# CS221 Exam 1 Solutions

Spring 2021

**Please read all of the following information before starting the exam:**

- This test has 19 questions on 18 pages for a total of 80 possible points.
- **You will have 100 minutes to complete and submit the exam.** The exam has been designed to take around 80 minutes, and the extra 20 minutes are provided to allow for submission upload.
- Note that different questions are worth different amounts of points. Budget your time accordingly!
- Keep your answers precise and concise. We may award partial credit so show all your work clearly and in order.
- Don't spend too much time on one problem. Read through all the problems carefully and do the easier ones first.
- If you are unsure about a problem statement when taking the exam, state your assumptions in the provided space on the last page. We will take all reasonable assumptions into account when grading.
- This exam is open-book; you may use any inanimate resources, including the course website.
- Being subject to the provisions of the Honor Code means in part that you must observe the rules established for this exam, which are: you may consult only inanimate sources. You may not consult or collaborate with anyone about the questions. Such collaboration is a violation of the Honor Code.
- Good luck!

| Problem | Part | Max Score | Score |
|---|---|---|---|
| 1 | a | 8 | |
| | b | 8 | |
| | c | 11 | |
| | **Total** | **27** | |
| 2 | a | 10 | |
| | b | 16 | |
| | c | 10 | |
| | **Total** | **36** | |
| 3 | a | 9 | |
| | b | 8 | |
| | **Total** | **17** | |

Total Score: ☐ + ☐ + ☐ = ☐

**0. Honor Code** (*0 points*) Please write or type down the honor code below and sign your name. Your exam will not be graded if this question is not completed.

*"I will not consult or collaborate with anyone about the questions. Such collaboration is a violation of the Honor Code."*

# 1. Machine Learning (*27 points*)

**a.** (*8 points*) **Loss Functions**

Consider the following loss function:

$$\text{Loss}(x, y, \mathbf{w}) = \frac{1}{3} \max \{4 - \mathbf{w} \cdot \phi(x) y, 0\}^3$$

Compute the gradient $\nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{w})$.

**Solution** Using the chain rule, we get the following gradient:

$$\nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{w}) = \begin{cases} -(4 - \mathbf{w} \cdot \phi(x) y)^2 \phi(x) y & 4 > \mathbf{w} \cdot \phi(x) y \\ 0 & \text{otherwise} \end{cases}$$

**b.** (*8 points*) **ML Design Decisions**

An important component of developing machine learning systems is making good design decisions to ensure that the trained model performs well. This problem will cover some important design decisions to be made when setting up a machine learning algorithm.

(i) [2 points] You want to train a binary classifier. However, before you begin training, you remember that there are some important decisions to be made! Recall that **hyperparameters** are design decisions that need to be made before running the learning algorithms, such as the hypothesis class, training objective, and optimization algorithm. Concisely describe a problem that can arise if you choose the values of hyperparameters based on their effect on the training error. Concisely describe a problem that can arise if you use the test error.

**Solution** Choosing hyperparameters that optimize training error results in overfitting and a model that includes all features, doesn't include regularization, and trains forever. Choosing hyperparameters that optimize test error are an unreliable estimate of a model's error on unseen data.

(ii) [2 points] Briefly describe what a validation set is and how it can be used to assist in choosing hyperparameters.

**Solution** A validation set is a portion of the data taken out of the training set that is used to optimize hyperparameters. We select the values of the hyperparameters based on which values yield the lowest error on the validation set. Using the validation set prevents the model from overfitting the training set and the test set.

(iii) [2 point] You decide to use stochastic gradient descent as your optimization algorithm and are selecting a value for the step size $\eta$. You are deciding between using a large step size or a small step size. Describe one advantage and one disadvantage of using each step size.

**Solution** A larger step-size results in faster convergence, but less stability. A smaller step-size results in slower convergence, but more stability.

(iv) [2 points] You train your binary classifier by running stochastic gradient descent using a regularized objective with regularization weight $\lambda$, and obtain a low training loss. However, you notice that the loss at test time is quite high. In 1-2 sentences, explain how you would change the hyperparameter $\lambda$ to improve performance at test time and why this change would help.

**Solution** The model is overfitting to the training data. You should increase the value of hyperparameter $\lambda$ in order to increase regularization by increasing the relative weight of the $\frac{1}{2}\mathbf{w} \cdot \mathbf{w}$. Increasing the regularization weight has the effect of reducing the hypothesis class by forcing the algorithm to pick weights from a smaller set that is closer to zero.

**c.** (*11 points*) **Unsupervised Learning**

(i) [8 points] Consider the following set of 1 dimensional points: {-5, -3, -1, 1, 2, 4, 6}. You want to cluster the data into two groups, so you decide to run the K-Means algorithm with two centroids.

Kem, one of your friends, receives the following centroids with the following clusterings: $\mu_1$ = -2 containing {-5, -3, -1, 1}, and $\mu_2$ = 4 containing {2, 4, 6}. You and Kem are competitive with each other, and you wish to find a better solution than Kem. Assume you initialize the centroids to be $\mu_1 = -7$ and $\mu_2 = 2$. **Report the final clusterings and the final centroid $\mu_1$ and $\mu_2$ after running K-Means until convergence. Between your solution and Kem's solution, which solution is better?** (Hint: Recall that our goal is to minimize the K-means objective, or the squared distance between a point and it's cluster)

**Solution** After the first iteration of K-means, we have the following cluster assignments: $\mu_1 = -4$ and $\mu_2 = 2.4$. Cluster 1 contains {-5, -3} and Cluster 2 contains {-1, 1, 2, 4, 6}.
After the second iteration of K-means, we have the following cluster assignments: $\mu_1 = -3$ and $\mu_2 = 3.25$. Cluster 1 contains {-5, -3, -1} and Cluster 2 contains {1, 2, 4, 6}. After this iteration, the algorithm converges as the cluster assignments do not change. Our solution is better than Kem's solution. The squared distance for our solution is 22.75, while the squared distance of Kem's solution is 28.

(ii) [3 points] You show your final cluster assignments to your friend Gabby and she wants to try computing the clusterings on her own to see if she can get a better solution than you and Kem. She decides to randomly initialize her centroids. Is Gabby guaranteed to reach a globally optimal solution? If yes, justify your answer. If no, what other guarantee does K-means provide?

**Solution** Gabby is not guaranteed to reach a globally optimal solution. K-Means only guarantees a local optima, so there is the possibility that her initialization does not yield the global optima.

## 2. Sabina's Travels (*36 points*)

**a.** (*10 points*) **Sabina's Errands**

Our friend Sabina (who you may remember from section!) once again needs our help navigating to different stops around town.

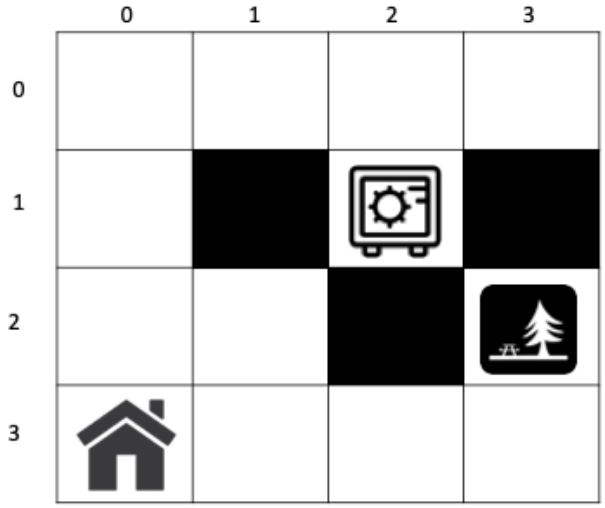



Figure 1: Sabina's town. Sabina's home is at (3, 0), the bank is at (1, 2), and the park is at (2, 3).

Our problem is as follows:

- Sabina will start at her house at $t = 0$ and needs to visit the bank and the park (in either order) before returning home.
- Sabina's goal is to minimize the total time spent on her journey.
- At a given location, Sabina can move north, south, east, or west, except if doing so would move into a blocked square.
- Additionally, Sabina can "jump" over a blocked square, moving a distance of 2 in a particular direction, but she can only perform this action twice.
- Each move or jump incurs a cost of 1.

(i) [5 points] To model our state for this problem, we will need to keep track of our current location in the grid, as well as some additional information.

- What additional information comprises the minimal state we need for our search problem?
- If there are n = 13 locations on the grid (as in the instance above), how many possible states are there?

**Solution** We need 3 additional pieces of information: Number of jumps remaining, **1**[Visited Park], **1**[Visited Bank]. This means we have 3 x 2 x 2 x n = 12n = 12 x 13 = 156 states.

(ii) [5 points] For each of the following search algorithms, state whether or not they could be used to find the optimal solution to our problem, along with a sentence or two explaining why or why not.

- Backtracking Search

  **Solution** Yes, backtracking search is always able to exhaustively explore the search tree to find the optimal solution.

- Depth First Search

  **Solution** No, our action costs are not all 0.

- Breadth First Search

  **Solution** Yes, all our action costs are constant = 1.

- Dynamic Programming

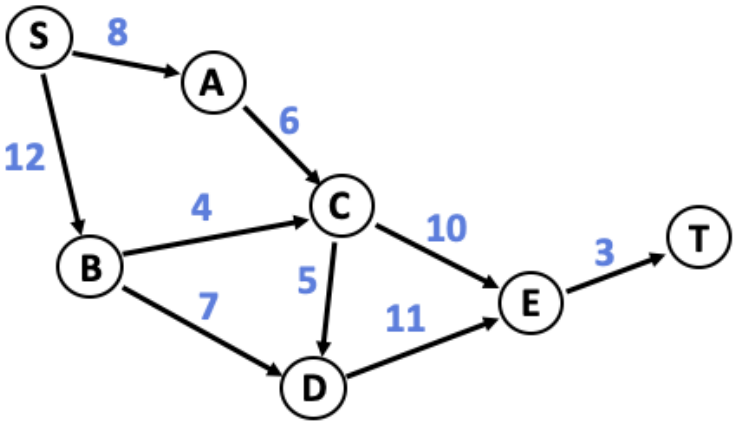
  **Solution** No, there are cycles.

- Uniform Cost Search

  **Solution** Yes, all our action costs are nonnegative.

**b.** (*16 points*) **A-Star Search**

Imagine we are provided the following search problem in graphical form, where the nodes represent states and the directed edges represent actions moving from one state to its successor (with the associated cost given as the edge weight). Our goal is to make it from our starting state S to our target state T.




(i) [8 points] We want to solve this problem with A-Star search, so the following table contains several candidate heuristics.

- For each heuristic, state whether or not it is consistent.
- If multiple heuristics are consistent, explain how we could combine them to perform A-Star search as efficiently as possible and why your strategy would lead to the best performance.

| | $h_1(x)$ | $h_2(x)$ | $h_3(x)$ |
|---|---|---|---|
| S | 22 | 20 | 24 |
| A | 16 | 15 | 20 |
| B | 14 | 13 | 16 |
| C | 10 | 11 | 14 |
| D | 8 | 9 | 10 |
| E | 2 | 1 | 3 |
| T | 0 | 0 | 0 |

**Solution** $h_1$ and $h_2$ are consistent. $h_3$ overestimates the distance from C.

We should use $\max(h_1, h_2)$ to get the best performance, as that is guaranteed to be consistent and will give us the largest possible heuristic value, which means we should expect to visit fewer states overall.

(ii) [8 points] Using the following heuristic, what is the order of states that we'll visit? And what states comprise the shortest path that we'll find?

| | $h_4(x)$ |
|---|---|
| S | 20 |
| A | 16 |
| B | 13 |
| C | 12 |
| D | 9 |
| E | 3 |
| T | 0 |

**Solution** The state ordering is S, A, B, C, E, T. The shortest path is S, A, C, E, T.

**c.** (*10 points*) **Sabina's Car Troubles**

We decide to check back with our friend Sabina and find that she's discovered an issue with her car. Now, whenever she attempts to move, there is some non-zero probability $p$ that the car will stall and Sabina will stay in the same location. Once again, Sabina will start at her home and attempt to make it to other locations in town. She really wants to go to the park, but it might be a better idea to go to the mechanic, and she only has time to do one.

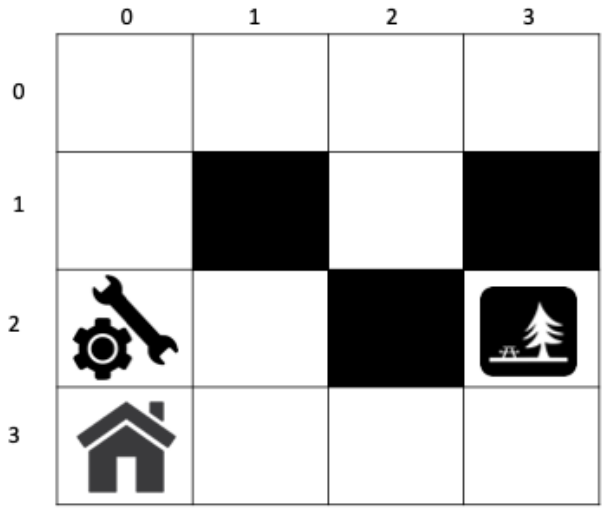



Figure 2: Sabina's town. Sabina's home is at (3, 0), the mechanic is at (2, 0), and the park is at (2, 3).

We decide to model this problem as an MDP. Now, instead of incurring a cost of 1 for a given move or jump action, we consider our action cost as a *negative reward*. Since Sabina would really like to go to the park, making it there earns a reward of 100, while making it to the mechanic earns a reward of 10 (although the max reward she could receive is 9, because reaching the mechanic in one step would incur a cost of -1). In either case, after making it to the mechanic or the park, the MDP terminates.

(i) [3 points] In the case where the probability of failure $p = 0$, what is the optimal behavior when the discount factor $\gamma = 1$? What is the optimal behavior when the discount factor $\gamma = 0$?

**Solution** If our discount factor is 0, then future rewards don't matter, and we would expect our agent to learn to head directly for the mechanic. If our discount factor is 1, then our agent will learn to maximize its reward by heading for the park.

(ii) [4 points] If we adopt the policy of always trying to navigate to the mechanic, what is the expected utility (average discounted sum of rewards) for this policy in terms of p when our discount factor is 0.5?

**Solution** $\mathbb{E}(\text{reward}) = (1-p)(-1+10) + p(-1+0.5\,\mathbb{E}(\text{reward}))$
$\mathbb{E}(\text{reward}) = \frac{9-10p}{1-0.5p}$

(iii) [3 points] We want to use an algorithm to estimate $Q_\pi$ for a policy $\pi$ that always proceeds to the park, using only data from that policy. Which algorithm would be a better choice and why: Model-based Monte Carlo or Model-free Monte Carlo?

**Solution** Model-free Monte Carlo is the best choice in this case because it's an on-policy method. We are not sufficiently exploring the MDP to use Model-based Monte Carlo.

## 3. Dividing the Cake (*17 points*)

Alice and Bob are sharing one cake. First, Alice divides the cake into two pieces, then Bob chooses one piece, leaving Alice with the other piece. Alice can either divide the cake with volume ratio 1:1 or 1:2.

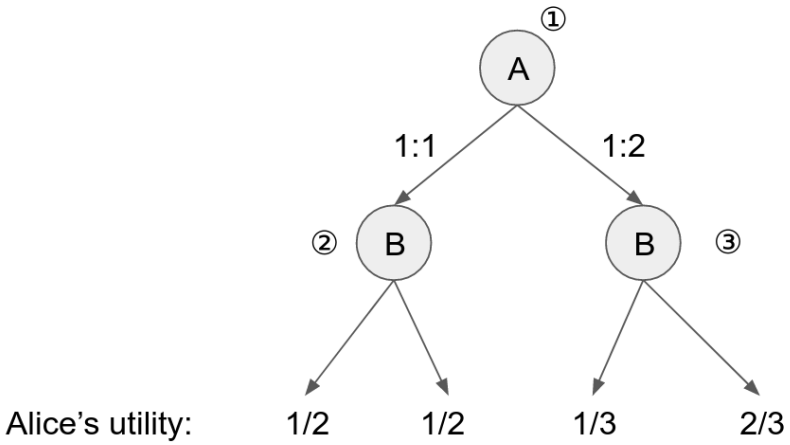



Figure 3: The answer game tree.

**a.** (*9 points*) **Know your Opponent**

Figure 3 shows the game tree. In each of the following scenarios, Bob chooses his piece using a different strategy. We assume that Alice always chooses the best possible policy to maximize her share of cake. For each scenario, please write down the value of each state node in the game tree above for Alice's policy.

We've labeled each node with an ID, shown as a circled number beside each node. We expect you to write your answer in the format of ⓧ**: y**, where x is the ID of the node, and y is the value of the node.

(i) [3 points] **Adversarial.** Bob plays adversarially (in other words, Bob is trying his best to get the largest possible share of cake):

**Solution** The minimax state values:

1: 1/2
2: 1/2
3: 1/3

(ii) [3 points] **Altruistic.** Bob plays collaboratively (in other words, Bob is trying his best to let Alice get the largest possible share of cake):

**Solution** All nodes are max nodes.

1: 2/3
2: 1/2
3: 2/3

(iii) [3 points] **Random.** Bob picks the larger piece of cake with probability $p$. Please write the state values in terms of $p$. How would you advise Alice to choose the dividing ratio according to the value of $p$?

**Solution** The expectimax state values:

1: $\max\{\frac{1}{2}, \frac{2}{3} - \frac{1}{3}p\}$
2: $\frac{1}{2}$
3: $\frac{1}{3}p + \frac{2}{3}(1-p) = \frac{2}{3} - \frac{1}{3}p$

Comparing the values of state 2 and 3, Alice should choose ratio 1:1 when $p > 0.5$, choose ratio 1:2 when $p < 0.5$, and choose either one when $p = 0.5$.

**b.** (*8 points*) **Inviting a New Friend**

Now Carol joins Alice and Bob. In this part, Alice divides the cake into three pieces. Bob chooses one piece, and then Carol chooses another. Both Bob and Carol would play adversarially to Alice (in other words, trying their best to minimize Alice's share of cake). Alice can either divide with volume ratio 1:1:1 or 1:2:3.

You want to help Alice find the best policy using alpha-beta pruning.

(i) [5 points] A game tree is shown in Figure 4. For Bob and Carol, the fractions labeled on the edges represent the percentage of cake they choose. In this game tree, how many **leaf nodes** are unvisited when running alpha-beta pruning? Note that a leaf node refers to a number showing Alice's utility at the bottom row of the tree.

**Solution** The rightmost 4 leaf nodes are unvisited. This is because (1) when traversing the left subtree of A, since we are essentially finding the global minimum in this subtree, we must visit every leaf node in it; (2) after visiting the first C node in the right subtree, we have the value of the B node in the right subtree $\leq \frac{1}{3}$, which does not non-trivially overlap with the value range of A, which is $\geq \frac{1}{3}$ after visiting the left subtree. Therefore the rest leaf nodes can be pruned.

(ii) [3 points] If we swap the order of the two subtrees of the root node A, do we end up with the same leaf node(s) being pruned? Justify your answer with 1-2 sentences.

**Solution** No. This is because we would have to solve for the global minimum within the original right subtree, therefore the original 4 rightmost leaf nodes won't be pruned.

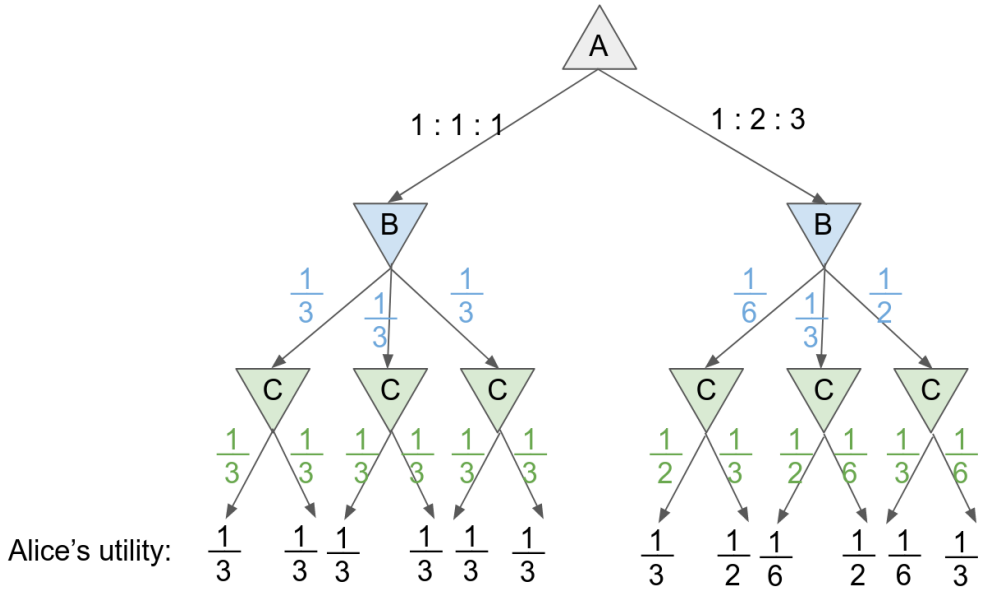



Figure 4: The game tree for Alice, Bob, and Carol.

**If you believe that any of the questions were ambiguous, please state your assumptions here.**