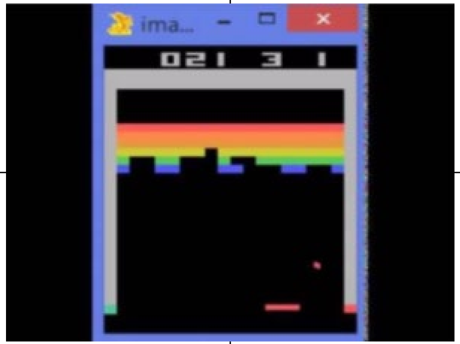
lecture 1

optimization

- Out of relative decisions → yield the decision with the best outcome

delayed consequences

- No immediate outcome feedback
- Induces credit assignment problem



exploration

- "Agent as a scientist"
- Reward predictable only for what the system has experienced (= outcomes based on previous decisions)
- Vs. exploitation

generalization

- Too much representations without generalization → require too much computing power
- Use a higher level representation of given task

lecture 1

Markov assumption

$$p(s_{t+1}|s_t, a_t) = p(s_{t+1}|h_t, a_t)$$

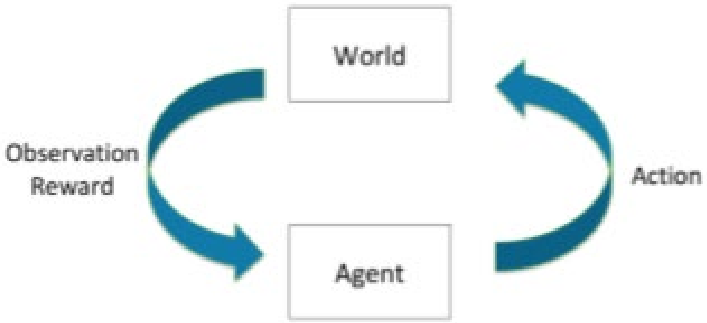
- Only require current state
- For predicting the future
- Independent of the past
- Although may use aggregate statistics (may be record of history : previous state, actions, rewards)

Think of other Markov systems:

- Knowledge of current blood pressure to determine medication control

lecture 1

sequential decision making



- Actions chosen in order to maximize total expected future reward

MDP

- Markov decision process
- Refer to Markov assumption

POMDP

- Partially observable MDP
- Many unknown factors of the world that can determine the observation & reward

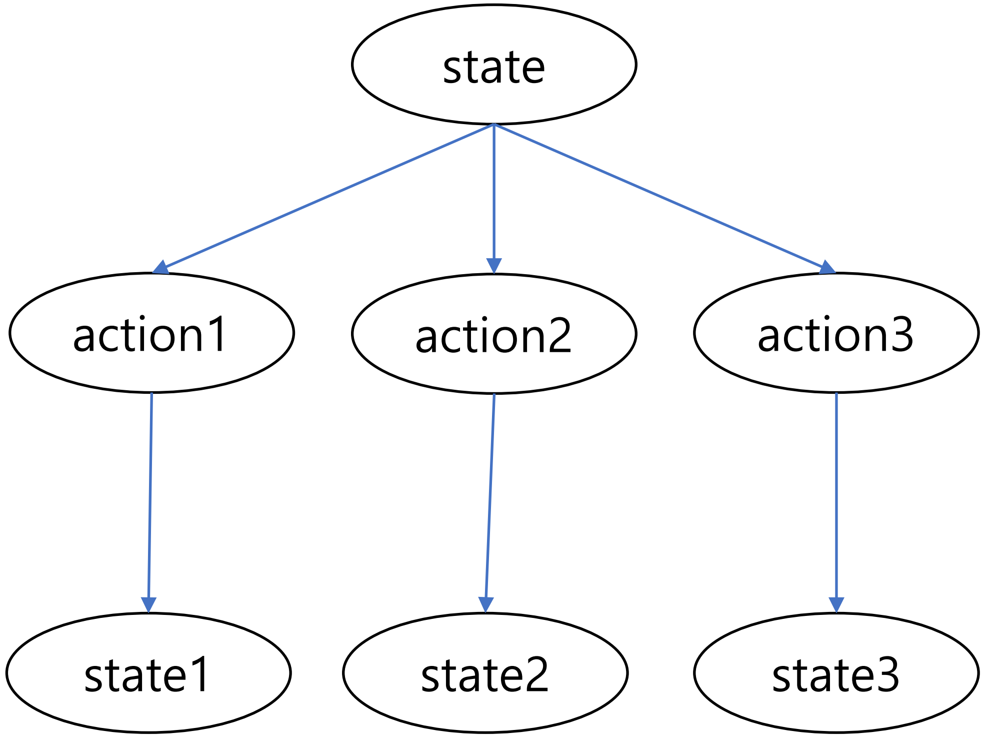
Bandit

- Actions have no influence on next observation & reward

lecture 1

deterministic policy

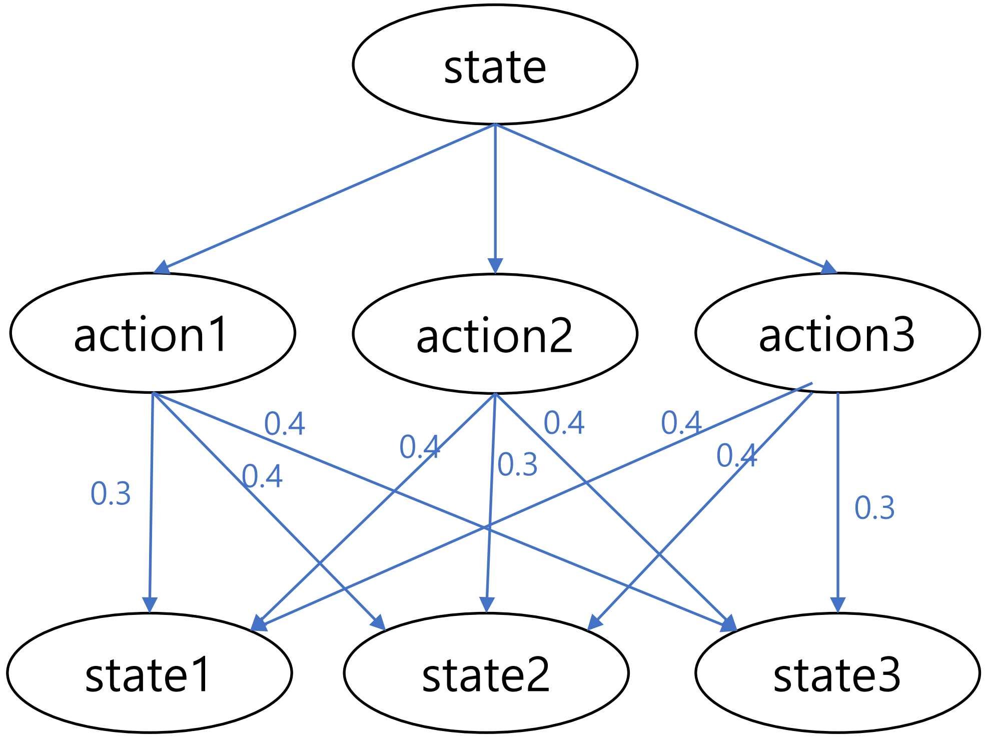
$$\pi(s) = a$$



- 100% certainty
- Definite next state

stochastic policy

$$\pi(a|s) = Pr(a_t = a|s_t = s)$$



- Many possible outcomes with relative probabilities
- Cannot be sure of next state

Value function

$$V^{\pi}(s_t = s) = \mathbb{E}_{\pi}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots | s_t = s]$$

- "expected discounted sum of future rewards under a particular policy"
- Discount factor gamma weighs immediate vs future rewards

lecture 2

lecture 2

Simple definitions

- Model : “Mathematical models of dynamics and reward”
  = expected rewards from particular action and current state

- Policy : “Function mapping agent’s states to actions”

- Model : “future rewards from being in a state and/or action when following a particular policy”
  = expected discount sum

lecture 2

Markov chain (S, P)

- Memoryless random process(not totally) with no rewards and no actions

$$P = \begin{pmatrix} P(s_1|s_1) & P(s_2|s_1) & \cdots & P(s_N|s_1) \\ P(s_1|s_2) & P(s_2|s_2) & \cdots & P(s_N|s_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(s_1|s_N) & P(s_2|s_N) & \cdots & P(s_N|s_N) \end{pmatrix}$$

Markov chain + rewards

MRP (S, P, R, gamma)

- Markov reward process
- No actions
- Value function = expected return

$$\begin{pmatrix} V(s_1) \\ \vdots \\ V(s_N) \end{pmatrix} = \begin{pmatrix} R(s_1) \\ \vdots \\ R(s_N) \end{pmatrix} + \gamma \begin{pmatrix} P(s_1|s_1) & \cdots & P(s_N|s_1) \\ P(s_1|s_2) & \cdots & P(s_N|s_2) \\ \vdots & \ddots & \vdots \\ P(s_1|s_N) & \cdots & P(s_N|s_N) \end{pmatrix} \begin{pmatrix} V(s_1) \\ \vdots \\ V(s_N) \end{pmatrix}$$

$$V = R + \gamma P V$$

- Simulation
- Analytic
- Iterative

Markov chain + rewards + actions

MRP + actions

MDP + policy

$(S, R^\pi, P^\pi, \gamma)$,

MDP (S, P, R, gamma, A)

- Markov decision process

$$P(s_{t+1} = s' | s_t = s, a_t = a)$$

$$R(s_t = s, a_t = a) = \mathbb{E}[r_t | s_t = s, a_t = a]$$

Within MDP…

There exists a unique optimal value function

=

Optimal policy in infinite horizon problem is deterministic

Policy Iteration

Q-function

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi}(s')$$

Policy improvement

$$\pi_{i+1}(s) = \arg\max_{a} Q^{\pi_i}(s,a) \; \forall s \in S$$

Note:

- Different to gradient based approaches – no problem with local minimum vs. global minimum / maximum

lecture 2

Policy Iteration

Policy improvement

$$\pi_{i+1}(s) = \arg\max_a Q^{\pi_i}(s, a) \ \forall s \in S$$

- Pi_i+1 only for the first action, and then follow pi_i
- Instead follow pi_i+1 onwards and it still monotonically improves

Policy Iteration

Monotonic improvement in policy value

$$
\begin{aligned}
V^{\pi_i}(s) &\le \max_a Q^{\pi_i}(s,a) \\
&= \max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi_i}(s') \\
&= R(s,\pi_{i+1}(s)) + \gamma \sum_{s' \in S} P(s'|s,\pi_{i+1}(s)) V^{\pi_i}(s') \quad \text{//by the definition of } \pi_{i+1} \\
&\le R(s,\pi_{i+1}(s)) + \gamma \sum_{s' \in S} P(s'|s,\pi_{i+1}(s)) \left(\max_{a'} Q^{\pi_i}(s',a')\right) \\
&= R(s,\pi_{i+1}(s)) + \gamma \sum_{s' \in S} P(s'|s,\pi_{i+1}(s)) \\
&\quad \left(R(s',\pi_{i+1}(s')) + \gamma \sum_{s'' \in S} P(s''|s',\pi_{i+1}(s')) V^{\pi_i}(s'')\right) \\
&\ \vdots \\
&= V^{\pi_{i+1}}(s)
\end{aligned}
$$

Q. How many iterations should be required?

Or

Q. How many iterations with improvement can there be?

Value iteration

" Idea : maintain optimal value of starting in a state s if have a finite number of steps k left in the episode."   = assuming finite horizon?

" value iteration update is equal to policy evaluation update "

" value iteration update is equal to Bellman optimality equation into an update rule "

" value iteration combines one sweep of policy evaluation and one sweep of policy improvement "

Sutton, 82-83

Value iteration

Bellman backup operator

$$BV(s) = \max_a \left(R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V(s') \right)$$

BV yields a new value function

Direction of understanding

Direction of application

Value iteration

$$V_{k+1} = BV_k$$

$$\pi_{k+1}(s) = \arg\max_a \left(R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s') \right)$$

Until V stops changing vs. no significant difference

$$V^\pi = B^\pi B^\pi \cdots B^\pi V$$

Policy iteration vs. Value iteration

“ value iteration update is equal to policy evaluation update ”

- Generally same thing
- But policy iteration is focused on updating the policy (given that value monotonically improves)
- And value iteration is focused on improving the value (utilizing a method that is same as updating and using the improved policy)

앞으로...

- 날짜 시간: 금 1400 ZOOM
- 어떻게 공부 할 것인가: 강의 2개 일단 다음부턴 무조건 3개씩
- 광훈 : 백준 100개 마스터 7월 31일까지
- 힘들면 편하게 얘기하기
- frozen lake 7월 31일까지
- 목표... 완강 assignment 따라하기

lecture 3

Policy evaluation

1. DP_Dynamic Programming
2. MC_Monte Carlo
3. TD_Temporal Difference

lecture 3

Policy evaluation

1. DP_Dynamic Programming

transition model

reward model

= model given

model used to
compute one timestep

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

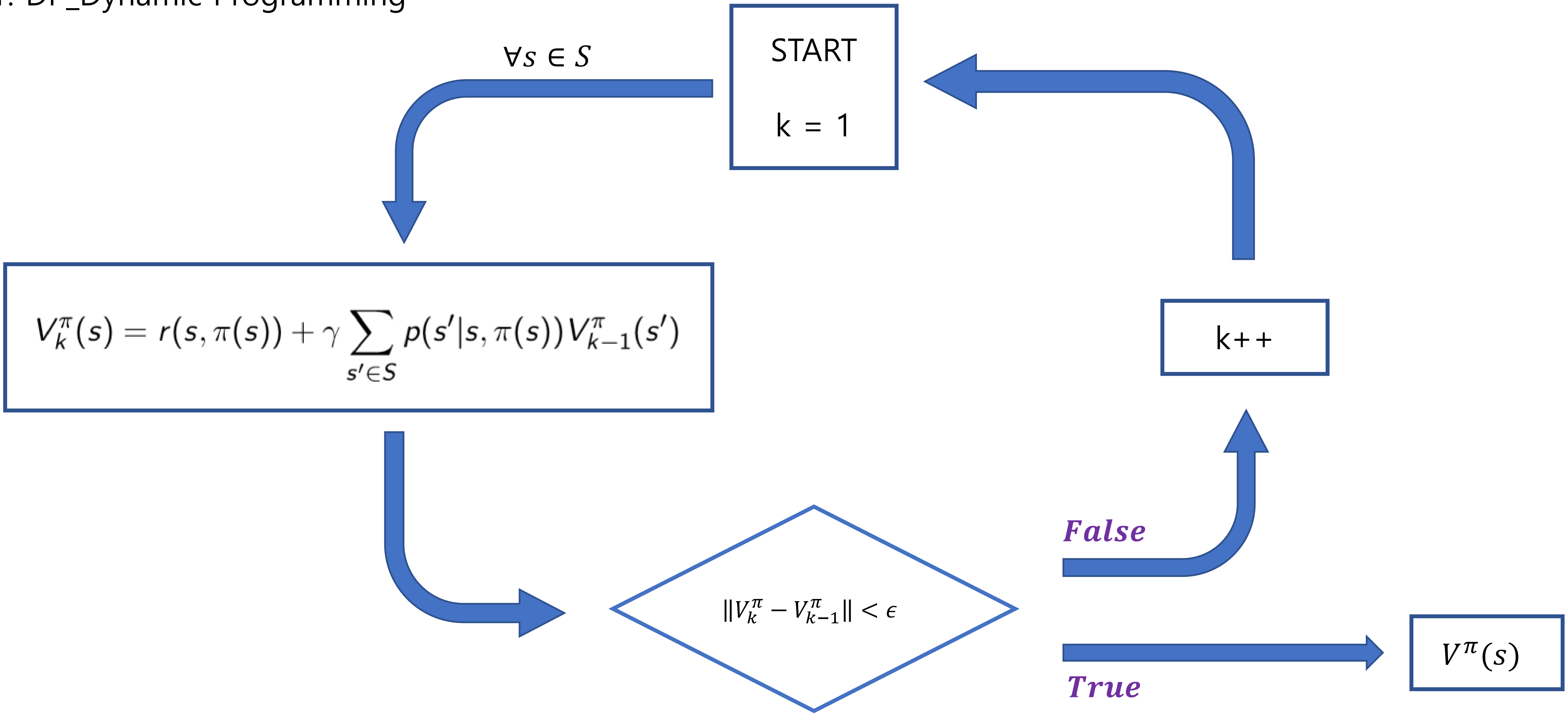
<u>Bootstrapping :</u>

1. Using the value estimate for $V_{k-1}^\pi(s')$
2. Using previously calculated $V_{k-1}^\pi(s')$
3. The expected value starting from state s' is identical no matter what timestep it starts from when considering infinite horizon

lecture 3

Policy evaluation

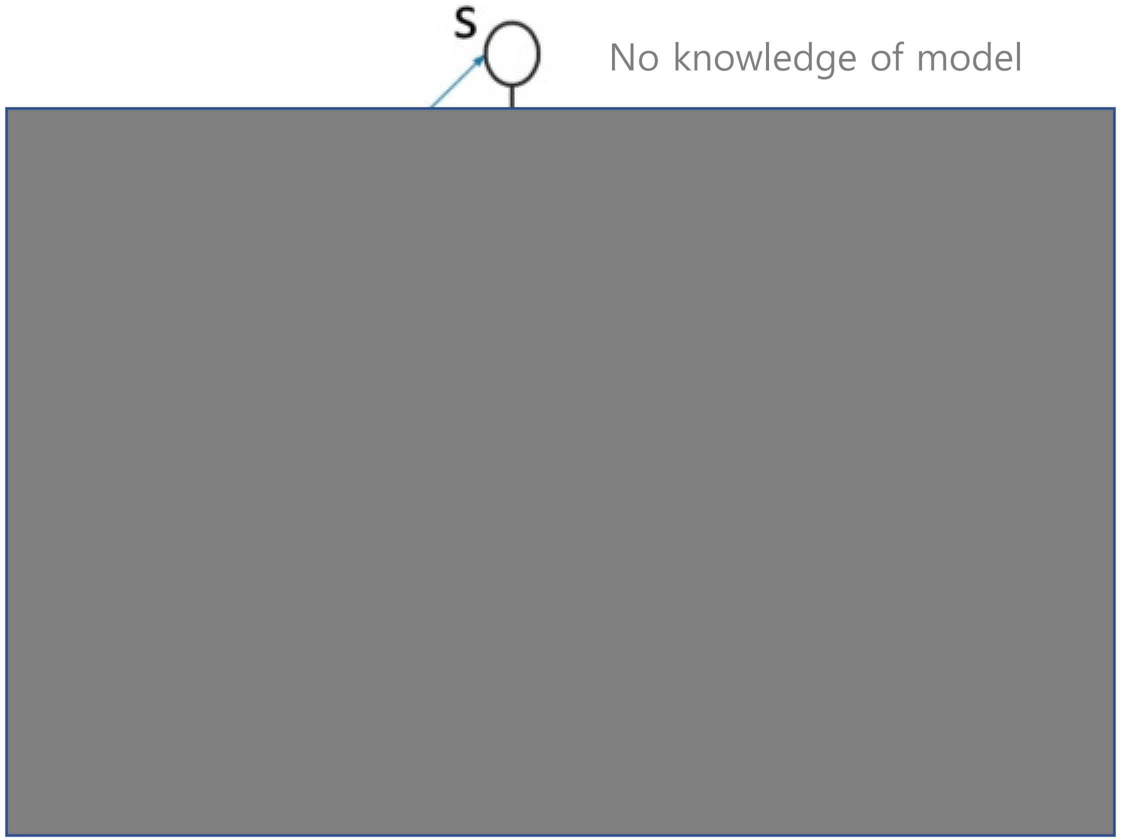
1. DP_Dynamic Programming



lecture 3

Policy evaluation

2. MC_Monte Carlo

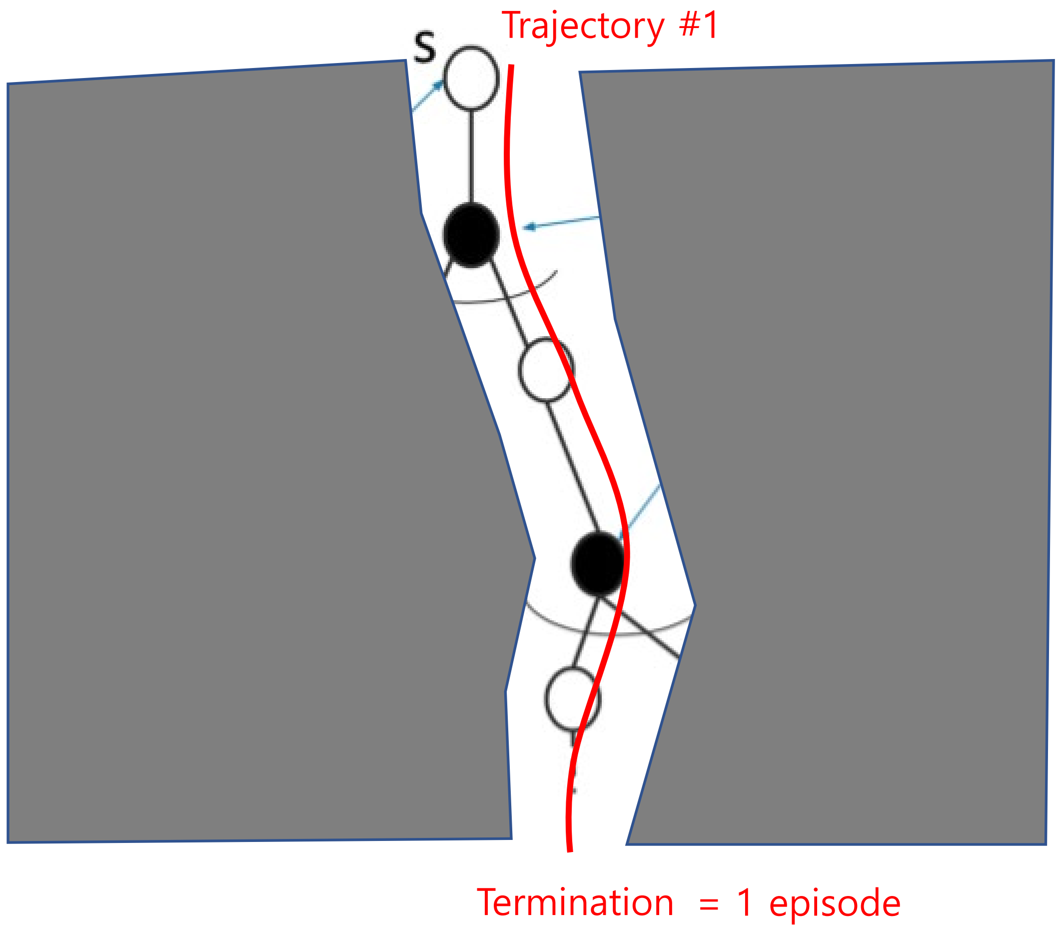


No knowledge of model

lecture 3

Policy evaluation

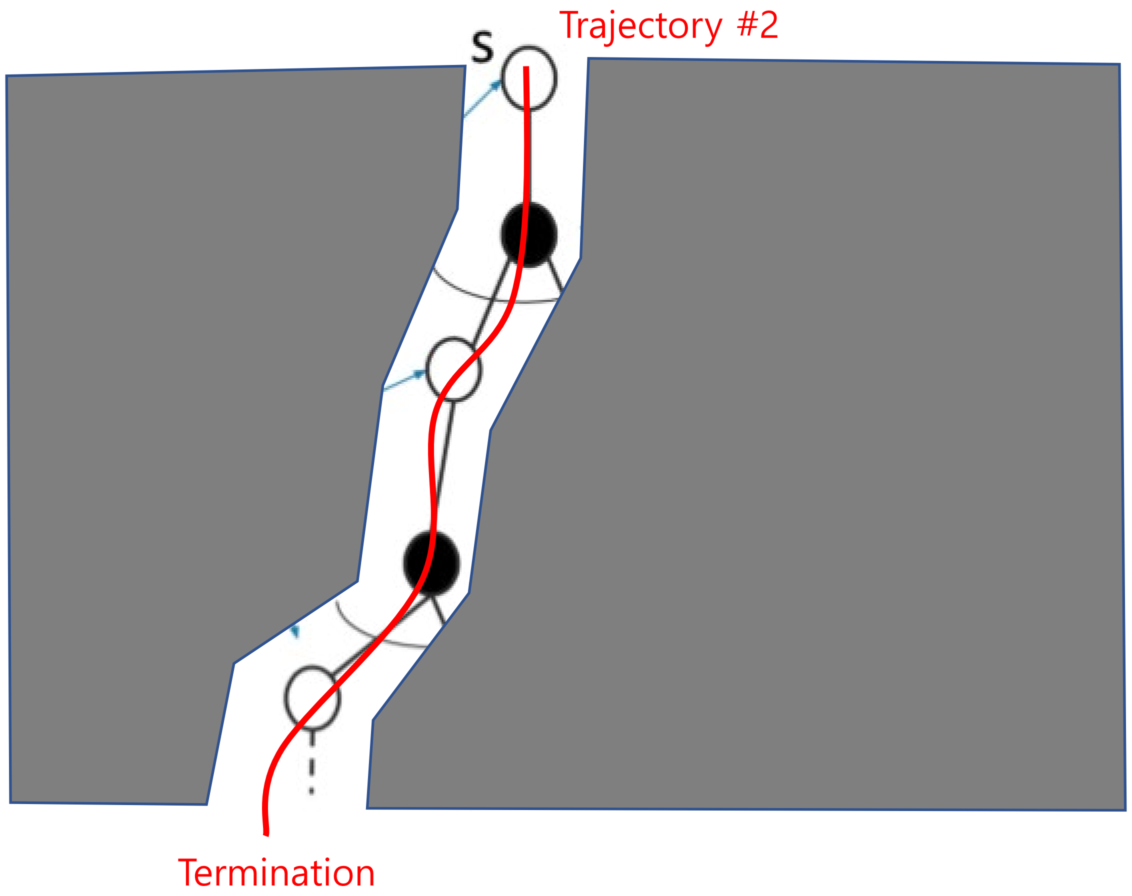
2. MC_Monte Carlo



lecture 3

Policy evaluation

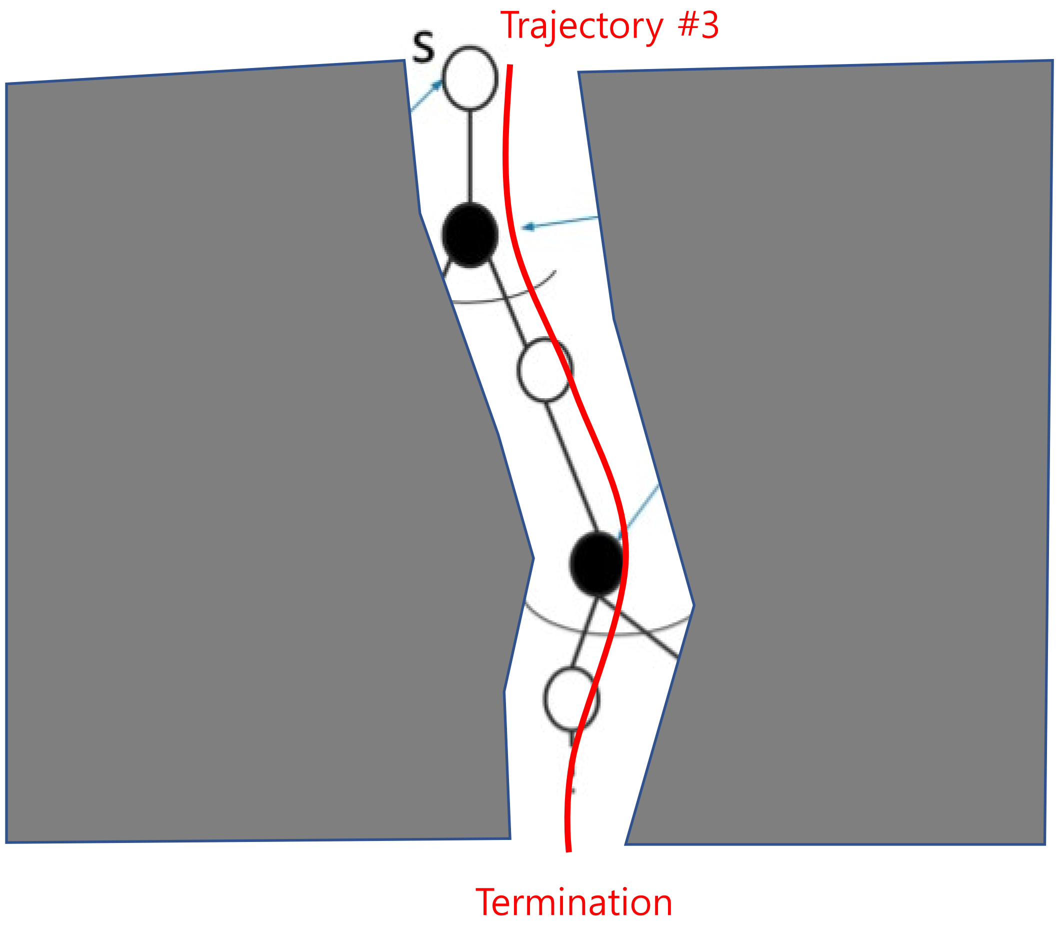
2. MC_Monte Carlo



lecture 3

Policy evaluation

2. MC_Monte Carlo



Value = average of
return from
all trajectories

Trajectory #1 … #N

lecture 3

Policy evaluation

2. MC_Monte Carlo

- Condition = episodic MDP = each episode must terminate
- Does not assume state is Markov = current state is all that is required to know what will happen next
- No bootstrapping

lecture 3

Policy evaluation

2. MC_Monte Carlo

- First-visit
- Every-visit
- Incremental

When $\alpha = 1/N(s)$: Incremental = every-visit
When $\alpha \neq 1/N(s)$: forget older data

lecture 3

Policy evaluation

2. MC_Monte Carlo

- First-visit
  - Unbiased
- Every-visit
  - Biased : during the same episode return for different states are correlated
  - Lower variance than first-visit : more data points
- But still requires a lot of data to reduce variance

Q1. Exactly why first-visit Monte Carlo is unbiased?

During one episode there can be multiple states encountered. And they will share similar return depending on the discount factor.

lecture 3

Policy evaluation

3. TD_Temporal Difference

Temporal Difference Learning :

Combination of MC & DP
= Bootstraps and samples

Bootstrapping = relying on previous data results that may not be true = biased

Available for both episodic or infinite horizon settings

Updates value each timestep

lecture 3

Policy evaluation

2. MC_Monte Carlo
3. TD_Temporal Difference

2. MC_Monte Carlo

$$G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \cdots \gamma^{T_i - 1} r_i$$

$$N(s) = N(s) + 1$$

$$G(s) = G(s) + G_{i,t}$$

$$V^\pi(s) = G(s)/N(s) \Longleftrightarrow V^\pi(s) = V^\pi(s) + \alpha(G_{i,t} - V^\pi(s))$$

Incremental factor

---

3. TD_Temporal Difference

$$V^\pi(s_t) = V^\pi(s_t) + \alpha([r_t + \gamma V^\pi(s_{t+1})] - V^\pi(s_t))$$

Update every timestep

lecture 3

Policy evaluation

1. DP_Dynamic Programming
2. MC_Monte Carlo
3. TD_Temporal Difference

	DP	MC	TD
Require model	Yeah	No	No
Require episodic	No	Yah	no
Require mark	No	Yahj	no
Consistent	ye	ye	ye
Unbiased	Biased	Unbiased	biased

With enough data

Q2. Difference between consistency and being unbiased

A2. With small amount of data the estimated value may be off from true value = biased

But with large amounts of data the expected value will converge to the true value

Q3. What is...

- Tabular representation
- Functional approximation

Q4. Explain how TD
exploits Markov structure.

lecture 3

Q5. Help on Certainty Equivalence...

lecture 4

Control

Control

1. Making decisions
2. Optimization : identify policy with high expected rewards
3. Explore : try different actions

Control

1. Making decisions

2. Optimization : identify policy with high expected rewards

3. Explore : try different actions

On-policy

Off-policy

Generalized policy improvement

$$\pi_{i+1}(s) = \arg\max_a Q^{\pi_i}(s, a)$$

ϵ-greedy policy improvement

$$\pi(a|s) = [\arg\max_a Q(s,a), \text{ w. prob } 1-\epsilon; \text{ a w. prob } \frac{\epsilon}{|A|}]$$

GLIE_Greedy in the Limit of Infinite Exploration

- All (s, a) is visited infinite number of times
- $\lim_{i \to \infty} \pi(a|s) \to \arg\max_a Q(s, a)$

SARSA Algorithm

ϵ-greedy policy improvement done for TD methods

Robbins-Munro sequence

SARSA → Q-Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma Q(s_{t+1}, a_{t+1})) - Q(s_t, a_t))$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma \max_{a'} Q(s_{t+1}, a')) - Q(s_t, a_t))$$

Bootstrapping

SARSA → Q-Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma Q(s_{t+1}, a_{t+1})) - Q(s_t, a_t))$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma \boxed{\max_{a'}} Q(s_{t+1}, a')) - Q(s_t, a_t))$$

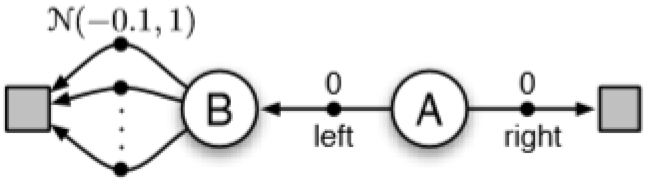
Leads to positive bias

= Maximization Bias

Q-Learning → Double Q-Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma \max_{a'} Q(s_{t+1}, a')) - Q(s_t, a_t))$$

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q_2\big(S_{t+1}, \arg\max_a Q_1(S_{t+1}, a)\big) - Q_1(S_t, A_t)\Big]$$



Q7. What is double Q-learning? And how does it overcome maximization bias?

Q8. For the example on the left, using Q-learning would always lead to "left" action from A?

Q9. Is double Q-learning basically bootstrapping from each other samples?

Q1. Exactly why first-visit Monte Carlo is unbiased and every-visit is biased.

A1.

First-visit 은 순수한 평균값이므로 unbiased

Every-visit은 혼합된 (불순한) 평균값이므로 biased

Q2. Difference between consistency and being unbiased

A2. With small amount of data the estimated value may be off from true value = biased

But with large amounts of data the expected value will converge to the true value

Q3. Bootstrapping

A3.

- 다음 값 계산 보다 예측 값을 갖고 옴
- 예측 값은 이전의 episode/trial에서 계산되었던 값 활용

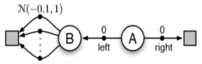
Q4. Explain how TD exploits Markov structure.

A4. Bootstrapping

Q7. What is double Q-learning? And how does it overcome maximization bias?

A7.

Maximization bias: 필연적인 게 아니라는 점 … 해당 예시에서



한 번 왼쪽 action 에서 0보다 큰 값이 나오면 지속적으로 argmax가 left action이 되는 문제가 생김 (true action(?) = right)

Double Q-learning 은 해결이 아닌 완화 method으로 이해…

파생 질문

이전 강의 내용 복습

Q5. difference between SARSA and Q-Learning

A5.

SARSA : On-policy
Q-Learning : Off-policy

Q6. What is Markov?

A6. 현재에 대한 정보로 미래를 예측할 수 있음.

답변 불충분…더 생각해 보기로

답변 불충분…더 생각해 보기로

답변 불충분…더 생각해 보기로

lecture 5

Previously…

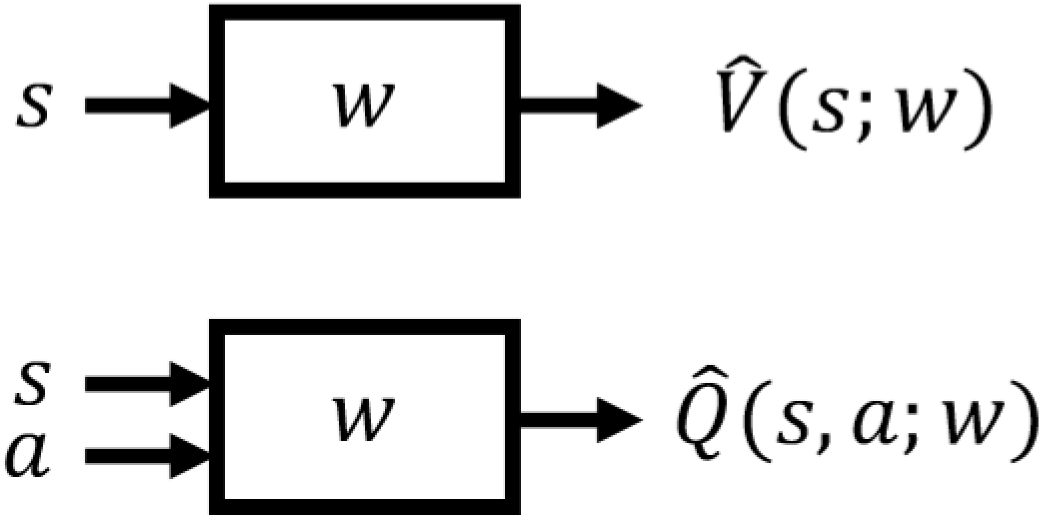
Q. Difference between tabular representation vs functional approximation?

Previously…

Q. Difference between tabular representation vs functional approximation?

A. Tabular representation is a table that holds probabilities/likelihood of every possible states as a result of current state + action, whilst functional approximation gives a more compact representation using parameters to represent the tabular representations.

Value Functional Approximation



Tabular Representation (lec2)

$$\begin{pmatrix} V(s_1) \\ \vdots \\ V(s_N) \end{pmatrix} = \begin{pmatrix} R(s_1) \\ \vdots \\ R(s_N) \end{pmatrix} + \gamma \begin{pmatrix} P(s_1|s_1) & \cdots & P(s_N|s_1) \\ P(s_1|s_2) & \cdots & P(s_N|s_2) \\ \vdots & \ddots & \vdots \\ P(s_1|s_N) & \cdots & P(s_N|s_N) \end{pmatrix} \begin{pmatrix} V(s_1) \\ \vdots \\ V(s_N) \end{pmatrix}$$

lecture 5

Value Functional Approximation

=

Generalization

1. Reduce memory required
2. Reduce computation
3. Reduce experience...

Q1. I understand that it can be also described as reducing data required.. But how does it reduce experience..?

lecture 5

From just policy evaluation → To Value Functional Approximation Prediction

From: Having a look up table of value estimates and then updating the value estimates each episode or steps

To: reapproximating function when every time new data is given (every step/run)

Feature vectors

$$\boldsymbol{x}(s) = \begin{pmatrix} x_1(s) \\ x_2(s) \\ \cdots \\ x_n(s) \end{pmatrix} \qquad \boldsymbol{x}(s,a) = \begin{pmatrix} x_1(s,a) \\ x_2(s,a) \\ \cdots \\ x_n(s,a) \end{pmatrix}$$

lecture 5

Update Linear VFA for Prediction with...

Oracle

$$\hat{V}(s;\boldsymbol{w}) = \sum_{j=1}^{n} x_j(s) w_j = \boldsymbol{x}(s)^T \boldsymbol{w}$$

$$J(\boldsymbol{w}) = \mathbb{E}_\pi[(V^\pi(s) - \hat{V}(s;\boldsymbol{w}))^2]$$

$$\Delta \boldsymbol{w} = -\frac{1}{2}\alpha \nabla_{\boldsymbol{w}} J(\boldsymbol{w})$$

$$\triangle \mathbf{w} = \boxed{-\frac{1}{2}\alpha\Big(2}\Big(\boxed{V^\pi(s) - \hat{V}(s;\mathbf{w})}\Big)\Big)\boxed{\mathbf{x}(s)}$$

Update = step-size x prediction error x feature value

lecture 5

Update Linear VFA for Prediction with…

Oracle

$$\triangle \mathbf{w} = -\frac{1}{2}\alpha\left(2\left(V^{\pi}(s) - \hat{V}(s;\mathbf{w})\right)\right)\mathbf{x}(s)$$

$$\triangle \mathbf{w} = \alpha\left(V^{\pi}(s) - \hat{V}(s;\mathbf{w})\right)\mathbf{x}(s)$$

Monte Carlo

$$\begin{aligned}\Delta \boldsymbol{w} &= \alpha(G_t - \hat{V}(s_t;\boldsymbol{w}))\nabla_{\boldsymbol{w}}\hat{V}(s_t;\boldsymbol{w}) \\ &= \alpha(G_t - \hat{V}(s_t;\boldsymbol{w}))\boldsymbol{x}(s_t) \\ &= \alpha(\boxed{G_t} - \boldsymbol{x}(s_t)^T\boldsymbol{w})\boldsymbol{x}(s_t)\end{aligned}$$

During algorithm for MC linear VFA policy evaluation

$G_t(s) = \sum_{j=t}^{L_k} r_{k,j}$

Gamma is set a 1, and this is no problem because MC itself is episodic = bound to terminate = return is bounded

lecture 5

Update Linear VFA for Prediction with…

Oracle

$$\triangle \mathbf{w} = -\frac{1}{2}\alpha\left(2\left(V^{\pi}(s) - \hat{V}(s;\mathbf{w})\right)\right)\mathbf{x}(s)$$

$$\triangle \mathbf{w} = \alpha\left(V^{\pi}(s) - \hat{V}(s;\mathbf{w})\right)\mathbf{x}(s)$$

Monte Carlo

$$\Delta \boldsymbol{w} = \alpha(G_t - \boldsymbol{x}(s_t)^T\boldsymbol{w})\boldsymbol{x}(s_t)$$

Temporal Difference

TD target

$$\begin{aligned}\Delta \boldsymbol{w} &= \alpha(\boxed{r + \gamma\hat{V}^{\pi}(s';\boldsymbol{w})} - \hat{V}^{\pi}(s;\boldsymbol{w}))\nabla_{\boldsymbol{w}}\hat{V}^{\pi}(s;\boldsymbol{w}) \\ &= \alpha(r + \gamma\hat{V}^{\pi}(s';\boldsymbol{w}) - \hat{V}^{\pi}(s;\boldsymbol{w}))\boldsymbol{x}(s) \\ &= \alpha(r + \gamma\boxed{\boldsymbol{x}(s')}^T\boldsymbol{w} - \boldsymbol{x}(s)^T\boldsymbol{w})\boldsymbol{x}(s)\end{aligned}$$

Q2. Can I point at this and say that bootstrapping is used?

lecture 5

Convergence Guarantees

"The Markov Chain defined by a MDP with a particular policy will eventually converge to a probability distribution over states d(s)"

Q3. 엥? 무슨 뜻?

Convergence Guarantees

"The Markov Chain defined by a MDP with a particular policy will eventually converge to a probability distribution over states d(s)"

Q4. 엥? 무슨 뜻?

$$d(s) = \sum_{s'} \sum_{a} \pi(s'|a) p(s'|s,a) d(s')$$

모든 action과 state을 거쳐서 합한 게 d(s)이고 이는 1이다. 다 1로 합해지는 것은 당연. 혹시 다른 의미가 있을까

lecture 5

Convergence Guarantees

Stationary distribution

Q5. what does it mean by "stationary" in stationary distribution?

$$MSVE(\boldsymbol{w}) = \sum_{s \in S} d(s)(V^{\pi}(s) - \hat{V}^{\pi}(s; \boldsymbol{w}))^2$$

$$MSVE(\boldsymbol{w}_{MC}) = \boxed{\min_{\boldsymbol{w}} \sum_{s \in S} d(s)(V^{\pi}(s) - \hat{V}^{\pi}(s; \boldsymbol{w}))^2}$$

same

$$MSVE(\boldsymbol{w}_{TD}) \leq \boxed{\frac{1}{1-\gamma}} \boxed{\min_{\boldsymbol{w}} \sum_{s \in S} d(s)(V^{\pi}(s) - \hat{V}^{\pi}(s; \boldsymbol{w}))^2}$$

Error from bootstrapping

Control using VFA

Interleave
1. Policy evaluation
2. E-greedy policy improvement

But unstable   Deadly Triad
- Functional approximation
- Bootstrapping
- Off-policy learning

Control using VFA

Incremental model-free approach

MC

$$\Delta \boldsymbol{w} = \alpha(G_t - \hat{Q}(s_t, a_t; \boldsymbol{w}))\nabla_{\boldsymbol{w}}\hat{Q}(s_t, a_t; \boldsymbol{w})$$

SARSA

$$\Delta \boldsymbol{w} = \alpha(r + \gamma\hat{Q}(s', a'; \boldsymbol{w}) - \hat{Q}(s, a; \boldsymbol{w}))\nabla_{\boldsymbol{w}}\hat{Q}(s, a; \boldsymbol{w})$$

Q-learning

$$\Delta \boldsymbol{w} = \alpha(r + \gamma \max_{a'} \hat{Q}(s', a'; \boldsymbol{w}) - \hat{Q}(s, a; \boldsymbol{w}))\nabla_{\boldsymbol{w}}\hat{Q}(s, a; \boldsymbol{w})$$

Control using VFA

Q6. What is the difference between linear and nonlinear VFA?

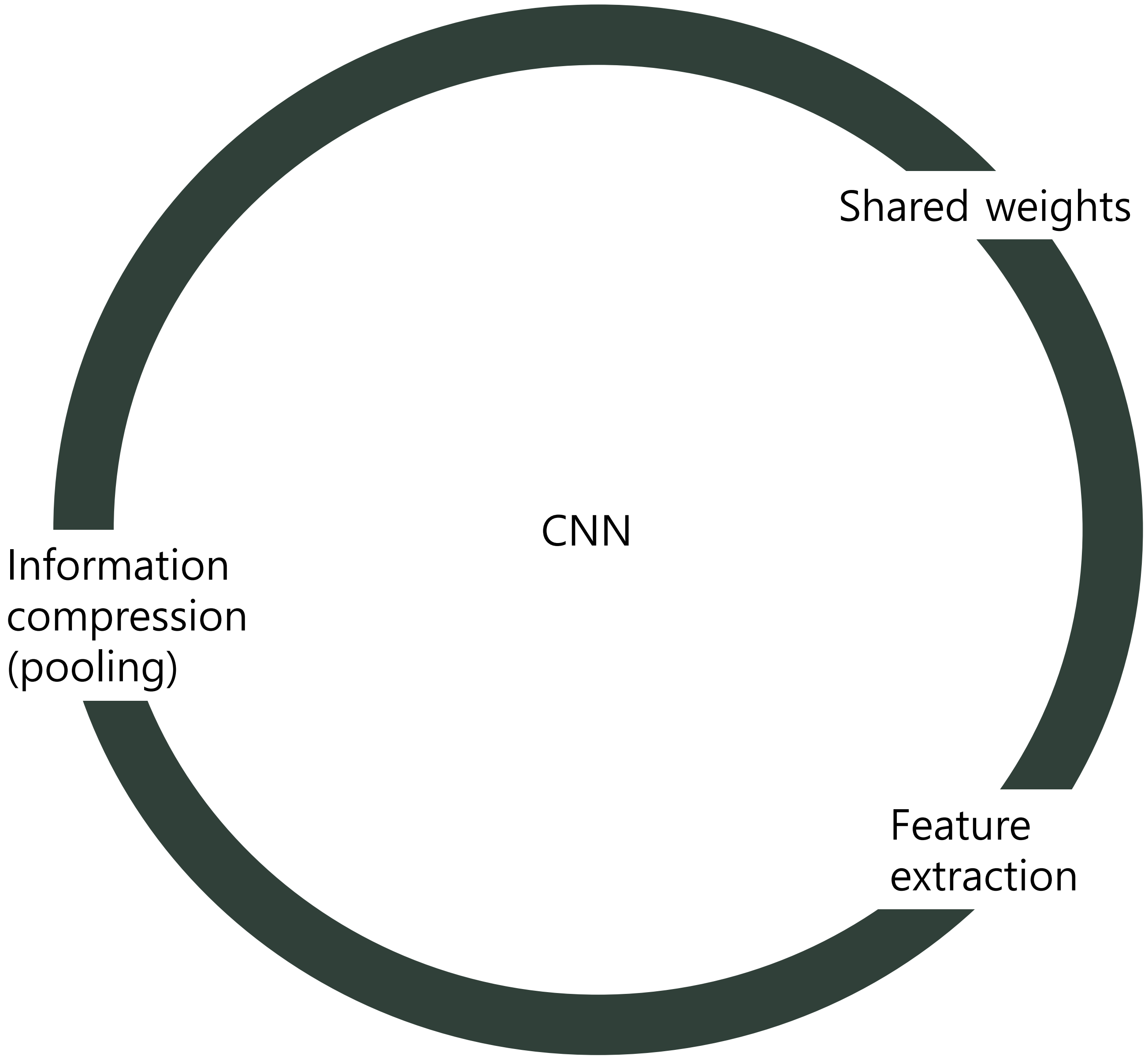
Algorithm	Tabular	Linear VFA	Nonlinear VFA
Monte-Carlo Control	converges	Converges but might have some oscillation	nope
SARSA	converges	Converges but might have some oscillation	Nope
Q-learning	converges	Nope	Nope

lecture 6

DNN

CNN

lecture 6


Shared weights
CNN
Information compression (pooling)
Feature extraction

Deep Q-Learning

Deep Reinforcement Learning

Deep Q-Networks

$$s \rightarrow \boxed{w} \rightarrow \hat{V}(s; w)$$

Q7. Don't need the Value function in the DQN's only Q function (parameterized) required. Isn't it?

$$\begin{matrix} s \rightarrow \\ a \rightarrow \end{matrix} \boxed{w} \rightarrow \hat{Q}(s, a; w)$$

From lecture 5

Control using VFA

Incremental model-free approach

MC

$$\Delta \boldsymbol{w} = \alpha(G_t - \hat{Q}(s_t, a_t; \boldsymbol{w}))\nabla_{\boldsymbol{w}}\hat{Q}(s_t, a_t; \boldsymbol{w})$$

SARSA

$$\Delta \boldsymbol{w} = \alpha(r + \gamma\hat{Q}(s', a'; \boldsymbol{w}) - \hat{Q}(s, a; \boldsymbol{w}))\nabla_{\boldsymbol{w}}\hat{Q}(s, a; \boldsymbol{w})$$

Q-learning

$$\Delta \boldsymbol{w} = \alpha(r + \gamma \max_{a'} \hat{Q}(s', a'; \boldsymbol{w}) - \hat{Q}(s, a; \boldsymbol{w}))\nabla_{\boldsymbol{w}}\hat{Q}(s, a; \boldsymbol{w})$$

lecture 6

DQN

Issues
- Correlation between samples
- Non-stationary targets

solutions
- Experience replay
- Fixed Q-targets

lecture 6

Double DQN

$$\Delta \boldsymbol{w} = \alpha(r + \gamma \overbrace{\hat{Q}(\underbrace{\arg\max_{a'} \hat{Q}(s', a'; \boldsymbol{w})}_{\text{Action selection: } \boldsymbol{w}}; \boldsymbol{w}^-)}^{\text{Action evaluation: } \boldsymbol{w}^-} - \hat{Q}(s, a; \boldsymbol{w}))$$

Prioritized order replay

$$p_i = \left| r + \gamma \max_{a'} Q(s_{i+1}, a'; \boldsymbol{w}^-) - Q(s_i, a_i; \boldsymbol{w}) \right|$$

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

Dueling DQN

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

Double DQN

$$\Delta \boldsymbol{w} = \alpha(r + \gamma \overbrace{\hat{Q}(\underbrace{\arg\max_{a'} \hat{Q}(s', a'; \boldsymbol{w})}_{\text{Action selection: } \boldsymbol{w}}; \boldsymbol{w}^{-})}^{\text{Action evaluation: } \boldsymbol{w}^{-}} - \hat{Q}(s, a; \boldsymbol{w}))$$

Q7. Difference between fixed Q-target.

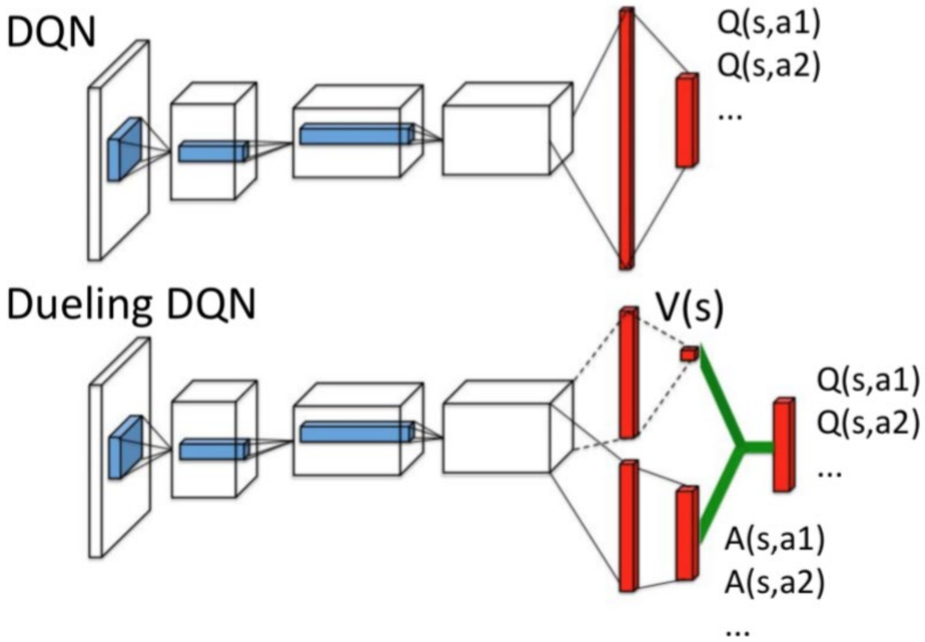
TD Error - Priority of a tuple is proportional to DQN error

Prioritized order replay

$$p_i = \boxed{\left| r + \gamma \max_{a'} Q(s_{i+1}, a'; \mathbf{w}^-) - Q(s_i, a_i; \mathbf{w}) \right|}$$

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

lecture 6



How much better or worse taking a particular action versus following the current policy

Identifiability : Whether there exists a unique Q for A and V given pi (policy)

Dueling DQN $A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$

lecture 7

DQN may require too large number of samples to learn a good policy

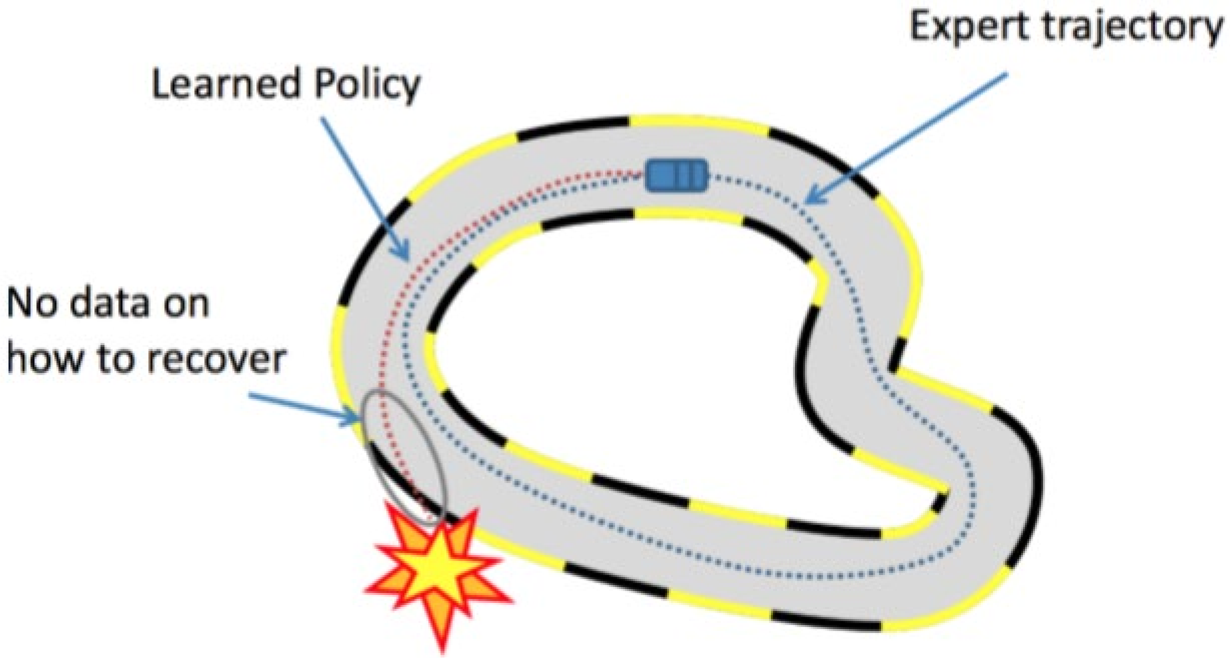
Or even a large number of samples used to learn might not promise a good policy improvement

→ Imitation learning

→ Imitation learning

→ Behavioral Cloning : Estimate policy from training examples

→ Problem : Compounding Error



→ Solution : DAGGER (or is it?)

lecture 7

→ Imitation learning

→ Apprenticeship learning via Inverse RL

Inverse RL

1. Identify weight vector given demonstrations (from teacher)
2. Use that to calculate function of policy pi ↔ Distribution of states
3. Mu : discounted sum of state features under policy pi

= Q8. Can I understand it as Indirectly gaining information on the policy?

Q9. So are we calculating anything at the 'just' Inverse RL stage? Or is it a method to combine with apprenticeship learning?

lecture 7

→ Imitation learning

→ Apprenticeship learning via Inverse RL

Possibly wrong

Inverse RL

1. Identify weight vector given demonstrations (from teacher)
2. Use that to calculate function of policy pi
3. Mu : discounted sum of state features under policy pi

Maybe room for some discussion?

→ Imitation learning

→ Apprenticeship learning via Inverse RL

Find w such that...

$$w^{*T}\mu(\pi^*) \geq w^{*T}\mu(\pi), \forall \pi \neq \pi^*$$

$$\arg\max_{\boldsymbol{w}} \max_{\gamma} s.t. w^T\mu(\pi^*) \geq w^T\mu(\pi) + \gamma \quad \forall \pi \in \{\pi_0, \pi_1, \ldots, \pi_{i-1}\}$$

Q10. So... how do we obtain mu? Mu is the demonstration itself? Only calculation of weight w is sufficient in finding out the reward function?

Ambiguity

- There is an infinite number of reward functions with the same optimal policy.
- There are infinitely many stochastic policies that can match feature counts
- Which one should be chosen?

Maybe room for some discussion?

Lecture 8

Value based RL	Policy based RL
Difference	
• SARSA, Q-learning • Find optimal value function and find the policy of optimal value (policy extraction) • Slow convergence but solves harder control problems	• REINFORCE • Policy evaluation and then policy improvement • Relatively faster convergence but appropriate for simpler control problems

https://www.researchgate.net/publication/329368817_Deep_Reinforcement_Learning_for_Soft_Robotic_Applications_Brief_Overview_with_Impending_Challenges

Value based RL	Policy based RL

Difference in procedure (iteration)

Value based RL

1. Approximate value function using parameters
2. Policy generated from value function using e-greedy

Policy based RL

- Collect set of data (trajectories) using the current policy
- Compute the policy gradient
- Apply gradient on SGD or ADAM

lecture 8

Value based RL	Policy based RL

Difference (simple)

Value based RL

- Learnt value function
- Implicit policy (cannot be certain with e chance of randomity)

Policy based RL

- No value function
- Learnt policy (current policy?)

Value based RL | Policy based RL

Advantages and disadvantages (in policy based RL perspective)

Disadvantages

1. Policy based RL typically converges to local optimum rather than the global optimum
2. Policy based RL typically is inefficient and high variance

Advantages

1. Better convergence properties
2. Effective in high-dimensional or continuous action spaces
3. Can learn stochastic properties (important)

Q1. Really? Why?

Value based RL | Policy based RL

Advantages and disadvantages (in policy based RL perspective)

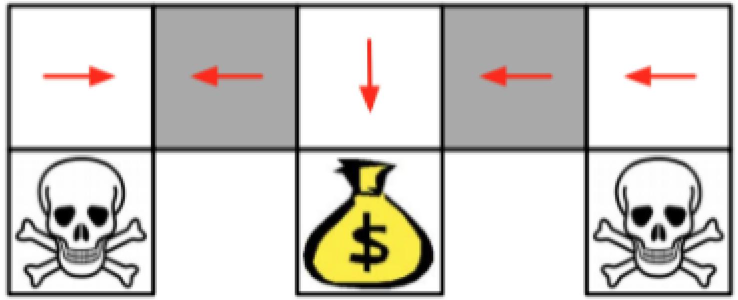
Disadvantages

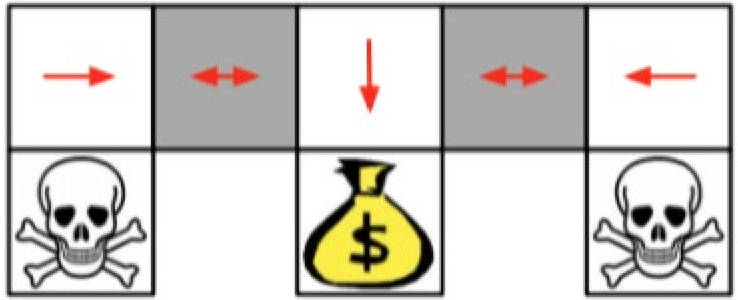
1. Policy based RL typically converges to local optimum rather than the global optimum
2. Policy based RL typically is inefficient and high variance

Advantages

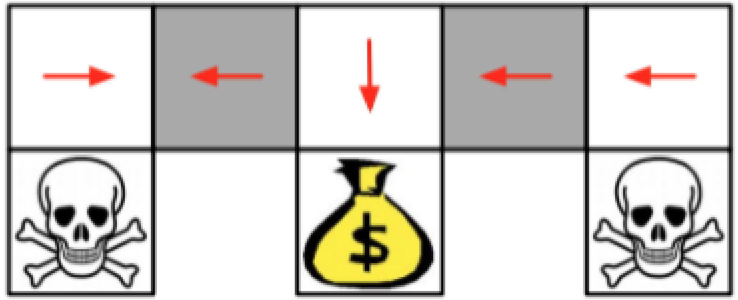
1. Better convergence properties
2. Effective in high-dimensional or continuous action spaces
3. Can learn stochastic properties (important)

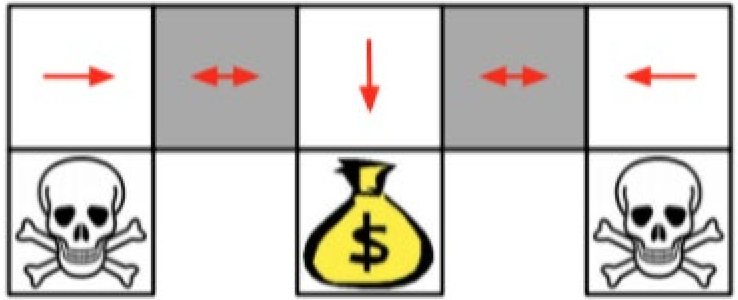
3. Policy based RL can learn stochastic properties





3. Policy based RL can learn stochastic properties





Trying to Understand Vanilla Policy Gradient Algorithm

Initialize policy parameter θ, baseline b
for iteration$=1, 2, \cdots$ **do**
  Collect a set of trajectories by executing the current policy
  At each timestep in each trajectory, compute
    the *return* $R_t = \sum_{t'=t}^{T-1} r_{t'}$, and
    the *advantage estimate* $\hat{A}_t = R_t - b(s_t)$.
  Re-fit the baseline, by minimizing $||b(s_t) - R_t||^2$,
    summed over all trajectories and timesteps.
  Update the policy, using a policy gradient estimate $\hat{g}$,
    which is a sum of terms $\nabla_\theta \log \pi(a_t|s_t, \theta)\hat{A}_t$.
    (Plug $\hat{g}$ into SGD or ADAM)
endfor

Can be modified better performance

Estimate gradient = delta value function over delta parameters

Trying to Understand Vanilla Policy Gradient Algorithm

1. Take one policy
2. Get data from it
3. Fit baseline (advantage)
4. Calculate policy gradient estimate
5. Update policy by step size using

Can be modified better performance

Trying to Understand Vanilla Policy Gradient Algorithm

1. Before estimation (calculation) of policy gradient, need to quantify the quality of the current policy = policy objective functions

Initialize policy parameter θ, baseline b
for iteration$=1, 2, \cdots$ **do**
Collect a set of trajectories by executing the current policy
At each timestep in each trajectory, compute
the *return* $R_t = \sum_{t'=t}^{T-1} r_{t'}$, and
the *advantage estimate* $\hat{A}_t = R_t - b(s_t)$.
Re-fit the baseline, by minimizing $||b(s_t) - R_t||^2$,
summed over all trajectories and timesteps.
Update the policy, using a policy gradient estimate $\hat{g}$,
which is a sum of terms $\nabla_\theta \log \pi(a_t | s_t, \theta)\hat{A}_t$.
(Plug $\hat{g}$ into SGD or ADAM)
endfor

Estimate gradient = delta value function over delta parameters

Different possible policy objective functions

1. Before estimation (calculation) of policy gradient, need to quantify the quality of the current policy = policy objective functions

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

for episodic environments

$$J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s,a) \mathcal{R}_s^a$$

For continuous environments

Stationary distribution

Trying to Understand Vanilla Policy Gradient Algorithm

1. Policy objective functions
2. Under assumption of episodic MDP, extend the idea with the following particular policy objective function…

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Policy objective functions
2. Under assumption of episodic MDP, extend the idea with the following particular policy objective function…

$$V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Policy objective functions
2. Under assumption of episodic MDP, extend the idea with the following particular policy objective function... then the objective is to optimize the given function

$$V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Policy objective functions
2. Under assumption of episodic MDP, extend the idea with the following particular policy objective function… then the objective is to optimize the given function. Although there are other methods that don't involve using gradient approaches, let's focus on the **policy gradient** approach

$$V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

Learning rate

$$\nabla\theta = \alpha\nabla_\theta V(\theta)$$

$$\nabla_\theta V(\theta) = \begin{pmatrix} \frac{\delta V(\theta)}{\delta\theta_1} \\ \vdots \\ \frac{\delta V(\theta)}{\delta\theta_n} \end{pmatrix}$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Policy objective functions
2. Optimization using policy gradient → first calculate policy gradient

$$V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

$$\nabla\theta = \alpha\nabla_\theta V(\theta)$$

$$\nabla_\theta V(\theta) = \begin{pmatrix} \frac{\delta V(\theta)}{\delta\theta_1} \\ \vdots \\ \frac{\delta V(\theta)}{\delta\theta_n} \end{pmatrix}$$

Similar to Q/V based approach but instead of deriving respect to parameters that define Q function, derive respect to parameters that define the policy

Little confusing with notation V(theta) but theta is parameter that defines pi and thus the value function becomes a function of theta (= value function for the policy)

Trying to Understand Vanilla Policy Gradient Algorithm

1. Policy objective functions
2. Calculate policy gradient (case: **finite difference policy gradient:** works even when policy is not differentiable, simple, may be inefficient but sometimes effective)

$$V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

$$\nabla\theta = \alpha \nabla_\theta V(\theta)$$

$$\nabla_\theta V(\theta) = \begin{pmatrix} \frac{\delta V(\theta)}{\delta \theta_1} \\ \vdots \\ \frac{\delta V(\theta)}{\delta \theta_n} \end{pmatrix} \longrightarrow \frac{\delta V(\theta)}{\delta \theta_k} \approx \frac{V(\theta + \epsilon u_k - V(\theta)}{\epsilon}$$

1. Perturb theta by epsilon in kth dimension
2. uk is a unit vector in kth component

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Likelihood Ratio Policy

1. **Policy objective functions**
2. Calculate policy gradient

(= policy value)

reset

$$V(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} R(s_t, a_t); \pi_\theta\right] = \sum_{\tau} P(\tau;\theta) R(\tau)$$

Expected discounted sum of rewards from given policy

Probability of observing particular trajectory

Reward of that trajectory

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Likelihood Ratio Policy

1. **Policy objective functions (goal)**
2. Calculate policy gradient

$$V(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} R(s_t, a_t); \pi_\theta\right] = \sum_{\tau} P(\tau;\theta)R(\tau)$$

(goal) $$\arg\max_{\theta} V(\theta) = \arg\max_{\theta} \sum_{\tau} P(\tau;\theta)R(\tau)$$

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Likelihood Ratio Policy

1. Policy objective function
2. **Calculate policy gradient**

$$V(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} R(s_t, a_t); \pi_\theta\right] = \sum_\tau P(\tau;\theta)R(\tau)$$

(goal) $\arg\max_\theta V(\theta) = \arg\max_\theta \sum_\tau P(\tau;\theta)R(\tau)$

$$\nabla_\theta V(\theta) = \nabla_\theta \sum_\tau P(\tau;\theta)R(\tau)$$

After some calculations …

$$\nabla_\theta V(\theta) = \sum P(\tau;\theta)R(\tau)\nabla_\theta \log P(\tau;\theta)$$

$$\nabla_\theta V(\theta) \approx \hat{g} = (1/m)\sum_{i=1}^{m} R(\tau^{(i)})\nabla_\theta \log P(\tau^{(i)};\theta)$$

Approximated with empirical estimates for m sample paths under given policy

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Score Function

1. Policy objective function
2. **Calculate policy gradient**

$$V(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} R(s_t, a_t); \pi_\theta\right] = \sum_{\tau} P(\tau;\theta)R(\tau)$$

(goal) $\arg\max_{\theta} V(\theta) = \arg\max_{\theta} \sum_{\tau} P(\tau;\theta)R(\tau)$

$$\begin{aligned}
\nabla_\theta V(\theta) &= \nabla_\theta \sum_{\tau} P(\tau;\theta)R(\tau) \\
\nabla_\theta V(\theta) &= \sum P(\tau;\theta)R(\tau)\nabla_\theta \log P(\tau;\theta) \\
\nabla_\theta V(\theta) &\approx \hat{g} = (1/m)\sum_{i=1}^{m} R(\tau^{(i)})\nabla_\theta \log P(\tau^{(i)};\theta)
\end{aligned}$$

Score function

$$\nabla_\theta \log P(\tau^{(i);\theta}) = \sum_{t=0}^{T-1} \underbrace{\nabla_\theta \log \pi_\theta(a_t|s_t)}_{\text{no dynamics model required!}}$$

$$\hat{g} = (1/m)\sum_{i=1}^{m} R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)})$$

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Score Function

1. Policy objective function
2. **Calculate policy gradient**

$$V(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} R(s_t, a_t); \pi_\theta\right] = \sum_{\tau} P(\tau;\theta)R(\tau)$$

(goal) $\arg\max_\theta V(\theta) = \arg\max_\theta \sum_\tau P(\tau;\theta)R(\tau)$

$$\nabla_\theta V(\theta) = \nabla_\theta \sum_\tau P(\tau;\theta)R(\tau)$$

$$\nabla_\theta V(\theta) = \sum P(\tau;\theta)R(\tau)\nabla_\theta \log P(\tau;\theta)$$

$$\nabla_\theta V(\theta) \approx \hat{g} = (1/m)\sum_{i=1}^{m} R(\tau^{(i)})\nabla_\theta \log P(\tau^{(i)};\theta)$$

$$\nabla_\theta \log P(\tau^{(i);\theta}) = \sum_{t=0}^{T-1} \underbrace{\nabla_\theta \log \pi_\theta(a_t|s_t)}_{\text{no dynamics model required!}}$$

: only require analytic form for derivative policy with respect to parameters

$$\hat{g} = (1/m)\sum_{i=1}^{m} R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)})$$

lecture 8

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Score Function

1. Policy objective function
2. **Calculate policy gradient**

$$V(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} R(s_t, a_t); \pi_\theta\right] = \sum_{\tau} P(\tau; \theta) R(\tau)$$

(goal) $\arg\max_{\theta} V(\theta) = \arg\max_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$

$$\begin{aligned}
\nabla_\theta V(\theta) &= \nabla_\theta \sum_{\tau} P(\tau; \theta) R(\tau) \\
\nabla_\theta V(\theta) &= \sum P(\tau; \theta) R(\tau) \nabla_\theta \log P(\tau; \theta) \\
\nabla_\theta V(\theta) &\approx \hat{g} = (1/m) \sum_{i=1}^{m} R(\tau^{(i)}) \nabla_\theta \log P(\tau^{(i)}; \theta)
\end{aligned}$$

$$\nabla_\theta \log P(\tau^{(i);\theta}) = \sum_{t=0}^{T-1} \underbrace{\nabla_\theta \log \pi_\theta(a_t|s_t)}_{\text{no dynamics model required!}}$$

$$\hat{g} = (1/m) \sum_{i=1}^{m} R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)})$$

Q1. meaning and importance of the score function? (review the lecture...)

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Score Function

1. Policy objective function
2. **Calculate policy gradient**

$$\nabla_\theta \log \pi_\theta(s, a)$$

A1. Not that much meaning, just labeling a specific part of an equation.

Q1. meaning and importance of the score function? (review the lecture…)

Trying to Understand ~~Vanilla Policy Gradient Algorithm~~ Policy Gradient Function

1. Decide policy objective function
2. Calculate policy gradient

Theorem

For any differentiable policy $\pi_\theta(s, a)$,
for any of the policy objective function $J = J_1$, (episodic reward), J_{avR} (average reward per time step), or $\frac{1}{1-\gamma} J_{avV}$ (average value),
the policy gradient is

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

Interesting that policy gradient can be independent on the type of policy objective function

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION1 : temporal structure

$$\hat{g} = (1/m)\sum_{i=1}^{m} R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)})$$

Sum of rewards expressed.

From every single one of the rewards – product with sum of the full trajectory of the derivative of the policy parameters

To product with sum of only the ones relevant to that particular reward

→ Lower variance achieved

$$\hat{g} = (1/m)\sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) G_t^{(i)}\ a_t^{(i)}|s_t^{(i)})$$

Reward at some time point is not influenced by later decisions made

= don't require the complete product of probability of action at given states

= future actions don't effect earlier rewards

Lecture 8, 1:00:18
Lecture 9, 17:25

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION1 : temporal structure

$$\hat{g} = (1/m)\sum_{i=1}^{m} R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)})$$

$$\left(\sum_{t=0}^{T-1} r_t\right)\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)$$

$$\sum_{t=0}^{T-1} r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

reorder

$$\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \sum_{t'=t}^{T-1} r_{t'}$$

$$\hat{g} = (1/m)\sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) G_t^{(i)}$$

Reward at some time point is not influenced by later decisions made

= don't require the complete product of probability of action at given states

= future actions don't effect earlier rewards

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION1 : temporal structure

$$\hat{g} = (1/m) \sum_{i=1}^{m} R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

$$\left(\sum_{t=0}^{T-1} r_t \right) \left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t | s_t) \right)$$

$$\sum_{t=0}^{T-1} r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t | s_t)$$

reorder

$$\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \sum_{t'=t}^{T-1} r_{t'}$$

$$\hat{g} = (1/m) \sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) G_t^{(i)}$$

lecture 8

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION1 : temporal structure

$$\hat{g} = (1/m) \sum_{i=1}^{m} R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

$$\left(\sum_{t=0}^{T-1} r_t \right) \left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t | s_t) \right)$$

$$\sum_{t=0}^{T-1} r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t | s_t)$$

reorder

$$\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \sum_{t'=t}^{T-1} r_{t'}$$

$$\hat{g} = (1/m) \sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) G_t^{(i)}$$

$\sum_{t'=t}^{T-1} r_{t'}^{(i)}$ is the return $G_t^{(i)}$

Q2. Given equation is confusing… shouldn't it be from 0 up to T-1 or t'? If it is an equation of reward up until given timestep??

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION1 : temporal structure = REINFORCE

Initialize policy parameters θ arbitrarily
for each episode $\{s_1, a_1, r_2, \cdots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
  for $t = 1$ to $T - 1$ **do**
    $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) G_t$
  endfor
endfor
return θ

Q3. Update after each episode because it uses MC methods, or as the algorithm implies, update each timestep???

lecture 9

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION2 : baseline

$$\hat{g} = (1/m)\sum_{i=1}^{m} R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)})$$

$$\hat{g} = (1/m)\sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) G_t^{(i)}$$

$$\hat{g} = (1/m)\sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \sum_{t'=t}^{T-1} r_{t'}$$

$$\hat{g} = (1/m)\sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t)\right)$$

Subtract by baseline that depends only on the state

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION2 : baseline

We can look at increasing log probability of an action proportional to how much better it is than a baseline

$$\hat{g} = (1/m) \sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right)$$

Good example of baseline can be a value function V(s)

Advantage estimate

Where baseline is simply the expected sum of rewards

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION2 : baseline

We can look at increasing log probability of an action proportional to how much better it is than a baseline

$$\hat{g} = (1/m) \sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right)$$

Good example of baseline can be a value function V(s)

Advantage estimate

Where baseline is simply the expected sum of rewards

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION2 : baseline

$$\hat{g} = (1/m) \sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right)$$

Refit baseline by...

$$\sum_i \sum_t \|b(s_t^i) - G_t^i\|$$

Q4. Is the intuition here that we are trying to develop baseline function to expected sum of rewards?

Therefore the advantage estimate will converge to 0 as the parameters become updated and the baseline becomes the (true)expected sum of rewards.

And the advantage estimate acts like a learning rate, which changes size relatively to how far off the sum of rewards from trajectory is from the (not true yet)expected sum of rewards?

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. **PROBLEM : too noisy**

SOLUTION2 : baseline

$$\hat{g} = (1/m)\sum_{i=1}^{m}\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t, s_t)\left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t)\right)$$

N-step estimator

Tradeoff between variance and bias

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, evaluate the possible new policy with the current data from using the current policy to find step size that guarantees monotonic improvement.

$$V(\tilde{\theta}) = V(\theta) + \sum_{s} \rho_{\tilde{\pi}}(s) \sum_{a} \tilde{\pi}(a|s) A_{\pi}(s, a)$$

But cannot calculate the stationary distribution given from the new policy

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, evaluate using **local approximation**

$$V(\tilde{\theta}) = V(\theta) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$$

$$L_\pi(\tilde{\pi}) = V(\theta) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$$

lecture 10

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, find the **lower-bound**

Theorem

Let $D_{TV}^{\max}(\pi_1, \pi_2) = \max_s D_{TV}(\pi_1(\cdot|s), \pi_2(\cdot|s))$. *Then*

$$V^{\pi_{new}} \geq L_{\pi_{old}}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}(D_{TV}^{\max}(\pi_{old}, \pi_{new}))^2$$

where $\epsilon = \max_{s,a} |A_\pi(s, a)|$.

$$D_{TV}(\pi_1(\cdot|s), \pi_2(\cdot|s)) = \max_a |\pi_1(a|s) - \pi_2(a|s)|$$

Maximum difference in the probability of an action under one policy versus another policy

Then D max is the biggest difference the two policies give for a particular action over all states

= where the two policies most differ

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, find the **lower-bound**

Theorem

Let $D_{TV}^{\max}(\pi_1, \pi_2) = \max_s D_{TV}(\pi_1(\cdot|s), \pi_2(\cdot|s))$. Then

$$V^{\pi_{new}} \geq L_{\pi_{old}}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}(D_{TV}^{\max}(\pi_{old}, \pi_{new}))^2$$

where $\epsilon = \max_{s,a} |A_\pi(s, a)|$.

$$D_{TV}(\pi_1(\cdot\,|s), \pi_2(\cdot\,|s)) = \max_a |\pi_1(a|s) - \pi_2(a|s)|$$

Q5. so this is the step size?

hange to KL divergence

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, find the **lower-bound**

Theorem

Let $D_{TV}^{\max}(\pi_1, \pi_2) = \max_s D_{TV}(\pi_1(\cdot|s), \pi_2(\cdot|s))$. *Then*

$$V^{\pi_{new}} \geq L_{\pi_{old}}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}(D_{TV}^{\max}(\pi_{old}, \pi_{new}))^2$$

where $\epsilon = \max_{s,a} |A_\pi(s,a)|$.

$$D_{TV}(\pi_1(\cdot|s), \pi_2(\cdot|s)) = \max_a |\pi_1(a|s) - \pi_2(a|s)|$$

$D_{TV}(p,q)^2 \leq D_{KL}(p,q)$ Change to KL divergence

$$D_{KL}^{\max}(\pi_1, \pi_2) = \max_s D_{KL}(\pi_1(\cdot|s), \pi_2(\cdot|s))$$

$$V^{\pi_{new}} \geq L_{\pi_{old}}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\pi_{old}, \pi_{new})$$

More practical

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **guarantee monotonic improvement**

Theorem

Let $D_{TV}^{\max}(\pi_1, \pi_2) = \max_s D_{TV}(\pi_1(\cdot|s), \pi_2(\cdot|s))$. Then

$$V^{\pi_{new}} \geq L_{\pi_{old}}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\pi_{old}, \pi_{new})$$

where $\epsilon = \max_{s,a} |A_\pi(s, a)|$.

$$M_i(\pi) = L_{\pi_i}(\pi) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\pi_i, \pi)$$

$$V^{\pi_{i+1}} - V^{\pi_i} \geq M_i(\pi_{i+1}) - M_i(\pi_i)$$

If the lower bound improves then there is monotonic improvement

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **problem & being more practical**

$$V^{\pi_{new}} \geq L_{\pi_{old}}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\pi_{old}, \pi_{new})$$

Optimize…

$$\max_{\theta} L_{\theta_{old}}(\theta_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\theta_{old}, \theta_{new}) = L_{\theta_{old}}(\theta_{new}) - C D_{KL}^{\max}(\theta_{old}, \theta_{new})$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **problem & being more practical**

Optimize…

$$\max_{\theta} L_{\theta_{old}}(\theta_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\theta_{old}, \theta_{new}) = L_{\theta_{old}}(\theta_{new}) - C D_{KL}^{\max}(\theta_{old}, \theta_{new})$$

Too small of a step size

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **problem & being more practical**

Optimize…

$$\max_{\theta} L_{\theta_{old}}(\theta_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\theta_{old}, \theta_{new}) = L_{\theta_{old}}(\theta_{new}) - C D_{KL}^{\max}(\theta_{old}, \theta_{new})$$

Instead

$$\max_{\theta} L_{\theta_{old}}(\theta)$$

$$\text{subject to } D_{KL}^{s\sim\rho_{\theta_{old}}}(\theta_{old}, \theta) \leq \delta$$

Constrain KL divergence on some trusted region

Use average KL instead of max KL divergence for practicality

Basically this means to maximize objective function (value of policy) subject to KL convergence bounded by some delta (range of trusted region)

Q6. would this be correct?

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **being more & more practical**

Optimize...

$$\max_\theta L_{\theta_{old}}(\theta_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\theta_{old}, \theta_{new}) = L_{\theta_{old}}(\theta_{new}) - C D_{KL}^{\max}(\theta_{old}, \theta_{new})$$

$$L_\pi(\tilde{\pi}) = V(\theta) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$$

1. Reweight according to states actually sampled by current policy

$$\sum_s \rho_\pi(s) \to \frac{1}{1-\gamma} \mathbb{E}_{s\sim\rho_{\theta_{old}}}[\ldots]$$

$$\sum_a \pi_\theta(a|s_n) A_{\theta_{old}}(s_n, a) \to \mathbb{E}_{a\sim q}\left[\frac{\pi_\theta(a|s_n)}{q(a|s_n)} A_{\theta_{old}}(s_n, a)\right]$$

$$A_{\theta_{old}} \to Q_{\theta_{old}}$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **being more & more practical**

Optimize…

$$\max_{\theta} L_{\theta_{old}}(\theta_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\theta_{old}, \theta_{new}) = L_{\theta_{old}}(\theta_{new}) - C D_{KL}^{\max}(\theta_{old}, \theta_{new})$$

$$L_{\pi}(\tilde{\pi}) = V(\theta) + \sum_s \rho_{\pi}(s) \sum_a \tilde{\pi}(a|s) A_{\pi}(s,a)$$

$$\sum_s \rho_{\pi}(s) \rightarrow \frac{1}{1-\gamma} \mathbb{E}_{s\sim\rho_{\theta_{old}}}[\ldots]$$

$$\sum_a \pi_{\theta}(a|s_n) A_{\theta_{old}}(s_n, a) \rightarrow \mathbb{E}_{a\sim q}\left[\frac{\pi_{\theta}(a|s_n)}{q(a|s_n)} A_{\theta_{old}}(s_n, a)\right]$$

$$A_{\theta_{old}} \rightarrow Q_{\theta_{old}}$$

2. Importance sampling : using how many time a particular action is taken with current policy and reweight them according to how many times the action would have been taken according to the new policy

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **being more & more practical**

Optimize...

$$\max_\theta L_{\theta_{old}}(\theta_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\theta_{old}, \theta_{new}) = L_{\theta_{old}}(\theta_{new}) - C D_{KL}^{\max}(\theta_{old}, \theta_{new})$$

$$L_\pi(\tilde{\pi}) = V(\theta) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$$

3. yeah

$$\sum_s \rho_\pi(s) \to \frac{1}{1-\gamma} \mathbb{E}_{s \sim \rho_{\theta_{old}}}[\ldots]$$

$$\sum_a \pi_\theta(a|s_n) A_{\theta_{old}}(s_n, a) \to \mathbb{E}_{a \sim q}\left[\frac{\pi_\theta(a|s_n)}{q(a|s_n)} A_{\theta_{old}}(s_n, a)\right]$$

$$A_{\theta_{old}} \to Q_{\theta_{old}}$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **being more & more practical**

Optimize…

$$L_\pi(\tilde{\pi}) = V(\theta) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$$

$$\max_\theta \mathbb{E}_{s\sim\rho_{\theta_{old}}, a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)} Q_{\theta_{old}}(s,a)\right]$$

$$\text{subject to } \mathbb{E}_{s\sim\rho_{\theta_{old}}} D_{KL}(\pi_{\theta_{old}}(\cdot|s), \pi_\theta(\cdot|s)) \le \delta$$

Trying to Understand Vanilla Policy Gradient Algorithm

1. Decide policy objective function
2. Calculate policy gradient
3. Temporal Structure & Baseline
4. Before applying step-sizes, **being more & more practical**

Optimize…

$$\max_{\theta} L_{\theta_{old}}(\theta)$$

$$\text{subject to } D_{KL}^{s\sim\rho_{\theta_{old}}}(\theta_{old}, \theta) \leq \delta$$

$$L_{\pi}(\tilde{\pi}) = V(\theta) + \sum_s \rho_{\pi}(s) \sum_a \tilde{\pi}(a|s) A_{\pi}(s,a)$$

Q7. where did this go?
Doesn't matter because it is constant no matter how the policy is changed?

$$\max_{\theta} \mathbb{E}_{s\sim\rho_{\theta_{old}}, a\sim q}\left[\frac{\pi_{\theta}(a|s)}{q(a|s)} Q_{\theta_{old}}(s,a)\right]$$

$$\text{subject to } \mathbb{E}_{s\sim\rho_{\theta_{old}}} D_{KL}(\pi_{\theta_{old}}(\cdot|s), \pi_{\theta}(\cdot|s)) \leq \delta$$

Trying to Understand Vanilla Policy Gradient Algorithm

Initialize policy parameter θ, baseline b
for iteration$=1, 2, \cdots$ **do**
  Collect a set of trajectories by executing the current policy
  At each timestep in each trajectory, compute
    the *return* $R_t = \sum_{t'=t}^{T-1} r_{t'}$, and
    the *advantage estimate* $\hat{A}_t = R_t - b(s_t)$.
  Re-fit the baseline, by minimizing $||b(s_t) - R_t||^2$,
    summed over all trajectories and timesteps.
  Update the policy, using a policy gradient estimate $\hat{g}$,
    which is a sum of terms $\nabla_\theta \log \pi(a_t|s_t, \theta)\hat{A}_t$.
    (Plug $\hat{g}$ into SGD or ADAM)
endfor

Trying to Understand Vanilla Policy Gradient Algorithm

for iteration$=1, 2, \ldots$ **do**

  Run policy for T timesteps or N trajectories

  At each timestep in each trajectory, compute target $Q^{\pi}(s_t, a_t)$, and baseline $b(s_t)$

  Compute estimated policy gradient $\hat{g}$

  Update the policy using $\hat{g}$, potentially constrained to a local region

end for

Lecture 11

Lecture 11-13 Fast Reinforcement Learning

Idea

Move towards algorithms that can be more computational-wise and sample-wise efficient?

Also mean that the optimal decision can be calculated quickly (require less time to react) and need less data to compute on.

Key terms

1. Bandits
2. Multi-armed bandits
3. Regret
4. Upper confidence bound (UCB)
5. Hoeffding's Inequality

Bandits

Actions have no influence on
next observation & reward

Slot machine

Bandits

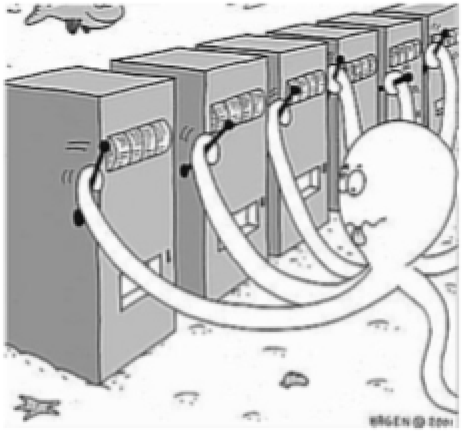
Actions have no influence on
next observation & reward

Slot machine = one-armed bandit

= each action is independent

Multi-armed Bandits

Multiple slot machine with multiple arms



Can choose which arm(action) to pull and the slot machine will return the rewards

Regret

Opportunity loss

Just an indicator for
algorithm evaluation

Regret

Opportunity loss

$$l_t = \mathbb{E}[V^* - Q(a_t)]$$

Difference of reward from current pull of the arm with the optimal value

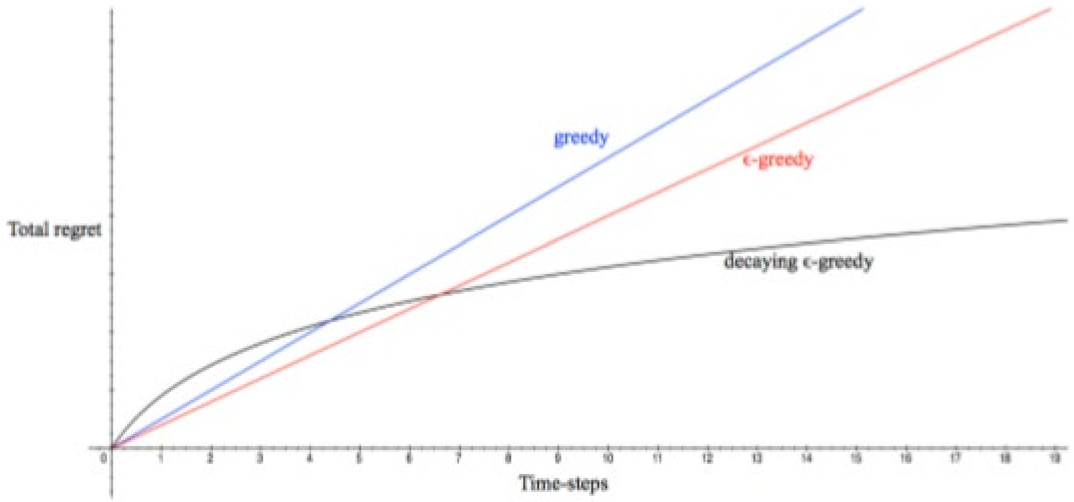
Or in other words this part is the Gap $\Delta_i = V^* - Q(a_i)$

Then calculate total regret = regret for all timesteps

$$\begin{aligned} L_t &= \mathbb{E}\left[\sum_{\tau=1}^{t} V^* - Q(a_\tau)\right] \\ &= \sum_{a \in \mathcal{A}} \mathbb{E}[N_t(a)](V^* - Q(a)) \\ &= \sum_{a \in \mathcal{A}} \mathbb{E}[N_t(a)]\Delta_a \end{aligned}$$

Count of the given action (arm pulled)

Upper confidence bound



Aim is to get sublinear regret

Because linear regret just means getting worse over time (continuously not picking the optimal action)

But first, In order to define the upper confidence bound...

Hoeffding's Inequality

$$\mathbb{P}\left[\mathbb{E}[X] > \bar{X}_n + u\right] \leq \exp(-2nu^2)$$

$$\bar{X}_n + u \geq E[X] \quad \text{w.prob} \geq 1 - \delta/t^2$$

Upper bound can deviate from the expected value by more than a certain amount

Hoeffding's Inequality

$$\mathbb{P}\left[\mathbb{E}[X] > \bar{X}_n + u\right] \leq \exp(-2nu^2)$$

$$\bar{X}_n + u \geq E[X] \quad \text{w.prob} \geq 1 - \delta/t^2$$

Where...

$$u = \sqrt{\frac{1}{2n}\log(t^2/\delta)}$$

Upper confidence bound

$$U_t(a_t) = \hat{Q}(a_t) + \sqrt{\frac{1}{2n(a_t)}\log\left(t^2/\delta\right)}$$

Upper confidence bound

$$U_t(a_t) = \hat{Q}(a_t) + \sqrt{\frac{1}{2n(a_t)} \log(t^2/\delta)}$$

UCB1 algorithm is choosing the action (arm) with the best upper bound value

$$a_t = \arg\max_{a \in \mathcal{A}} [\hat{Q}(a) + \sqrt{\frac{2 \log t}{N_t(a)}}]$$

Upper confidence bound

$$U_t(a_t) = \hat{Q}(a_t) + \sqrt{\frac{1}{2n(a_t)} \log(t^2/\delta)}$$

UCB1 algorithm is choosing the action (arm) with the best upper bound value

$$a_t = \arg\max_{a \in \mathcal{A}} [\hat{Q}(a) + \sqrt{\frac{2 \log t}{N_t(a)}}]$$

Key terms of lecture 12

1. Bayesians bandit
2. Thompson Sampling
3. Probably Approximately Correct (PAC)

Bayesian

$$\overset{\text{Posterior}}{P(A|B)} = \frac{\overset{\text{Likelihood}}{P(B|A)} * \overset{\text{Prior}}{P(A)}}{\underset{\text{Evidence}}{P(B)}}$$

Bayesian bandit

$$\underbrace{P(A|B)}_{\text{Posterior}} = \frac{\overbrace{P(B|A)}^{\text{Likelihood}} * \overbrace{P(A)}^{\text{Prior}}}{\underbrace{P(B)}_{\text{Evidence}}}$$

$$p(\phi_i|r_{i1}) = \frac{p(r_{i1}|\phi_i)p(\phi_i)}{\int_{\phi_i} p(r_{i1}|\phi_i)p(\phi_i)d\phi_i}$$

Bayesian bandit

Likelihood, Prior, Posterior

$$\underset{\text{Posterior}}{P(A|B)} = \frac{\overset{\text{Likelihood}}{P(B|A)} * \overset{\text{Prior}}{P(A)}}{\underset{\text{Evidence}}{P(B)}}$$

Data likelihood

Reward of particular action depends on this parameter

$$p(\phi_i|r_{i1}) = \frac{p(r_{i1}|\phi_i)p(\phi_i)}{\int_{\phi_i} p(r_{i1}|\phi_i)p(\phi_i)d\phi_i}$$

Distribution of rewards

Bayesian bandit

Posterior, Likelihood, Prior, Evidence

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Conjugate....

$$p(\phi_i|r_{i1}) = \frac{p(r_{i1}|\phi_i)p(\phi_i)}{\int_{\phi_i} p(r_{i1}|\phi_i)p(\phi_i)d\phi_i}$$

Bayesian bandit

$$Regret(\mathcal{A}, T; \theta) = \sum_{t=1}^{T} \mathbb{E}\left[Q(a^*) - Q(a_t)\right]$$

$$BayesRegret(\mathcal{A}, T; \theta) = \mathbb{E}_{\theta \sim p_\theta}\left[\sum_{t=1}^{T} \mathbb{E}\left[Q(a^*) - Q(a_t) | \theta\right]\right]$$

Thompson Sampling

Because computing optimal action from posterior can be difficult, a simpler approach would be as following

Thompson Sampling

1: Initialize prior over each arm a, $p(\mathcal{R}_a)$
2: **loop**
3: For each arm a **sample** a reward distribution $\mathcal{R}_a$ from posterior
4: Compute action-value function $Q(a) = \mathbb{E}[\mathcal{R}_a]$
5: $a_t = \arg\max_{a \in \mathcal{A}} Q(a)$ ←
6: Observe reward r
7: Update posterior $p(\mathcal{R}_a | r)$ using Bayes law
8: **end loop**

Easier understood through Bernoulli toy example

In attempt to understand…

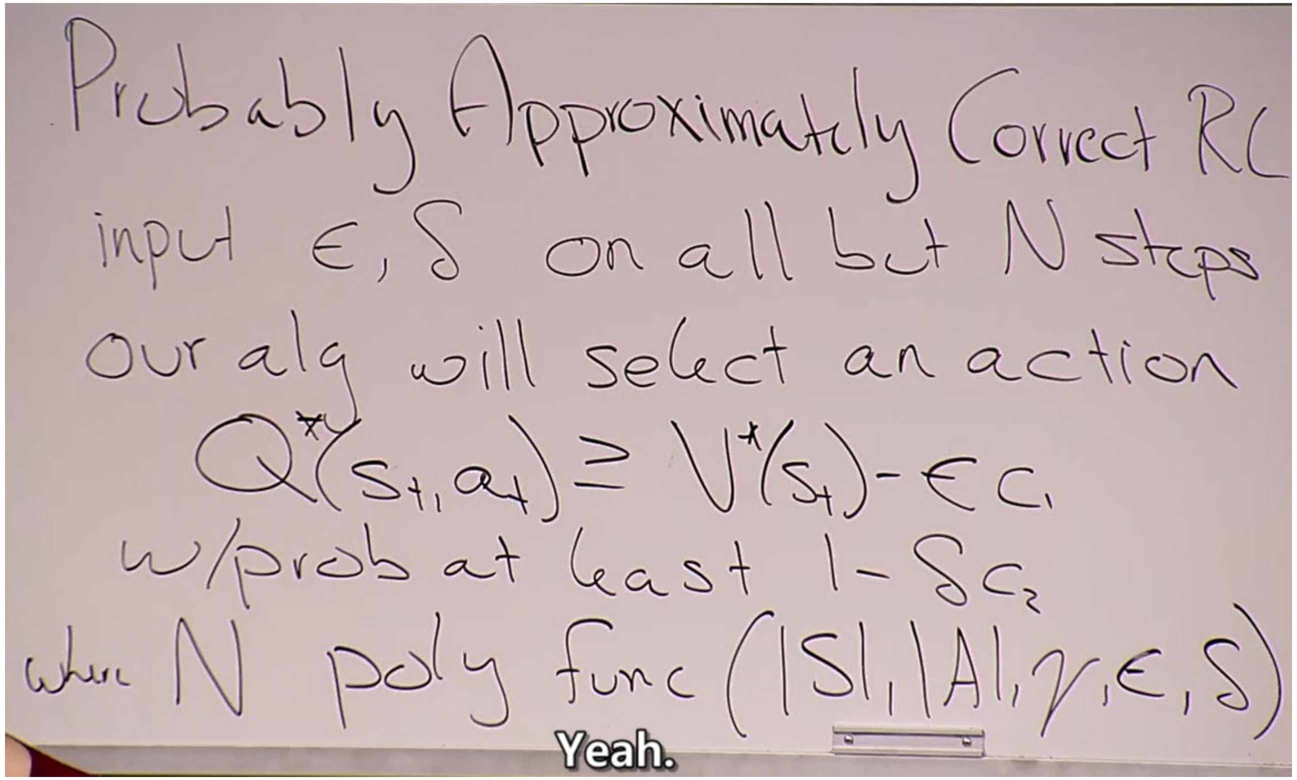
PAC Probably Approximately Correct

Input epsilon and delta in all but N steps

The algorithm selects action which its true Q value will be greater than the best possible value of that state subtracted by epsilon

With probability of at least 1 – delta

N is a polynomial function of size of S, A and gamma, epsilon and delta

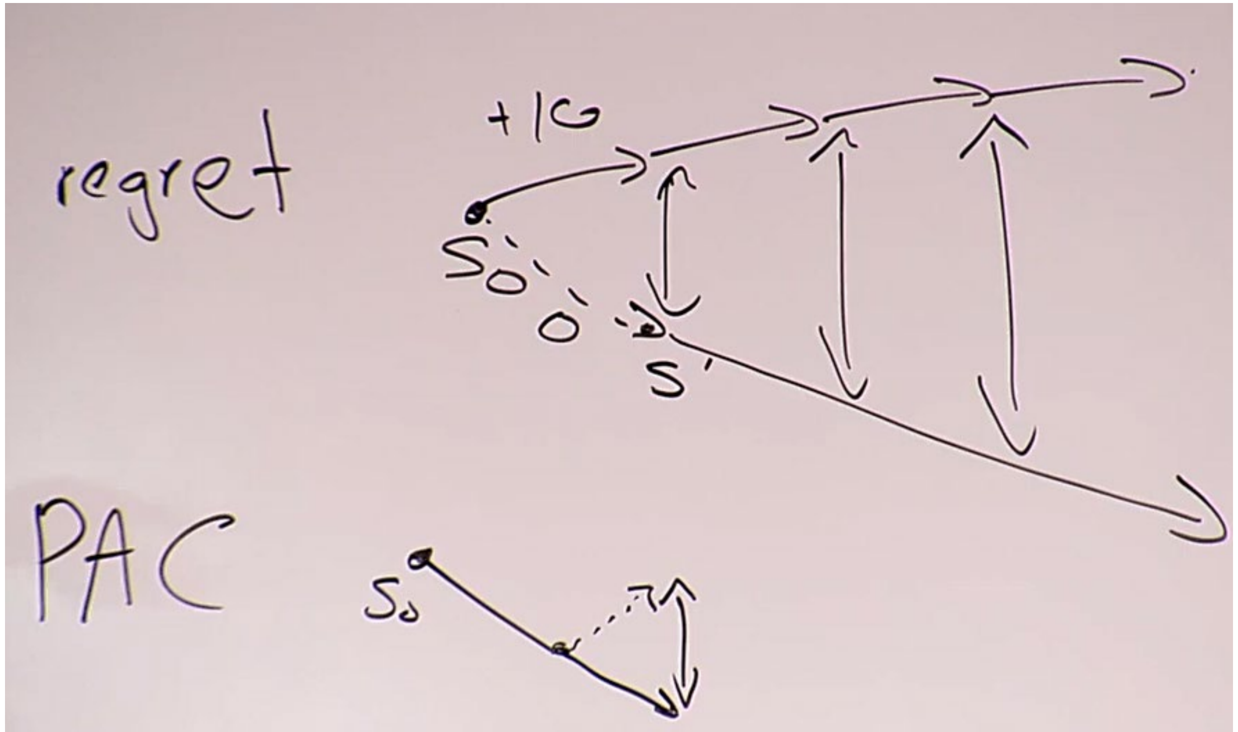


In attempt to understand…

PAC Probably Approximately Correct

PAC can be though of as making the most out of its circumstances

While regret judges on whether good decisions have been made from the beginning or not.



In attempt to understand...

PAC Probably Approximately Correct

Sufficient condition for PAC

1. Optimism $\underbrace{Q_t(s,a)}$ $\geq Q^*(s,a) - \epsilon \quad \forall s,a,t$

computed value should be optimistic with respect to real Q values

2. Accuracy $V_t(s) - \underbrace{V^{\pi_t}_{\mathcal{M}}}(s) \leq \epsilon$

will define further. MDP related to true MDP & MDP defined in MBIE-EB

3) Bounded learning complexity:
- total # of updates to Q
- # times visit an "unknown" (s,a) pair

bounded by $\xi(\epsilon, \delta)$

In attempt to understand...

MBIE-EB Model Based Interval Estimation with Exploration Bonus

Given ϵ, δ, m

$\beta = \frac{1}{1-\gamma}\sqrt{0.5\ln(2|S||A|m/\delta}$

$n_{sas}(s,a,s') = 0 \; s \in S,\; a \in A,\; s' \in S$

$rc(s,a) = 0,\; n_{sa}(s,a) = 0,\; \tilde{Q}(s,a) = 1/(1-\gamma) \; \forall\, s \in S,\; a \in A$

$t = 0,\; s_t = s_{init}$

loop

  $a_t = \arg\max_{a \in \mathcal{A}} Q(s_t, a)$

  Observe reward r_t and state s_{t+1}

  $n_{sa}(s_t,a_t) = n(s_t,a_t) + 1,\; n_{sas}(s_t,a_t,s_{t+1}) = n_{sas}(s_t,a_t,s_{t+1}) + 1$

  $rc(s_t,a_t) = \frac{rc(s_t,a_t)n_{sa}(s_t,a_t)+r_t}{(n_{sa}(s_t,a_t)+1)}$

  $\hat{R}(s,a) = \frac{rc(s_t,a_t)}{n(s_t,a_t)}$ and $\hat{T}(s'|s,a) = \frac{n_{sas}(s_t,a_t,s')}{n_{sa}(s_t,a_t)} \; \forall s' \in S$

  while not converged **do**

Backup　　　Bonus

    $\tilde{Q}(s,a) = \hat{R}(s,a) + \gamma \sum_{s'} \hat{T}(s'|s,a) \max_{a'} \tilde{Q}(s',a') + \underbrace{\frac{\beta}{\sqrt{n_{sa}(s,a)}}} \; \forall\, s \in S,\; a \in A$

Transition model

  end while

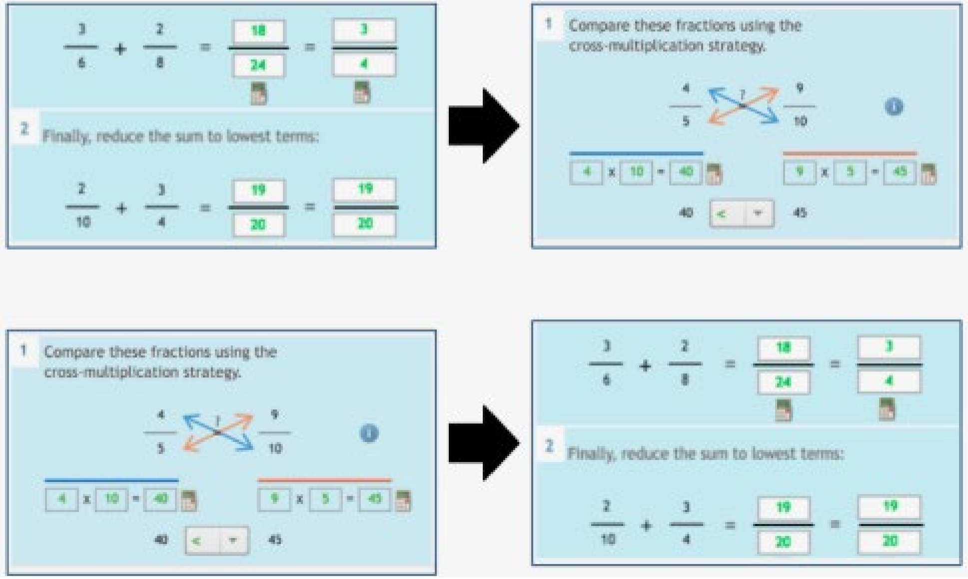
end loop

Lecture 15 : Batch RL

Why

A Group

B Group



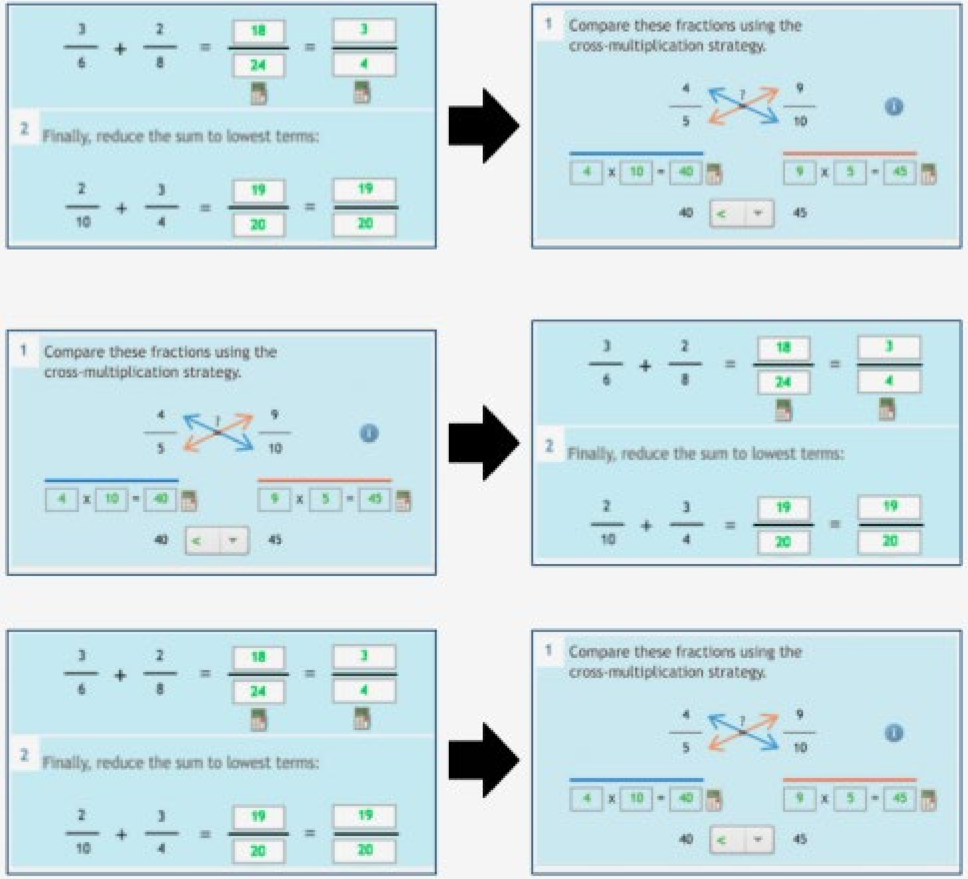
Avg Score: 95

Can we say that method applied on group A is good enough?

Avg Score: 92

Nope, you don't know what would have happened if group B went through same sequence of interventions as group A

Why



A Group — Avg Score: 95

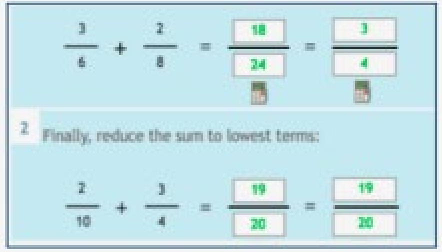
B Group — Avg Score: 92

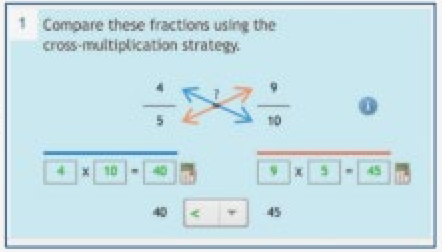
B Group — ???

Test the first method by trying it and comparing it to what group B previously had done

Why

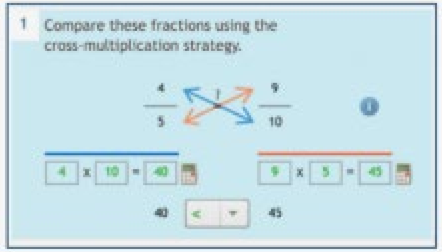
A Group

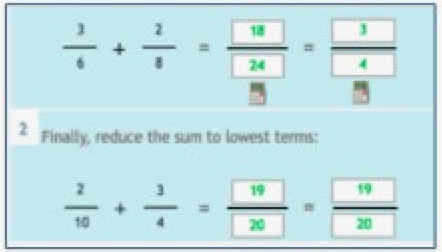




Avg Score: 95

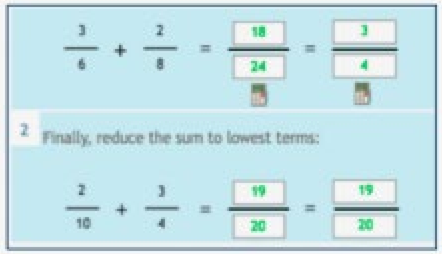
B Group

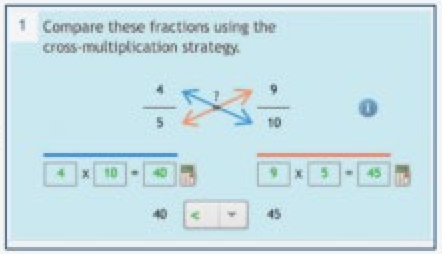




Avg Score: 92

B Group





???

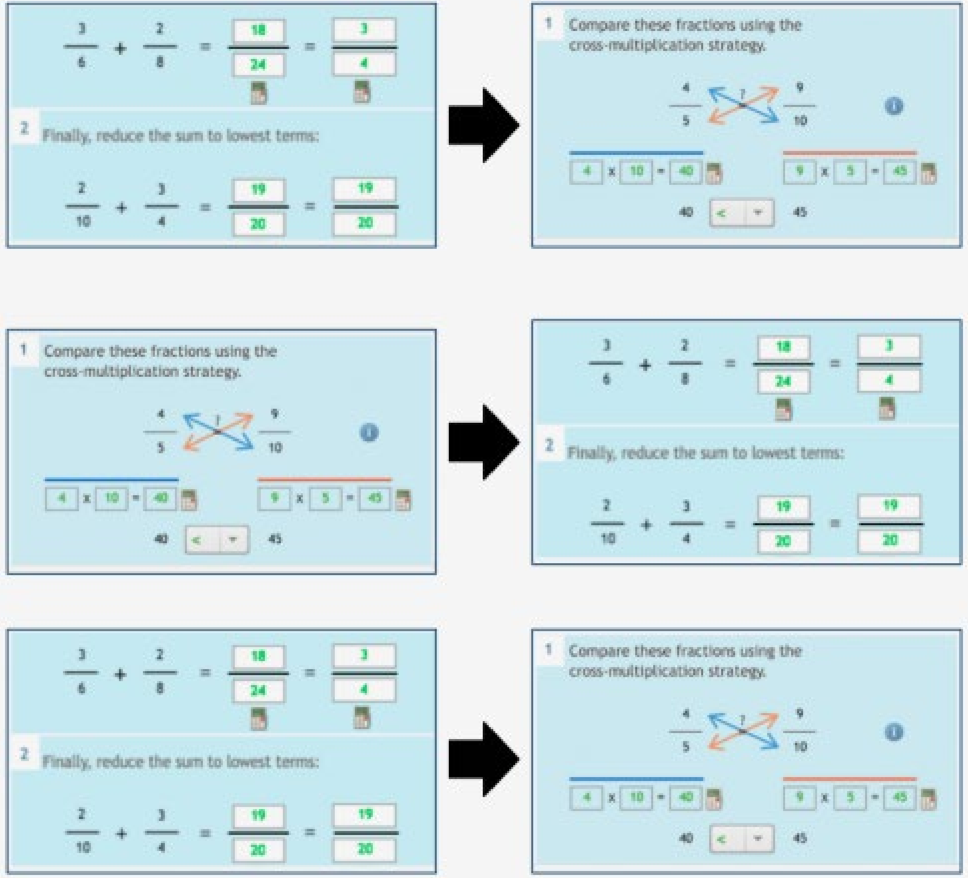
So is this good enough?

Why

A Group — Avg Score: 95

B Group — Avg Score: 92

B Group — ???



So is this good enough?

Why

Generalization

Fair, but can improve

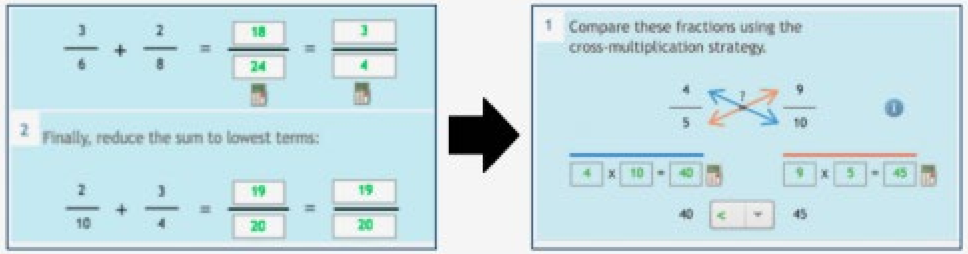
Q1. check on the meaning

In this case, generalization would mean making sure all candidate methods have been used to all possible groups to get rid of that factor that certain methods return unusual results when applied on certain groups

So, we want to reach towards a point where we know that all candidate methods can be generalized: know what averaged return they will give when applied to almost all of the possible groups to be tested on.

Improvement would include testing group A in the sequence that group B originally has gone through.

B Group



???

Why

Generalization

Connects...

The idea of using the data that already has been collected and making the most out of it to make decisions on moving forward

Especially when making bad decisions can be costly or dangerous

Similarly if obtaining more data itself is costly or not possible

Q2 Would this mean: obtaining the most generalized result from given data?

Why

Generalization

Connects...

The idea of using the data that already has been collected and making the most out of it to make decisions on moving forward

Especially when making bad decisions can be costly or dangerous

Similarly if obtaining more data itself is costly or not possible

→ Safe batch reinforcement learning

Safe Batch Reinforcement Learning

Some probability that the new policy will not be worse than the old policy

$$\Pr(V^{\mathcal{A}(\mathcal{D})} \geq V^{\pi_b}) \geq 1 - \delta$$

Easier to calc

$$\Pr(V^{\mathcal{A}(\mathcal{D})} \geq V_{min}) \geq 1 - \delta$$

Safe batch reinforcement learning

Safe Batch Reinforcement Learning

Off-policy Policy Evaluation (OPE)

Use data to make estimate of its Value function

High-confidence Off-policy Policy Evaluation (HCOPE)

Add the idea of confidence on the lower bound of value function

Safe Policy Improvement (SPI)

Safe batch reinforcement learning algorithm = which would mean finding improved policies

Safe Batch Reinforcement Learning

Historical Data, D
Proposed Policy, π_e

→ Estimate of V^{π_e} ➡ Importance Sampling (IS)

Historical Data, D
Proposed Policy, π_e
Probability, $1-\delta$

→ $1-\delta$ confidence lower bound on $J(\pi_e)$

Historical Data, D
Probability, $1-\delta$

→ New policy π, or `No Solution Found`

Importance Sampling

$$IS(D) = \frac{1}{n}\sum_{i=1}^{n}\left(\prod_{t=1}^{L}\frac{\pi_e(a_t \mid s_t)}{\pi_b(a_t \mid s_t)}\right)\left(\sum_{t=1}^{L}\gamma^t R_t^i\right)$$

Importance Sampling (IS)

Importance Sampling

$$IS(D) = \frac{1}{n}\sum_{i=1}^{n}\left(\prod_{t=1}^{L}\frac{\pi_e(a_t \mid s_t)}{\pi_b(a_t \mid s_t)}\right)\left(\sum_{t=1}^{L}\gamma^t R_t^i\right)$$

$$\prod \frac{p(a_j \mid s_j)^{\pi_e}}{p(a_j \mid s_j)^{\pi_b}} \qquad G(h_j)$$

Importance Sampling

$$IS(D) = \frac{1}{n}\sum_{i=1}^{n}\left(\prod_{t=1}^{L}\frac{\pi_e(a_t \mid s_t)}{\pi_b(a_t \mid s_t)}\right)\left(\sum_{t=1}^{L}\gamma^t R_t^i\right)$$

Q3.

n = number of batches (epochs?)

L = number of timesteps within batch(epoch?)

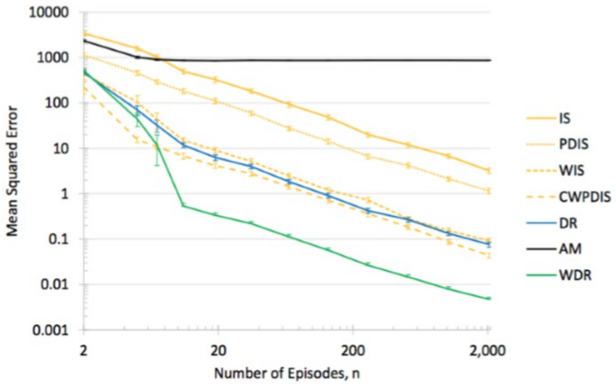
Importance Sampling

$$IS(D) = \frac{1}{n}\sum_{i=1}^{n}\left(\prod_{t=1}^{L}\frac{\pi_e(a_t \mid s_t)}{\pi_b(a_t \mid s_t)}\right)\left(\sum_{t=1}^{L}\gamma^t R_t^i\right)$$

$$PSID(D) = \sum_{t=1}^{L}\gamma^t \frac{1}{n}\sum_{i=1}^{n}\left(\prod_{\tau=1}^{t}\frac{\pi_e(a_\tau \mid s_\tau)}{\pi_b(a_\tau \mid s_\tau)}\right) R_t^i$$

$$WIS(D) = \frac{1}{\sum_{i=1}^{n} w_i}\sum_{i=1}^{n} w_i\left(\sum_{t=1}^{L}\gamma^t R_t^i\right)$$

$$DR(\pi_e \mid D) = \frac{1}{n}\sum_{i=1}^{n}\sum_{t=0}^{\infty}\gamma^t w_t^i(R_t^i - \hat{q}^{\pi_e}(S_t^i, A_t^i)) + \gamma^t \rho_{t-1}^i \hat{v}^{\pi_e}(S_t^i)$$



WDR

MAGIC

Importance Sampling

$$IS(D) = \frac{1}{n}\sum_{i=1}^{n}\left(\prod_{t=1}^{L}\frac{\pi_e(a_t \mid s_t)}{\pi_b(a_t \mid s_t)}\right)\left(\sum_{t=1}^{L}\gamma^t R_t^i\right)$$

$$PSID(D) = \sum_{t=1}^{L}\gamma^t \frac{1}{n}\sum_{i=1}^{n}\left(\prod_{\tau=1}^{t}\frac{\pi_e(a_\tau \mid s_\tau)}{\pi_b(a_\tau \mid s_\tau)}\right) R_t^i$$

temporal

$$WIS(D) = \frac{1}{\sum_{i=1}^{n} w_i}\sum_{i=1}^{n} w_i \left(\sum_{t=1}^{L}\gamma^t R_t^i\right)$$

weighted

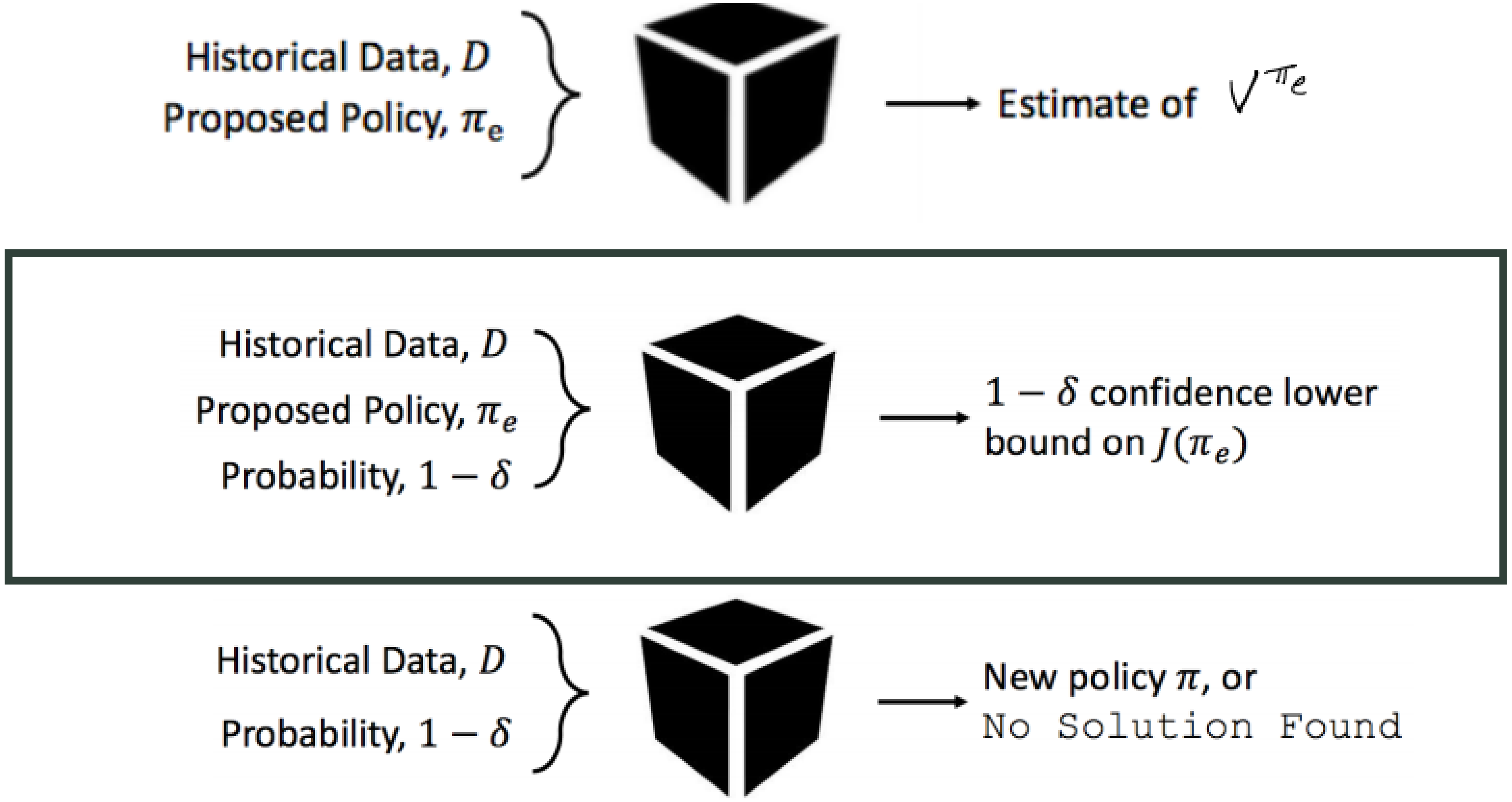
$$DR(\pi_e \mid D) = \frac{1}{n}\sum_{i=1}^{n}\sum_{t=0}^{\infty}\gamma^t w_t^i (R_t^i - \hat{q}^{\pi_e}(S_t^i, A_t^i)) + \gamma^t \rho_{t-1}^i \hat{v}^{\pi_e}(S_t^i),$$

approximated model + IS

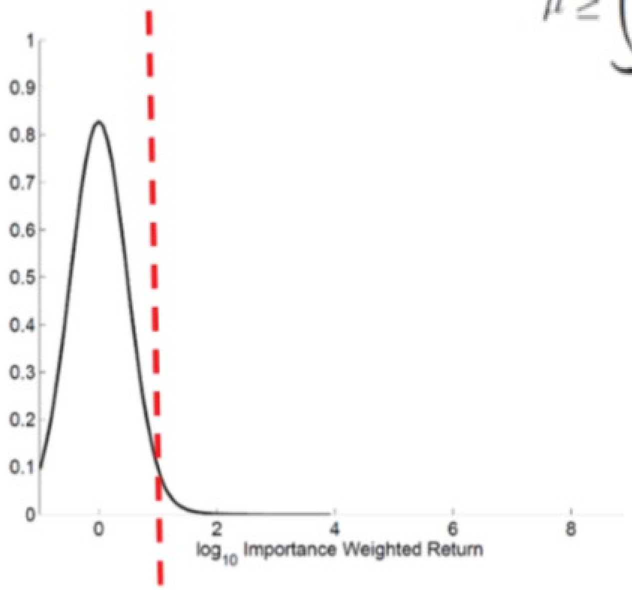
WDR weighted DR

MAGIC

High-confidence off-policy policy evaluation (HCOPE)



High-confidence off-policy policy evaluation (HCOPE)



$$\mu \geq \underbrace{\left(\sum_{i=1}^{n} \frac{1}{c_i}\right)^{-1} \sum_{i=1}^{n} \frac{Y_i}{c_i}}_{\text{empirical mean}} - \underbrace{\left(\sum_{i=1}^{n} \frac{1}{c_i}\right)^{-1} \frac{7n \ln(2/\delta)}{3(n-1)}}_{\text{term that goes to zero as } 1/n \text{ as } n \to \infty} - \underbrace{\left(\sum_{i=1}^{n} \frac{1}{c_i}\right)^{-1} \sqrt{\frac{\ln(2/\delta)}{n-1} \sum_{i,j=1}^{n} \left(\frac{Y_i}{c_i} - \frac{Y_j}{c_j}\right)^2}}_{\text{term that goes to zero as } 1/\sqrt{n} \text{ as } n \to \infty}.$$

1. Use some of the data to cutoff / tune the confidence interval
2. Compute lower bound (value function)

Frozen Lake

Used tensorflow and gym

Used e-greedy or random noise for just frozen lake (not slippery) deterministic

Apply
learning rate
for slow
learning

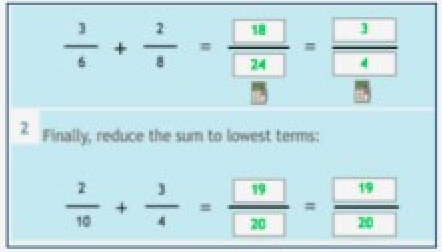
Used e-greedy or random noise for slippery and windy frozen lake stochastic

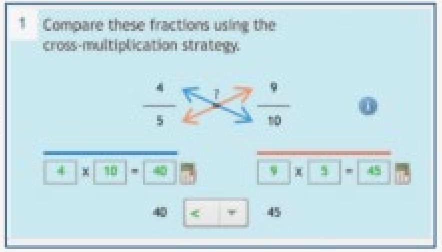
Used q-network for slippery and windy frozen lake

Lecture 15 : Batch RL

Why

A Group

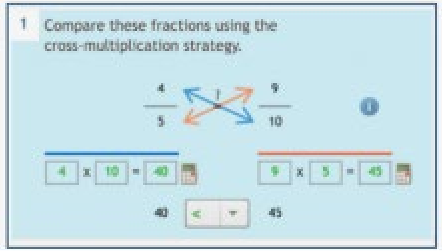


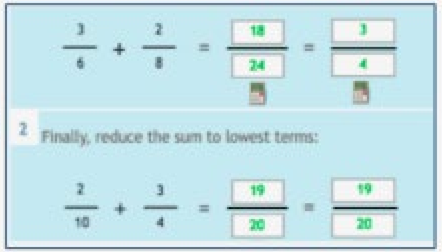


Avg Score: 95

Can we say that method applied on group A is good enough?

B Group

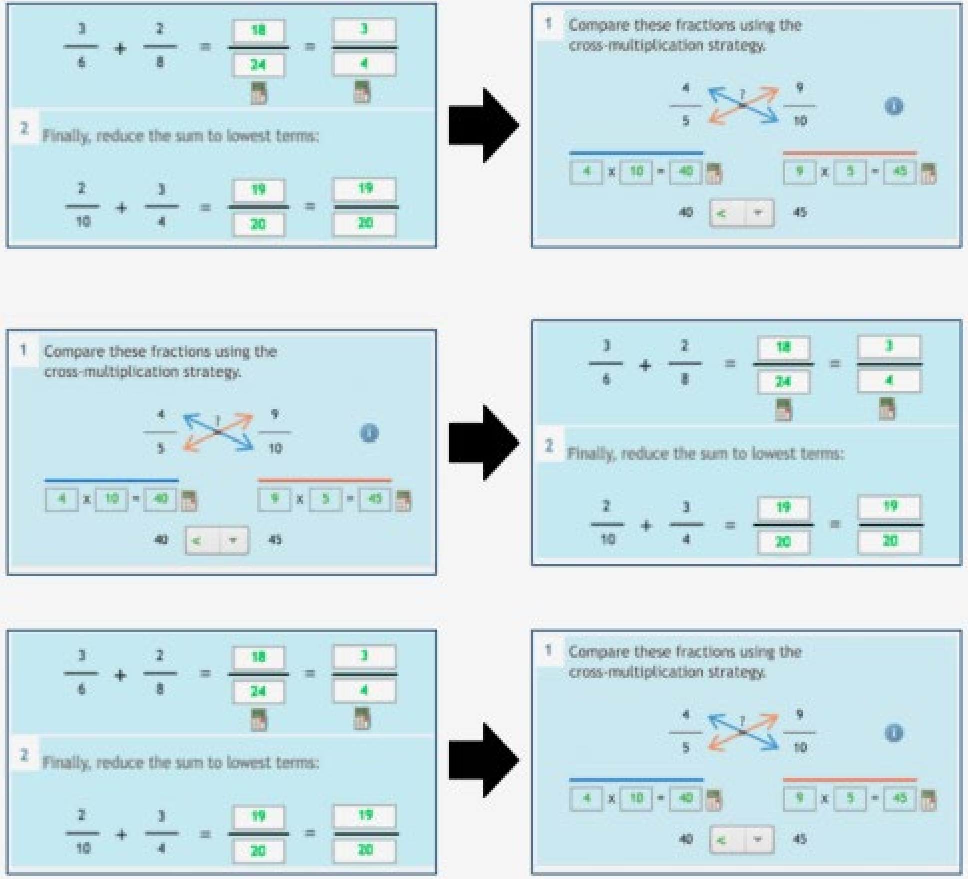




Avg Score: 92

Nope, you don't know what would have happened if group B went through same sequence of interventions as group A

Why



A Group — Avg Score: 95

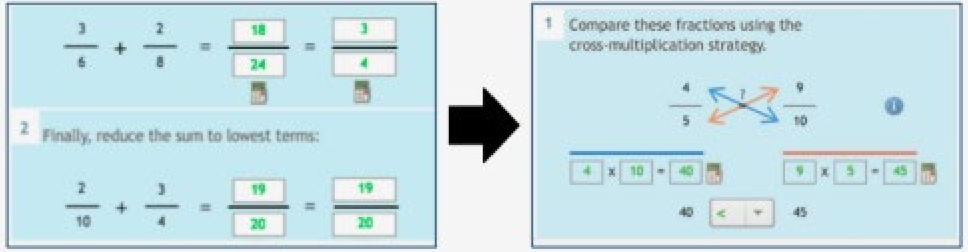
B Group — Avg Score: 92

B Group — ???

Test the first method by trying it and comparing it to what group B previously had done

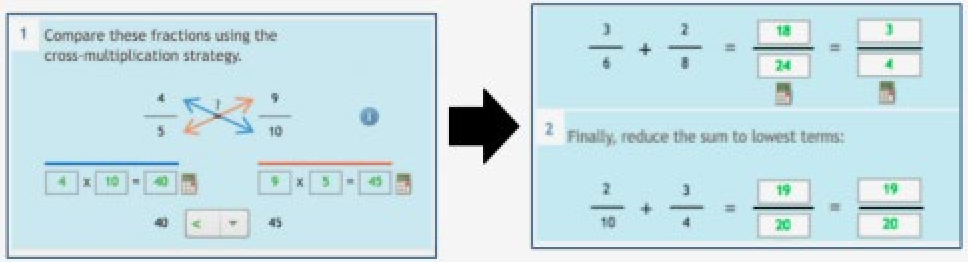
Why

A Group



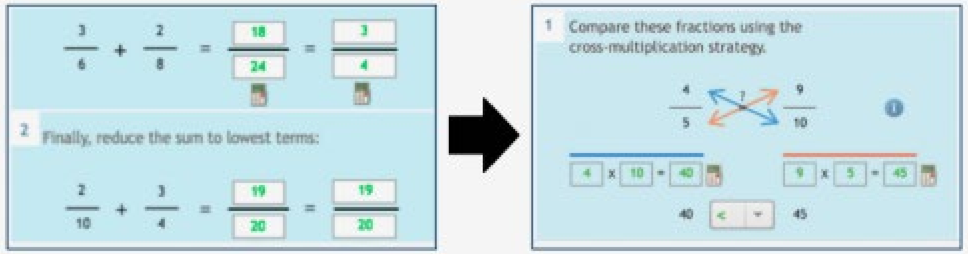
Avg Score: 95

B Group



Avg Score: 92

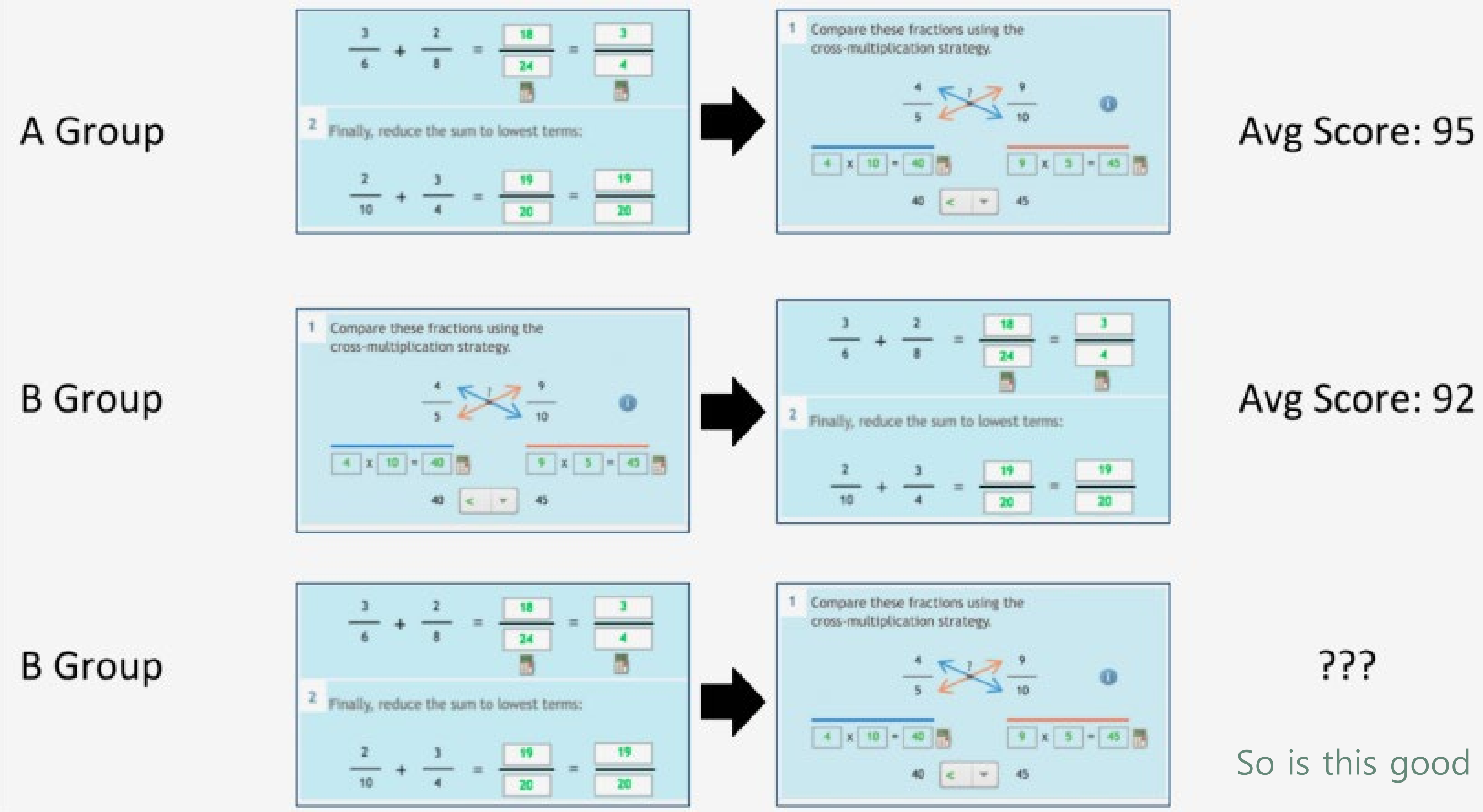
B Group



???

So is this good enough?

Why



So is this good enough?

Why

Generalization

Fair, but can improve

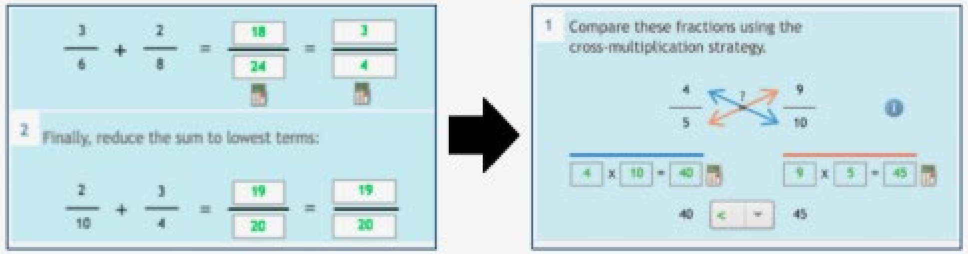
Q1. check on the meaning

In this case, generalization would mean making sure all candidate methods have been used to all possible groups to get rid of that factor that certain methods return unusual results when applied on certain groups

So, we want to reach towards a point where we know that all candidate methods can be generalized: know what averaged return they will give when applied to almost all of the possible groups to be tested on.

Improvement would include testing group A in the sequence that group B originally has gone through.

B Group



???

Why

Generalization

Connects...

The idea of using the data that already has been collected and making the most out of it to make decisions on moving forward

Especially when making bad decisions can be costly or dangerous

Similarly if obtaining more data itself is costly or not possible

Q2 Would this mean: obtaining the most generalized result from given data?

Why

Generalization

Connects...

The idea of using the data that already has been collected and making the most out of it to make decisions on moving forward

Especially when making bad decisions can be costly or dangerous

Similarly if obtaining more data itself is costly or not possible

→ Safe batch reinforcement learning

Safe Batch Reinforcement Learning

Some probability that the new policy will not be worse than the old policy

$$\Pr(V^{\mathcal{A}(\mathcal{D})} \geq V^{\pi_b}) \geq 1 - \delta$$

Easier to calc

$$\Pr(V^{\mathcal{A}(\mathcal{D})} \geq V_{min}) \geq 1 - \delta$$

Safe batch reinforcement learning

Safe Batch Reinforcement Learning

Off-policy Policy Evaluation (OPE)

Use data to make estimate of its Value function

High-confidence Off-policy Policy Evaluation (HCOPE)

Add the idea of confidence on the lower bound of value function

Safe Policy Improvement (SPI)

Safe batch reinforcement learning algorithm = which would mean finding improved policies

Safe Batch Reinforcement Learning

Historical Data, D
Proposed Policy, π_e

→ Estimate of V^{π_e} ➡ Importance Sampling (IS)

Historical Data, D
Proposed Policy, π_e
Probability, $1-\delta$

→ $1-\delta$ confidence lower bound on $J(\pi_e)$

Historical Data, D
Probability, $1-\delta$

→ New policy π, or `No Solution Found`

Importance Sampling

$$IS(D) = \frac{1}{n} \sum_{i=1}^{n} \left(\prod_{t=1}^{L} \frac{\pi_e(a_t \mid s_t)}{\pi_b(a_t \mid s_t)} \right) \left(\sum_{t=1}^{L} \gamma^t R_t^i \right)$$

Importance Sampling (IS)

Importance Sampling

$$IS(D) = \frac{1}{n}\sum_{i=1}^{n}\left(\prod_{t=1}^{L}\frac{\pi_e(a_t \mid s_t)}{\pi_b(a_t \mid s_t)}\right)\left(\sum_{t=1}^{L}\gamma^t R_t^i\right)$$

$$\prod \frac{p(a_j \mid s_j)^{\pi_e}}{p(a_j \mid s_j)^{\pi_b}} \qquad G(h_j)$$

Importance Sampling

$$IS(D) = \frac{1}{n}\sum_{i=1}^{n}\left(\prod_{t=1}^{L}\frac{\pi_e(a_t \mid s_t)}{\pi_b(a_t \mid s_t)}\right)\left(\sum_{t=1}^{L}\gamma^t R_t^i\right)$$

Q3.

n = number of batches (epochs?)

L = number of timesteps within batch(epoch?)

Importance Sampling

$$IS(D) = \frac{1}{n}\sum_{i=1}^{n}\left(\prod_{t=1}^{L}\frac{\pi_e(a_t \mid s_t)}{\pi_b(a_t \mid s_t)}\right)\left(\sum_{t=1}^{L}\gamma^t R_t^i\right)$$

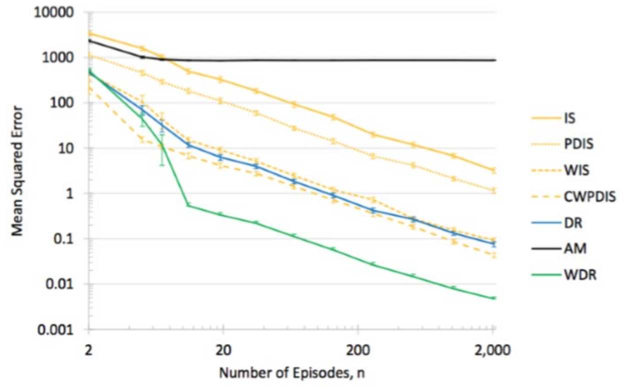
$$PSID(D) = \sum_{t=1}^{L}\gamma^t \frac{1}{n}\sum_{i=1}^{n}\left(\prod_{\tau=1}^{t}\frac{\pi_e(a_\tau \mid s_\tau)}{\pi_b(a_\tau \mid s_\tau)}\right) R_t^i$$

$$WIS(D) = \frac{1}{\sum_{i=1}^{n} w_i}\sum_{i=1}^{n} w_i\left(\sum_{t=1}^{L}\gamma^t R_t^i\right)$$

$$DR(\pi_e \mid D) = \frac{1}{n}\sum_{i=1}^{n}\sum_{t=0}^{\infty}\gamma^t w_t^i(R_t^i - \hat{q}^{\pi_e}(S_t^i, A_t^i)) + \gamma^t \rho_{t-1}^i \hat{v}^{\pi_e}(S_t^i).$$

WDR

MAGIC



Importance Sampling

$$IS(D) = \frac{1}{n}\sum_{i=1}^{n}\left(\prod_{t=1}^{L}\frac{\pi_e(a_t \mid s_t)}{\pi_b(a_t \mid s_t)}\right)\left(\sum_{t=1}^{L}\gamma^t R_t^i\right)$$

$$PSID(D) = \sum_{t=1}^{L}\gamma^t \frac{1}{n}\sum_{i=1}^{n}\left(\prod_{\tau=1}^{t}\frac{\pi_e(a_\tau \mid s_\tau)}{\pi_b(a_\tau \mid s_\tau)}\right) R_t^i$$

temporal

$$WIS(D) = \frac{1}{\sum_{i=1}^{n} w_i}\sum_{i=1}^{n} w_i \left(\sum_{t=1}^{L}\gamma^t R_t^i\right)$$

weighted

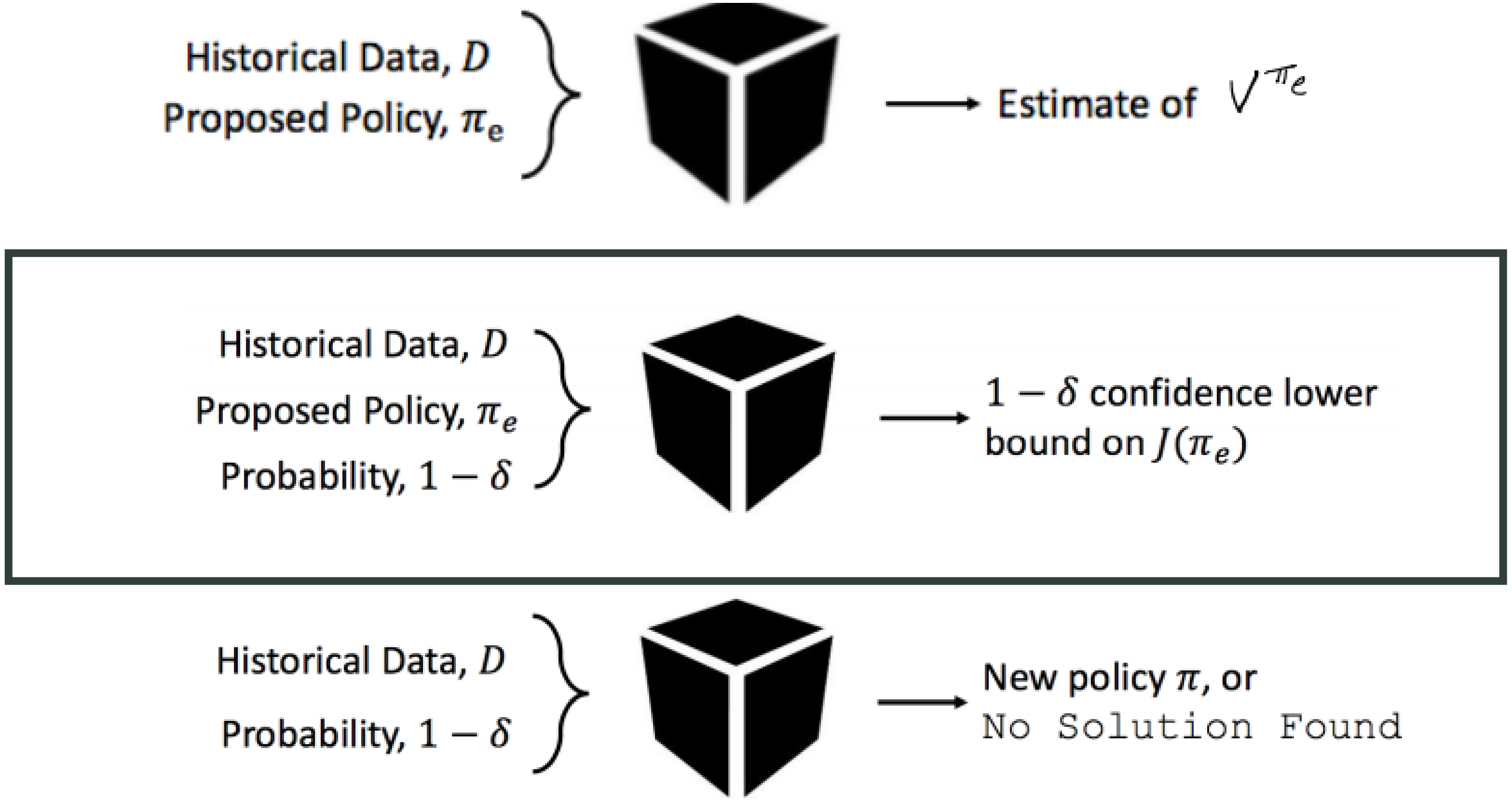
$$DR(\pi_e \mid D) = \frac{1}{n}\sum_{i=1}^{n}\sum_{t=0}^{\infty}\gamma^t w_t^i (R_t^i - \hat{q}^{\pi_e}(S_t^i, A_t^i)) + \gamma^t \rho_{t-1}^i \hat{v}^{\pi_e}(S_t^i),$$

approximated model + IS

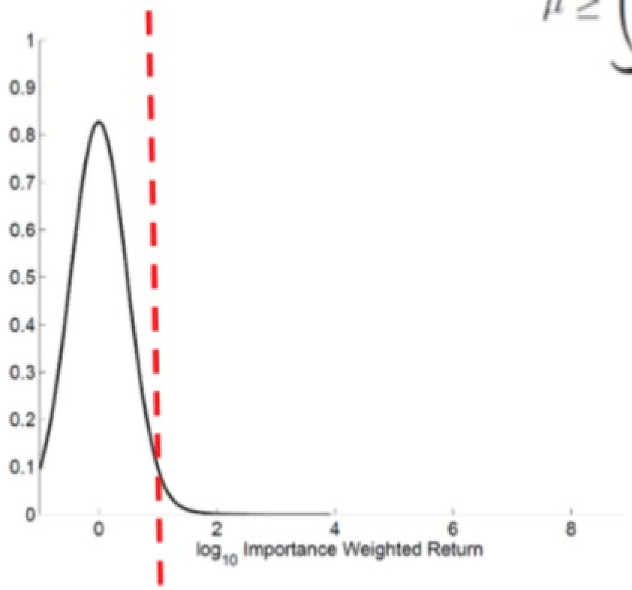
WDR weighted DR

MAGIC

High-confidence off-policy policy evaluation (HCOPE)



High-confidence off-policy policy evaluation (HCOPE)



$$\mu \geq \underbrace{\left(\sum_{i=1}^{n}\frac{1}{c_i}\right)^{-1}\sum_{i=1}^{n}\frac{Y_i}{c_i}}_{\text{empirical mean}} - \underbrace{\left(\sum_{i=1}^{n}\frac{1}{c_i}\right)^{-1}\frac{7n\ln(2/\delta)}{3(n-1)}}_{\text{term that goes to zero as } 1/n \text{ as } n\to\infty} - \underbrace{\left(\sum_{i=1}^{n}\frac{1}{c_i}\right)^{-1}\sqrt{\frac{\ln(2/\delta)}{n-1}\sum_{i,j=1}^{n}\left(\frac{Y_i}{c_i}-\frac{Y_j}{c_j}\right)^2}}_{\text{term that goes to zero as } 1/\sqrt{n} \text{ as } n\to\infty}.$$

1. Use some of the data to cutoff / tune the confidence interval
2. Compute lower bound (value function)

lecture 15

Frozen Lake

Used tensorflow and gym

Used e-greedy or random noise for just frozen lake (not slippery) deterministic

Apply
learning rate
for slow
learning

Used e-greedy or random noise for slippery and windy frozen lake stochastic

Used q-network for slippery and windy frozen lake