9 - MCTS & Reward Shaping & Q-Function Approximation

知识点 & 题目

MCTS

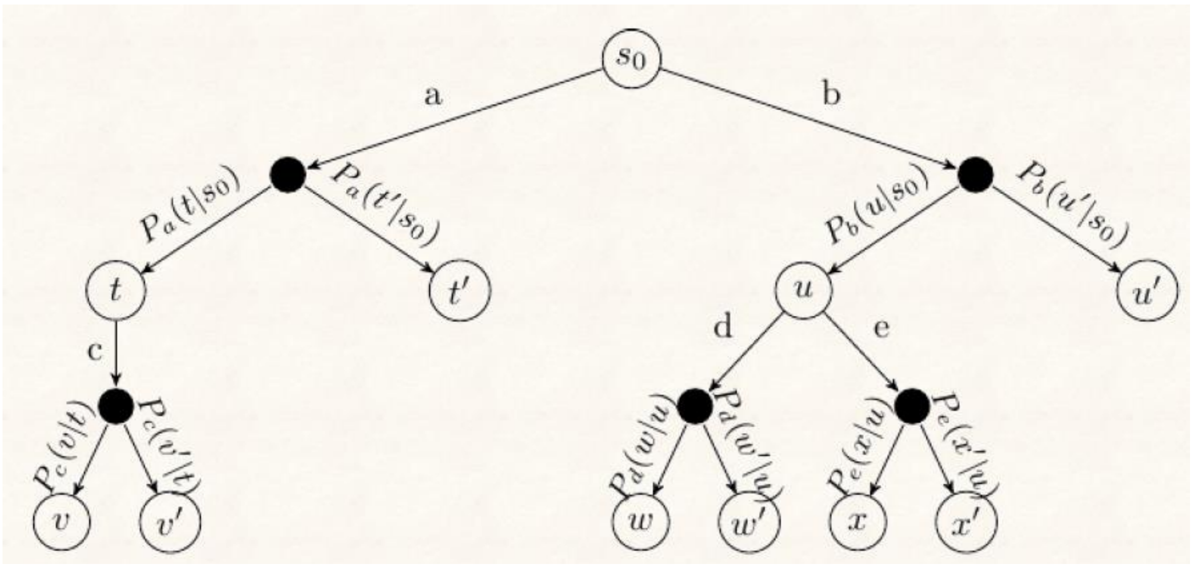
Value iteration and TD learning work will in some situations, however

- Value iteration learns a policy, but it requires a model; and also learns a policy for every state → lots of states, X
- TD-learning learns a policy, but really only works well for repetitive tasks OR requires extensive training (e.g. up to weeks) → cannot react to things that were not explored a lot!

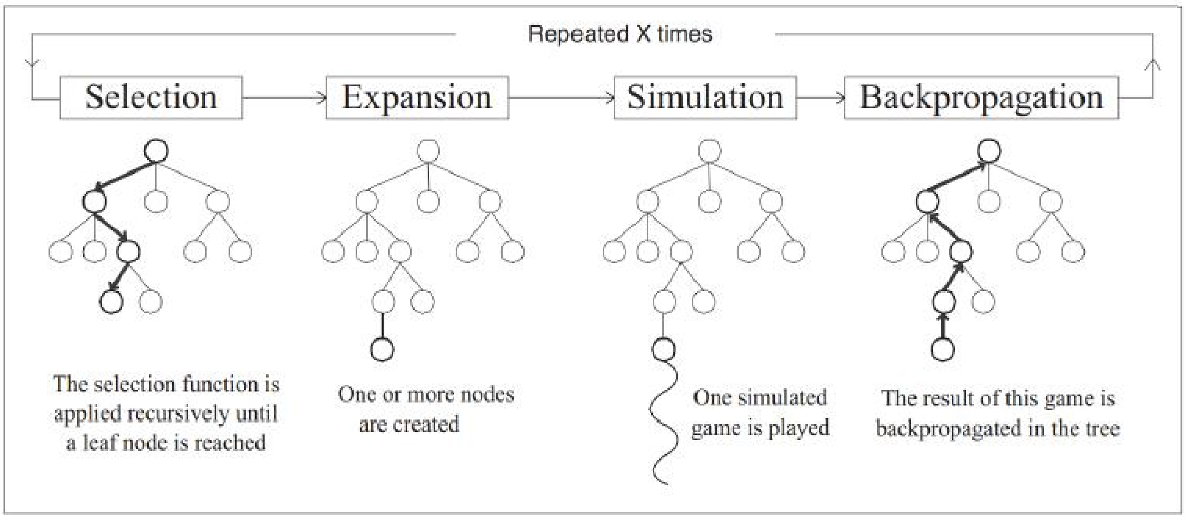
They are both **offline methods**: policies must be trained in advance.

MCTS: Similar to Monte-Carlo reinforcement learning (based on simulation), but uses a tree to prioritise more favourable moves

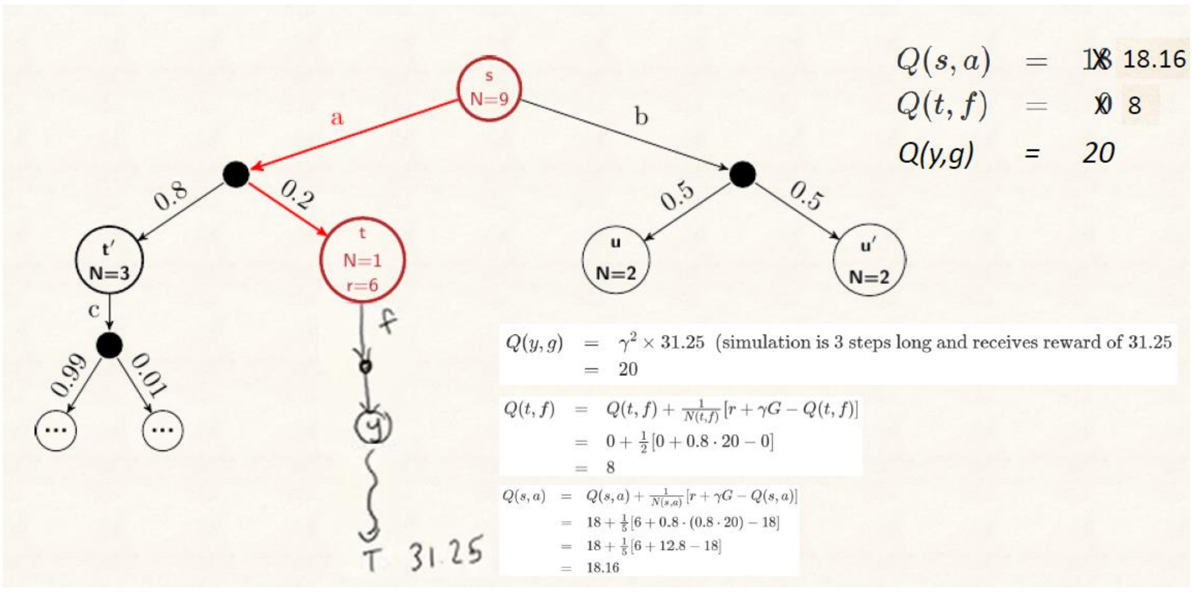
MCTS Foundation: MDPs as Expecti-Max Trees



MCTS Overview


Repeated X times
Selection
Expansion
Simulation
Backpropagation
The selection function is applied recursively until a leaf node is reached
One or more nodes are created
One simulated game is played
The result of this game is backpropagated in the tree

Example



$$\begin{aligned} Q(s,a) &= \cancel{18}\ 18.16 \\ Q(t,f) &= \cancel{0}\ 8 \\ Q(y,g) &= 20 \end{aligned}$$

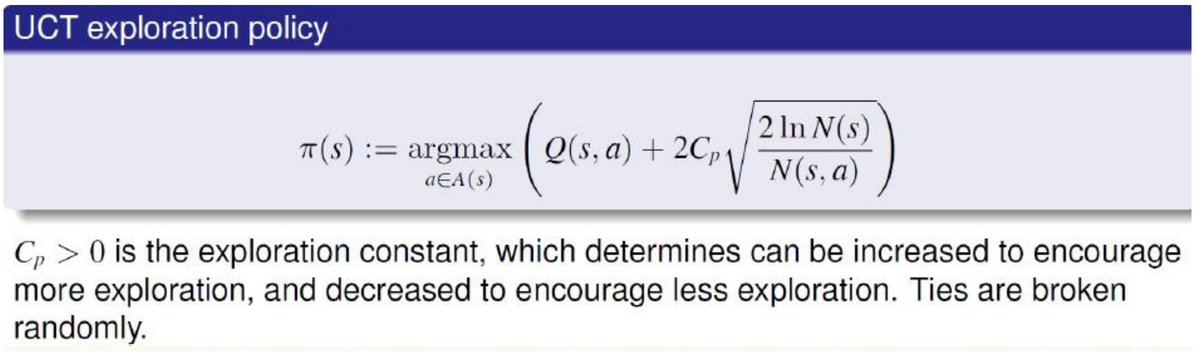
$$\begin{aligned} Q(y,g) &= \gamma^2 \times 31.25 \ \text{(simulation is 3 steps long and receives reward of 31.25} \\ &= 20 \end{aligned}$$

$$\begin{aligned} Q(t,f) &= Q(t,f) + \tfrac{1}{N(t,f)}[r + \gamma G - Q(t,f)] \\ &= 0 + \tfrac{1}{2}[0 + 0.8 \cdot 20 - 0] \\ &= 8 \end{aligned}$$

$$\begin{aligned} Q(s,a) &= Q(s,a) + \tfrac{1}{N(s,a)}[r + \gamma G - Q(s,a)] \\ &= 18 + \tfrac{1}{5}[6 + 0.8 \cdot (0.8 \cdot 20) - 18] \\ &= 18 + \tfrac{1}{5}[6 + 12.8 - 18] \\ &= 18.16 \end{aligned}$$

Upper Confidence Trees: effective as the multi-armed bandit

UCT = MCTS + UCB1



UCT exploration policy

$$\pi(s) := \operatorname*{argmax}_{a \in A(s)} \left(Q(s,a) + 2C_p \sqrt{\frac{2 \ln N(s)}{N(s,a)}} \right)$$

$C_p > 0$ is the exploration constant, which determines can be increased to encourage more exploration, and decreased to encourage less exploration. Ties are broken randomly.

MCTS Large Example L9.1 P9

MCTS is particularly effective in massive state/search spaces when resources and time are not available.

Reward Shaping

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \underbrace{F(s,s')}_{\text{additional reward}} + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

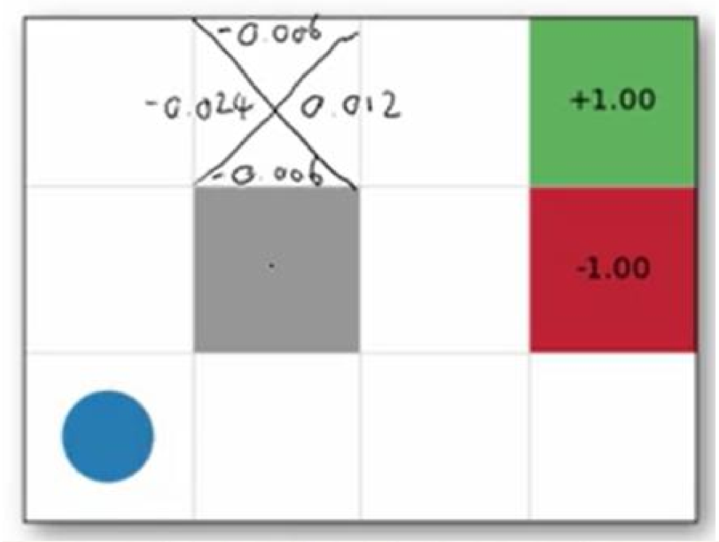
$F : S \times S \to \mathbb{R}$

$F : S \times S \to \mathbb{R}$

$$F(s,s') = \gamma\Phi(s') - \Phi(s)$$

Example

Assume $\gamma = 0.9$ and $\alpha = 0.1$

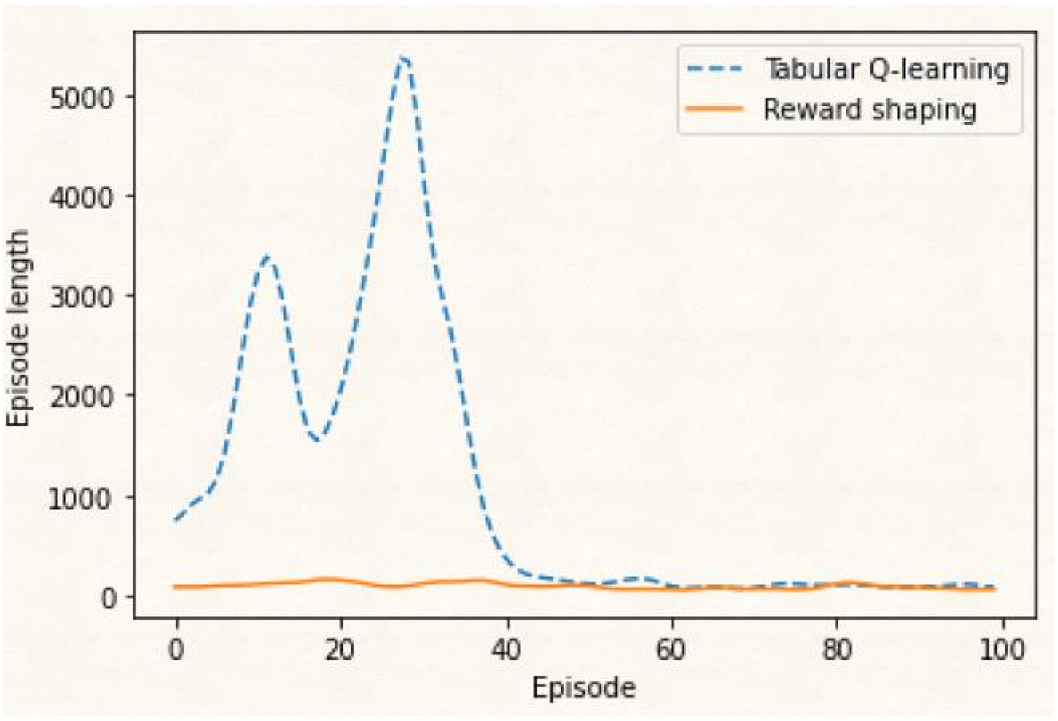


$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \underbrace{F(s,s')}_{\text{additional reward}} + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

$$\Phi(s) = 1 - \frac{|x(g) - x(s)| + |y(g) - y(s)|}{width + height - 2}$$

Q((1,2), Right) =

$$\begin{aligned} F((1,2),(2,2)) &= \gamma\Phi(2,2) - \Phi(1,2) \\ &= 0.9 \cdot (1 - \tfrac{1}{5}) - (1 - \tfrac{2}{5}) \\ &= 0.12 \end{aligned}$$



Q-Function Initialisation

Q(s, a) = F(s, s’)

$$
\begin{aligned}
Q((1,2),Up) &= 0.9(1-\tfrac{2}{5})-(1-\tfrac{2}{5}) &= -0.06\\
Q((1,2),Down) &= 0.9(1-\tfrac{2}{5})-(1-\tfrac{2}{5}) &= -0.06\\
Q((1,2),Right) &= 0.9(1-\tfrac{1}{5})-(1-\tfrac{2}{5}) &= 0.12\\
Q((1,2),Left) &= 0.9(1-\tfrac{3}{5})-(1-\tfrac{2}{5}) &= -0.24
\end{aligned}
$$

$$\Phi(s) = V_0(s)$$

- Small, ‘fake’ rewards can frequently be easily defined using domain knowledge.
- If defined as a potential functions, guaranteed to converge.
- A well defined potential function can significantly reduce the length of episodes early in learning.
- Q-function initialisation - similar technique, shown to have equivalent outcomes if potential function is static.

Q-Function Approximation

Using Q-function approximation, we can scale reinforcement learning by approximating Q functions, rather than storing complete Q tables.

Linear Q-Function Approximation

Use simple linear methods in which we select features and learn weights are effective and guarantee convergence.

Representation:

$$f(s,a) = \begin{pmatrix} f_1(s,a) \\ f_2(s,a) \\ \cdots \\ f_{n\times|A|}(s,a) \end{pmatrix}$$

$$\begin{aligned} Q(s,a) &= f_1(s,a)\cdot w_1^a + f_2(s,a)\cdot w_2^a + \ldots + f_n(s,a)\cdot w_n^a \\ &= \textstyle\sum_{i=0}^{n} f_i(s,a) w_i^a \end{aligned}$$

Freeway example

$$Q(s,Up) = f_1(s,Up)\cdot 0.31 + \ldots + f_6(s,Up)\cdot 0.04$$

Update:

For each state-action feature i

$$w_i^a \leftarrow w_i^a + \alpha\cdot\delta\cdot\ f_i(s,a)$$

- w: weight

Example: Freeway L9.3 P5

Defining state-action features

$$f_{i,k}(s,a) = \begin{cases} f_i(s) & \text{if } a = a_k \\ 0 & \text{otherwise} \quad 1\le i\le n, 1\le k\le |A| \end{cases}$$

$$f(s,a_1) = \begin{pmatrix} f_{1,a_1}(s,a) \\ f_{2,a_1}(s,a) \\ 0 \\ 0 \\ 0 \\ 0 \\ \cdots \end{pmatrix} \quad f(s,a_2) = \begin{pmatrix} 0 \\ 0 \\ f_{1,a_2}(s,a) \\ f_{2,a_2}(s,a) \\ 0 \\ 0 \\ \cdots \end{pmatrix} \quad f(s,a_3) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ f_{1,a_3}(s,a) \\ f_{2,a_3}(s,a) \\ \cdots \end{pmatrix} \quad \cdots$$

Deep Q-Learning

Offers alternatives in which we do not need to select features, making it more suitable for unstructured data. But requires more training data (more episodes) and has no convergence guarantees.

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

$$\theta \leftarrow \theta + \alpha \cdot \delta \cdot \nabla_\theta Q(s, a; \theta)$$

- theta - > gradient

Strengths ✔ & Limitations ❌

Q-function approximation

- Compact representation ✔
- Q-value propagation ✔
- They are approximations ❌

Linear

- Convergence ✔

Deep

- Minimal feature engineering ✔
- Unstructured input ✔
- Data hungry (high sampling complexity) ❌

题目

Quiz

Question 1

0 / 1 pts

Interleaved action selection (planning) and action execution is known as what?

- [✓ Offline planning]
- Internet planning
- Online planning
- MCTS

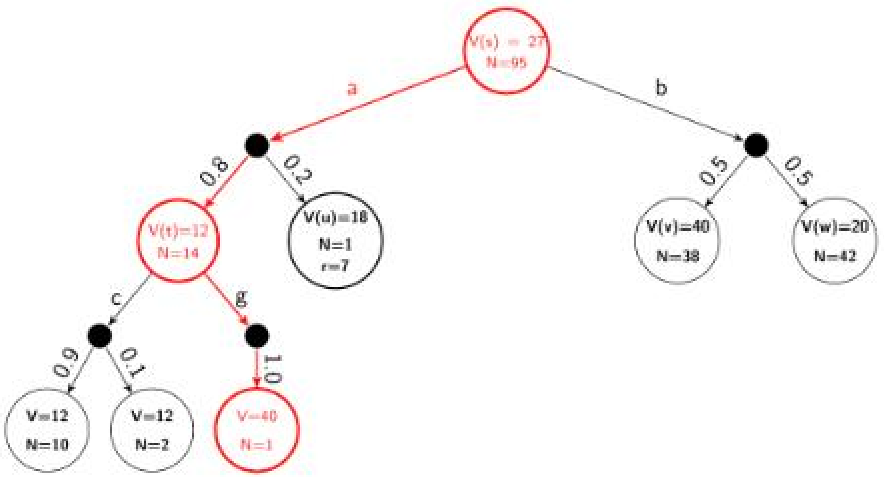
Question 2

2 / 2 pts

The four steps in each iteration of MCTS are:

1. select
2. expand
3. simulate
4. backpropagate

The following three questions refer to the expectimax tree below:



Assume an MCTS algorithm that has just completed the steps of selection (the red path), expansion of node "t", generating with the action "g", and simulated from the new node, resulting in a value of 40 for the new node.

Perform the backpropagation step to calculate the new values for V(t) and (Vs).

Question 3

1 / 1 pts

Assuming $\gamma = 0.9$, what is the new value of V(t)?

rrect!

36

t Answers

36 (with margin: 0)

$$
\begin{aligned}
V(t) &= \max_{a' \in \{c,g\}} \sum_{t' \in children(t)} P'_a(t' \mid t)[r(t,a',t') + \gamma V(st)] \\
&= \max(0.9(0 + 0.9 \times 12) + 0.1(0 + 0.9 \times 12), \qquad \text{(action c)} \\
&\qquad 0.1(0 + 0.9 \times 40)) \qquad \text{(action g)} \\
&= \max(10.8 + 36) \\
&= 36
\end{aligned}
$$

Question 4

0 / 1 pts

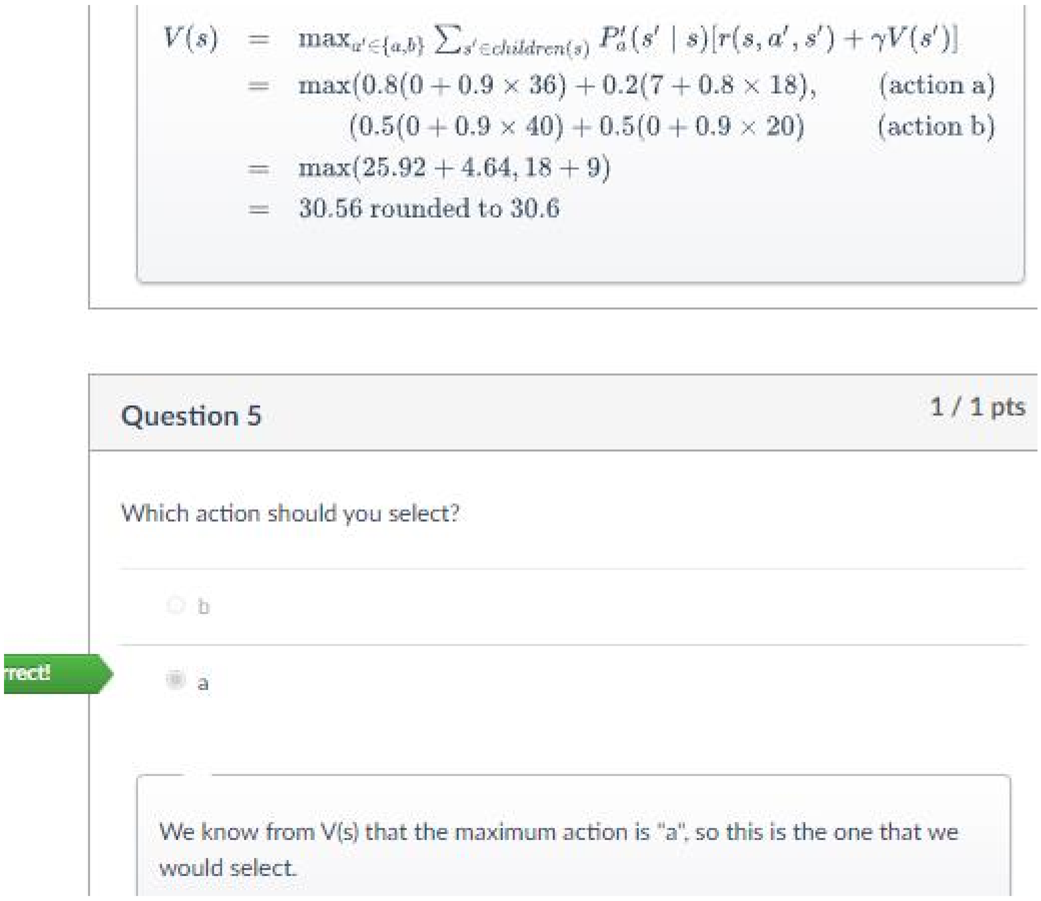
Assuming $\gamma = 0.9$, what is the new value of V(s) (to one decimal place)?

nswered

29.2

t Answers

30.6 (with margin: 0.1)



$$\begin{aligned} V(s) &= \max_{a' \in \{a,b\}} \sum_{s' \in children(s)} P'_a(s' \mid s)[r(s,a',s') + \gamma V(s')] \\ &= \max(0.8(0 + 0.9 \times 36) + 0.2(7 + 0.8 \times 18), \qquad \text{(action a)} \\ &\qquad (0.5(0 + 0.9 \times 40) + 0.5(0 + 0.9 \times 20) \qquad \text{(action b)} \\ &= \max(25.92 + 4.64, 18 + 9) \\ &= 30.56 \text{ rounded to } 30.6 \end{aligned}$$

Question 5

1 / 1 pts

Which action should you select?

- b
- [✓ a]

rrect!

We know from V(s) that the maximum action is "a", so this is the one that we would select.

Question 6

1 / 1 pts

Consider an MDP with four actions (A, B, C, D) and three variables:

x : boolean
y : boolean
z : [1, 10]

How many cells would we have in a Q-table?

- 3
- [✓ 160]
- 120
- 32
- 12
- 4

For variable x and y, each has two possible values, while for variable z, it has 10 possible values. Therefore, the number of states is 2 x 2 x 10=40.

The number of actions is four.

Therefore, the number of cells in the Q-table is 160.

Question 7

1 / 1 pts

Consider again the above MDP with four actions A, B, C, and D, in which we select two features to represent the state.

How many elements will be in our weight vector w?

- 2: One for each feature
- 4: One for each action
- 6: One for each action plus one for each state
- [✓ 8: One for each state-action pair]
- 0: Weights are independent of features.

Our feature vector has an entry for every state-action pair. If there are two features and four actions, the total number of elements of the state-action feature vector is 8.

Each element in the state-action feature vector has a weight, so the answer is 8.

Question 8

2 / 2 pts

Consider the Freeway example from lectures. If we have just four features:

1) r: row number relative to the final row, which is normalised to [0,1]
2) dc: distance to the nearest car in the current row, which is normalised to [0,1]
3) da: distance to the nearest car in the row *above*, which is normalised to [0,1]
4) db: distance to the nearest car in the row *below*, which is normalised to [0,1]

Consider we have just two actions: Up and Down.

If we have the weight vector w = (0.4, 0.3, 0.2, 0.01,   0.2, 0.2, 0.01, 0.2), in which the first four elements are for action Up, and the rest for action Down.

Given state s that is row 4, one car in each row that are 2 columns away, what will *f(s, Down)* return if we assume the order of actions in the element is *Up* and then *Down*?

(0.4, 0.4, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2)

(0.4, 0.2, 0.2, 0.2, 0.4, 0.2, 0.2, 0.2)

(0.4, 0.2, 0.2, 0.2)

ect! [✓ (0, 0, 0, 0, 0.4, 0.2, 0.2, 0.2)]

(0.4, 0.2, 0.2, 0.2, 0, 0, 0, 0)

The vector will have 8 elements: one for each state-action pair. The entire vector is returned.

Down is the 2nd set of actions in the vector, and all other parts of the vector (which are the values for action Down), are set to zero because they are not relevant to the Down action; so the answer is (0, 0, 0, 0, 0.4, 0.2, 0.2, 0.2)

The vector is ordered by action, so the answer starting with (0.4, 0.4, 0.2, ...) is not correct this interleaves the values for the action.

Question 9

2 / 2 pts

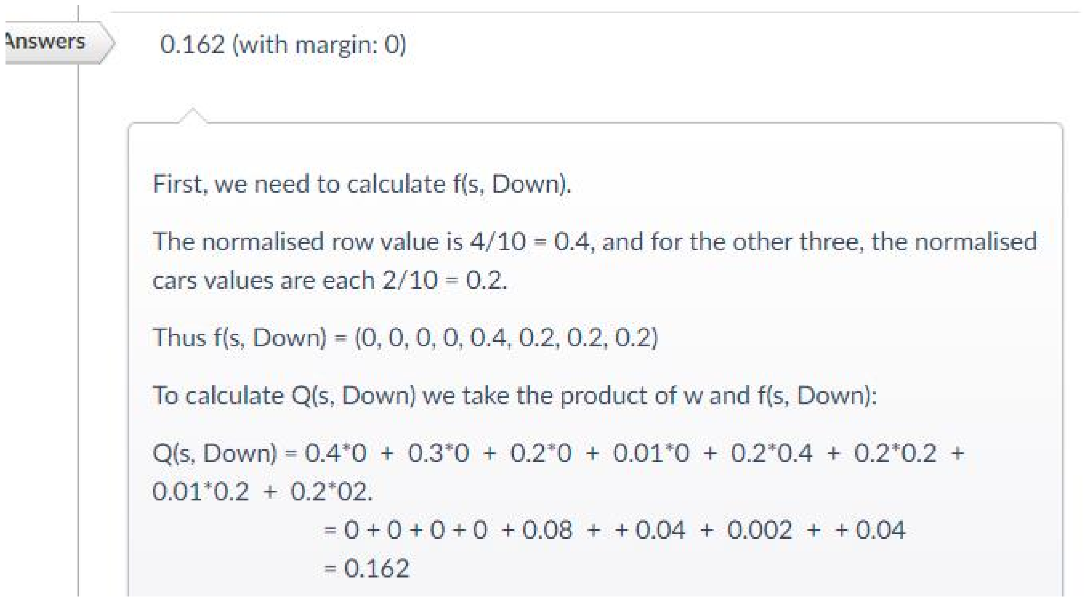
Consider the above example of Freeway with four features.

If we have the weight vector w = (0.4, 0.3, 0.2, 0.01,   0.2, 0.2, 0.01, 0.2), in which the first four elements are for action Up, and the rest for action Down, and we are in the state s that is row 4, one car in each row that are 2 columns away, what is Q(s, Down)?

Assume that rows and columns are both min-max normalised 10 rows and 10 columns; so if we are in row 6, the feature value would be 6/10 = 0.6.

ect!

0.162



Answers 0.162 (with margin: 0)

First, we need to calculate f(s, Down).

The normalised row value is 4/10 = 0.4, and for the other three, the normalised cars values are each 2/10 = 0.2.

Thus f(s, Down) = (0, 0, 0, 0, 0.4, 0.2, 0.2, 0.2)

To calculate Q(s, Down) we take the product of w and f(s, Down):

Q(s, Down) = 0.4*0 + 0.3*0 + 0.2*0 + 0.01*0 + 0.2*0.4 + 0.2*0.2 + 0.01*0.2 + 0.2*02.
= 0 + 0 + 0 + 0 + 0.08 + + 0.04 + 0.002 + + 0.04
= 0.162

Question 10

4 / 4 pts

Consider the above example of Freeway with four features.

If we have the weight vector w = (0.4, 0.3, 0.2, 0.01,   0.2, 0.2, 0.01, 0.2), in which the first four elements are for action Up, and the rest for action Down, and we are in the state s that is row 6, one car in each row that are 2 columns away.

Assume that rows and columns are both min-max normalised 10 rows and 10 columns; so if we are in row 6, the feature value would be 6/10 = 0.6, and that Q(s, Down) = 0.162.

If $\alpha = 0.4$ and $\gamma = 0.9$, and the action Down is executed (going to row 5), and receives a reward of -1. What is the new weight vector using a Q-learning update, assuming that max_a' Q(s',a') = 0.162? Round to three decimal places.

- w = (-0.005, 0.132, -0.058, 0.132, 0.4, 0.3, 0.2, 0.01)
- w = (-0.005, 0.132, -0.058, 0.132)
- w = (0.4, 0.3, 0.2, 0.01, 0.005, 0.132, 0.058, 0.132)
- Correct! [✓ w = (0.4, 0.3, 0.2, 0.01, -0.005, 0.132, -0.058, 0.132)]

The update rule is:

$$w_i^{Down} \leftarrow w_i^{Down} + \alpha[r + \gamma \max_a Q(s', a') - Q(s,a)] f_i(s,a)$$

We only need to update weights for the $Down$ action. For the row feature, this is:

$$\begin{aligned} w_r^{Down} &\leftarrow 0.2 + 0.4[-1 + 0.9 \times 0.162]0.6 \\ &= 0.2 + 0.4[-0.8542]0.6 \\ &= 0.2 - 0.205 \\ &= 0.005 \end{aligned}$$

The 0.6 is $f_{row}(s, Down)$ normalised. Remember that the first four elements in $f(s,a)$ are for the Up action!

The term inside the square brackets is the same for other weights, so we can calculate these as:

$$\begin{aligned} w_{dc}^{Down} &\leftarrow 0.2 + 0.4[-0.8542]0.2 &= 0.132 \\ w_{da}^{Down} &\leftarrow 0.01 + 0.4[-0.8542]0.2 &= -0.058 \\ w_{db}^{Down} &\leftarrow 0.2 + 0.4[-0.8542]0.2 &= 0.132 \end{aligned}$$

The weights for the Up do not change, so the next vector is $w = (0.4, 0.3, 0.2, 0.01, 0.005, 0.132, -0.058, 0.132)$

Question 11

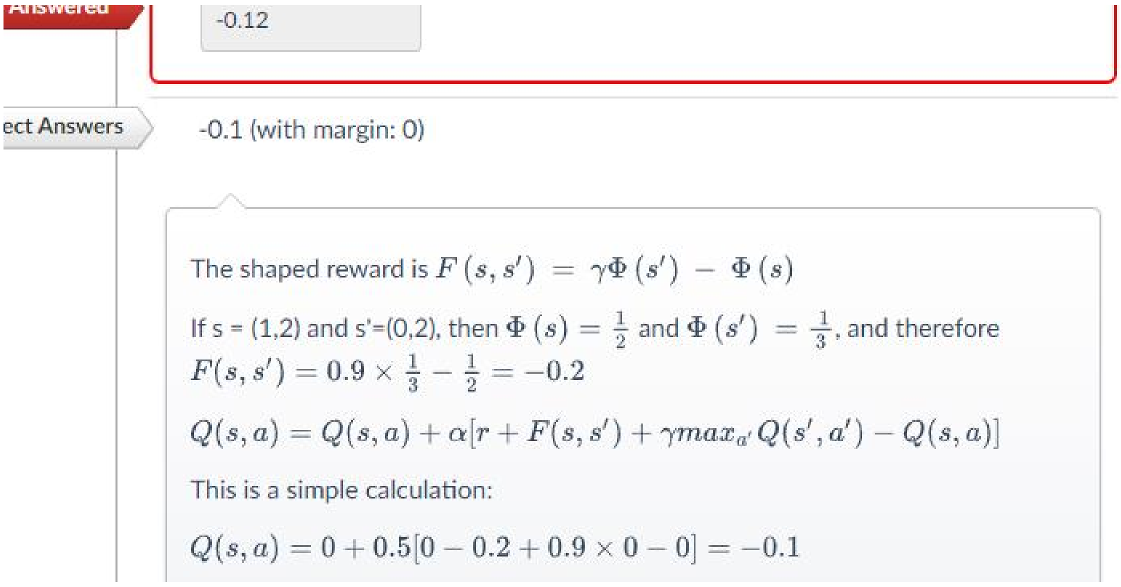
0 / 1 pts

Consider the GridWorld example from the notes.

Using the inverse Manhattan distance as a potential reward function, calculate Q(s, West) for state s = (1,2) and state s' = (0,2), receiving no immediate reward.

Assume $\alpha = 0.5$ and $\gamma = 0.9$ and Q(s,a)=0 for all states and actions.

Answered



Answered

-0.12

ect Answers

-0.1 (with margin: 0)

The shaped reward is $F\left(s,s'\right) \;=\; \gamma\Phi\left(s'\right) \;-\; \Phi\left(s\right)$

If s = (1,2) and s'=(0,2), then $\Phi\left(s\right) = \frac{1}{2}$ and $\Phi\left(s'\right) \;=\; \frac{1}{3}$, and therefore $F(s,s') = 0.9 \times \frac{1}{3} - \frac{1}{2} = -0.2$

$$Q(s,a) = Q(s,a) + \alpha[r + F(s,s') + \gamma max_{a'} Q(s',a') - Q(s,a)]$$

This is a simple calculation:

$$Q(s,a) = 0 + 0.5[0 - 0.2 + 0.9 \times 0 - 0] = -0.1$$