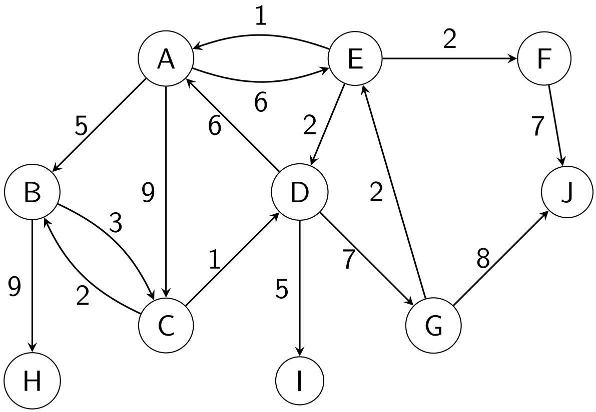
CS-583: Deep Learning
Reinforcement Learning

Abdul Rafae Khan
Department of Computer Science
Stevens Institute of Technology
akhan4@stevens.edu

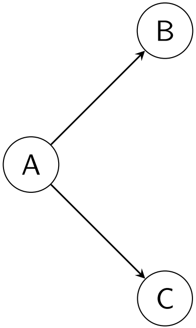
November 9, 2023

Search Problem



Deterministic Actions

Actions from a give state are deterministic
$Succ(s, a)$ is always the same state s'

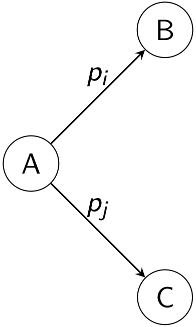


Stochastic Actions

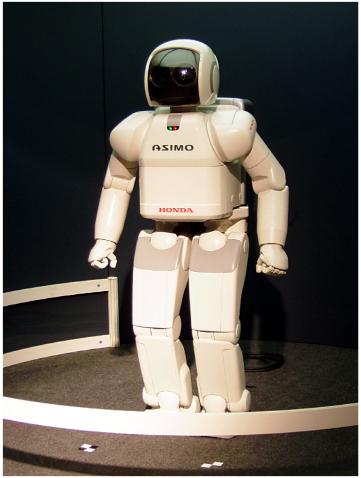
Actions from a give state are probabilistic (stochastic)

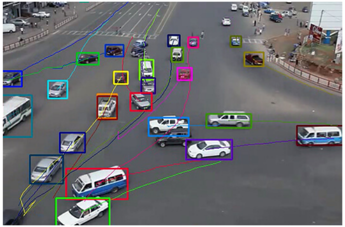
$Succ(s, a, t)$ denotes the next state given the current state s and action a taken at the time t_i

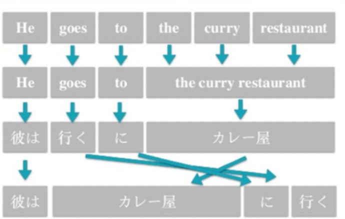
It can either be state B with a probability p_i or state C with probability p_j



Applications







Markov Decision Process

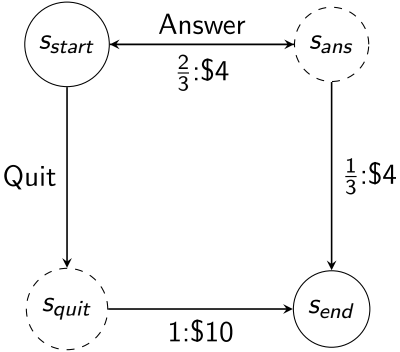
Game:

The player starts with \$0 as the prize money.

In each round, the player can take two steps:

- Quit and take \$10
- Answer a question
  - Correctly answer with a probability of $\frac{2}{3}$, get \$4 prize and move to the next round
  - Otherwise get \$4 prize and end the game

Markov Decision Process



Markov Decision Process

Gameshow:

The player starts with 0 as the prize money.

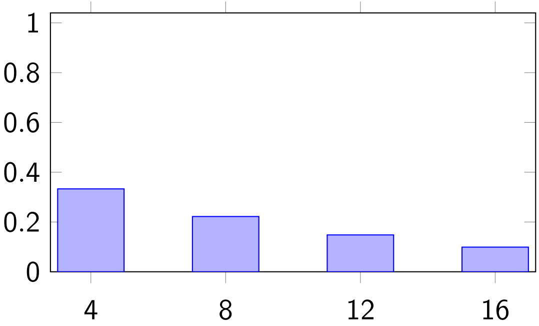
In each round, the player can take two steps:

- Quit and take \$10
- Answer a question
  - Correctly answer with a probability of $\frac{2}{3}$ and move to the next round
  - Otherwise take \$4 and end the game

What is the best strategy for the game?

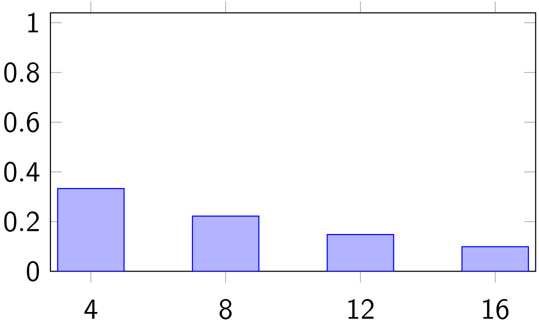
Markov Decision Process

If our policy is to ‘answer’:



Markov Decision Process

If our policy is to 'answer':

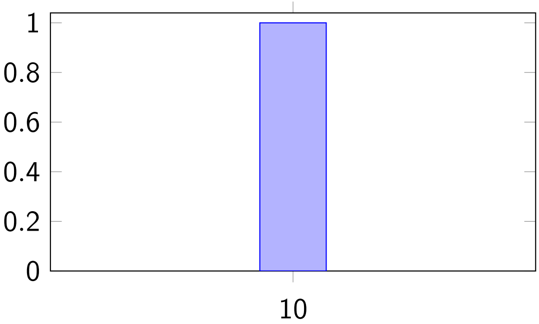


Expected Utility:

$$\frac{1}{3}(4) + \frac{2}{3} \cdot \frac{1}{3}(8) + \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{1}{3}(12) + \cdots = 12$$

Markov Decision Process

If our policy is to 'quit':



Expected Utility:

$$1(10) = 10$$

Search problem

s_{start}: start state
Actions(s**)**: all possible actions from state s
Succ(s, a**)**: next possible states given action a is taken from state s
Cost(s, a**)**: cost of transition from state s by taking action a
IsEnd(s**)**: is s a goal state

Markov Decision Process

s_{start}: start state
Actions(s): all possible actions from state s
T(s, a, s'): probability of s' if action a is taken from state s
Reward(s, a, s'): reward from the transition s to s'
IsEnd(s): is s a goal state
$0 \leq \gamma \leq 1$: discount factor (default: 1)

Markov Decision Process

Total transition probability: $\sum_{s'} T(s, a, s') = 1$
Discount factor γ is based on how much we value the future reward

Markov Decision Process

Succ$(s, a) \rightarrow$ T(s, a, s')
Succ(s, a) can be considered as a special case of transition probability

$$T(s, a, s') = \begin{cases} 1 \text{ if } s' = Succ(s, a) \\ 0 \text{ otherwise} \end{cases}$$

Markov Decision Process

$\text{Cost}(s, a) \rightarrow \text{Reward}(s, a, s')$
Instead of minimizing the cost, we maximize the reward
Negating one is equivalent to the other

Markov Decision Process

$\mathbf{T}(s, a, s')$: probability of s' if action a is taken from state s

s	a	s'	$T(s, a, s')$
s_{start}	$Quit$	s_{end}	1
s_{start}	$Question$	s_{end}	$1/3$
s_{start}	$Question$	s_{start}	$2/3$

Markov Decision Process

$\mathbf{T}(s, a, s')$: probability of s' if action a is taken from state s

s	a	s'	$\mathrm{T}(s, a, s')$
s_{start}	$Quit$	s_{end}	1
s_{start}	$Question$	s_{end}	$1/3$
s_{start}	$Question$	s_{start}	$2/3$

To re-iterate:
Sum of probabilities from a given state s by making an action a is 1

$$\sum_{s' \in states} T(s, a, s') = 1$$

Successors: states s' where $T(s, a, s') > 0$

Markov Decision Process

$\mathbf{T}(s, a, s')$: probability of s' if action a is taken from state s

s	a	s'	$T(s, a, s')$
s_{start}	$Quit$	s_{end}	1
s_{start}	$Question$	s_{end}	$^1/_3$
s_{start}	$Question$	s_{start}	$^2/_3$

Sum of probabilities from a given state s by making an action a is 1

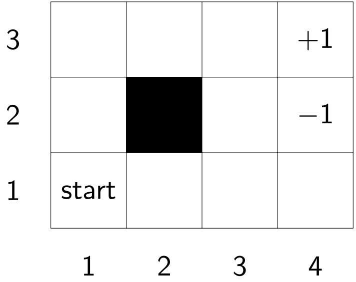
Policy

Policy: gives an action a for a given $\pi : s \rightarrow s$

For deterministic search problems, we wanted the optimal sequence of actions from start to goal
For MDP, we want the optimal policy $\pi^* : s \rightarrow a$ which maximizes the reward Reward(s, a, s')

Grid World!

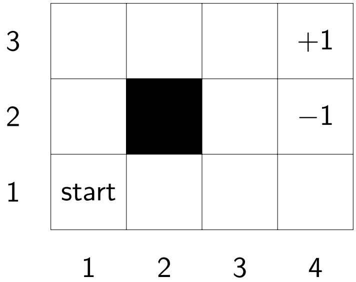
Our world is 3×4 grid
Start state is at (0,0)
Reward $+1$ at (4,3)
Reward -1 at (4,2)



Grid World!

For any state, three possible moves

- up: 0.8
- left: 0.1
- right: 0.1



Grid World!

3	$\rightarrow$	$\rightarrow$	$\rightarrow$	$+1$
2	$\uparrow$		$\uparrow$	-1
1	$\uparrow$	$\leftarrow$	$\leftarrow\uparrow$	$\leftarrow$
	1	2	3	4

Optimal policy for $\gamma < -0.04$

There are two optimal policies for state (3,1)

Discount

Additive discount utility

Let say the path is $s_0, a_1 r_1 s_1, a_2 r_2 s_2, \cdot$ (sequence of state, action, and reward)

The utility with discount γ is:

$\mathrm{R}(s, a, s') + \gamma \mathrm{R}(s, a, s') + \gamma^2 \mathrm{R}(s, a, s') + \cdots$ where $\gamma \in [0, 1]$

γ is based on how important current reward is compared to the future reward

Discount

Solving the problem of infinite stream of rewards
Geometric series: $1 + \gamma + \gamma^2 + \ldots = 1/(1-\gamma)$
Assume rewards bounded by $\pm R_{max}$
Then $r_0 + \gamma_1 r_1 + \gamma_2 r_2 + \ldots$ is bounded by $\pm R_{max}/(1-\gamma)$

Policy Evaluation

The **utility** is the discounted sum of rewards on the path.
Optimal policy: $\pi^*(s)$ = optimal actions from state s
It gives highest $U_\pi(s)$ for any π

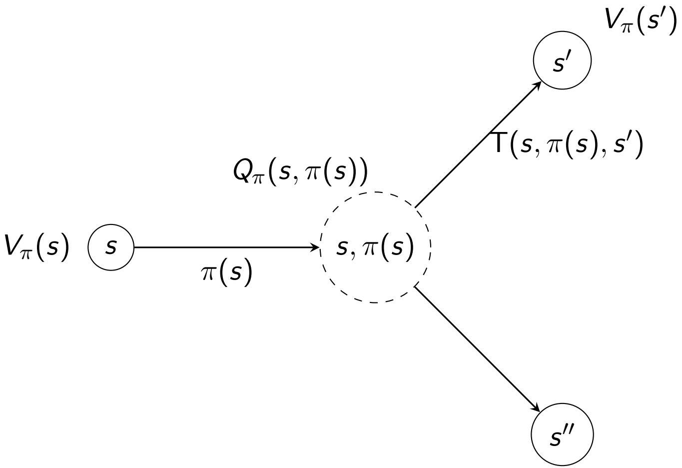
$$U_\pi(s) = R(s, a, s') + \gamma R(s, a, s') + \gamma^2 R(s, a, s') + \cdots$$

Policy Evaluation

For a given policy π, we have two variable associated with it:

- Value of the policy $V_\pi(s)$
- Q-value of the policy $Q_\pi(s, \pi(s))$

Markov Decision Process



Policy Evaluation

For a given policy π, we have two variable associated with it:

- Value of the policy $V_\pi(s)$
- Q-value of the policy $Q_\pi(s, \pi(s))$

The value can be thought of as the label for the nodes representing the states and the Q-value as the label for the chance nodes

Policy Evaluation

Value is the expected utility from following policy π from state s
Q-value is the expected utility of taking action a from state s, and then following policy π.

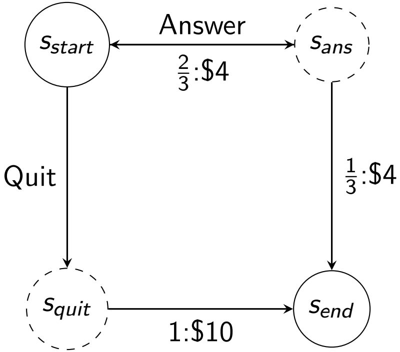
$$V_\pi(s) = E[V_\pi(s)] = \begin{cases} 0 \text{ if } \mathit{isEnd}(s) \\ Q_\pi(s) \text{ otherwise} \end{cases}$$

$$Q_\pi(s) = \sum_{s'} T(s'|s,a)[R(s,a,s') + \gamma V(s')]$$

Policy Evaluation

Let the policy π be '*Answer*':

$$\begin{aligned} V_\pi(s_{end}) &= 0 \\ V_\pi(s_{start}) &= Q_\pi(s_{start}, Answer) \\ &= \frac{1}{3}(4 + V_\pi(s_{end})) + \frac{2}{3}(4 + V_\pi(s_{start})) \\ \implies V_\pi(s_{start}) &= \frac{1}{3}(4) + \frac{2}{3}(4 + V_\pi(s_{start})) \end{aligned}$$



Closed form solution:

$$\begin{aligned} 3V_\pi(s_{start}) &= 4 + 2 \cdot 4 + 2V_\pi(s_{start}) \\ V_\pi(s_{start}) &= 12 \end{aligned}$$

Policy Evaluation

Given the recursion $V^*(s) = \max_a Q^*(s, a)$

Value:

$$V^*(s) = \max_{a \in Actions(s)} \sum_{s'} \{P(s'|s, a)[R(s, a, s') + \gamma V(s')]\}$$

Q-value:

$$\begin{aligned} Q^*(s, a) &= \sum_{s'} \{P(s'|s, a)[R(s, a, s') + \gamma V(s')]\} \\ &= \sum_{s'} \{P(s'|s, a)[R(s, a, s') + \gamma \max_{a'} Q(s', a')]\} \end{aligned}$$

Markov Decision Process

Solving MDPs:

- Value Iteration
- Policy Iteration

Policy Iteration

$V_\pi^{(0)}(s) \leftarrow 0$

$\texttt{for}\ \ i = 1 \cdots t_{max}$

$\quad\quad \texttt{for each state}\ \ s$

$\quad\quad\quad\quad V_\pi^{(t)}(s) \leftarrow \sum_{s'} T(s'|s,a)[R(s,\pi(s),s') + \gamma V_\pi^{(t-1)}(s')]$

Policy Iteration

$V_\pi^{(0)}(s) \leftarrow 0$

$\texttt{for} \;\; i = 1 \cdots t_{max}$

$\qquad \texttt{for each state} \;\; s$

$$V_\pi^{(t)}(s) \leftarrow \underbrace{\sum_{s'} T(s'|s,a)[R(s,\pi(s),s') + \gamma V_\pi^{(t-1)}(s')]}_{Q_\pi^{(t-1)}(s)}$$

Policy Evaluation

How many iterations (t_{max})?
Repeat until there is no/very little change

$$\max_{s \in states} |V_\pi^{(t)}(s) - V_\pi^{(t-1)}(s)| \leq \epsilon$$

Only save the last two iterations, $V_\pi^{(t)}$ & $V_\pi^{(t-1)}$

Policy Iteration

$V_\pi^{(0)}(s) \leftarrow 0$
for $i = 1 \cdots t_{max}$
    for each state s
        $V_\pi^{(t)}(s) \leftarrow \sum_{s'} T(s'|s,a)[R(s,\pi(s),s') + \gamma V_\pi^{(t-1)}(s')]$

Total states: S
Actions per state: A
Total successor (with $\mathrm{T}(s'|s,a) > 0$): S'

Complexity: $O(SS' t_{max})$

Policy Iteration

Let the policy π be '*Answer*':

$$V_\pi^{(t)}(s_{end}) = 0$$

$$V_\pi^{(t)}(s_{start}) = \frac{1}{3}(4 + V_\pi^{(t-1)}(s_{end})) + \frac{2}{3}(4 + V_\pi^{(t-1)}(s_{start}))$$

Iteration (t)	$V_\pi^{(t)}(s_{end})$	$V_\pi^{(t)}(s_{start})$
0	0.00	0.00
1	0.00	4.00
2	0.00	6.67
3	0.00	8.44
100	0.00	12.00

$$V_\pi^{(t)}(s_{start}) = 12$$

Optimal Value

Goal: try to get directly at maximum expected utility
$V_{opt}(s)$ = is the maximum value obtained by any policy

Optimal Value

Given the recursion $V_{opt}(s) = \max_a Q_{opt}(s, a)$

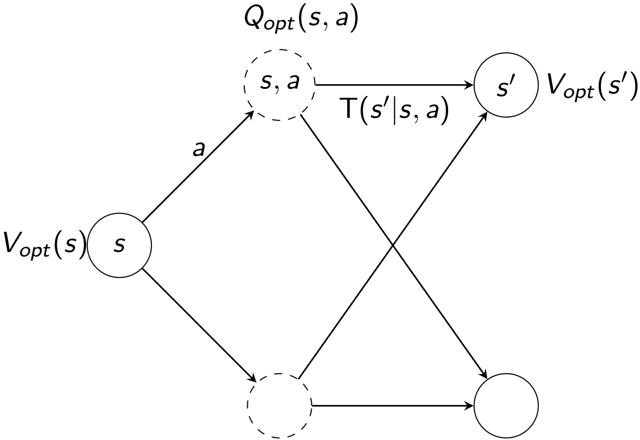
Value:

$$V_{opt}(s) = \max_{a \in Actions(s)} \sum_{s'} \{T(s'|s, a)[R(s, a, s') + \gamma V_{opt}(s')]\}$$

Q-value:

$$\begin{aligned} Q_{opt}(s, a) &= \sum_{s'} \{T(s'|s, a)[R(s, a, s') + \gamma V_{opt}(s')]\} \\ &= \sum_{s'} \{T(s'|s, a)[R(s, a, s') + \gamma \max_{a'} Q_{opt}(s', a')]\} \end{aligned}$$

Optimal Value



Optimal Value

Policy evaluation used the action from a fixed policy π
Now we pick the action which maximizes the Q-value $Q_{opt}(s)$

$$V_{opt}(s) = \begin{cases} 0 \text{ if } \mathit{isEnd}(s) \\ \max_{a \in Actions(s)} Q_{opt}(s) \text{ otherwise} \end{cases}$$

$$Q_{opt}(s) = \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V_{opt}(s')]$$

Optimal Policy

As for any state s, $Q_\pi(s)$ gives you the value of taking the policy $\pi(s)$
Therefore, **Optimal policy** π_{opt} in state s is the one which gives the largest value for $Q_{opt}(s)$

$$\pi_{opt}(s) = \underset{a \in Actions(s)}{\arg\max}\ Q_{opt}(s)$$

Value Iteration

$V_{opt}^{(0)}(s) \leftarrow 0$

$\texttt{for } i = 1 \cdots t_{max}$

$\quad\quad \texttt{for each state } s$

$\quad\quad\quad\quad V_{opt}^{(t)}(s) \leftarrow \max_{a \in Actions(s)} \sum_{s'} T(s,a,s')[R(s,\pi(s),s') + \gamma V_{opt}^{(t-1)}(s')]$

Value Iteration

$V_{opt}^{(0)}(s) \leftarrow 0$

$\texttt{for}\ \ i = 1 \cdots t_{max}$

$\qquad \texttt{for each state}\ \ s$

$$V_{opt}^{(t)}(s) \leftarrow \max_{a \in Actions(s)} \underbrace{\sum_{s'} T(s, a, s')[R(s, \pi(s), s') + \gamma V_{opt}^{(t-1)}(s')]}_{Q_{opt}^{(t-1)}(s)}$$

Value Iteration

$V_{opt}^{(0)}(s) \leftarrow 0$
for $i = 1 \cdots t_{max}$
    for each state s
        $V_{opt}^{(t)}(s) \leftarrow \max_{a \in Actions(s)} \sum_{s'} T(s,a,s')[R(s,\pi(s),s') + \gamma V_{opt}^{(t-1)}(s')]$

Total states: S
Actions per state: A
Total successor: S'

Complexity: $O(SAS't_{max})$

Value Iteration

$V_{opt}^{(0)}(s) \leftarrow 0$
`for` $i = 1 \cdots t_{max}$
    `for each state` s
        $V_{opt}^{(t)}(s) \leftarrow \max_{a \in Actions(s)} \sum_{s'} T(s, a, s')[R(s, \pi(s), s') + \gamma V_{opt}^{(t-1)}(s')]$

argmax instead of **max** will give the optimal policy π_{opt}

Value Iteration

Iteration (t)	$V_{opt}^{(t)}(s_{end})$	$V_{opt}^{(t)}(s_{start})$	$\pi_{opt}(s_{end})$	$\pi_{opt}(s_{start})$
0	0.00	0.00	-	-
1	0.00	10.00	-	Quit
2	0.00	10.67	-	Answer
3	0.00	11.11	-	Answer
100	0.00	12.00	-	Answer

$$V_{\pi}^{(t)}(s_{start}) = 12$$

Recap

s_{start}: start state
Actions(s): all possible actions from state s
T(s, a, s'): probability of s' if action a is taken from state s
Reward(s, a, s'): reward from the transition s to s'
IsEnd(s): is s a goal state
$0 \leq \gamma \leq 1$: discount factor (default: 1)

Unknown Transitions & Reward

s_{start}: start state
Actions(s): all possible actions from state s
~~**T(s, a, s')**: probability of s' if action a is taken from state s~~
~~**Reward(s, a, s')**: reward from the transition s to s'~~
IsEnd(s): is s a goal state
$0 \leq \gamma \leq 1$: discount factor (default: 1)

Unknown Transitions & Reward

s_{start}: start state
Actions(s): all possible actions from state s
~~**T(s, a, s')**: probability of s' if action a is taken from state s~~
~~**Reward(s, a, s')**: reward from the transition s to s'~~
IsEnd(s): is s a goal state
$0 \leq \gamma \leq 1$: discount factor (default: 1)

Reinforcement Learning!

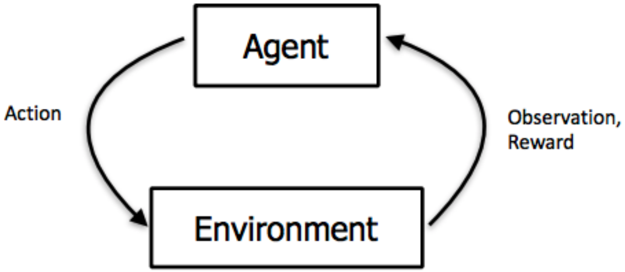
Unknown Transitions & Reward

MDPs:
Know how the word works: Environment is observable
Find a policy which maximizes the reward

Reinforcement learning:
Do not know about the world: Environment is not observable
Find a policy which maximizes the reward
Perform actions and collect the reward

Reinforcement Learning

The agent performs actions and observes the rewards
This feedback loop helps learn the missing values (transition probabilities and reward)



Reinforcement Learning

Overall algorithm

```
for t = 1, 2, 3, ···
    Choose action a_t = π_act(s_{t-1})
    Get reward r_t and new state s_t
    Update parameters
```

Model-based Monte Carlo

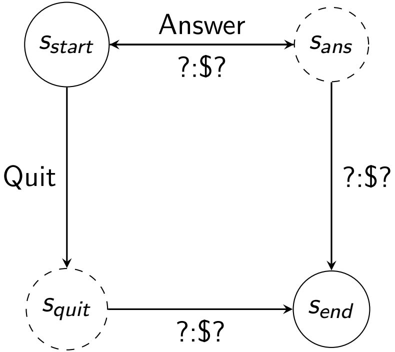
Data: $s_0; a_1 r_1 s_1; a_2, r_2, s_2; a_3, r_3, s_3; \cdots$
Estimate $T(s, a, s')$ & $R(s, a, s')$

$$\hat{T}(s, a, s') = \frac{\text{No. of times } s, a, s' \text{ occurs}}{\text{No. of times } s, a \text{ occurs}}$$

$$\hat{R}(s, a, s') = \text{reward observed by } s, a, s'$$

Model-based Monte Carlo

Iteration: 0

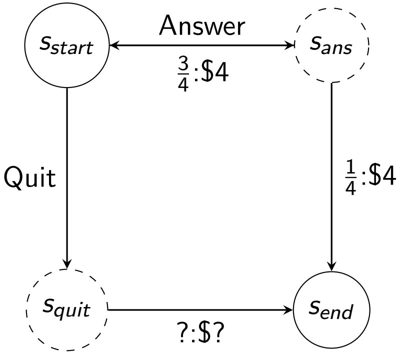


Model-based Monte Carlo

Policy π is Answer

Iteration: 1

Data:s_{start}; $Ans, 4, s_{start}$; $Ans, 4, s_{start}$; $Ans, 4, s_{start}$; $Ans, 4, s_{end}$

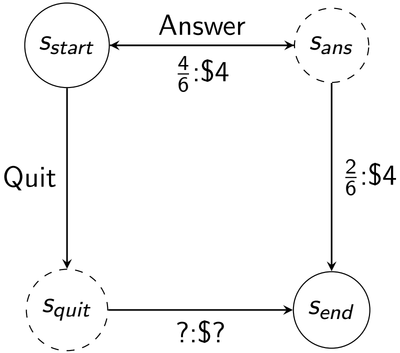


Model-based Monte Carlo

Policy π is Answer

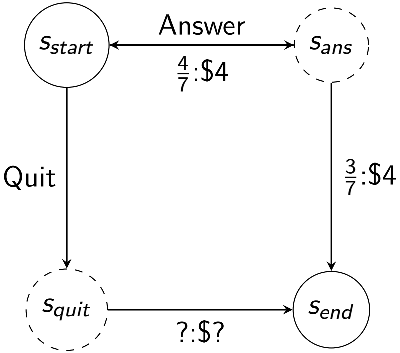
Iteration: 2

Data:s_{start}; $Ans, 4, s_{start}$; $Ans, 4, s_{end}$

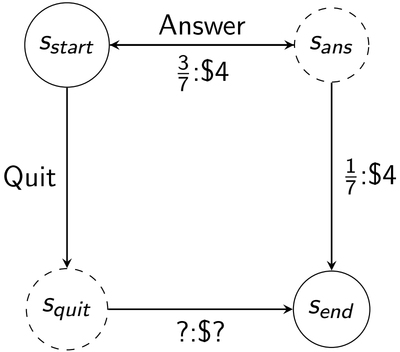


Model-based Monte Carlo

Policy π is Answer
Iteration: 3
Data: s_{start}; $Ans, 4, s_{end}$



Model-based Monte Carlo

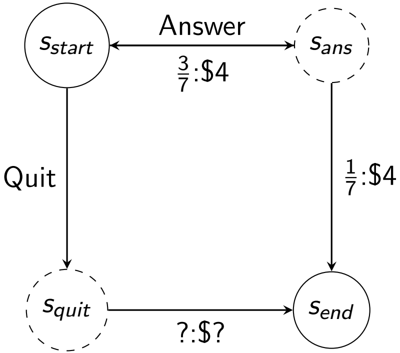


Can converge to true values

Compute policy using value Iteration for the estimated MDP (with $\hat{T}$ and $\hat{R}$)

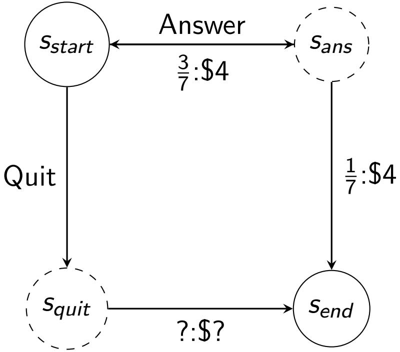
Model-based Monte Carlo

If $a \neq \pi(s)$ ($a = Quit$), s, a will not be seen



Model-based Monte Carlo

Exploration: try unknown actions to get information



Model-based Monte Carlo

We can use the computed transitions and rewards
And compute the optimal Value and Q-value

$$\hat{V}_{opt}(s) = E[\hat{V}_{opt}(s)] = \begin{cases} 0 \text{ if } isEnd(s) \\ \hat{Q}_{opt}(s) \text{ otherwise} \end{cases}$$

$$\hat{Q}_{opt}(s,a) = \sum_{s'} \hat{T}(s,a,s')[\hat{R}(s,a,s') + \gamma \hat{V}_{opt}(s')]$$

Model-based Monte Carlo

Pros:

- Makes efficient use of experiences

Cons:

- May not scale to large state spaces
  - Learns model one state-action pair at a time
  - Cannot solve MDP for very large $|S|$

Model-based vs Model-free

Goal: Compute the age of CS students

$P(A)$ is known

$$
\begin{aligned}
\mathbb{E}[A] &= \sum_a P(A) \cdot a \\
&= 0.35 \times 20 + \cdots
\end{aligned}
$$

Model-based vs Model-free

Without $P(A)$, collect samples $[a_1, a_2, \cdots, a_N]$

Unknown $P(A)$: *Model-based*

$$\hat{P}(A) = \frac{num(a)}{N}$$

$$\mathbb{E}[A] \approx \sum_a \hat{P}(A)$$

Because, eventually the correct model is learnt

Unknown $P(A)$: *Model-free*

$$\mathbb{E}[A] \approx \frac{1}{N} \sum_i a_i$$

Because, samples appear with right frequencies

Model-based vs Model-free

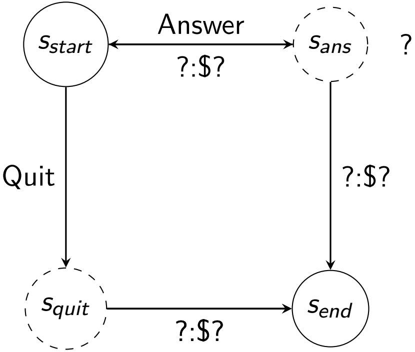
Model based vs. Model free:
Do we estimate $T(s, a, s')$ and $R(s, a, s')$, or just learn values/policy directly

Online vs Batch:
Learn while exploring the world, or learn from fixed batch of data

Active vs Passive:
Does the learner actively choose actions to gather experience? or, is a fixed policy provided?

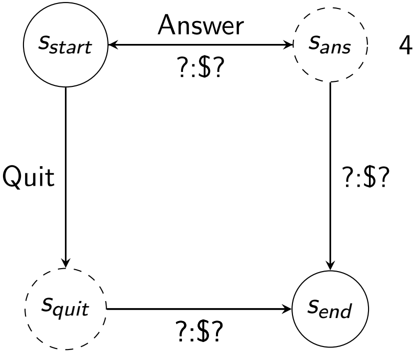
Model-free Monte Carlo

Policy π is Answer
Iteration: 0
Data:



Model-free Monte Carlo

Policy π is Answer
Iteration: 1
Data: $s_{start}; Ans, 4, s_{end}$

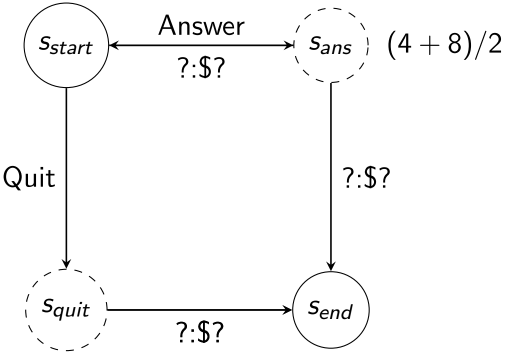


Model-free Value Iteration

Policy π is Answer

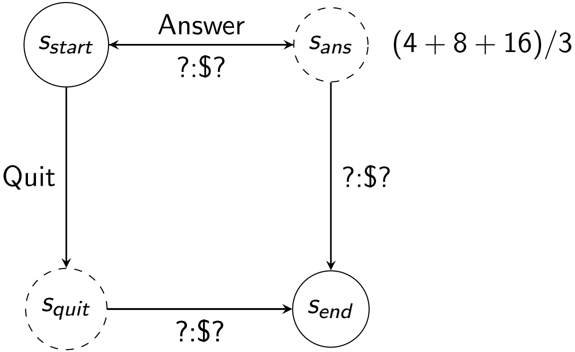
Iteration: 2

Data: $s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end}$



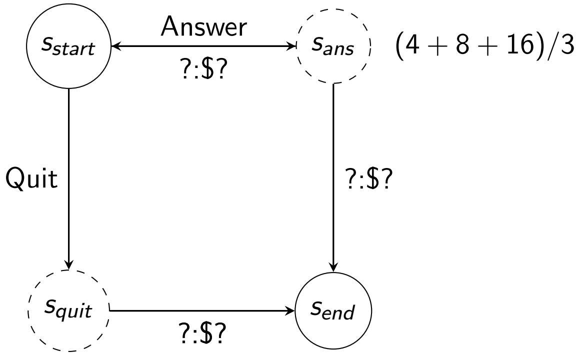
Model-free Value Iteration

Policy π is Answer
Iteration: 3
Data: $s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end}$



Model-free Value Iteration

We are estimating Q_π and not Q_{opt}



Model-free Value Iteration

Policy π is Answer
Data: $s_1; a_1, r_1, s_1; a_2, r_2, s_2; \cdots; a_n, r_n, s_n$

$$\hat{Q}(s, a) = \text{average of } u_t \text{ where } s_{t-1} = s, a_t = a$$

Equivalent formulation (convex combination)

$$\begin{aligned}
&\text{for each } (s, a, u) \\
&\qquad \eta = \frac{1}{1 + \text{No. of updates } (s, a)} \\
&\qquad \hat{Q}_\pi(s, a) \leftarrow (1 - \eta)\hat{Q}_\pi(s, a) + \eta u
\end{aligned}$$

Model-free Value Iteration

Convex combination:

$$\text{for each } (s, a, u)$$
$$\hat{Q}_\pi(s, a) \leftarrow (1 - \eta)\hat{Q}_\pi(s, a) + \eta u$$

Stochastic Gradient:

$$\text{for each } (s, a, u)$$
$$\hat{Q}_\pi(s, a) \leftarrow \hat{Q}_\pi(s, a) - \eta[\underbrace{\hat{Q}_\pi(s, a)}_{\textit{prediction}} - \underbrace{u}_{\textit{target}}]$$

Objective (Least squares): $(\hat{Q}_\pi(s, a) - u)^2$

Using the Utility

Policy π is Answer

Data:

$$s_{start}; Ans, 4, s_{end} \qquad u = 4$$

$$s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end} \qquad u = 8$$

$$s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end} \qquad u = 12$$

$$s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end} \qquad u = 16$$

Model-free Monte Carlo:

$$\text{for each } (s, a, u)$$

$$\hat{Q}_\pi(s, a) \leftarrow (1 - \eta)\hat{Q}_\pi(s, a) + \eta \underbrace{u}_{data}]$$

Using the reward+Q-value

Current estimate: $Q_\pi(s, Ans) = 11$
Data:

$s_{start}; Ans, 4, s_{end}$	$4 + 0$
$s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end}$	$4 + 11$
$s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end}$	$4 + 11$
$s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end}$	$4 + 11$

SARSA:

$$\text{for each } (s, a, r, s', a')$$
$$\hat{Q}_\pi(s, a) \leftarrow (1 - \eta)\hat{Q}_\pi(s, a) + \eta[\underbrace{r}_{data} + \gamma \underbrace{\hat{Q}_\pi(s', a')}_{estimate}]$$

Model-free Monte Carlo vs SARSA

Model-free Monte Carlo:

$$\text{for each } (s, a, u)$$
$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta \underbrace{u}_{data}]$$

SARSA:

$$\text{for each } (s, a, r, s', a')$$
$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta[\underbrace{r}_{data} + \gamma \underbrace{\hat{Q}_\pi(s',a')}_{estimate}]$$

SARSA uses $\hat{Q}_\pi(s,a)$ instead of raw data u
SARSA doesn't have to wait till it reaches the terminal node to update

Model-free Monte Carlo vs SARSA

Output	**MDP**	**Reinforcement Learning**
Q_π	Policy Evaluation	Model-free Monte Carlo, SARSA
Q_{opt}	Value Iteration	*Q-Learning*

Q-Learning

Bellman optimality equation:

$$Q_{opt}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_{opt}(s')]$$

Q-Learning:

for each (s,a,r,s')

$$\hat{Q}_{opt}(s,a) \leftarrow (1-\eta)\underbrace{\hat{Q}_{opt}(s,a)}_{prediction} + \eta\underbrace{(r + \gamma V_{opt}(s'))}_{target}$$

Q-Learning

Recall (Bellman optimality equation):

$$Q_{opt}(s, a) = \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V_{opt}(s')]$$

Q-Learning:

for each (s, a, r, s')

$$\hat{Q}_{opt}(s, a) \leftarrow (1 - \eta) \underbrace{\hat{Q}_{opt}(s, a)}_{prediction} + \eta(\underbrace{r + \gamma V_{opt}(s')}_{target})$$

$$\hat{V}_{opt}(s') = \max_{a' \in Actions(s')} \hat{Q}_{opt}(s', a')$$

SARSA vs Q-Learning

SARSA:

$$\text{for each } (s, a, r, s', a')$$
$$\hat{Q}_\pi(s, a) \leftarrow (1 - \eta)\hat{Q}_\pi(s, a) + \eta\big[r + \gamma \hat{Q}_\pi(s', a')\big]$$

Q-Learning:

$$\text{for each } (s, a, r, s')$$
$$\hat{Q}_{opt}(s, a) \leftarrow (1 - \eta)\hat{Q}_{opt}(s, a) + \eta\big(r + \gamma \max_{a' \in Actions(s')} \hat{Q}_{opt}(s', a')\big)$$

Reinforcement Learning

On-policy: evaluate or improve the data-generating policy
Off-policy: evaluate or learn using data from another policy

	On-Policy	**Off-Policy**
Policy Evaluation (Q_π)	Monte-Carlo, SARSA	
Policy Optimization (Q_{opt})		Q-Learning

Reinforcement Learning

Algorithm	**Estimating**	**Based On**
Model-Based Monte Carlo	$\hat{T}, \hat{R}$	$s_0, a_1, r_1, s_1, \cdots$
Model-Free Monte Carlo	$\hat{Q}_\pi$	u
SARSA	$\hat{Q}_\pi$	$r + \hat{Q}_\pi$
Q-Learning	$\hat{Q}_{opt}$	$r + \hat{Q}_{opt}$

Reinforcement Learning

Overall algorithm

```
for t = 1, 2, 3, ···
    Choose action a_t = π_act(s_{t-1})
    Get reward r_t and new state s_t
    Update parameters
```

Reinforcement Learning

Overall algorithm

```
for t = 1, 2, 3, ...
    Choose action a_t = π_act(s_{t-1}) (how?)
    Get reward r_t and new state s_t
    Update parameters (how?)
```

$s_0; a_1, r_1, s_1; a_2, r_2, s_2; a_3, r_3, s_3, \cdots; a_n, r_n, s_n$

What policy π_{act} should be used?

Choosing the policy

Option1: Select the best policy
$\pi_{act}(s) = \arg\max_{a \in Actions(s)} \hat{Q}_{\pi}(s, a)$
Problem: $\hat{Q}_{\pi}(s, a)$ estimates are inaccurate. Too greedy

Option2: Select a random policy
$\pi_{act}(s) = $ random from $Actions(s)$
Problem: Exploration is not guided

Epsilon-Greedy Policy

$$\pi_{act}(s) = \begin{cases} \arg\max_{a \in Actions(s)} \hat{Q}_\pi(s, a) & \text{probability } 1 - \epsilon \\ \text{random from } Actions(s) & \text{probability } \epsilon \end{cases}$$

A balance between the two!

Function Approximation

Stochastic Gradient update:

$$\hat{Q}_{opt}(s,a) \leftarrow (1-\eta)\hat{Q}_{opt}(s,a) + \eta\big[\underbrace{\hat{Q}_{opt}(s,a)}_{prediction} - \underbrace{(r + \gamma \hat{V}_{opt}(s',a'))}_{target}\big]$$

How to generalize to unseen states/actions

Function Approximation

Linear Regression:

Use features $\phi(s, a)$ and weights $\mathbf{w}$

$$\hat{Q}_{opt}(s, a; \mathbf{w}) = \mathbf{w} \cdot \phi(s, a)$$

Grid World:

$$\phi_1(s, a) = 1[a = Up]$$
$$\phi_2(s, a) = 1[a = Left]$$
$$\ldots$$

$$\phi_7(s, a) = 1[s = (1, *)]$$
$$\phi_8(s, a) = 1[s = (*, 2)]$$
$$\ldots$$

Function Approximation

Q-Learning with Function Approximation:

$$\text{for each } (s, a, r, s') :$$
$$\mathbf{w} \leftarrow \mathbf{w} - \eta \Big[\underbrace{\hat{Q}_{opt}(s, a; \mathbf{w})}_{prediction} - \underbrace{(r + \gamma \hat{V}_{opt}(s'))}_{target} \Big] \phi(s, a)$$

Objective Function:

$$\Big(\underbrace{\hat{Q}_{opt}(s, a; \mathbf{w})}_{prediction} - \underbrace{(r + \gamma \hat{V}_{opt}(s'))}_{target} \Big)^2$$

Recap

Deterministic vs Stochastic

Markov Decision Process

- Transition
- Reward
- Policy
- Discount

Policy value & Q-value

Solving MDPs

- Policy Iteration
- Value Iteration

Recap

Reinforcement Learning
Model-based Monte Carlo Learning
Model-free Monte Carlo Learning
SARSA
Q-Learning
Epsilon-Greedy
Function Approximation

References

Stuart Russell and Xiaodong Song (2021)
CS 188 — Introduction to Artificial Intelligence
University of California, Berkeley

Chelsea Finn and Nima Anari (2021)
CS221 — Artificial Intelligence: Principles and Techniques
Stanford University

The End