RBE 595 — Reinforcement Learning
Midterm Exam

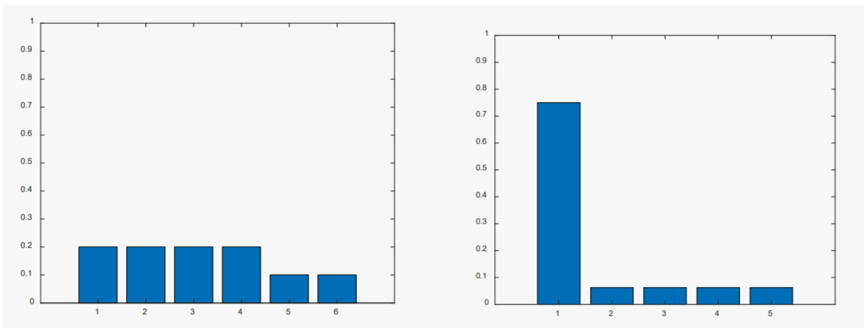
Arjan Gupta

Problem 1

Consider two random variables with distributions below:

$$p = \{0.2, 0.2, 0.2, 0.2, 0.1, 0.1\}$$

$$q = \{0.75, 0.0625, 0.0625, 0.0625, 0.0625\}$$



A. [4 points] Calculate the entropy for each variable.
B. [4 points] Intuitively how can you tell which variable has a higher entropy without calculating the entropy numerically? What does higher entropy mean?

Answer

A. The entropy for each variable is given by:

$$\begin{aligned} H(p) &= -\sum_{i=1}^{6} p_i \log_2 p_i \\ &= -\left(0.2\log_2 0.2 + 0.2\log_2 0.2 + 0.2\log_2 0.2 + 0.2\log_2 0.2 + 0.1\log_2 0.1 + 0.1\log_2 0.1\right) \\ &= -(4 * 0.2\log_2 0.2 + 2 * 0.1\log_2 0.1) \\ &= 2.5219 \end{aligned}$$

$$\begin{aligned} H(q) &= -\sum_{i=1}^{5} q_i \log_2 q_i \\ &= -\left(0.75\log_2 0.75 + 4 * 0.0625\log_2 0.0625\right) \\ &= 1.3112 \end{aligned}$$

B. Entropy is defined as the lack of expected information, or the 'surprise'/uncertainty of a random variable. We can tell that q has a higher entropy than p without calculating the entropy, because the histogram for q shows that there is a 'surprising' value of 0.75, which goes against the trend of the other values (which are all 0.0625). On the other hand, the histogram for p shows that all the values are fairly close to each other. In general, higher entropy means that the random variable has more uncertainty, so the likelihood of encountering a value closer to the expected value is lower.

Problem 2

2- [5 points] Which equation is correct? draw the corresponding backup-diagram and explain.

A. $v_*(s) = \sum_{r,s} \pi(a|s) p(s', r|s, a)[r + \gamma q_*(s')]$

B. $v_*(s) = \max_a q_*(s, a)$

C. $v_*(s) = \max_a \sum_{r,s} p(s', r|s, a)[r + \gamma q_*(s', a)]$

D. $v_*(s) = \max_a \sum_{r,s'} p(s', r|s, a)[r + \gamma v_*(s')]$

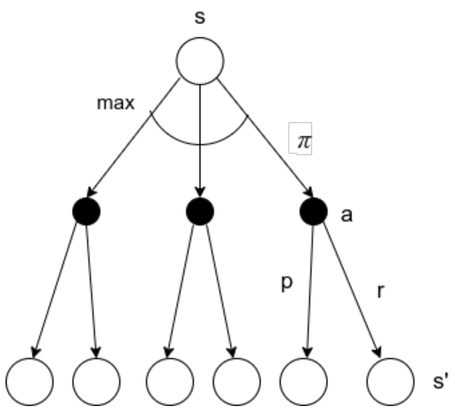
E. None of the above

F. B and D

Answer

Option F (B and D) is correct.

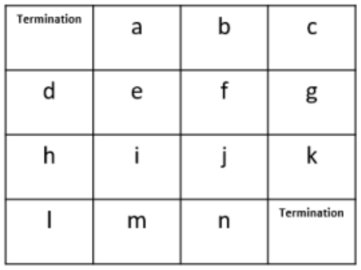
I have drawn the backup diagram for v_* as given below:



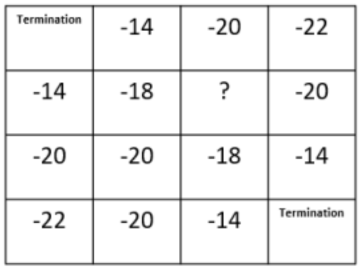
Explanation
v_* is the optimal state-value function. The backup diagram for v_* shows us that, if we start at a state s, we choose the action that maximizes the value of the state-action pair, and then we take the action. The action choice is taken using our current policy, π. Once we take the action, we end up in a new state, s', and we get a reward, r. The resultant state s' as well as the reward r are chosen according to the dynamics of the environment, $p(s', r|s, a)$, which is not something we can control. In summary, the optimal state value function chooses the action that maximizes the value of the state-action value.

Problem 3

Consider a vehicle with 4 actions (left, right, up, down). There's no uncertainty in the outcome of the action (i.e. when left is commanded, the left state is achieved). The actions that cause the vehicle outside the grid, leave the state unchanged. The reward for all transition is -1 except when the goal is reached where the reward is zero. Discount factor $\gamma = 1$.
The figure on left shows the name of the states and figure on the right shows the state-value $V(s)$, for each state under a uniform random policy.



Termination	a	b	c
d	e	f	g
h	i	j	k
l	m	n	Termination



Termination	-14	-20	-22
-14	-18	?	-20
-20	-20	-18	-14
-22	-20	-14	Termination

A. [4 points] What is $q(k, down)$?
B. [4 points] What is $q(g, down)$?
C. [4 points] What is $V(f)$?

Answer

A. We know the equation for $q(s, a)$ is given by,

$$q(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma V(s')\right]$$

We can calculate $q(k, down)$ as follows:

$$\begin{aligned} q(k, down) &= \sum_{s',r} p(s',r|k,down)\left[r + \gamma V(s')\right] \\ &= 1\left[0 + 1 \cdot 0\right] \\ &= 0 \end{aligned}$$

B. We can calculate $q(g, down)$ as follows:

$$\begin{aligned} q(g, down) &= \sum_{s',r} p(s',r|g,down)\left[r + \gamma V(s')\right] \\ &= 1\left[-1 + 1 \cdot -14\right] \\ &= -15 \end{aligned}$$

C. We know the equation for $V(s)$ is given by,

$$V(s) = \sum_{a} \pi(a|s) q(s,a)$$

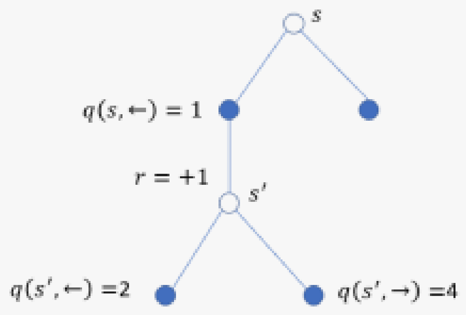
Problem 3 continued on next page. . .

We can calculate $V(f)$ as follows:

$$\begin{aligned}
V(f) &= \sum_{s',r} p(s', r | f, \pi(f)) \left[r + \gamma V(s') \right] \\
&= 0.25 \left[-1 + 1 \cdot -18 \right] + 0.25 \left[-1 + 1 \cdot -20 \right] + 0.25 \left[-1 + 1 \cdot -20 \right] + 0.25 \left[-1 + 1 \cdot -18 \right] \\
&= 0.25 \left[-19 \right] + 0.25 \left[-21 \right] + 0.25 \left[-21 \right] + 0.25 \left[-19 \right] \\
&= 0.25 \left[-80 \right] \\
&= -20
\end{aligned}$$

Problem 4

4- Consider the state-action interaction below. The $q(s,a)$ written next to each action (left and right) is the initial estimate.



consider discount factor $\gamma = 0.5$ and learning rate $\alpha = 0.1$:

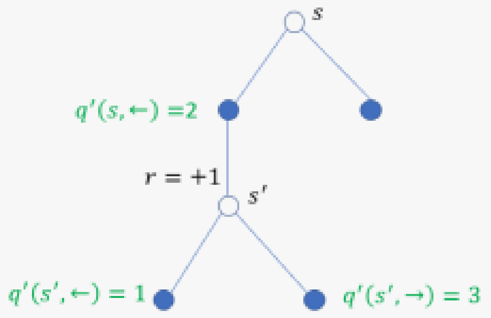
A. [4 points] What is the target value as well as updated value of $q(s,\leftarrow)$ using SARSA algorithm if the action at s' was the left action. What is the target value as well as updated value of $q(s,\leftarrow)$ using SARSA algorithm if the action at s' was the right action.

B. [4 points] Assume that the action at s' has a distribution in such a way that it is 30 percent left action and 70 percent right action. What is the expected SARSA target value and as well expected value of $q(s,\leftarrow)$ under SARSA algorithm?

C. [4 points] What is the target value as well as updated value of $q(s,\leftarrow)$ using Q-learning algorithm.

D. [4 points] Does the distribution of the action at s' have an effect on the Q-learning target value?

Consider now we have another batch of initial estimates for the action-values. We call then $q'(s,a)$. They are shown in the diagram below in green color:



E. [4 points] What is the updated value of $q(s,\leftarrow)$ using double-Q learning algorithm? (You need to use both green and black initial estimates)

F. [4 points] What is the updated value of $q'(s,\leftarrow)$ using double-Q learning algorithm? (You need to use both green and black initial estimates)

Problem 4 continued on next page. . .

Answer

A. The target value for SARSA is given by $R + \gamma Q(S_{t+1}, A_{t+1})$.
The update rule for SARSA is given by $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\,[target - Q(S_t, A_t)]$.

For left action at s':
The target SARSA value at $q(s, left)$ is given by $1 + 0.5 \cdot 2 = 2$
The update value for SARSA is given by $1 + 0.1 \cdot (2 - 1) = 1.1$

For right action at s':
The target SARSA value at $q(s, right)$ is given by $1 + 0.5 \cdot 4 = 3$
The update value for SARSA is given by $1 + 0.1 \cdot (3 - 1) = 1.2$

B. The target value for Expected SARSA is given by $R + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$.
The update rule for Expected SARSA is given by $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\,[target - Q(S_t, A_t)]$.

The policy distribution for s' is given by $\pi(left|s') = 0.3$ and $\pi(right|s') = 0.7$.

The target Expected SARSA value at $q(s, left)$ is given by $1 + 0.5(0.3 \cdot 2 + 0.7 \cdot 4) = 2.7$
The update value for Expected SARSA is given by $1 + 0.1 \cdot (2.7 - 1) = 1.17$

C. The target value for Q-learning is given by $R + \gamma \max_a Q(S_{t+1}, a)$.
The update rule for Q-learning is given by $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\,[target - Q(S_t, A_t)]$.

The target Q-learning value at $q(s, left)$ is given by $1 + 0.5 \cdot 4 = 3$
The update value for Q-learning is given by $1 + 0.1 \cdot (3 - 1) = 1.2$

D. The distribution of the action at s' does not affect the target value for Q-learning, because the target value for Q-learning is given by $R + \gamma \max_a Q(S_{t+1}, a)$, which does not depend on the action distribution at s'.

E. With the new batch of initial estimates, and applying double Q-learning,
our Q_1 target is given by $R + \gamma Q_2(S_{t+1}, \arg\max_a Q_1(S_{t+1}, a))$
and our Q_2 target is given by $R + \gamma Q_1(S_{t+1}, \arg\max_a Q_2(S_{t+1}, a))$

The update rule for Q_1 is given by $Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha\,[targetQ_1 - Q_1(S_t, A_t)]$.
The update rule for Q_2 is given by $Q_2(S_t, A_t) \leftarrow Q_2(S_t, A_t) + \alpha\,[targetQ_2 - Q_2(S_t, A_t)]$.

The target value for $q(s, left)$ is given by $1 + 0.5 \cdot q'(s', right) = 1 + 0.5 \cdot 3 = 2.5$
The update value for $q(s, left)$ is given by $1 + 0.1 \cdot (2.5 - 1) = 1.15$

F.
The target value for $q'(s, left)$ is given by $1 + 0.5 \cdot q(s', right) = 1 + 0.5 \cdot 4 = 3$
The update value for $q'(s, left)$ is given by $2 + 0.1 \cdot (3 - 2) = 2.1$

Problem 5

As we discussed, n-step off-policy return via bootstrapping can be written as:

$$G_{t:h} = \rho_t \left(R_{t+1} + \gamma G_{t+1:h}\right) \tag{1}$$

where $\rho_t = \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$ is the importance sampling ratio between target and behavior policy.
Using control variates, the return can be written as:

$$G_{t:h} = \rho_t \left(R_{t+1} + \gamma G_{t+1:h}\right) + (1-\rho_t)V_{h-1}(S_t) \tag{2}$$

• [8 points] Prove that introducing the control variates in equation (2) does not add any bias to the original return (i.e. equation (1)) in expectation.

Answer

If equation (2) does not add any bias to the original return, then we would be able to state that,

$$\mathbb{E}[\rho_t \left(R_{t+1} + \gamma G_{t+1:h}\right)] = \mathbb{E}[\rho_t \left(R_{t+1} + \gamma G_{t+1:h}\right) + (1-\rho_t)V_{h-1}(S_t)] \tag{3}$$

This is what we have to prove. Let us first consider the right hand side of the equation (3). We can expand it as follows:

$$\begin{aligned}
&\mathbb{E}[\rho_t \left(R_{t+1} + \gamma G_{t+1:h}\right)] + \mathbb{E}[(1-\rho_t)V_{h-1}(S_t)] \\
=&\mathbb{E}[\rho_t \left(R_{t+1} + \gamma G_{t+1:h}\right)] + \mathbb{E}[(V_{h-1}(S_t) - \rho_t V_{h-1}(S_t)] \\
=&\mathbb{E}[\rho_t \left(R_{t+1} + \gamma G_{t+1:h}\right)] + \mathbb{E}[V_{h-1}(S_t)] - \mathbb{E}[\rho_t V_{h-1}(S_t)]
\end{aligned}$$

However, ρ_t is independent of $V_{h-1}(S_t)$ because the ratio of the target policy and the behavior policy is independent of the value function. Therefore, we can further write the right hand side of the equation (3) as,

$$\mathbb{E}[\rho_t \left(R_{t+1} + \gamma G_{t+1:h}\right)] + \mathbb{E}[V_{h-1}(S_t)] - \mathbb{E}[\rho_t]\,\mathbb{E}[V_{h-1}(S_t)] \tag{4}$$

Furthermore, the expectation of the importance sampling ratio is given by,

$$\mathbb{E}[\rho_t] = \mathbb{E}\left[\frac{\pi(A_t|S_t)}{b(A_t|S_t)}\right] = \sum_a b(a|S_t)\frac{\pi(a|S_t)}{b(a|S_t)} = \sum_a \pi(a|S_t) = 1$$

Which we can substitute into equation (4) to get,

$$\begin{aligned}
&\mathbb{E}[\rho_t \left(R_{t+1} + \gamma G_{t+1:h}\right)] + \mathbb{E}[V_{h-1}(S_t)] - \mathbb{E}[V_{h-1}(S_t)] \\
=&\mathbb{E}[\rho_t \left(R_{t+1} + \gamma G_{t+1:h}\right)]
\end{aligned}$$

Which is the same as the left hand side of the equation (3). Therefore, we have proved that equation (2) does not add any bias to the original return in expectation.

Problem 6

[8 points] If you were to design an algorithm called TD-search by modifying the Monte-Carlo Tree Search, how do you go about it. In particular, which part of the MCTS you would modify and how?

Answer

TD methods use bootstrapping, whereas MCTS does not. Therefore, to design TD-search by modifying MCTS, we can incorporate bootstrapping into MCTS. Specifically, we can modify the simulation phase of MCTS to use TD methods.

By default, MCTS uses the rollout algorithm to simulate the environment. In rollout, we simulate trajectories by starting from a state, then choosing actions randomly until we reach a terminal state. The randomness of the action choice is often a uniform distribution. After taking each action, we record the reward observed. Once we reach a terminal state, we take the average of the rewards observed in the trajectory, and use that as the estimated reward for the state we started from. So, suppose we took 3 actions before reaching a terminal state, where we observed rewards of 1, 2, and 3. Then, we would take the average of these rewards: $(1 + 2 + 3)/3 = 2$. Then we can proceed to backup this value in the tree.

However, if we want to use TD methods, we would not choose actions according to a uniform random policy like the way traditional rollout works. Instead, we would choose actions according to the policy that we are trying to learn, i.e. derived from the Q-values. However, we can still make this policy ϵ-greedy, so that we can still explore in the planning phase. Additionally, since this is a simulation, there is no harm in using an on-policy method like SARSA (because we are not actually taking actions in the real environment, so we do not have to worry about 'bad' results from the on-policy method). So, for the simulation phase of TD-search, we would take the following algorithmic steps:

1. Start at the state S we are simulating from.
2. Choose the action from our current policy, i.e. the one that maximizes the Q-value of the state-action pair, or if the random number is less than ϵ, then choose a random action (this is normal ϵ-greedy exploration).
3. Take the action, and observe the reward R and the record next state S'. Use R to update our current average reward estimate for the state we started from. Say that this average reward estimate is R_{avg}.
4. Update the Q-value of the state-action pair using SARSA, i.e. given by the following update rule:

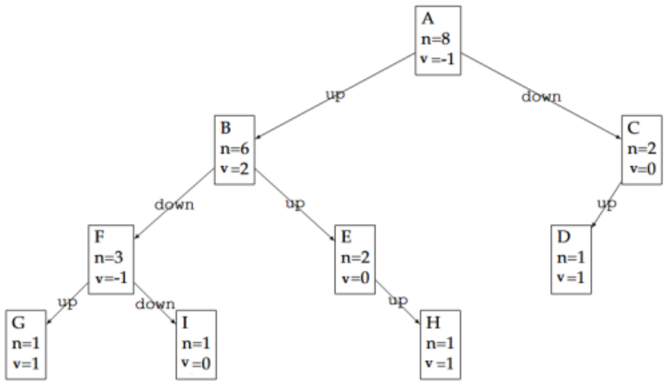
$$Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma Q(S', A') - Q(S, A)\right]$$

5. Set $S = S'$, set $A = A'$, and repeat steps 2-4 until we reach a terminal state.
6. Backup the value of the state we started from.

The backup step is similar to backup step in traditional MCTS, i.e. we take the R_{avg} that we calculated in step 3 of the TD simulation, and we backup this value in the tree. This way, we can use the benefits of bootstrapping in MCTS.

Problem 7

7- Assume we are using Monte Carlo Tree Search (MCTS) to decide on the next action for a game with two actions at each state (up and down). The state of the tree at time t is as follows:



Each state has a name (A, B, ..), a return value v and n value. Assume that the constant *c* in UCB1 is 0.5.

A. [5 points] What is the node that is selected next? Show your work.

B. [2 points] What action is next for the selected node? This action will make a new state call it "J".

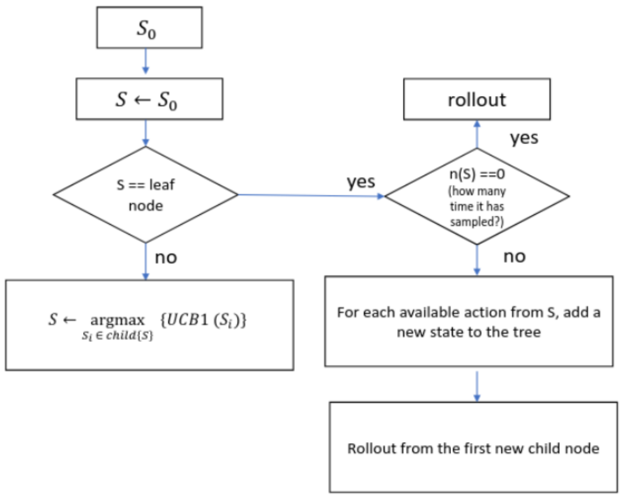
C. [2 points] What is the "n" and "v" value for the new state "J"?

D. [5 points] if the simulation rollout from the expanded node gives a value of 1, backup that value to all of the affected nodes.

E. [4 points] assume that after this final rollout, we've run out of time to run the MCTS simulation and must now choose an action from state A. What action will be chosen and why if we use the greedy tree policy.

Answer

A. We can use the following flowchart to choose the next node:


S_0
$S \leftarrow S_0$
S == leaf node
yes
no
$S \leftarrow \underset{S_i \in child(S)}{\text{argmax}} \{UCB1(S_i)\}$
rollout
yes
n(S) ==0 (how many time it has sampled?)
no
For each available action from S, add a new state to the tree
Rollout from the first new child node

Problem 7 continued on next page. . .

The UCB1 formula is given by,

$$UCB1(S_i) = \frac{v_i}{n_i} + C\sqrt{\frac{\ln N_i}{n_i}}$$

Where,
n_i is the number of times we have visited node S_i,
N_i is the number of times we have visited the parent of node S_i,
v_i is the value of node S_i, and
C is a constant that we choose.

Starting at A, we choose B because by the UCB1 algorithm,

$$\frac{2}{6} + 0.5\sqrt{\frac{\ln 8}{6}} > \frac{0}{2} + 0.5\sqrt{\frac{\ln 8}{2}}$$
$$0.62 > 0.51$$

Then, at B, we pick E because,

$$\frac{-1}{3} + 0.5\sqrt{\frac{\ln 6}{3}} < \frac{0}{2} + 0.5\sqrt{\frac{\ln 6}{2}}$$
$$0.053 < 0.473$$

Then, at E, we pick H because that is the only option. It is possible that H is the only option because there is no valid down action from E.

Hence, H is the next node we pick.

B. At H, we assume that both up and down actions are valid, since the prompt of this question does not specify otherwise. Therefore, we add both up and down nodes from H. Since the first action is up, we choose it. Ideally we would choose the action that maximizes the UCB1 value, but since we have no information about the value of the nodes, we choose the first action. This can also be randomly selected according to the uniform distribution (like a coin flip). This up action creates a new state, which is the 'J' node.

C. The 'n' and 'v' of J are 0 and 0 currently because we have not visited J yet, and we have not rolled out from it yet.

D. The simulation from J gives a value of 1. So now we can backup the tree and update the 'n' and 'v' of J and its ancestors. The new values are given below:
J: $n = 1$, $v = 1$
H: $n = 2$, $v = 2$
E: $n = 3$, $v = 1$
B: $n = 7$, $v = 3$
A: $n = 9$, $v = 0$

E. Now that we have run out of time, we compare the up and down actions at A. With up, we reach B, which has an average reward is $3/7 = 0.42$. With down, we reach C, which has an average reward of $0/2 = 0$. Therefore we choose up because in the greedy tree policy we choose the action that maximizes the average reward.

Problem 8

In this question we are comparing mean and variance of the estimate reward for on-policy and off-policy algorithms. Consider the on-policy scenario below where actions are taken under target policy π:

Action	Reward	Probability of Action under policy π
right	+10	0.9
left	+20	0.1

A. [2 points] What is the expected value (mean) of estimated reward?
B. [4 points] What is the variance of the estimate reward?
Hint: use the equation $Var[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$

Now assume that there is a behavior policy that is taking actions with following distribution:

Action	Reward	Probability of Action under policy π	Probability of Action under policy b
right	+10	0.9	0.5
left	+20	0.1	0.5

C. [3 points] Assume that we don't know the distribution of the target policy and we only know the ratio $\frac{\pi}{b}$ (importance sampling ratio).
Numerically calculate the expected reward assuming that the action are taken by the behavior policy.
D. [5 points] Calculate the variance of the expected reward?
E. [3 points] What is the intuition behind higher variance in off-policy in this example? Can you explain how the importance sampling could increase the variance in off-policy learning?

Answer

A. The expected value of the estimated reward is given by,

$$\begin{aligned}\mathbb{E}[X] &= \sum_x xp(x)\\ &= 10 \cdot 0.9 + 20 \cdot 0.1\\ &= 11\end{aligned}$$

B. The variance of the estimated reward is given by,

$$\begin{aligned}Var[X] &= \mathbb{E}[X^2] - \mathbb{E}[X]^2\\ &= (10^2 \cdot 0.9 + 20^2 \cdot 0.1) - 11^2\\ &= 9\end{aligned}$$

C. The expected reward under the behavior policy is given by,

$$\begin{aligned}\mathbb{E}[X] &= \sum_x xp(x)\\ &= 10 \cdot 0.5 + 20 \cdot 0.5\\ &= 15\end{aligned}$$

And the overall importance sampling ratio is given by,

$$\begin{aligned}\prod \frac{\pi}{b} &= \frac{0.9}{0.5} \cdot \frac{0.1}{0.5} \\ &= 0.36\end{aligned}$$

Therefore, the expected reward is,

$$\begin{aligned}\mathbb{E}[X] &= 15 \cdot 0.36 \\ &= 5.4\end{aligned}$$

D. The variance of the expected reward is given by,

$$\begin{aligned}Var[X] &= \mathbb{E}[X^2] - \mathbb{E}[X]^2 \\ &= (10^2 \cdot 0.5 + 20^2 \cdot 0.5) \cdot 0.36 - 5.4^2 \\ &= 60.84\end{aligned}$$

E. The intuition behind higher variance in off-policy learning is that, the behavior policy is not the same as the target policy, so the actions taken by the behavior policy are not the same as the actions taken by the target policy. Therefore, the expected reward is different for the behavior policy and the target policy.

Importance sampling increases the variance in off-policy learning because there could be actions that the behavior policy takes that the target policy does not take. This would result in a higher variance in the estimated reward, because if we graphed the accumulated reward over time, the graph would be more 'jagged' than if we were using the target policy. The jaggedness comes from the times when importance sampling goes to 0 because of actions not taken by the target policy.