CS 188
Summer 2020 MT1 Review

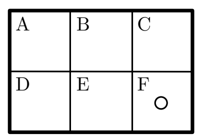
MDPs Solutions

Q1. MDP

Pacman is using MDPs to maximize his expected utility. In each environment:

- Pacman has the standard actions {North, East, South, West} unless blocked by an outer wall
- There is a reward of 1 point when eating the dot (for example, in the grid below, $R(C, South, F) = 1$)
- The game ends when the dot is eaten

(a) Consider the following grid where there is a single food pellet in the bottom right corner (F). The **discount** factor is 0.5. There is no living reward. The states are simply the grid locations.



(i) What is the optimal policy for each state?

State	$\pi(state)$
A	East or South
B	East or South
C	South
D	East
E	East

(ii) What is the optimal value for the state of being in the upper left corner (A)? Reminder: the discount factor is 0.5.

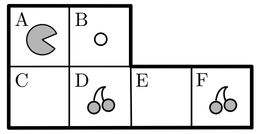
$V^*(A) =$ 0.25

k	V(A)	V(B)	V(C)	V(D)	V(E)	V(F)
0	0	0	0	0	0	0
1	0	0	1	0	1	0
2	0	0.5	1	0.5	1	0
3	0.25	0.5	1	0.5	1	0
4	0.25	0.5	1	0.5	1	0

(iii) Using value iteration with the value of all states equal to zero at k=0, for which iteration k will $V_k(A) = V^*(A)$?

$k =$ 3 (see above)

(b) Consider a new Pacman level that begins with cherries in locations D and F. Landing on a grid position with cherries is worth 5 points and then the cherries at that position disappear. There is still one dot, worth 1 point. The game still only ends when the dot is eaten.



(i) With no discount ($\gamma = 1$) and a living reward of -1, what is the optimal policy for the states in this level's state space?

State	$\pi(state)$
A	South
C	East
D, F_{Cherry}=true	East
D, F_{Cherry}=false	North
E, F_{Cherry}=true	East
E, F_{Cherry}=false	West
F	West

Larger state spaces with equivalent states and actions are possible too. For example with the state representation of (grid, D-cherry, F-cherry), there could be up to 24 different states, where all four with A are the same, etc.

(ii) With no discount ($\gamma = 1$), what is the range of living reward values such that Pacman eats exactly one cherry when starting at position A?

Valid range for the living reward is (-2.5,-1.25).

Let x equal the living reward.

The reward for eating zero cherries {A,B} is $x + 1$ (one step plus food).

The reward for eating exactly one cherry {A,C,D,B} is $3x + 6$ (three steps plus cherry plus food).

The reward for eating two cherries {A,C,D,E,F,E,D,B} is $7x + 11$ (seven steps plus two cherries plus food).

x must be greater than -2.5 to make eating at least one cherry worth it ($3x + 6 > x + 1$).

x must be less than -1.25 to eat less than one cherry ($3x + 6 > 7x + 11$).

(c) Quick reinforcement learning questions [PLEASE WRITE CLEARLY]:

(i) What is the difference between value-iteration and TD-learning?

Value iteration has explicity models for transitions and rewards, while TD-learning relies on active samples.

(ii) What is the difference between TD-learning and Q-learning?

TD-learning stores and updates V(s) while Q-learning stores and updates Q(s,a). Also, Q-learning is able to learn quality policies despite random or suboptimal actions, while TD-learning values are affected by the actions taken.

(iii) What is the purpose of using a learning rate (α) during Q-learning?

The learning rate allows us to average information from previous iterations with the current sample. It allows us to step towards a solution at an incremental rate. This allows us to incorporate random samples while moving away from poor initial estimates.

(iv) In value iteration, we store the value of each state. What do we store during *approximate* Q-learning?

We update and store the weights associated with the features.

(v) Give one advantage and one disadvantage of using approximate Q-learning rather than standard Q-learning.

Pros: Feature representation scales to very large or infinite spaces; learning process generalizes from seen states to unseen states.

Cons: True Q may not be representable in the chosen form; learning may not converge; need to design feature functions.

Q2. Markov Decision Processes

Consider a simple MDP with two states, S_1 and S_2, two actions, A and B, a discount factor γ of $1/2$, reward function R given by

$$R(s,a,s') = \begin{cases} 1 & \text{if } s' = S_1; \\ -1 & \text{if } s' = S_2; \end{cases}$$

and a transition function specified by the following table.

s	a	s'	$T(s,a,s')$
S_1	A	S_1	$1/2$
S_1	A	S_2	$1/2$
S_1	B	S_1	$2/3$
S_1	B	S_2	$1/3$
S_2	A	S_1	$1/2$
S_2	A	S_2	$1/2$
S_2	B	S_1	$1/3$
S_2	B	S_2	$2/3$

(a) Perform a single iteration of value iteration, filling in the resultant Q-values and state values in the following tables. Use the specified initial value function V_0, rather than starting from all zero state values. Only compute the entries not labeled "skip".

s	a	$Q_1(s,a)$
S_1	A	1.25
S_1	B	1.50
S_2	A	skip
S_2	B	skip

s	$V_0(s)$	$V_1(s)$
S_1	2	1.50
S_2	3	skip

$$Q_1(s) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_0(s')]$$

$$V_1(s) = \max_a Q_1(s,a)$$

(b) Suppose that Q-learning with a learning rate α of $1/2$ is being run, and the following episode is observed.

s_1	a_1	r_1	s_2	a_2	r_2	s_3
S_1	A	1	S_1	A	-1	S_2

Using the initial Q-values Q_0, fill in the following table to indicate the resultant progression of Q-values.

s	a	$Q_0(s,a)$	$Q_1(s,a)$	$Q_2(s,a)$
S_1	A	$-1/2$	$1/4$	$-1/8$
S_1	B	0	(0)	(0)
S_2	A	-1	(-1)	(-1)
S_2	B	1	(1)	(1)

Here is the only update for the first observed tuple (s_1, a_1, r_1, s_2):

$$Q_1(s_1,a_1) = Q_0(s_1,a_1) + \alpha(r_1 + \gamma \max_a Q_0(s_2,a) - Q_0(s_1,a_1)$$

There is another observed tuple, so there is another update to get to Q_2.

(c) Given an arbitrary MDP with state set S, transition function $T(s, a, s')$, discount factor γ, and reward function $R(s, a, s')$, and given a constant $\beta > 0$, consider a modified MDP (S, T, γ, R') with reward function $R'(s, a, s') = \beta \cdot R(s, a, s')$. Prove that the modified MDP (S, T, γ, R') has the same set of optimal policies as the original MDP (S, T, γ, R).

$V^\pi_{\text{modified}} = \beta \cdot V^\pi_{\text{original}}$ satisfies the Bellman equation

$$\begin{aligned}
\beta \cdot V^\pi_{\text{original}}(s) = V^\pi_{\text{modified}}(s) \\
&= \sum_{s'} T(s, \pi(s), s')[R'(s, \pi(s), s') + \gamma \cdot V^\pi_{\text{modified}}(s')] \\
&= \sum_{s'} T(s, \pi(s), s')[\beta \cdot R(s, \pi(s), s') + \gamma \cdot \beta \cdot V^\pi_{\text{original}}(s')] \\
&= \beta \cdot \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma \cdot V^\pi_{\text{original}}(s')] \\
&= \beta \cdot V^\pi_{\text{original}}(s').
\end{aligned}$$

It follows that for any state s, the set of policies π that maximize V^π_{original} is precisely the same set of policies that maximize V^π_{modified}.

Intuitively, you should understand that scaling the reward function does not affect the arg max that ultimately determines the policy.

(d) Although in this class we have defined MDPs as having a reward function $R(s, a, s')$ that can depend on the initial state s and the action a in addition to the destination state s', MDPs are sometimes defined as having a reward function $R(s')$ that depends only on the destination state s'. Given an arbitrary MDP with state set S, transition function $T(s, a, s')$, discount factor γ, and reward function $R(s, a, s')$ that *does depend* on the initial state s and the action a, define an *equivalent* MDP with state set S', transition function $T'(s, a, s')$, discount factor γ', and reward function $R'(s')$ that depends only on the destination state s'.

By *equivalent*, it is meant that there should be a one-to-one mapping between state-action sequences in the original MDP and state-action sequences in the modified MDP (with the same value). **You do not need to give a proof of the equivalence.**

States: $S' = S \times S \times A$, where A is the set of actions.

Transition function:

$$T'(s, a, s') = \begin{cases} T(s''', a, s'''') & \text{if } s = (s'', a', s''') \text{ and } s' = (s''', a, s''''); \\ 0 & \text{otherwise.} \end{cases}$$

Discount factor: $\gamma' = \gamma$

Reward function: $R'(s') = R(s, a, s'')$, where $s' = (s, a, s'')$.