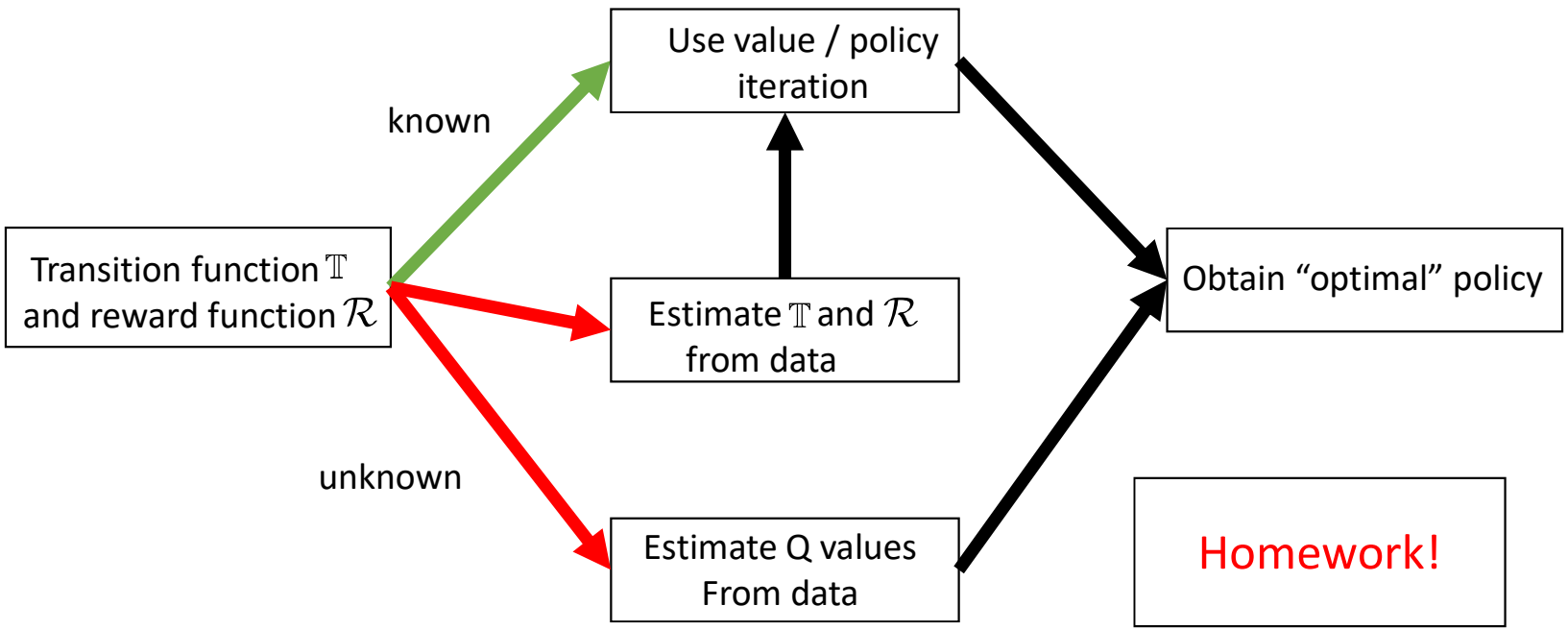
Getting to the optimal policy



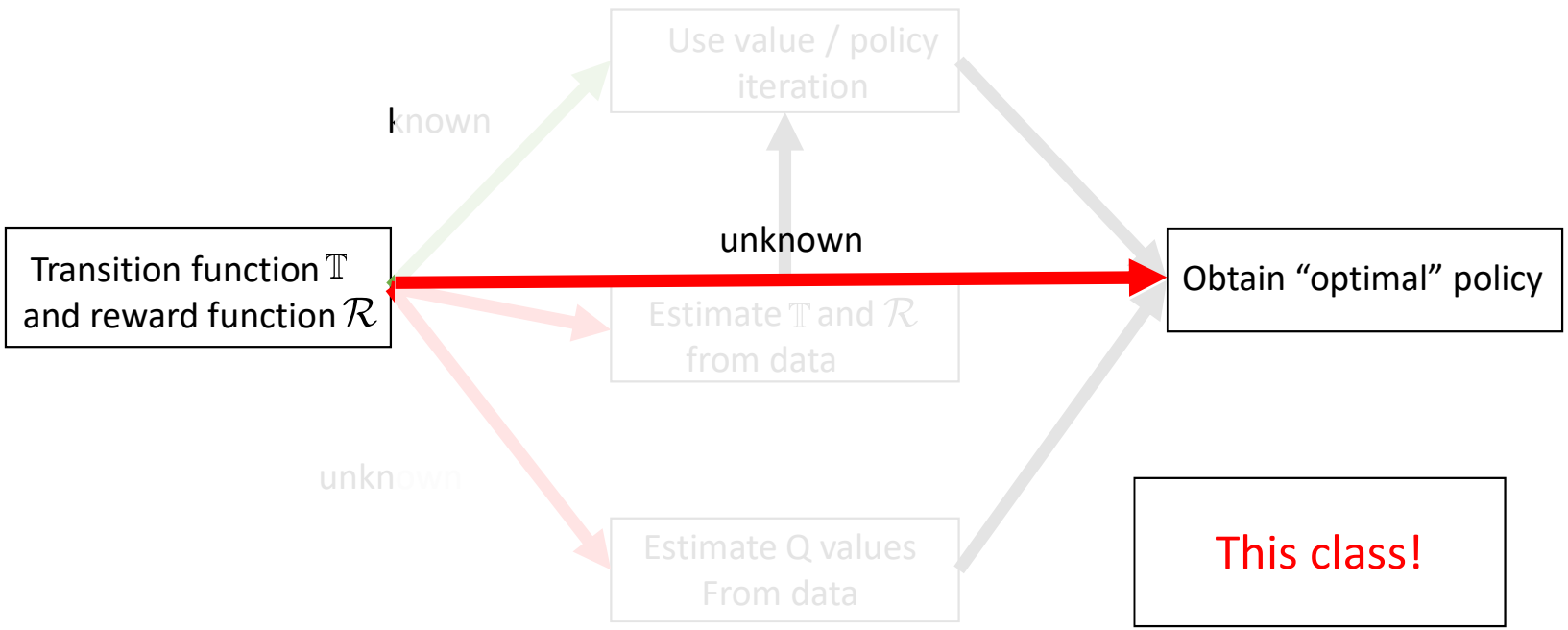
Getting to the optimal policy



Getting to the optimal policy



Getting to the optimal policy



Learning the optimal policy

- Class of policies defined by parameters θ

$$\pi_\theta(a|s) : \mathcal{S} \to \mathcal{A}$$

  - Eg: θ can be parameters of linear transformation, deep network, etc.

Learning the optimal policy

- Class of policies defined by parameters θ

$$\pi_\theta(a|s) : \mathcal{S} \to \mathcal{A}$$

  - Eg: θ can be parameters of linear transformation, deep network, etc.
- Want to maximize:

$$J(\pi) = \mathbb{E}\left[\sum_{t=1}^{T} \mathcal{R}(s_t, a_t)\right]$$

Learning the optimal policy

- Class of policies defined by parameters θ

$$\pi_\theta(a|s) : \mathcal{S} \to \mathcal{A}$$

  - Eg: θ can be parameters of linear transformation, deep network, etc.
- Want to maximize:

$$J(\pi) = \mathbb{E}\left[\sum_{t=1}^{T} \mathcal{R}(s_t, a_t)\right]$$

- In other words,

$$\pi^* = \arg\max_{\pi:\mathcal{S}\to\mathcal{A}} \mathbb{E}\left[\sum_{t=1}^{T} \mathcal{R}(s_t, a_t)\right] \Rightarrow \theta^* = \arg\max_{\theta} \mathbb{E}\left[\sum_{t=1}^{T} \mathcal{R}(s_t, a_t)\right]$$

Learning the optimal policy

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\mathcal{R}(\tau) \right]$$

$$= \mathbb{E}_{a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)} \left[\sum_{t=0}^{T} \mathcal{R}(s_t, a_t) \right]$$

Sample a few trajectories $\{\tau_i\}_{i=1}^{N}$ by acting according to π_θ

$$\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} r(s_t^i, a_t^i)$$

REINFORCE algorithm

Mathematically, we can write:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[r(\tau) \right]$$
$$= \int_{\tau} r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

Where r(τ) is the reward of a trajectory $\tau = (s_0, a_0, r_0, s_1, \ldots)$

REINFORCE algorithm

Expected reward:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[r(\tau) \right]$$
$$= \int_{\tau} r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

REINFORCE algorithm

Expected reward: $$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right] = \int_{\tau} r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

Now let's differentiate this: $$\nabla_\theta J(\theta) = \int_{\tau} r(\tau) \nabla_\theta p(\tau;\theta) \mathrm{d}\tau$$

REINFORCE algorithm

Expected reward: $$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right] = \int_{\tau} r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

Now let's differentiate this: $$\nabla_\theta J(\theta) = \int_{\tau} r(\tau) \nabla_\theta p(\tau;\theta) \mathrm{d}\tau$$

Intractable! Expectation of gradient is problematic when p depends on θ

REINFORCE algorithm

Expected reward: $$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right] = \int_{\tau} r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

Now let's differentiate this: $$\nabla_\theta J(\theta) = \int_{\tau} r(\tau) \nabla_\theta p(\tau;\theta) \mathrm{d}\tau$$

Intractable! Expectation of gradient is problematic when p depends on θ

However, we can use a nice trick: $$\nabla_\theta p(\tau;\theta) = p(\tau;\theta) \frac{\nabla_\theta p(\tau;\theta)}{p(\tau;\theta)} = p(\tau;\theta) \nabla_\theta \log p(\tau;\theta)$$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

REINFORCE algorithm

Expected reward: $$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right] = \int_{\tau} r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

Now let's differentiate this: $$\nabla_\theta J(\theta) = \int_{\tau} r(\tau) \nabla_\theta p(\tau;\theta) \mathrm{d}\tau$$

Intractable! Expectation of gradient is problematic when p depends on θ

However, we can use a nice trick: $$\nabla_\theta p(\tau;\theta) = p(\tau;\theta) \frac{\nabla_\theta p(\tau;\theta)}{p(\tau;\theta)} = p(\tau;\theta) \nabla_\theta \log p(\tau;\theta)$$

If we inject this back:

$$\nabla_\theta J(\theta) = \int_{\tau} \left(r(\tau) \nabla_\theta \log p(\tau;\theta)\right) p(\tau;\theta) \mathrm{d}\tau = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau) \nabla_\theta \log p(\tau;\theta)\right]$$

Can estimate with Monte Carlo sampling

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

REINFORCE algorithm

Can we compute those quantities without knowing the transition probabilities?

We have: $p(\tau;\theta) = \prod_{t \geq 0} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$

REINFORCE algorithm

Can we compute those quantities without knowing the transition probabilities?

We have: $p(\tau;\theta) = \prod_{t\geq 0} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$

Thus: $\log p(\tau;\theta) = \sum_{t\geq 0} \log p(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t)$

REINFORCE algorithm

Can we compute those quantities without knowing the transition probabilities?

We have: $p(\tau;\theta) = \prod_{t\geq 0} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$

Thus: $\log p(\tau;\theta) = \sum_{t\geq 0} \log p(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t)$

And when differentiating: $\nabla_\theta \log p(\tau;\theta) = \sum_{t\geq 0} \nabla_\theta \log \pi_\theta(a_t|s_t)$

Doesn't depend on transition probabilities!

$$\nabla_\theta J(\theta) = \int_\tau (r(\tau)\nabla_\theta \log p(\tau;\theta))\, p(\tau;\theta)\mathrm{d}\tau$$
$$= \mathbb{E}_{\tau\sim p(\tau;\theta)}\left[r(\tau)\nabla_\theta \log p(\tau;\theta)\right]$$

REINFORCE algorithm

Can we compute those quantities without knowing the transition probabilities?

We have: $p(\tau;\theta) = \prod_{t\geq 0} p(s_{t+1}|s_t,a_t)\pi_\theta(a_t|s_t)$

Thus: $\log p(\tau;\theta) = \sum_{t\geq 0} \log p(s_{t+1}|s_t,a_t) + \log \pi_\theta(a_t|s_t)$

And when differentiating: $\nabla_\theta \log p(\tau;\theta) = \sum_{t\geq 0} \nabla_\theta \log \pi_\theta(a_t|s_t)$ Doesn't depend on transition probabilities!

Therefore when sampling a trajectory τ, we can estimate J(θ) with

$$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} r(\tau)\nabla_\theta \log \pi_\theta(a_t|s_t)$$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Policy Gradients

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[\underbrace{\nabla_\theta \log \pi_\theta(\tau)}\mathcal{R}(\tau)]$$

$$\nabla_\theta \left[\cancel{\log p(s_0)} + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \sum_{t=1}^{T} \cancel{\log p(s_{t+1} \mid s_t, a_t)} \right]$$

Doesn't depend on Transition probabilities!

Policy Gradients

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[\underbrace{\nabla_\theta \log \pi_\theta(\tau)}_{\nabla_\theta \left[\cancel{\log p(s_0)} + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \sum_{t=1}^{T} \cancel{\log p(s_{t+1} \mid s_t, a_t)}\right]}\mathcal{R}(\tau)]$$

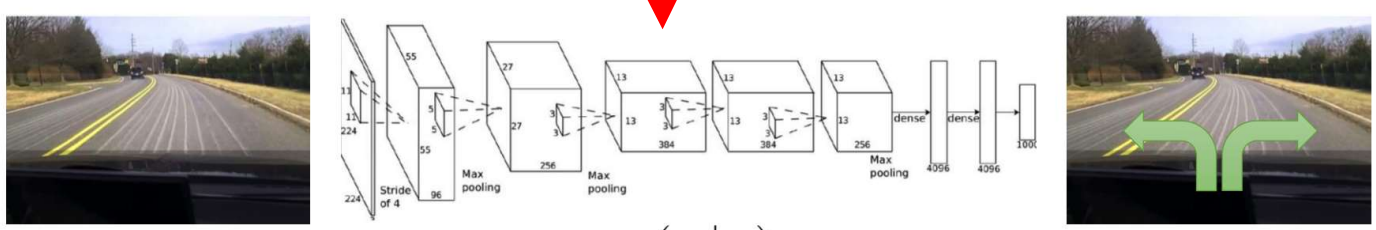
$$= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \sum_{t=1}^{T} \mathcal{R}(s_t, a_t) \right]$$

Policy Gradients

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[\underbrace{\nabla_\theta \log \pi_\theta(\tau)}\mathcal{R}(\tau)]$$

$$\nabla_\theta \left[\cancel{\log p(s_0)} + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \sum_{t=1}^{T} \cancel{\log p(s_{t+1} \mid s_t, a_t)} \right]$$

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



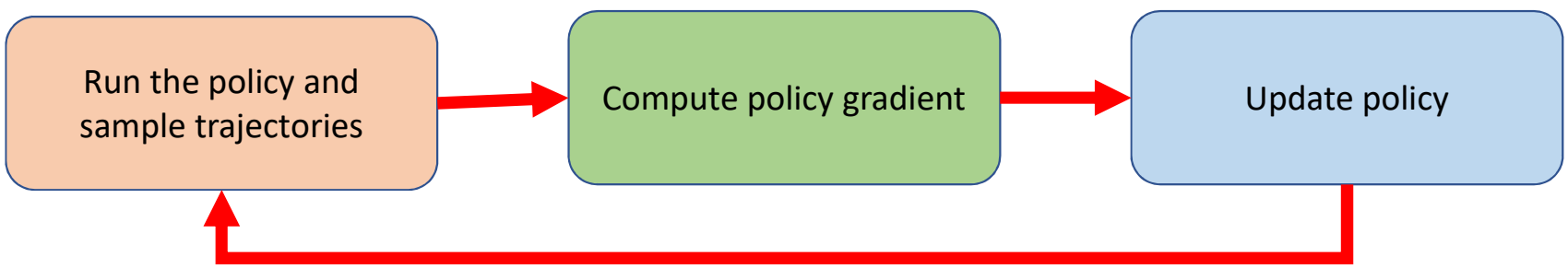
$\mathbf{s}_t$ $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ $\mathbf{a}_t$

REINFORCE

1. Sample trajectories $\tau_i = \{s_1, a_1, \ldots s_T, a_T\}_i$ by acting according to π_θ
2. Compute policy gradient as

$$\nabla_\theta J(\theta) \approx \sum_i \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i) \cdot \sum_{t=1}^{T} \mathcal{R}(s_t^i \mid a_t^i) \right]$$

3. Update policy $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



Slide credit: Sergey Levine

Pong from pixels

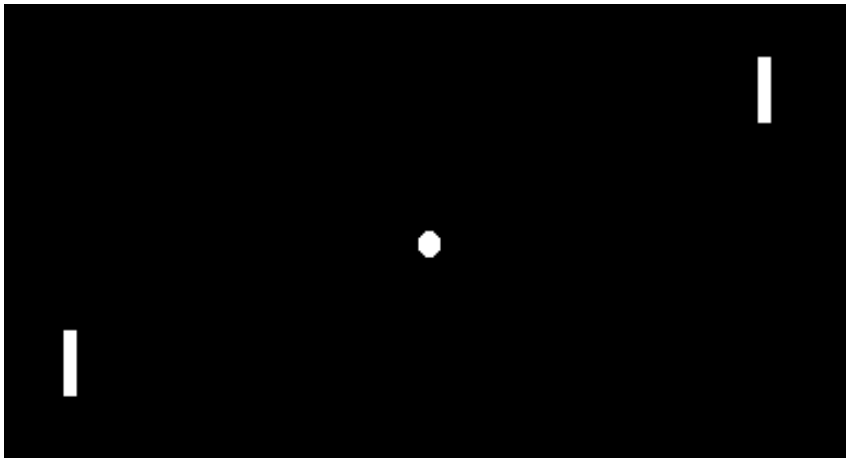


Image Credit: http://karpathy.github.io/2016/05/31/rl/

Pong from pixels

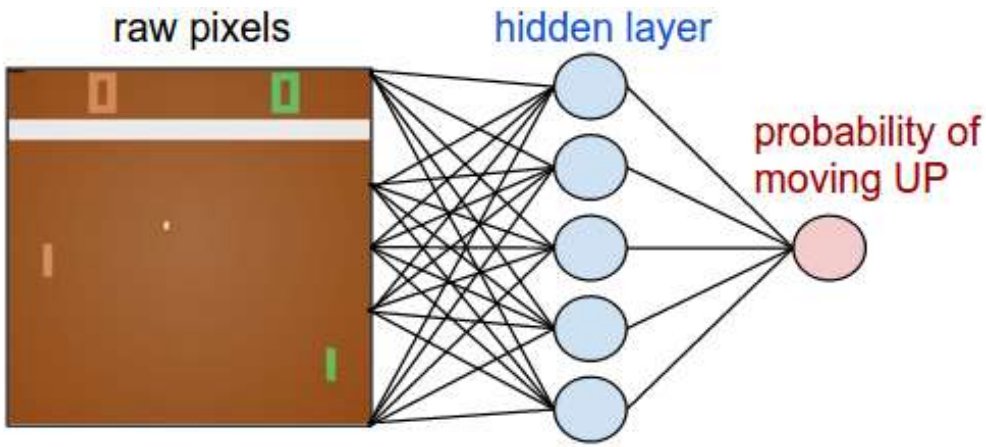
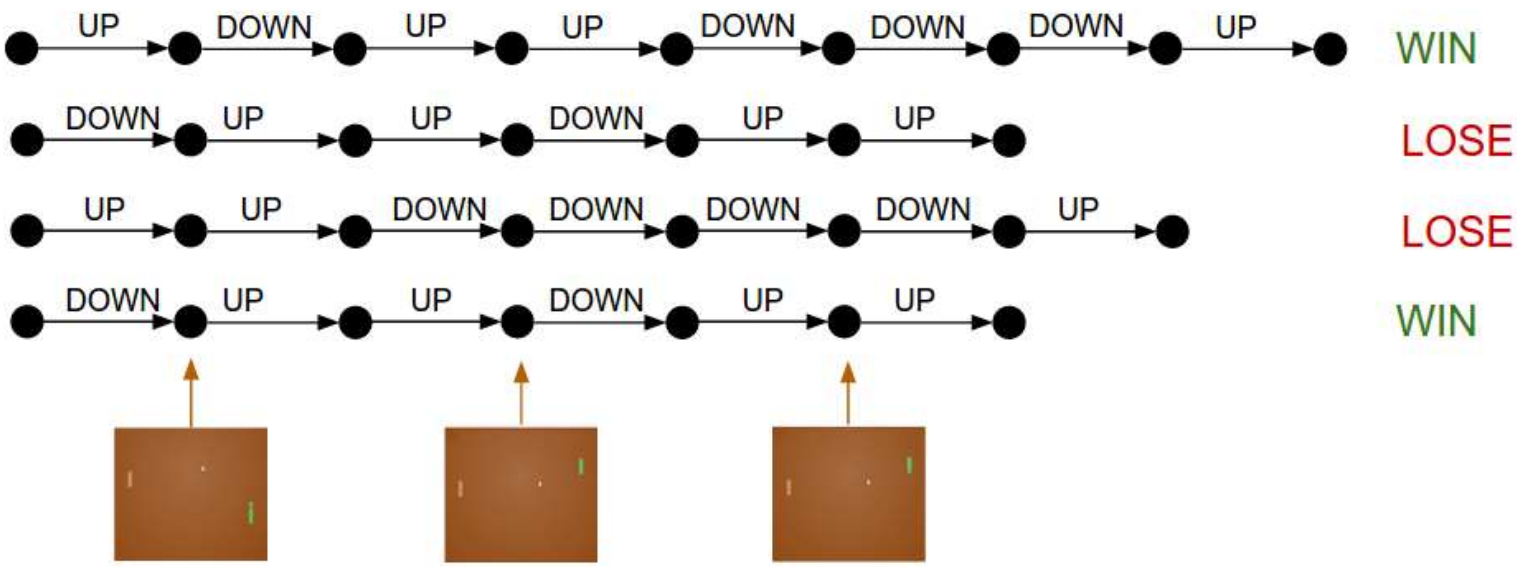


Image Credit: http://karpathy.github.io/2016/05/31/rl/

Intuition


UP DOWN UP UP DOWN DOWN DOWN UP
WIN
DOWN UP UP DOWN UP UP
LOSE
UP UP DOWN DOWN DOWN DOWN UP
LOSE
DOWN UP UP DOWN UP UP
WIN

Policy Gradients

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[\underbrace{\nabla_\theta \log \pi_\theta(\tau)}\mathcal{R}(\tau)]$$

$$\nabla_\theta \left[\cancel{\log p(s_0)} + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \sum_{t=1}^{T} \cancel{\log p(s_{t+1} \mid s_t, a_t)} \right]$$

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

Formalizes notion of "trial and error":

- If reward is high, probability of actions seen is increased
- If reward is low, probability of actions seen is reduced
- But in expectation, it averages out

Issues with Policy Gradients

- Credit assignment is hard!
  - Which specific action led to increase in reward
  - Suffers from high variance → leading to unstable training
- How to reduce the variance?
  - Subtract a constant from the reward!

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \sum_{t=1}^{T} \mathcal{R}(s_t, a_t) - b \right]$$

Issues with Policy Gradients

- Credit assignment is hard!
  - Which specific action led to increase in reward
  - Suffers from high variance → leading to unstable training
- How to reduce the variance?
  - Subtract a constant from the reward!

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \sum_{t=1}^{T} \mathcal{R}(s_t, a_t) - b \right]$$

  - Why does it work?
  - What is the best choice of b?

Homework!

Variance reduction

Gradient estimator: $$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

First idea: Push up probabilities of an action seen, only by the cumulative future reward from that state

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} r_{t'} \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Variance reduction

Gradient estimator: $$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

First idea: Push up probabilities of an action seen, only by the cumulative future reward from that state

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} r_{t'} \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Second idea: Use discount factor γ to ignore delayed effects

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Issues with Policy Gradients

- Credit assignment is hard!
  - Which specific action led to increase in reward
  - Suffers from high variance → leading to unstable training

How to choose the baseline?

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

How to choose the baseline?

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

A: Q-function and value function!

How to choose the baseline?

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

A: Q-function and value function!

Intuitively, we are happy with an action a_t in a state s_t if $Q^\pi(s_t, a_t) - V^\pi(s_t)$ is large. On the contrary, we are unhappy with an action if it's small.

How to choose the baseline?

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

A: Q-function and value function!

Actor-Critic

- Learn both policy and Q function
  - Use the “actor” to sample trajectories
  - Use the Q function to “evaluate” or “critic” the policy

Actor-Critic

- Learn both policy and Q function
  - Use the "actor" to sample trajectories
  - Use the Q function to "evaluate" or "critic" the policy

- REINFORCE: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)\mathcal{R}(s,a)\right]$

- Actor-critic: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)Q^{\pi_\theta}(s,a)\right]$

Actor-Critic

- Learn both policy and Q function
  - Use the "actor" to sample trajectories
  - Use the Q function to "evaluate" or "critic" the policy

- REINFORCE: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)\mathcal{R}(s,a)\right]$

- Actor-critic: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)Q^{\pi_\theta}(s,a)\right]$

- Q function is unknown too! Update using $\mathcal{R}(s,a)$

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$
- For each step:
  - Sample reward $\mathcal{R}(s, a)$ and next state $s' \sim p(s'|s, a)$

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$
- For each step:
  - Sample reward $\mathcal{R}(s, a)$ and next state $s' \sim p(s'|s, a)$
  - evaluate "actor" using "critic" $Q_\beta(s, a)$

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$
- For each step:
  - Sample reward $\mathcal{R}(s, a)$ and next state $s' \sim p(s'|s, a)$
  - evaluate "actor" using "critic" $Q_\beta(s, a)$ and update policy:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a \mid s) Q_\beta(s, a)$$

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$
- For each step:
  - Sample reward $\mathcal{R}(s,a)$ and next state $s' \sim p(s'|s,a)$
  - evaluate "actor" using "critic" $Q_\beta(s,a)$ and update policy:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a \mid s) Q_\beta(s,a)$$

  - Update "critic":
    - Recall Q-learning

$$\text{MSE Loss} := \left(Q_{new}(s,a) - (r + \max_a Q_{old}(s',a)) \right)^2$$

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$
- For each step:
  - Sample reward $\mathcal{R}(s,a)$ and next state $s' \sim p(s'|s,a)$
  - evaluate "actor" using "critic" $Q_\beta(s,a)$ and update policy:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a \mid s) Q_\beta(s,a)$$

  - Update "critic":
    - Recall Q-learning

$$\text{MSE Loss} := \left(Q_{new}(s,a) - (r + \max_a Q_{old}(s',a)) \right)^2$$

    - Update β Accordingly
- $a \leftarrow a', s \leftarrow s'$

Actor-critic

- In general, replacing the policy evaluation or the "critic" leads to different flavors of the actor-critic
  - REINFORCE:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)\mathcal{R}(s,a)\right]$$

  - Q – Actor Critic

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)Q^{\pi_\theta}(s,a)\right]$$

How to choose the baseline?

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

A: Q-function and value function!

Intuitively, we are happy with an action a_t in a state s_t if $Q^\pi(s_t, a_t) - V^\pi(s_t)$ is large. On the contrary, we are unhappy with an action if it's small.

Using this, we get the estimator: $\nabla_\theta J(\theta) \approx \sum_{t \geq 0} (Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)) \nabla_\theta \log \pi_\theta(a_t|s_t)$

Actor-critic

- In general, replacing the policy evaluation or the "critic" leads to different flavors of the actor-critic
  - REINFORCE:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)\mathcal{R}(s,a)\right]$$

  - Q – Actor Critic

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)Q^{\pi_\theta}(s,a)\right]$$

  - Advantage Actor Critic:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)A^{\pi_\theta}(s,a)\right]$$

$$= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$

"how much better is an action than expected?

Summary

- Policy Learning:
  - Policy gradients
  - REINFORCE
  - Reducing Variance (Homework!)
- Actor-Critic:
  - Other ways of performing “policy evaluation”
  - Variants of Actor-critic