CS6601 Final – Spring 2020

Please read the following instructions thoroughly.

Fill out this PDF form and submit it on **Gradescope**. Remember to also submit on Canvas for backup purposes.

You have unlimited resubmissions until the deadline. You can: **(a)** type directly into the form – we highly recommend using Adobe Reader DC (or Master PDF on Linux). Other programs may not save your answers, so **please keep a backup**; or **(b)** print, hand-write & scan. You can combine the methods as well.

Submit only a single PDF – no phone pictures, please! (You may use an app like CamScanner or Office Lens if you do not have scanner access.) Do not add pages unless absolutely necessary; if you do, please add them at the end of the exam **only**, and clearly label **both** the extra page and the original question page. Submit **ALL** pages of the exam, not only the completed ones.

Do not forget to fill the checklist at the end before turning in the exam. The exam may not be graded if it is left blank.

The exam is open-book, open-note, open video lectures, with no time limit aside from the open period. No internet use is allowed, except for e-text versions of the textbook, this semester's CS6601 course materials, Piazza, and any links provided in the PDF itself. No resources outside this semester's 6601 class should be used. Do not discuss the exam on Piazza, Slack, or any other form of communication. If there is a question for the teaching staff, **please make it private on Piazza and tag it as Final Exam with the question number in the subject line** (for example, a question on Search would be "Final Exam #2").

Please round all your final answers to 6 decimal places and do not round intermediate results. You can use the `round(your_number,6)` function in Python for help.
You will not receive full credit if your answers are not given to the specified precision.

Point breakdown (Each question has sub-parts with varying points):

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Total
Pts	**7**	**7**	**10**	**10**	**12**	**10**	**12**	**8**	**12**	**12**	**100**

1. Game Playing - Giant Probabilistic TicTacToe

(7 points)

This is a version of TicTacToe that incorporates randomness. The first main difference from regular TicTacToe is that the board is 4x4 instead of 3x3. The rules to win are the same as regular TicTacToe (i.e.), you need 3 consecutive markers vertically, horizontally or diagonally to win. The other main difference is that on your turn, the marker being placed may not end up where it was intended to be placed. The marker has a probability of being placed in the intended location with a probability of 0.8. It may end up in any of the 4 squares adjacent to the intended location with a probability of 0.05 each.

The probability of the final marker location is:

- Intended position with probability 0.8
- Left of intended position with probability 0.05
- Below intended position with probability 0.05
- Right of intended position with probability 0.05
- Above intended position with probability 0.05

In addition, if your intended position is adjacent to any blocked positions of the board, the probability of those blocked positions gets added to the probability of the given position. Here, a blocked position could be a position that's already occupied, or a position that is off the board. A couple of examples are given below:

In the following empty board, if the intended position is **D2**, a marker can't be placed below **D2** because **D2** is at the edge of the board. So the 0.05 that was intended for the below cell is now assigned to **D2**, and so the probability of the marker being placed at **D2** becomes 0.85. The distribution of probabilities would be as shown below:

	1	2	3	4
A				
B				
C		0.05		
D	0.05	0.85	0.05	

In the following empty board, if the intended position is **A4**, the distribution of probabilities would be as shown:

	1	2	3	4
A			0.05	0.9
B				0.05
C				
D				

In the following partially filled board, if the intended position is **B3**, the distribution of probabilities would be as shown:

	1	2	3	4
A	X		0.05	
B		O	0.95	O
C	O		X	
D				

In the following partially filled board, if the intended position is **B1**, marker would go to the intended position with a probability of 1:

	1	2	3	4
A	X			
B	1	O		O
C	O		X	
D				

Question A: Evaluating the Board

In this part, we will evaluate different moves, using different evaluation metrics.

The first metric we will use is the number of similar markers neighboring the position in question. For example, if it is X's turn to place a marker, a cell's score is the number of X's in the 8 cells that are neighbors to the cell in consideration. (A cell's neighbors include cells that are adjacent vertically, horizontally and diagonally).

For example, in the given partially filled board, the scores of some of the scores have been filled, given it's X's turn and that X is the maximizing player.

	1	2	3	4
A	**X**		0	
B	1	**O**	1	**O**
C	**O**	2	**X**	
D			**X**	

Q1.A.1 Given the following board state, answer the following questions given that it is O's turn and that O is the maximizing player:

	1	2	3	4
A		X	O	
B			O	
C		X		O
D	X			

Q1.A.1.a. What is the score of cell B2? **(0.25 points)**

_____

Q1.A.1.b. What is the score of cell B4? **(0.25 points)**

_____

Q1.A.1.c. What is the score of cell C3? **(0.25 points)**

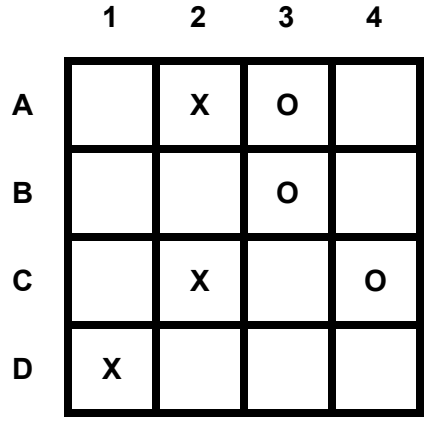
_____

Q1.A.1.d. According to this metric, which is the best cell to place an O? **(0.25 points)**

_____

Do you think this evaluation metric is the most suitable? As you may have guessed, it probably isn't the best, because it doesn't consider opponent markers. It seems that placing a marker at B2 is the better option, as it not only blocks X from possibly winning in their next turn, but also opens the possibility of O winning in their next turn. So let us try a metric that considers the opponent markers as well. The next evaluation metric we consider is the total number of neighboring markers.

Q1.A.2 Apply this new evaluation metric to the same board and, answer the following questions given that it is O's turn and that O is the maximizing player



Q1.A.2.a. What is the score of cell B2? **(0.25 points)**

_____

Q1.A.2.b. What is the score of cell B4? **(0.25 points)**

_____

Q1.A.2.c. What is the score of cell C3? **(0.25 points)**

_____

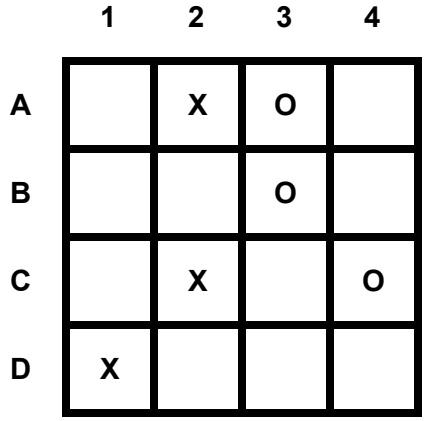
Q1.A.2.d. According to this metric, which is the best cell to place an O? **(0.25 points)**

_____

Clearly, this metric is a little better than the previous one. Still, is it the best? You must've noticed that placing an O at C3 would likely cause O to win, and yet this is not captured in either evaluation metric as above. So to further improve the evaluation metric, we can use something like this:

- If a cell causes immediate victory, assign it a massive score (say 10) which is greater than the largest possible score from the first two metrics
- If there is a position which could lead to opponent victory in the next move, blocking that move should be the next highest priority. Assign it a value (say 9) that is greater than the largest possible score from the first two metrics, but lesser than the score assigned to a possible immediate victory
- Else, score is equal to the number of neighboring markers, i.e., the second evaluation function.

Q1.A.3 Apply this new evaluation metric to the same board and, answer the following questions given that it is O's turn and that O is the maximizing player:



Q1.A.3.a. What is the score of cell B2? **(0.25 points)**

_____

Q1.A.3.b. What is the score of cell B4? **(0.25 points)**

_____

Q1.A.3.c. What is the score of cell C3? **(0.25 points)**

_____

Q1.A.3.d. According to this metric, which is the best cell to place an O? **(0.25 points)**

_____

Hopefully, this has helped you understand better how to construct a suitable evaluation function for the given game. If you look into it more deeply, you will notice that the above evaluation function has some flaws, and would fail to produce the best move in some cases, especially when the game is close to its end. For those who are further interested, there are other things you can consider to build up an evaluation function for this game:

- Different weights for player and opponent's markers while counting neighbors
- Weighting edge and corner cells differently from other cells
- Considering the cases where the intended position may not be reached because of the probabilistic nature of the game

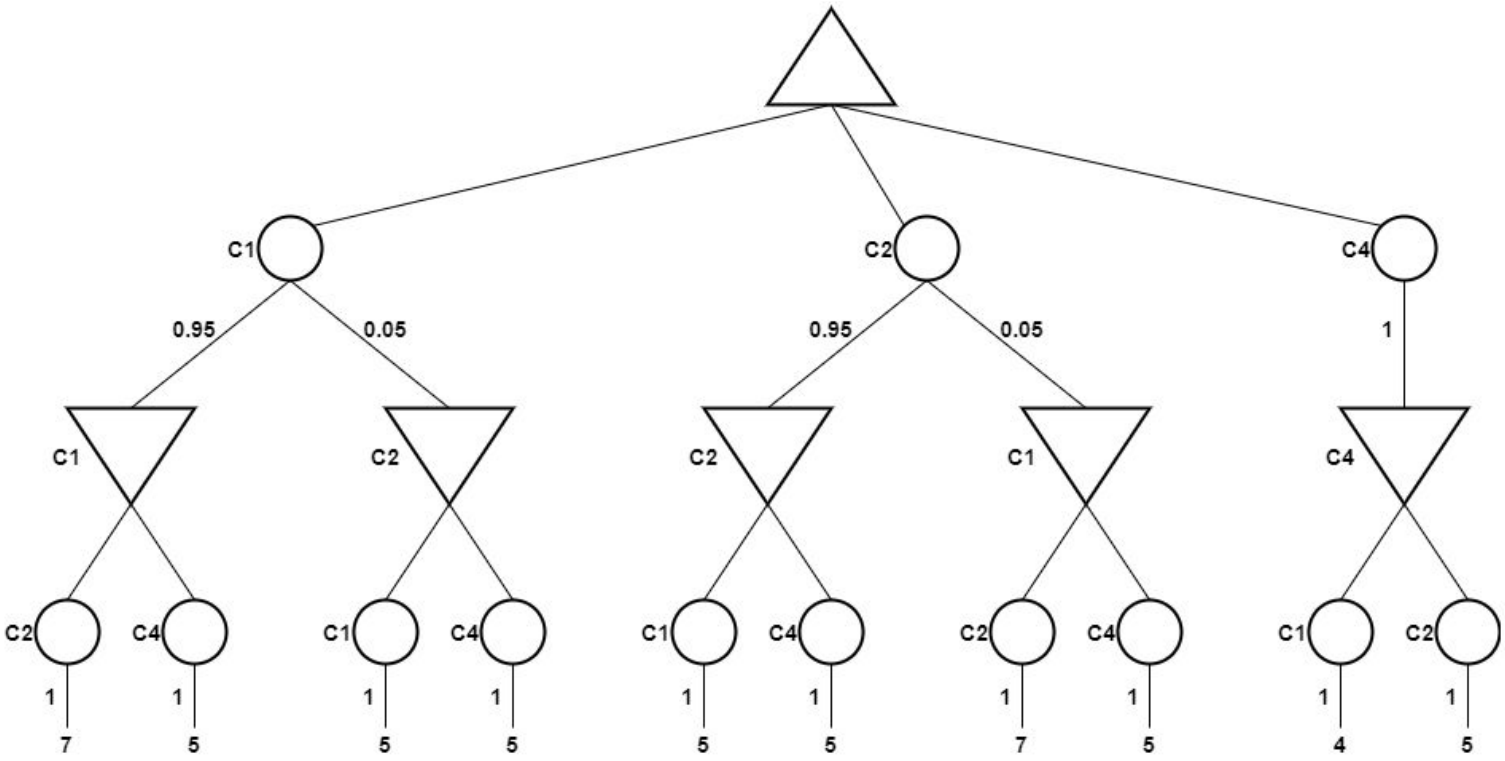
Question B: Expectimax

Expectimax is a version of the Minimax algorithm that takes into account probabilistic events.

Consider the following board. Assume that it is O's turn to place a marker, and that O is the maximizing player.

	1	2	3	4
A	**O**	**X**	**X**	**O**
B	**X**	**X**	**O**	**X**
C			**O**	
D	**O**	**X**	**X**	**O**

The three possible moves for O are cells C1, C2, and C4. Obviously, the marker may not end up in the intended position. All this has been captured in the expectimax tree given below. The intended cell is given next to the probability circles, and the actual cell taken is given next to the triangles.



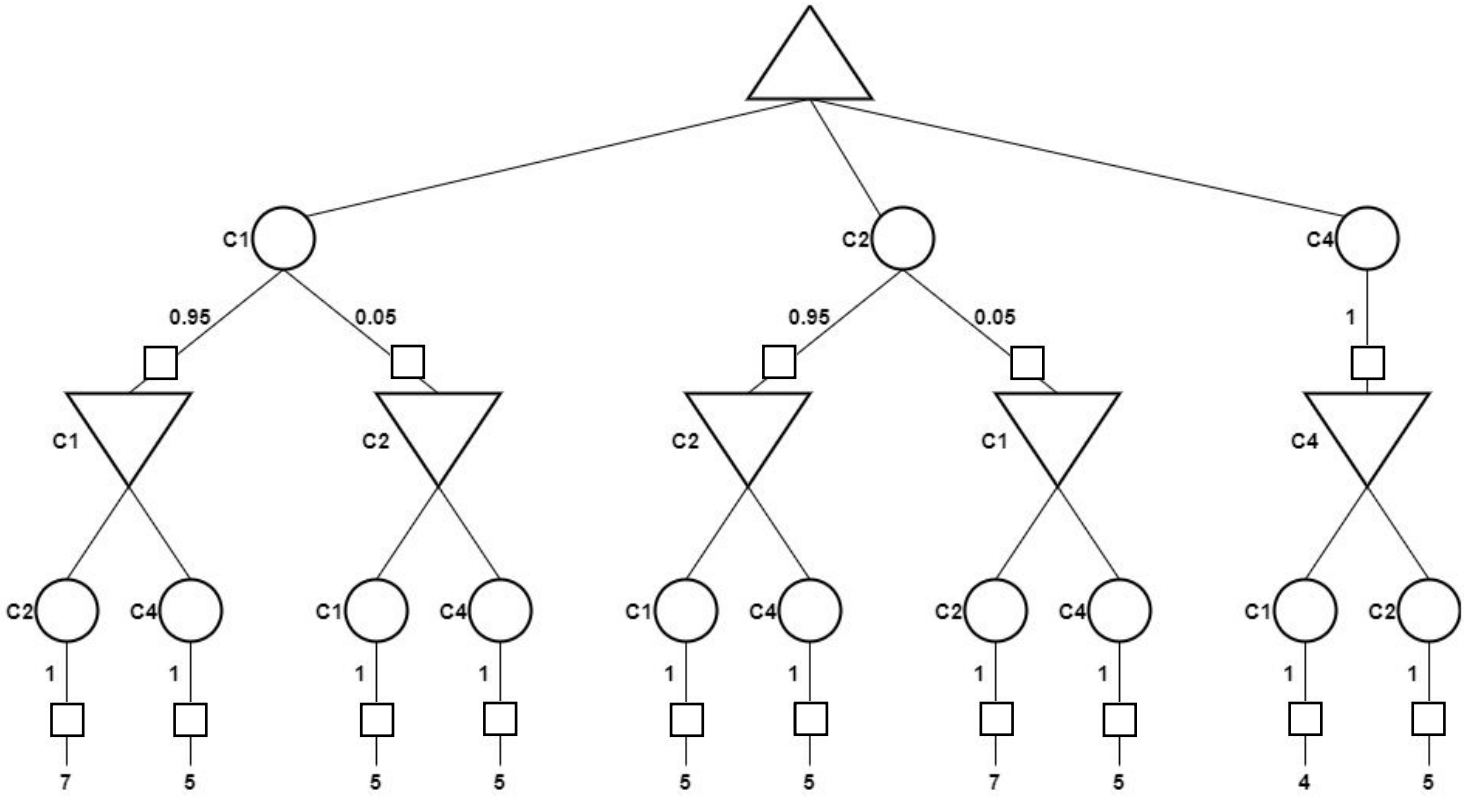
Q1.B.1. Please fill in the tree (without pruning). Make sure to use inequality signs wherever necessary. Answer the following questions:

Q1.B.1.a What is the value of the root node? **(0.5 points)**

_____

Q1.B.1.b Which move does O select based on this tree? **(0.5 points)**

_____



Q1.B.2 Please fill in the tree (with pruning). Make sure to use inequality signs wherever necessary. Answer the following questions:

Q1.B.2.a What is the value of the root node? **(0.25 points)**

_____

Q1.B.2.b Which move does O select based on this tree? **(0.25 points)**

_____

Q1.B.2.c How many leaf nodes are pruned? **(0.5 points)**

_____

Question C: General Game Playing

Q1.C.1 Which of these change automatically when you go from regular minimax to minimax with Alphabeta Pruning? (Select all that apply) **(1 point)**

- [ ] The move selected by agent
- [ ] The value of the root node
- [ ] The number of computations performed to determine the best move
- [ ] The evaluation function
- [ ] The number of nodes explored during the search

Q1.C.2 Which of the following are true about Iterative Deepening, in the context of game playing? (Select all that apply) **(1 point)**

- [ ] It is a DFS technique
- [ ] It gives the same value for root node irrespective of the depth currently being searched
- [ ] It is used in cases where the time allotted to select a move is limited
- [ ] If a timeout occurs, you return the best move from the current level being searched, even if the current level is not fully searched
- [ ] It makes use of the fact that better moves are predicted if you increase the search depth

After all that you have learnt through this question, maybe you can build an unbeatable AI agent for Giant Probabilistic TicTacToe!

2. Search

(7 points)

Maks is an undergraduate student at Georgia Tech and is getting close to finishing his coursework and graduating. To better prepare for the future, Maks decides to do some career planning. There are many job options he could explore for his career, so he creates a graph of the job options in which he starts at node **G = Georgia Tech** and his final goal is to reach node **R = Retire**, with a bunch of options in between. The job options in the graph are connected by edges that represent the "cost of transition" between these job options. You could think of the "cost of transition" between two job options as the opportunity cost for Maks to transfer from one job to another (don't think too hard about what this means).

Below is the career graph that Maks has made, where **W = Wall Street | A = Graduate School | P = Professor | L = Lecturer | D = Civil Servant | E = Engineer | T = Entrepreneur | F = Freelancer | B = Writer | J = Journalist.**

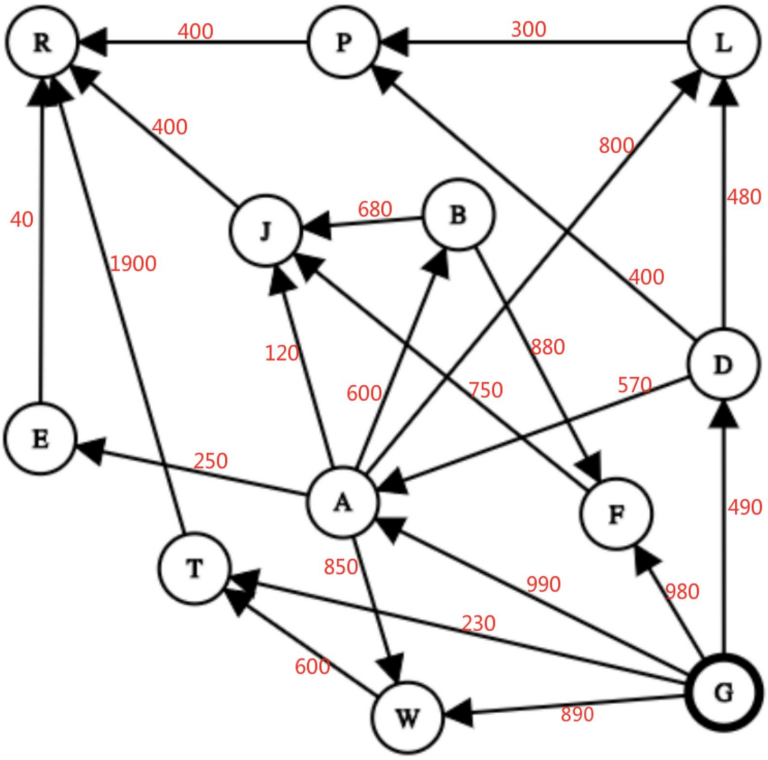


Figure 1. Career graph

Note: In all search problems asked below, please use alphabetical order to break ties when deciding the priority to use for extending nodes.

Q2.1.a. Assume Maks wants to retire after doing the least number of different jobs. Of all the basic search algorithms listed below, which one is guaranteed to find the path with the minimum nodes (**Note**: we want you to find the shortest path with the number of nodes as the length of the path) from G to R when it reaches the goal node for the first time? (**1 point**)

- ◯ Depth first search
- ◯ Breadth first search
- ◯ Uniform cost search

Q2.1.b. Now Maks is interested in finding a path which has the least "cost of transition". Which search algorithm should Maks apply to the graph in order to find the path? (**1 point**).

- ◯ Depth first search
- ◯ Breadth first search
- ◯ Uniform cost search

Q2.1.c. What is the least "cost of transition" path using the search algorithm you chose above? Please enter your answer as a comma-separated list using the node labels in the graph (e.g., the answer format should like: G, T, R) (**2 points**).

_____

As a new undergraduate student, Maks was not very confident about his own judgement for the career graph he plotted. Therefore, he went to talk to one of the career advisors in the Center for Career Discovery & Development at Georgia Tech. The career advisor gave Maks an additional cost --- heuristic node-to-goal costs for each job option, which represent the preconceptions about the "potential cost" (**Note: we assume "potential cost" has the same unit with "cost of transition" in the career graph**) he has to work until he retires.

For example, Maks thinks it will take 350 “potential costs” to go from G (Georgia Tech) to retirement (R), 285 “potential costs” from Grad School (A), but only 120 “potential costs” from Entrepreneur (T). The heuristic node-to-goal costs table is listed below.

Job options	Heuristic node-to-goal costs
G	350
W	150
F	400
D	320
A	285
T	120
E	35
J	330
B	390
L	400
P	310
R	0

Using the node-to-goal cost in the above table as a heuristic, the cost of a node in the below questions is: $f(x) = g(x) + h(x)$

Q2.2.a. How many nodes were explored (including the start and end nodes) from G to R using A* search? **Note: A node is considered as explored if this node is popped out from the frontier and added to the explored set.** (**2 points**)

_____

Q2.2.b. The heuristic function, h(n), in the A* algorithm is: (**1 points**)

- ◯ Cost of the cheapest path from current node to goal
- ◯ Estimated cost of cheapest path from current node to goal
- ◯ Average cost of all paths from current node to goal

3. Optimization Algorithms

(10 points)

The data used for answering the questions in this section is attached as a spreadsheet at the end of this question on page 22. For the sake of convenience, this spreadsheet is also provided to you as a Google Sheet that you can download and utilize on your local machine:

https://docs.google.com/spreadsheets/d/1bUCva8a7n3MKf-SC4OlUT5rfsRIvanp8wsKQnxyomPM/edit?usp=sharing

Background

A standard optimization problem is defined by two elements: an **objective function** and **constraints**. The objective function is typically the function that you want to explore, whether you want to find its minimum or maximum. The constraints specify the range of the domain of your search, as most searches don't extend into an infinite domain. If you have an optimization problem and can formulate a clear objective function and constraints, you're in a position to apply a wide range of optimization methods to find your answer such as genetic algorithms, simulated annealing, stochastic beam search, or gradient descent. One example of a standard formulation of an optimization problem is:

$$\begin{cases} \text{minimize} & y - ax \\ \text{subject to} & x + 2y \leq 4, -y + 2x \leq 3, x \geq 0, y \geq 0 \end{cases}$$

For some formulations of an objective function, we can apply methods to guarantee a globally optimum result, such as Convex Optimization. However, it is often the case that the objective function is unknown or can't be modeled in a way where we can apply these guaranteed methods, and we must turn to other approaches. Gradient Descent is an intuitive and effective method used to explore an objective function over a certain domain that can be visualized as dropping a ping-pong ball on a curved surface. Depending on its initial position, the ball will follow the force of gravity and roll downhill until it rests at a local minimum, where it can't go downhill by moving in any immediate direction. In this toy example, we could consider the distance from the ball to the ground as our objective function that we want to minimize. We can abstract this process into the following algorithm:

1. Choose an initial point to start your search
2. Calculate the gradient of your objective function at your current position.
3. If you determine you have reached a local minimum, stop your search. Else, move according to the gradient and repeat step 2.

You may notice that we need to define how we move in this algorithm. If your domain is enormous, it may be necessary to quickly traverse the space by moving in large steps at a time. If you want to carefully explore a small domain, however, you probably want to take smaller steps in each iteration. While the gradient will determine the direction of your step, the magnitude of your step is called the **step size**, and is an important parameter to tune according to a problem's context.

Problem

Laura is a material scientist working closely with an architecture firm to build a new skyscraper right in the middle of Manhattan. The design and location has been decided, and all that's left for her job is to determine the alloy used in constructing the support beams within the structure. She decides to use an alloy composed of Iron, Metal X, and Metal Y, but she isn't quite sure of the proportion of the latter two metals. The field of Material Science relies heavily on optimization algorithms to solve difficult problems involving the design of physical structures and their material composition, and she is familiar with the fundamentals of gradient descent. Laura decides to formulate and approach this scenario as a gradient descent problem.

First, she knows to define her objective function and constraints. To encourage stability, she decides to use a measurement of the building's shear strain (how much it wobbles) under normal conditions as her objective function to minimize. Luckily, she owns software such that she can simulate the construction of the building and measure the values in a somewhat reasonable amount of time. When defining the constraints, she uses her domain knowledge to narrow the search down to including anywhere from 1 to 10 ounces of each metal per square meter of alloy. Thus, her optimization problem is formulated as:

$$\texttt{minimize } \; shear(x, y)$$
$$\texttt{subject to } \; 1 \leq x \leq 10, \; 1 \leq y \leq 10$$

Where x and y represent the ounces per square meter of alloy of metal X and metal Y respectively.

The shear measurements of the different combinations of metals are shown in the spreadsheet on page X, but these values are unknown to Laura and she must traverse this space using the following gradient descent algorithm:

1. Choose an initial position (x, y) and a step size s.
2. Calculate the shear values at the four nearest cells: the cell immediately above your current position, immediately below, immediately left, and immediately right.
3. Calculate the direction of the X variable gradient, $d(x)$, as either -1, 0, or 1.
4. Calculate the direction of the Y variable gradient, $d(y)$, as either -1, 0, or 1.
5. If $d(x) = 0$ and $d(y) = 0$, terminate your search and return (x, y)
6. Move to the position $(x - s \times d(x), y - s \times d(y))$ and repeat step 2.

To calculate the direction of a gradient, we simply look for the slope of three values. Intuitively, we want to calculate the general direction of the slope of the function at a point, whether it's going towards the positive or negative direction for each axis. For a mathematical interpretation, consider a list of three values x_1, x_2, x_3 that represent the shear values at the left of the current position, current position, and right of the current position respectively. The value of $d(x)$ is then:

$$
\begin{aligned}
&+1 \text{ if } x_1 < x_2 \leq x_3 \text{ or } x_1 \leq x_2 < x_3 \\
&\ 0 \text{ if } x_2 < x_1 \text{ and } x_2 < x_3 \\
&\ 0 \text{ if } x_2 > x_1 \text{ and } x_2 > x_3 \\
&\ 0 \text{ if } x_1 = x_2 = x_3 \\
&-1 \text{ if } x_3 < x_2 \leq x_1 \text{ or } x_3 \leq x_2 < x_1
\end{aligned}
$$

The same calculation can be used to find $d(y)$ where y_1, y_2, y_3 represent the shear values below the current position, at the current position, and above the current position respectively.

The details for some edge cases are described below:

Edge Bordering: In the case that your current location borders an edge, you only have two values to compare when calculating $d(x)$ or $d(y)$. In this scenario, you should duplicate the border value to fill in the missing cell. For example, the three values we should use to calculate $d(x)$ at cell (10, 10) are (0.21, 0.23, 0.23), as the border value is duplicated. The value of $d(x)$ will be +1 accordingly. Similarly, the values we use to calculate $d(y)$ at cell (10, 10) are (0.23, 0.23, 0.26), and $d(y)$ will be -1.

Step Out-of-Bounds: In the case where a step would have led you outside the domain, stop at the border of the domain for that direction. For example, if we wanted to move from cell (10, 10) with a step size of 2, we could calculate our values of $d(x)$ and $d(y)$ to be +1 and -1 accordingly. Our new position should be (8, 12), but 12 is outside our domain. Because of this, we stop at (8, 10) and use this as our new position instead.

See the example below for a demonstration of all these rules:

Say we're currently exploring the cell (9, 9) and we want to move with a step size of 3. To calculate the x gradient direction, we use cell (9, 9) as a reference and examine the right, center, and left cells. The values of these cells are $(.12, .25, .26)$ respectively. We plug these three values into the criteria for $d(x)$ shown above and get a value of +1. When calculating the y gradient direction, we examine the below, center, and above cells to get the values $(.31, .25, .21)$. Our value for $d(y)$ is -1. When taking our step with step size 3, our new location is:

$$(x - s \times d(x),\ y - s \times d(y))$$

Which, filling in the values we've calculated, results in:

$$(9 - 3 \times 1,\ 9 - 3 \times (-1)) = (6, 12)$$

We can't go beyond our border, so we stop at (6, 10) and finish our movement.

Q3.a. Use the information and details described above to perform 5 steps of the gradient descent algorithm and record the results **(2 points):**

```
Initial Position: (x,y) = (4, 10)
Step Size = 2
```

Step Number	Current X Position	Current Y Position	Value of Objective Function
1	4	10	0.48
2			
3			
4			
5			

You may realize that our method of movement can greatly influence the output of the algorithm. Changing our step size can lead to a large difference in performance, and finding the best value is dependent on the problem context at hand as well as your desired precision for an output. See how the two different values below lead to significantly different outputs for the same optimization formulation:

Q3.b. Fill the table below using the below parameters for Gradient Descent **(2 points):**

```
Initial Position: (x,y) = (2, 9)
Step Size = 2
```

Step Number	Current X Position	Current Y Position	Value of Objective Function
1	2	9	0.34
2			
3			
4			
5			
6			

Q3.c. Fill the table below using the below parameters for Gradient Descent **(2 points):**

```
Initial Position: (x,y) = (2, 9)
Step Size = 1
```

Step Number	Current X Position	Current Y Position	Value of Objective Function
1	2	9	0.34
2			
3			
4			
5			
6			

You may realize that even with proper parameter values, standard gradient descent has some flaws. One of the most notable is the tendency to get stuck in a local optimum and failing to find the global optimum. This tendency is a result of only looking at the gradient of the current position when searching the domain, and there are some ways to alleviate this weakness. One popular approach is the 'random restart' method, where we perform multiple iterations of gradient descent with a different initial position every time. By increasing the number of iterations we complete alongside covering the domain of initial positions (the number and diversity of restarts), we can more confidently say that we have found the global optimum compared to standard gradient descent. Answer the question below to demonstrate this concept:

Hint: There are various ways to answer this question, some more tedious than others.

Iteration Number	X Initial Position	Y Initial Position
1	7	7
2	10	6
3	1	3
4	4	9
5	3	1
6	5	8
7	7	2

Fig 1: Initial Positions for Random-Restart Gradient Descent

Q3.d. Of the 7 iterations described above, which iteration number finds the smallest value of the objective function when using a step size of 1? **(2 points)**

Using the exact same random-restart method as above, Laura is able to find effective values of the above variables and construct the desired support beams with great success. As with most optimization algorithms, however, the solution produced can't be realistically guaranteed to be the global optimum.

Q3.e. Of the below decisions, which could potentially help Laura find the true global minimum of the shear strain function? Consider each decision as a single change from the above random restart method. Select all that apply. **(2 points)**

- ☐ Run a larger number of iterations of gradient descent from a common starting point
- ☐ Increasing the search space from [1, 10] to [1, 20] for both the x and y variables
- ☐ Discretize the search space into smaller buckets by searching from values 1 to 10 in increments of 0.5 instead of 1 for each variable
- ☐ Increase the step size to traverse the search space quicker.

10	0.43	0.51	0.49	0.48	0.47	0.31	0.19	0.04	0.21	0.23
9	0.31	0.34	0.45	0.43	0.46	0.29	0.23	0.12	0.25	0.26
8	0.23	0.29	0.41	0.39	0.45	0.28	0.25	0.15	0.31	0.29
7	0.15	0.11	0.39	0.35	0.38	0.24	0.27	0.19	0.45	0.31
6	0.12	0.09	0.21	0.19	0.21	0.22	0.29	0.28	0.41	0.35
5	0.09	0.07	0.19	0.15	0.14	0.21	0.32	0.33	0.29	0.41
4	0.07	0.02	0.16	0.13	0.03	0.12	0.31	0.31	0.19	0.43
3	0.21	0.19	0.17	0.29	0.11	0.19	0.27	0.29	0.11	0.21
2	0.09	0.09	0.21	0.31	0.19	0.21	0.21	0.19	0.02	0.16
1	0.01	0.08	0.18	0.32	0.21	0.23	0.23	0.22	0.15	0.18
Y / X	1	2	3	4	5	6	7	8	9	10

Table 3.1: Datasheet for Optimization Algorithms Questions

4. Constraint Satisfaction Problem

(10 points)

StarnTech has announced a soccer fantasy league, where students can submit a fantasy team. A fantasy team is a hypothetical team consisting of real-world players. This fantasy team gets scores based on the individual performances of each player when they play actual soccer games. Being a soccer enthusiast, you are very excited and start preparing for the competition. The rules of the game are:

- Each participant's team should have exactly 5 different players.
- The players must be selected from the roster available.
- The total budget available for the team is $120. You can spend less than the budget but cannot go over it.
- There must be exactly one goalkeeper in the team.
- There has to be at least one defender, one midfielder and one forward in the team.
- The fifth player, henceforth referred to as the Star Player, could be a defender, midfielder or a forward.
- There cannot be more than 2 players that belong to the same real-world club. For example, you cannot choose G1, D1 and M1 to be in your team together since all three of them belong to Arsenal.

You set out to form your best team. You start off selecting your goalkeeper, defender, midfielder, and striker in that order. And finally, you select the star player. You start with modeling this as a Constraint Satisfaction Problem.

- Each player position is a variable - G, D, M, F and S referring to Goalkeeper, Defender, Midfielder, Forward and Star Player. Let the team be denoted by T = {G,D,M,F,S}. See the available players in the tables below to assign values to these variables.
- The domain for each variable at any point of time consists of the set of allowable values for a variable. For example, at the onset when no players have been assigned, the domain of G is {G1, G2, G3, G4, G5}.

Q4.a. Select **all correct statements** below regarding the setup of this problem. **(2 points)**

- ☐ When no players have been assigned to any variable, the domain of M is {D1, D2, M3, F4, D5}.
- ☐ If f(X) denotes the club of the player X, $\forall X, Y, Z \varepsilon\, T;\ X \neq Y,\ X \neq Z,\ Y \neq Z$, if $f(X) == f(Y) \Rightarrow f(Z) \neq f(X)$ is a constraint for this problem.
- ☐ $\exists X, Y \varepsilon\, T\ s.t.\ X == Y$ is a constraint for this problem.
- ☐ If G and D have been assigned the values G5 and D5 respectively, and no other constraints have been applied, then the domain of M is {M1, M2, M3, M4}.

You are given a roster of 20 players from which you select your team. There are 4 attributes given for every player - their club, their cost, their overall rating and current form. Overall rating is a comparison metric used to compare the overall skill set of different players based on their entire playing history. The current form is a metric used to compare players based on their recent performances. There are four categories of players - Goalkeepers, Defenders, Midfielders and Forwards.

Goalkeepers:

Name	Club	Cost	Overall Rating	Current Form(out of 5)
G1	Arsenal	$15	84	3.4
G2	Liverpool	$23	89	2.9
G3	Manchester City	$25	88	4.2
G4	Manchester United	$28	89	4.9
G5	Tottenham Hotspurs	$19	88	3.8

Defenders:

Name	Club	Cost	Overall Rating	Current Form(out of 5)
D1	Arsenal	$19	82	4.1
D2	Liverpool	$29	90	4.3
D3	Manchester City	$22	87	4.0
D4	Manchester United	$27	83	4.8
D5	Tottenham Hotspurs	$20	87	4.0

Midfielders:

Name	Club	Cost	Overall Rating	Current Form(out of 5)
M1	Arsenal	$23	83	3.5
M2	Liverpool	$17	84	1.8
M3	Manchester City	$28	91	5.0
M4	Manchester United	$24	85	3.2
M5	Tottenham Hotspurs	$26	83	4.8

Forwards:

Name	Club	Cost	Overall Rating	Current Form(out of 5)
F1	Arsenal	$30	88	4.7
F2	Liverpool	$28	90	4.3
F3	Manchester City	$23	89	3.1
F4	Manchester United	$25	83	4.8
F5	Tottenham Hotspurs	$20	89	4.3

Selecting the Team

An expert of fantasy soccer games shares her secret about doing well in these competitions. According to her, teams having players with higher overall rating perform better than others. She uses **Backtracking Search** to find a team so that there aren't any rule violations. She also uses a secret heuristic and a secret inference technique to make the search easier. These are defined here for your reference.

Backtracking Search is a depth-first search that chooses values for one variable at a time and backtracks when a variable has no legal values left to assign. The value you assign to a variable has to be consistent with the earlier assignments.

Secret Heuristic: The secret heuristic helps decide the order in which variables are chosen. You select the variables G, D, M, F and S in that order and assign values to each of them. The order in which values are assigned is as follows:

1. You select the player having the best overall rating
2. If it's a tie in step 1, you select the player with the best current form
3. If it's a tie in step 2, you select the player with lesser cost
4. If it's a tie in step 3, you select the player alphabetically & numerically. (Ex: select G1 before G2, and select D5 before G1).

Secret Inference: This entails that if a particular player 'A' assignment for a variable X violates the budget constraint, you would remove all the players having cost greater than or equal to player 'A' from the domain of X while Backtracking. Please note that this inference is to be applied only due to the violation of the budget constraint (either only budget constraint being violated or multiple constraints being violated due to an assignment with the budget constraint being one of them).

For example, if you observe that selecting the player D5 for variable D leads to a situation with no legal values for variable S, then you remove the values D2, D3 and D4 from the domain of D.

You run the Backtracking Search to select a team. Please answer questions Q2, Q3, and Q4 based on that. It is recommended to go over the questions once before you start with the search since these questions also ask you about some of the details of your search.

Q4.b. What was the domain size for selecting the variable S during the final step when a value is assigned to it? **(1 point)**

_____

Q4.c. How many distinct assignments do you end up making for the variable M during the search? **(1.5 points)**

_____

Q4.d. Fill in the entire team you end up selecting. **(2.5 points)**

Position	Player Selected
G	
D	
M	
F	
S	

Q4.e. Which of the following statements are true regarding Backtracking search? Mark all that apply. **(1.5 points)**

- ☐ Backtracking search analyses all the possible solutions and returns the best one out of them.
- ☐ Use of forward checking will always return the same solution as the case without forward checking.
- ☐ Minimum values remaining heuristic helps minimize the number of nodes in the search tree by pruning larger parts of the tree.

Q4.f. Which of the following statements are true with regards to Constraint Satisfaction Problems and Optimization Problems? Mark all that apply. **(1.5 points)**

- ☐ CSP solvers can be faster than state-space searchers since a CSP solver can eliminate large swatches of the search space.
- ☐ If we were to model this problem as a Genetic algorithm, a function mapping a team to the inverse of the sum of the overall ratings of the players in that team can be a candidate fitness function.
- ☐ If we were to model this problem as a Genetic algorithm, a function mapping a team to the sum of the squares of overall rating would be a candidate fitness function.

5. Probability

(12 points)

Language Modelling

Statistical language modeling enables us to capture regularities in language sequences for the purpose of improving the performance of various natural language applications. By and large, language modeling is the process of estimating the probability distribution of various linguistic units, such as words, sentences, and whole documents. Although language models were originally developed for the problem of speech recognition, they are still widely used in a variety of language technology applications such as machine translation, document classification, information retrieval, handwriting recognition, spelling correction, and many more.

More formally, *a language model consists of a finite set of words* ($w_1, w_2, \ldots w_n$) *also termed as a vocabulary* (**V**) *and a probability function* $P(w_1, w_2, \ldots w_n)$ *such that:*

1. *For any sequence of words* $W = \{ w_1, w_2, \ldots w_n \} \in V$, $P(w_1, w_2, \ldots w_n) \geq 0$
2. $\sum_{\{ w_1, w_2, \ldots w_n \} \in V} P(w_1, w_2, \ldots w_n) = 1$

Hence $P(w_1, w_2, \ldots w_n)$ *is a probability distribution over the sentences in V.*

In addition to this, probability of a sequence of $W = \{ w_1, w_2, \ldots w_n \}$, with the chain rule, can be estimated as the product of probabilities of each word given the history, as shown:

$$\begin{aligned} P(\mathbf{W}) &= P(w_1, w_2 \ldots w_n) \\ &= P(w_1)\ P(w_2|w_1)\ P(w_3|w_1, w_2) \ldots P(w_n|w_1, w_2 \ldots w_{n-1}) \\ &= \prod_{i=1}^{n} P(w_i|w_1^{i-1}) \end{aligned}$$

For example,

P("Stay home stay safe") = P(Stay) x P(home | Stay) x P(stay | Stay home) x P(safe | Stay home stay)

where, $P(\text{safe} \mid \text{Stay home stay}) = \dfrac{\text{count(Stay home stay safe)}}{\text{count(Stay home stay)}}$

This means the number of times the words “Stay home stay safe” occur in sequence in the vocabulary divided by the number of times the words “Stay home stay” occur in sequence in the vocabulary.

N-Gram Models

However the above model performs poorly. In particular, it will assign a probability of 0 to any sentence not seen in the training corpus. Thus it fails to generalize to sentences that have not been seen in the vocabulary.

For this reason, we’ll need to introduce smarter ways of estimating the probability of a word *w* given a history *h*, or the probability of an entire word sequence *W*.

Enter Markov models! Markov models are the class of probabilistic models that assume we can predict the probability of some future unit without looking too far into the past. Thus the intuition behind an n-gram model is that instead of computing the probability of a word given its entire history, we can approximate the history with just the last few words.

The bi-gram model, for instance, approximates the probability of a word given all the previous words by using only the conditional probability of only the preceding word.

Thus, *P(“Stay home stay safe”) = P(Stay | BOS) x P(home | Stay) x P(stay | home) x P(safe | stay) x P(EOS | safe)*

Similarly, we can generalize the bi-gram (which looks one word into the past) to the tri-gram (which looks two words into the past) and thus to the n-gram (which looks at $n - 1$ words into the past).

NOTE:

BOS (Beginning of a sentence) and *EOS* (End of a sentence) are special tokens added in the n-gram for ease of calculation. For instance, in a bi-gram calculation the sentence will be converted to: *BOS Stay home stay safe EOS* and in the case of tri-gram calculation it will be: *BOS BOS Stay home stay safe EOS EOS*. Also, with the addition of these tags, your vocabulary size will increase by 2. Please make use of these tags for **ALL** the questions in the Probability section.

Smart Compose for Piazza

Now let us build a simple system using the concepts learned above for generating real-time interactive suggestions in Piazza that can assist other students in CS6601 for reducing their repetitive typing efforts. After carrying out a quick data scrape on the most asked questions on Piazza, you have created the below vocabulary:

1. **unit tests are failing**
2. **unit tests are passing but gradescope is failing**
3. **the gradescope submission does not terminate**
4. **please cancel my gradescope submission**
5. **all the unit tests are passing**
6. **unit tests pass with warning**
7. **all the unit tests are failing**
8. **please resolve issues with unit tests**
9. **the unit tests are not running**
10. **unit tests are taking forever**
11. **my gradescope submission is running forever**
12. **the unit tests are running into issues**
13. **failure in unit tests**
14. **unexpected failure in gradescope submissions**
15. **please terminate my gradescope submission**
16. **however the tests are failing**
17. **running into issues with the unit tests**
18. **the unit tests are buggy**
19. **my unit tests are terminating with errors**
20. **bug in the gradescope tests**
21. **the unit tests are terminating with errors**
22. **gradescope submission is terminating with errors**
23. **unit tests are not terminating**
24. **the tests are passing but gradescope is failing**

WHEN YOU TRAIN PREDICTIVE MODELS ON INPUT FROM YOUR USERS, IT CAN LEAK INFORMATION IN UNEXPECTED WAYS.

Q5.1.a. A student types in this query on Piazza: ***"the unit tests are"***
Based on the bi-gram model, which one of the following is the most probable next word that your system will predict? **(1.5 points)**

- terminating
- failing
- passing
- not
- running

Q5.1.b. What was the maximal probability for the above-predicted word? **(1.5 points)**

_____

Q5.1.c. If your system is trained on a tri-gram model what is the probability of the below query:
"please terminate the unit tests" *?* **(2 points)**

_____

Smoothing

Congratulations, you have now built a basic language model! However, your model will assign a probability of 0 for words that exist in the vocabulary but appear in the test query in an unseen context. To prevent this, you will have to readjust the probability estimates of the more frequent events such that you distribute some probability mass to these unseen events. This modification is called smoothing. The simplest way to do smoothing is to add one to all the n-gram counts before we normalize them into probabilities. All the counts that used to be zero will now have a count of 1, the counts of 1 will be 2, and so on. This algorithm is called **Laplace (add one) smoothing**. The bi-gram probability is therefore given by:

$$P(w_n \mid w_{n-1}) = \frac{\text{count}(w_{n-1}\, w_n) + 1}{\text{count}(w_{n-1}) + |V|}$$

where $|V|$ is the size of the vocabulary

As you can see we will need to adjust the denominator to take into account the extra *V* observations. It is also important to note that the size of the vocabulary is given by the number of unique words present in the given vocabulary.

Q5.2. What is the probability of the below query that uses the Laplace smoothing technique with the bi-gram language model?

"terminating my gradescope submission is failing" (2 points)

Perplexity

Now that you have built your own language model, let us go ahead and evaluate it! Your evaluation method should verify whether your language model assigns a higher probability to grammatically correct and frequent sentences than those sentences which are rarely encountered or have some grammatical error. However, the probability of a sentence tends to get extremely small with larger vocabulary size. Therefore, we employ a commonly used metric called perplexity, which is defined as the multiplicative inverse probability of the test set normalized by the number of unique words in the test set. Intuitively, higher probability corresponds to lower perplexity and thereby a better language model.
More formally, perplexity can be defined in the following way:

Consider test sentence S : { w_1, w_2, ..., w_N } ;
N = total number of words in sentence S

$$\text{Perplexity(S)} = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

NOTE: For the questions below please use the Laplace smoothing defined above.

Q5.3.a. Given a bi-gram model trained on the same initial vocabulary, how would you compare the perplexity of the model on the sentence ***"unexpected failure in unit tests"*** with respect to the perplexity of the same model on ***"my gradescope submission is failing"*** ? **(2 points)**

- ☐ Perplexity on "***unexpected failure in unit tests***" is greater
- ☐ The perplexities are equal
- ☐ Perplexity on ***"my gradescope submission is failing"*** is greater

Q5.3.b. Fill in the above-calculated perplexity values**: (3 points)**

"***unexpected failure in unit tests***" : _____

"my gradescope submission is failing" : _____

To learn more about Gmail's Smart compose refer to the paper below by Google Research: https://arxiv.org/pdf/1906.00080.pdf

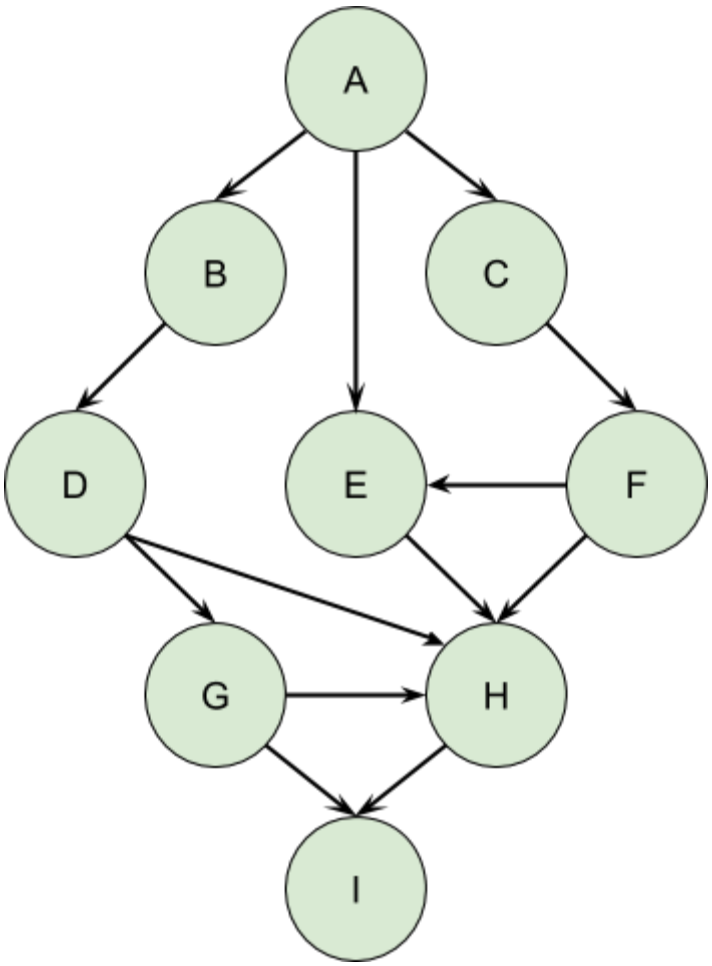
6. Bayes Nets

(10 points)

Question 1: D-Separation

In this question, we use D-separation to determine conditional [in]dependence among nodes, given some evidence variables. Read each question carefully (the information being asked for may vary per question) before answering.

Consider the following Bayes Net, represented by a Directed Acyclic Graph.



Using the above Bayes Net, answer questions 1 - 5, which concern conditional [in]dependence among nodes. Remember to use D-Separation Techniques!

Q6.1.a. Consider the claim: “Nodes A and I are conditionally independent given a set of nodes {X,Y, …, Z}”. For which of the following sets of nodes is the claim true? Choose all that apply. **(1 point)**

- ☐ G,H
- ☐ D,F
- ☐ G,E,C
- ☐ D,H
- ☐ B,E,C

Q6.1.b. Consider the claim: “Nodes A and F are conditionally independent given a set of nodes {X,Y, …, Z}”. For which of the following sets of nodes is the claim true? Choose all that apply. **(1 point)**

- ☐ C,H
- ☐ C,G
- ☐ C,E
- ☐ C,I
- ☐ C

Q6.1.c. Consider the claim: “Nodes A and H are conditionally independent given a set of nodes {X,Y, …, Z}”. For which of the following sets of nodes is the claim true? Choose all that apply. **(1 point)**

- ☐ E
- ☐ E,F
- ☐ E,F,D
- ☐ G,F
- ☐ E,F,B

Q6.1.d. Given the nodes I and D as evidence, which of the following pairs of nodes are **conditionally dependent** on one another? Choose all that apply. **(1 point)**

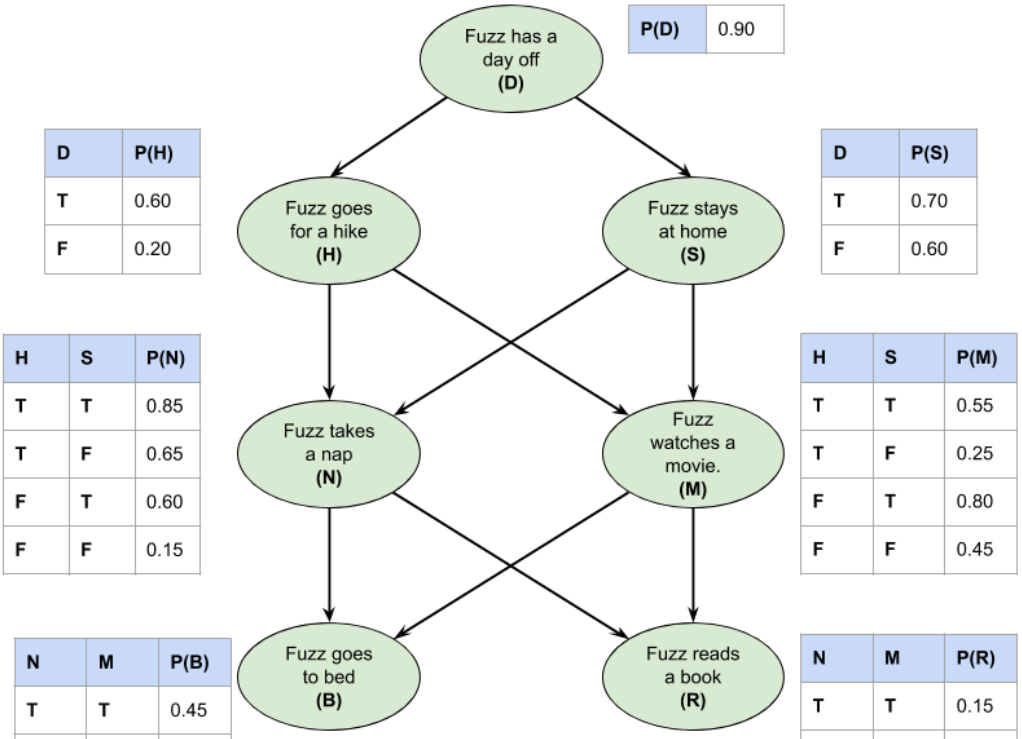
- ☐ E,F
- ☐ G,H
- ☐ B,G
- ☐ B,E
- ☐ C,G

Q6.1.e. Given the nodes B and C as evidence, which of the following pairs of nodes are **conditionally independent** of each other? Choose all that apply. **(1 point)**

- ☐ A,D
- ☐ A,I
- ☐ D,F
- ☐ G,F
- ☐ D,E

Question 2: Computation

In this question, we pose a scenario with multiple possible events and ask you to compute the probability of an outcome given some evidence. The scenario is as follows: Fuzz may have a day off in the future, and she can plan to do several things. We trace out the possibilities using a Bayesian Network.



P(D)	0.90

D	P(H)
T	0.60
F	0.20

D	P(S)
T	0.70
F	0.60

H	S	P(N)
T	T	0.85
T	F	0.65
F	T	0.60
F	F	0.15

H	S	P(M)
T	T	0.55
T	F	0.25
F	T	0.80
F	F	0.45

N	M	P(B)
T	T	0.45
T	F	0.35
F	T	0.75
F	F	0.90

N	M	P(R)
T	T	0.15
T	F	0.80
F	T	0.50
F	F	0.95

Using the Bayes Net, answer the following questions. Please show your work.

Q6.2.a. What is the probability that Fuzz reads a book **(R)** given that she goes for a hike **(H)** and has a day off **(D)**? **(2 points)**

Please use space below to show your work:

Q6.2.b. What is the probability that Fuzz stays at home **(S)** given that she goes to bed **(B)** and watches a movie **(M)**? **(3 points)**

Please use space below to show your work:

7. Machine Learning

(12 points)

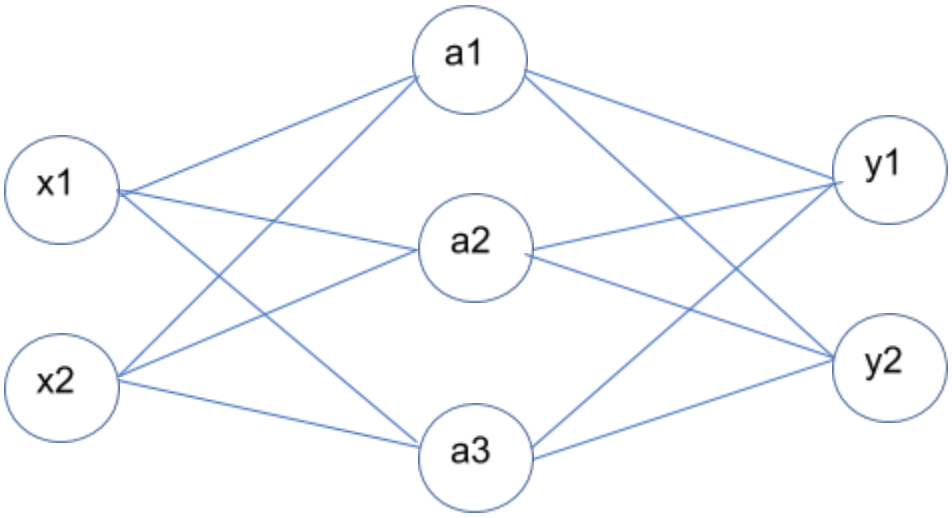
Question 1:

With this question, we will look at backpropagation, a technique used by neural networks to try and adjust their weights to better predict the data that they are given. Think of backpropagation as "learning from error". We will guide you through computing one iteration of backpropagation and also try to give some explanation as to what the algorithm is actually trying to do.

Q7.1.a. Before we start doing any computations, let's get warmed up a bit first. One of the common activation functions that are used in neural networks is the sigmoid function: $\frac{1}{1+e^{-z}} = \sigma(z)$. What is the derivative of $\sigma(z)$? **(1 points)**

- $\sigma'(z) = \sigma(z)^{-1} * e^{-z}$
- $\sigma'(z) = \sigma(z) * (1- \sigma(z))$
- $\sigma'(z) = \sigma(z) * (\sigma(z)-1)$
- $\sigma'(z) = \sigma(z)$

Now consider the following network:



Input Layer **Hidden Layer** **Output Layer**

Here, the sigmoid function activation is applied to the output of the hidden layer. Every $a_i = \sigma(z)$ where $z = \sum w_i * x_i$. **For the output layer the activation function is linear (y = x where x is the input to the neuron and y is the output of the neuron)**.

For the remaining questions, we will be using the following notation:

$$\mathbf{W}^{(0)} = \begin{bmatrix} w_{11}^{(0)} & w_{12}^{(0)} & w_{13}^{(0)} \\ w_{21}^{(0)} & w_{22}^{(0)} & w_{23}^{(0)} \end{bmatrix} \qquad \mathbf{W}^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \hat{\mathbf{Y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Notice that something like $w_{11}^{(1)}$ refers to the weight of the connection between the first node in the hidden layer and first node in the output layer. In addition, let $E_{out} = 0.5 * || \widehat{Y} - Y ||_2^2$ be the error of the neural network and ∂ be the symbol for partial derivatives.

For the remainder of the question, we will be using the following values. When it comes to rounding, only round when filling in your answers. However, you should use the rounded values from previous problems in subsequent ones (i.e. using the solution to #3 in #4). **Be careful though, as no partial credit will be given.** In addition, **make sure to fill in the boxes with numerical answers**:

$$\mathbf{X} = \begin{bmatrix} 7 \\ 2 \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} 8 \\ 3 \end{bmatrix}$$

$$\mathbf{W}^{(0)} = \begin{bmatrix} 0.11 & 0.73 & 0.72 \\ 0.09 & 0.75 & 0.79 \end{bmatrix} \qquad \mathbf{W}^{(1)} = \begin{bmatrix} 0.80 & 0.38 \\ 0.72 & 0.66 \\ 0.23 & 0.87 \end{bmatrix}$$

Q7.1.b. Before doing anything backpropagation-related, let's first see what the forward pass would look like through a neural network. Compute the values of **Ŷ**. Use the note below to learn how to perform a forward pass **(1 points)**:

Let **x** be the input vector of size 2x1, W_0 be the weights of the first hidden layer of size 2x3 and W_1 be the weights of the second hidden layer of size 3x2, then we perform the forward pass in the following way:

$$\mathbf{a} = \mathbf{W}_0^T \cdot \mathbf{x}$$

$$\mathbf{h} = \sigma(a)$$

$$\mathbf{y} = \mathbf{W}_1^T \cdot \mathbf{h}$$

$$\widehat{Y} = \mathbf{y}$$

$\hat{y}_1$ =
$\hat{y}_2$ =

Q7.1.c. With this done, what we need to do is learn with respect to the error. On that note, calculate the following derivatives **(remember we want only numerical answers in the boxes) (1 points)**:

$\frac{\partial E_{Out}}{\partial \hat{y}_1}$ =
$\frac{\partial E_{Out}}{\partial \hat{y}_2}$ =

Q7.1.d. The key to backprop is that we learn based on the error that we get. So what we want to do is learn how the parameters of the network that we tune contribute to the error of a network. Suppose we wanted to know how an arbitrary weight, w_x, has an impact on our overall error method. We can do so with the following equation (known as a Chain rule) **(1.5 points)**:

$$\frac{\partial E_{Out}}{\partial w_x} = \frac{\partial E_{Out}}{\partial a(z)} * \frac{\partial a(z)}{\partial z} * \frac{\partial z}{\partial w_x}$$

where a(z) is the activation function of the output node(s) (in our case, it is linear). Given that we have the above, compute the values of the above derivative for six of the network weights:

$\frac{\partial E_{Out}}{\partial w_{11}^{(1)}} =$	$\frac{\partial E_{Out}}{\partial w_{12}^{(1)}} =$
$\frac{\partial E_{Out}}{\partial w_{21}^{(1)}} =$	$\frac{\partial E_{Out}}{\partial w_{22}^{(1)}} =$
$\frac{\partial E_{Out}}{\partial w_{31}^{(1)}} =$	$\frac{\partial E_{Out}}{\partial w_{32}^{(1)}} =$

Q7.1.e. Now we have what we need to re-compute the new weights for backpropagation. Let the learning rate be 0.3. Think about how we want to update these weights. If we were to graph the error as a function of the weights, how would we want to move along that error surface to come up with better weights? This gives us the following weight update formula for w_x:

$$w_x = w_x - \alpha \frac{\partial E_{out}}{\partial w_x}$$

where ***α*** is the learning rate

Provide the new weights after a single gradient step update **(1.5 points)**:

$w_{11}^{(1)}$ =	$w_{12}^{(1)}$ =
$w_{21}^{(1)}$ =	$w_{22}^{(1)}$ =
$w_{31}^{(1)}$ =	$w_{32}^{(1)}$ =

Q7.1.f. At this point, you should have a good idea how backprop works, so why don't you go ahead and try applying it on to the next set of weights (i.e. between the hidden layer and input layer)? Fill in all your answers below **(3 points)**:

$\frac{\partial E_{Out}}{\partial w_{11}^{(0)}} =$	$\frac{\partial E_{Out}}{\partial w_{12}^{(0)}} =$	$\frac{\partial E_{Out}}{\partial w_{13}^{(0)}} =$
$\frac{\partial E_{Out}}{\partial w_{21}^{(0)}} =$	$\frac{\partial E_{Out}}{\partial w_{22}^{(0)}} =$	$\frac{\partial E_{Out}}{\partial w_{23}^{(0)}} =$

$w_{11}^{(0)} =$	$w_{12}^{(0)} =$	$w_{13}^{(0)} =$
$w_{21}^{(0)} =$	$w_{22}^{(0)} =$	$w_{23}^{(0)} =$

Congratulations! You have now completed a single round of backpropagation. Hopefully you learned a little more about what goes on when a neural network is trying to learn a function, and now you can start exploring some of the various types of neural networks out there with a little more intuition as to how the internals work.

Question 2.

Q7.2.a. One of the algorithms you were asked to implement in assignment 5 is the K-means algorithm. It is a fairly straightforward algorithm, but it has its weaknesses, and you may be wondering whether there is a better algorithm out there. One such algorithm that tries to address the K-means' weaknesses is called the K-medoids algorithm. We can think of medoids as analogous to cluster centers in the original k-means algorithm. The only difference is that these medoids must be members of the original data set. The algorithm works as follows:

a. Initialize: greedily select k of the n data points as the medoids to minimize the cost
b. Associate each data point to the closest medoid.
c. While the cost of the configuration decreases:
    i. For each medoid m, and for each non-medoid data point o:
        1. Consider the swap of m and o, and compute the cost change (the cost of a cluster is defined as the pairwise distance between the medoid m and all of its cluster's non-medoid data points)
        2. If the cost change is the current best, remember this m and o combination
    ii. Perform the best swap of m_{best} and o_{best} if it decreases the cost function. Otherwise, the algorithm terminates.

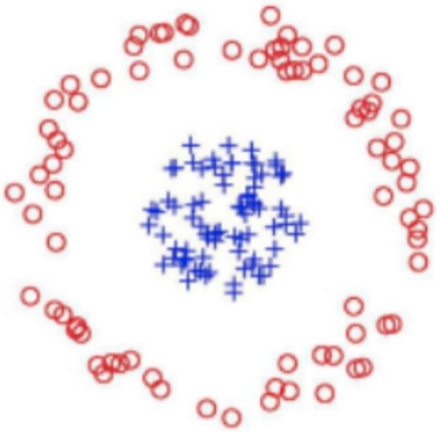
There are other variations of this algorithm, but the above is an approach known as PAM (partitioning around medoids).

With the above information, which one of the options below best addresses why one may prefer the K-medoids algorithm over the K-means algorithm? Consider only the given K-medoids algorithm **(1 points)**:

- ◯ K-medoids always finds the minimum cost cluster assignment.
- ◯ K-medoids is more robust to outliers
- ◯ K-medoids initialization method will guarantee faster convergence
- ◯ None. That is, K-medoids changes how clusters are defined and assigned among points, but that does not help in addressing any weakness of K-means.

Q7.2. The Support Vector Machine (SVM) uses the kernel trick to try and project data into a higher dimension in order to make the data linearly separable. A lot of times, this detail is hidden away by the model that is doing the projection, so let's see if we can get a visual representation of what the kernel trick is doing by trying to be a kernel ourselves.

Suppose we have the following data:



Q7.2.a. In what way can we project the data so that an SVM can classify the data perfectly? That is, suppose we project the data to the k^{th} dimension to get the SVM to classify the data linearly. What could the data look like in this k^{th} dimension (ignore the number of points in the answers, we just want the shape of the data)? **(1 points)**

- ◯'s and +'s on opposite sides? (example below)
  ◯◯◯◯◯◯◯◯◯◯◯◯◯
  ++++++++++++
- Alternating concentric circles, where we have all ◯'s in the innermost circle, then +'s in the circle outside that, then ◯'s outside that, etc.
- No projection of the data is needed. The data can be separated as is.
- ◯'s and +'s alternate but line up with each other? (example below)
  ◯+◯+◯+◯+◯+◯+
  +◯+◯+◯+◯+◯+◯

Q7.2.b. Which of the following properties are true regarding SVMs (select all that apply) **(1 points)**:

- ☐ These models create a linear-separating hyperplane to classify data
- ☐ SVMs are parametric
- ☐ These models minimize the distance between support vectors and the decision boundary
- ☐ On the training data, SVMs attempt to minimize expected generalization loss as opposed to expected empirical loss

8. Logic and Planning

(8 points)

Question 1: Logic

Consider the following:

F: (A ⇒ D) ∨ (B ⇒ D) ∨ (C ⇒ D) ∧ E
G: (A ∧ B ∧ C) ⇒ D

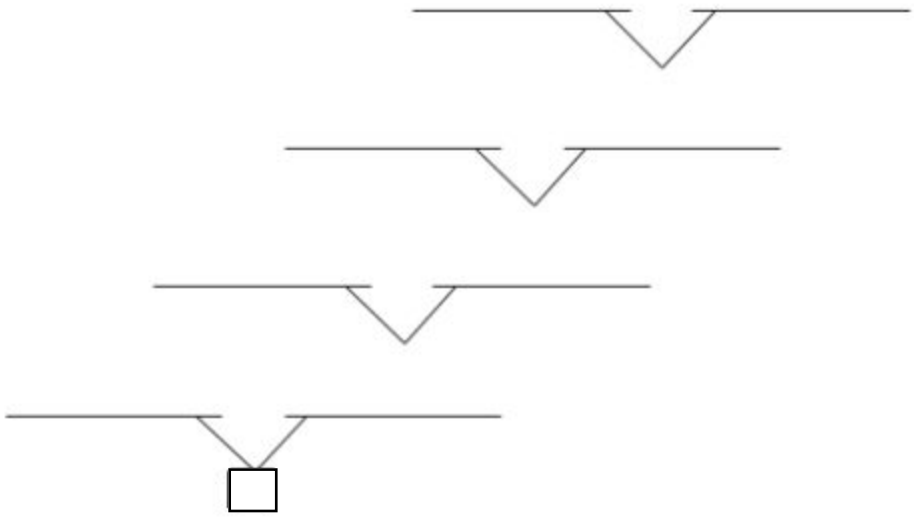
Q8.1.a. Write out the conjunctive normal form of **F (1 points)**:

_____

Q8.1.b. Write out the conjunctive normal form of **G (1 points)**:

_____

Q8.1.c. Complete the following resolution tree with your unification pairs in order to attempt to show **F ⇒ G** is either valid or unsatisfiable. You may not have to complete the entire tree:



F ⇒ G is **(1 points)**:

- ◯ Valid
- ◯ Unsatisfiable

Q8.1.d. Which of the following sets of clauses are satisfiable (here ¬ is “Not” operator) **(2 points)**:

- ☐ {(a, b),(c,¬e),(¬b, c),(¬d, ¬a),(¬a, ¬b),(¬c, ¬d),(d, e),(¬c, b)}
- ☐ {(x), (¬x, y), (¬x, ¬z), (z, ¬x, ¬y)}
- ☐ {(a, b), (¬a, c), (¬a, ¬c), (a, ¬b)}
- ☐ {(y), (¬x, y), (y, ¬z), (x, z)}
- ☐ {(c, b, ¬d), (¬c, e, f), (d, ¬e), (d, ¬f), (a, ¬b, c), (¬a, f)}

Question 2: Planning

In the middle of the night, you get a call from a doctor who is running low on protective gloves. There are 3 patients, P1, P2, and P3; and 2 sets of gloves, G1 and G2 and the doctor needs to use one set (one pair of gloves) for each patient. A glove Gx has two sides, Gxa and Gxb, and can be inverted Invert(Gxx) from one side to the other. The doctor can Wear(Gxx) gloves facing in either direction and may stack gloves on top of each other. The doctor can Remove(Gxx) gloves. The doctor always operates with the outermost side of the topmost glove worn. The doctor does not want to contaminate herself, so she will never have a dirty glove against her own skin. The doctor would like to Treat(Px, Gxx) every patient, but Treating a patient contaminates the outermost facing side of the outermost glove being worn. Contaminated gloves contaminate surfaces with which they come into contact, and may not be used to treat a patient, but Remove(Gxx) and Invert(Gxx) do not contaminate anything. The doctor will treat the patients in numerical order, and should use gloves in numerical order.

For example, the sequence “Wear glove G2a and treat patient P2, then invert the glove” would be: `Wear(G2a); Treat(P2, G2a); Invert(G2a);`

Q8.2.a. What is an action sequence that moves us from the initial state to the goal state? **(3 points)**:

_____

9. Pattern Recognition Through Time

(12 points)

You have recently joined a Tech Startup and have been matched to a team that is building a new voice assistant. Fortunately, you took the CS6601 class and have a couple of tools in your arsenal to ace this task!

Suppose you are given a recorded audio sample of a user saying "`Georgia Tech`" and your task is to transcribe it into text for further processing by the voice assistant. Additionally, you are given a database of common English words and phrases and their corresponding audio recording. How would you quantify the similarity between the audio sample you are given and the corresponding audio recording present in your database?

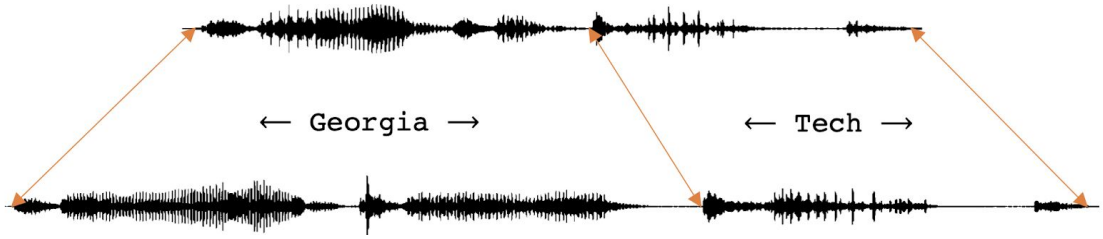


Figure 1: Matching of User Recording (top) and Database Recording (bottom)

You recall that one way to approach this is to create distance measures based on time series analysis, and then perform Dynamic Time Warping (DTW), one of the most widely used algorithms that measure similarities between two time series. It allows for more flexibility than the Euclidean distance metric, and is used on data that has a large amount of variance. See the figure below for a visual depiction of how these two algorithms compare two similar time-series by comparing individual data points. Note that the Dynamic Time Warping algorithm matches data points that we would consider to be most suitable for comparison, while the Euclidean matching algorithm is a bit more naive.

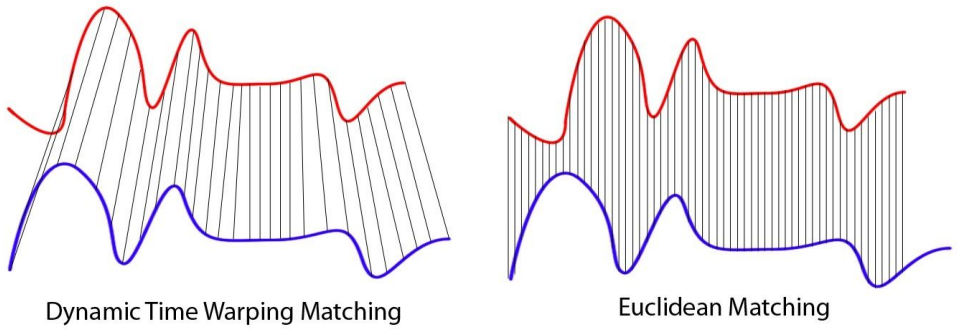


Figure 2: Dynamic Time Warping vs Euclidean Matching

How is the DTW algorithm executed? We can perform the task of finding the distance between two time series by using a distance matrix, where every possible warping route between the two time series is a path through the matrix. The goal of the algorithm is to find a path with the lowest final distance that connects the bottom-left square to the top-right square.

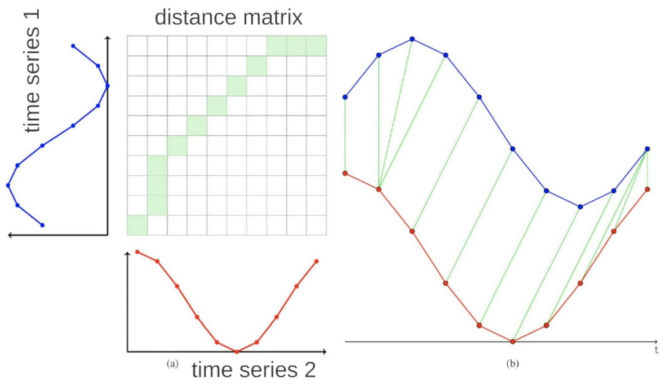


Figure 3: Example of a warping path

Each square $\gamma(i, j)$ in the grid is computed by the following formula:

$$\gamma(i,j) = |A_i - B_j| + \min\Big(\gamma(i-1,j), \gamma(i,j-1), \gamma(i-1,j-1)\Big)$$

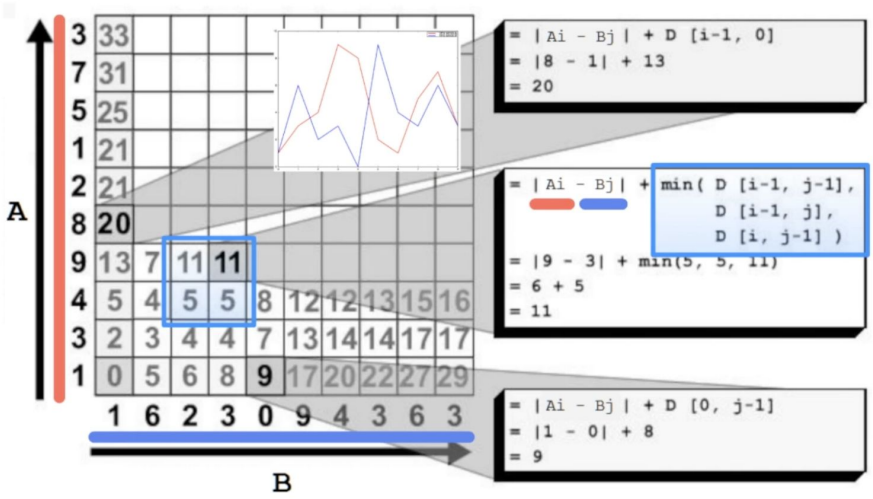


Figure 4: Computing the DTW matrix between the time series A and B

To find which data points have been matched together you can work the algorithm backward. Determine the **shortest warping path** by starting from the (top-right) end point and move towards the (bottom-left) starting point, following each neighbor that you used to calculate the current element. At each step, you can either move diagonally, left, or down. In the case of a tie with the same values, you should move diagonally. The shortest warping path for this example is `0, 3, 4, 4, 8, 8, 9, 11, 13, 14, 15, 15.` The **total distance** between two time series is the value of the top-right square. In this example, the total distance between the time series A and B is `15`.

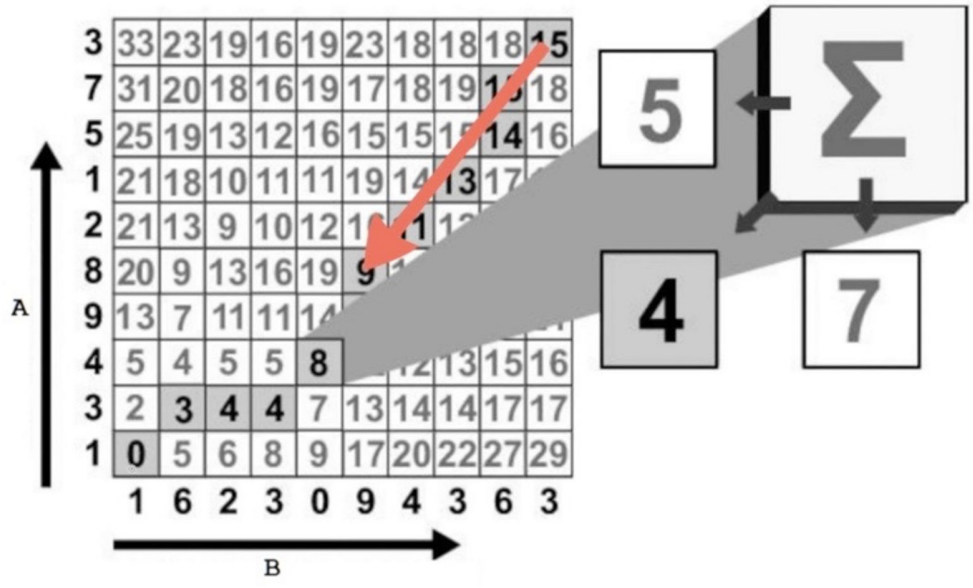


Figure 5: Determining the shortest warping path between the time series A and B

Question A. Dynamic Time Warping

You are given a recording of an unknown word and you want to determine which word it is.

Time series of the unknown word in the recording:

`[7, 15, 15, 18, 10, 10, 10, 1, 1]`

You then run an efficient script developed by your colleague that takes in as input the supplied recording file, searches the entire database, and returns the top 2 labeled audio samples that are similar to it. After execution, this script returned the audio samples for the words "`George`" and "`Georgia`".

Time series of the word "George" from the database:

`[7, 5, 15, 18, 18, 7, 7, 3]`

Time series of the word "Georgia" from the database:

`[7, 5, 15, 18, 18, 7, 7, 9, 1]`

TASK 1. Compute the total distance and the shortest warping path between the unknown recording and the word "George".

Q9.A.1.a. What is the total distance between both time series? **(1 points)**

Q9.A.1.b. What is the shortest warping path between both time series? **(2 points)**

Answer Format: `0, 3, 4, 4, 8, 8, 9, 11, 13, 14, 15, 15`

1								
1								
10								
10								
10								
18								
15								
15								
7								
	7	5	15	18	18	7	7	3

TASK 2. Compute the total distance and the shortest warping path between the unknown recording and the word "`Georgia`".

Q9.A.2.a. What is the total distance between both time series? **(1 points)**

Q9.A.2.b. What is the shortest warping path between both time series? **(2 points)**

Answer Format: `0, 3, 4, 4, 8, 8, 9, 11, 13, 14, 15, 15`

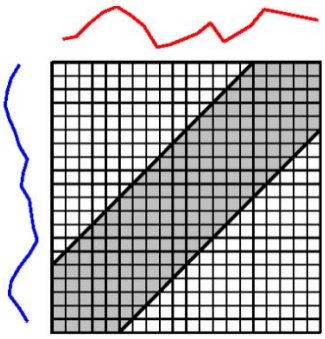
1									
1									
10									
10									
10									
18									
15									
15									
7									
	7	5	15	18	18	7	7	9	1

TASK 3. Determine the text contained in the unknown audio recording.

Q9.A.3. Now that you have used DTW and computed the distance between the unknown audio recording and both words "`George`" and "`Georgia`", you were able to determine that the unknown recording corresponds to the word with the **lowest total distance**. Which word is it? **(1.5 points)**

- ◯ `George`
- ◯ `Georgia`

Question B. Sakoe-Chiba Bands



This naive implementation of DTW has a time complexity of $O(N \cdot M)$ and a space complexity of $O(N \cdot M)$, where N and M are the lengths of the two time sequences. This is one of the most cited reasons to not use DTW. Based on your results from Question A, you should observe that a lot of grids, mostly grids around the bottom-right and top-left corners, were left unused. We can exploit this observation to reduce unnecessary computation using **Sakoe-Chiba bands**. In this question, you are going to compute the distance between two time series with different Sakoe-Chiba band widths, where only the values contained within the bands are used to compute DTW, and those outside of the bands are ignored.

Compute the total distance between the time series for the word "`Georgia`" and the unknown recording from Question A with different widths of Sakoe-Chiba bands. For each of the following questions, the matrix has been provided to you with some grayed-out regions. These grayed-out grids are outside of the width of the Sakoe-Chiba bands and therefore these values are not taken into consideration while performing DTW.

Q9.B.1. What is the total distance using Sakoe-Chiba bands of degree $N = 0$? **(1.5 points)**

1									
1									
10									
10									
10									
18									
15									
15									
7									
	7	5	15	18	18	7	7	9	1

Q9.B.2. What is the total distance using Sakoe-Chiba bands of degree $N = 1$? **(1.5 points)**

1									
1									
10									
10									
10									
18									
15									
15									
7									
	7	5	15	18	18	7	7	9	1

The purpose of this question is to show you how important the width of the Sakoe-Chiba bands is. When the width is zero (without any warping), you are basically performing Euclidean Matching between the two sequences. With some warping allowed, we can reduce the runtime complexity of the DTW and yet obtain a reasonable calculation of the distance between two sequences. In addition, limiting the size of the bands limits local warping and can help reduce “unreasonable” matches.

Question C. Different Kinds of Bands

As we learnt in the previous question, lower bounding the DTW distance is helpful to reduce unreasonable matches. The two most common constraints in the literature to achieve this are the Sakoe-Chiba Band and the Itakura Parallelogram.

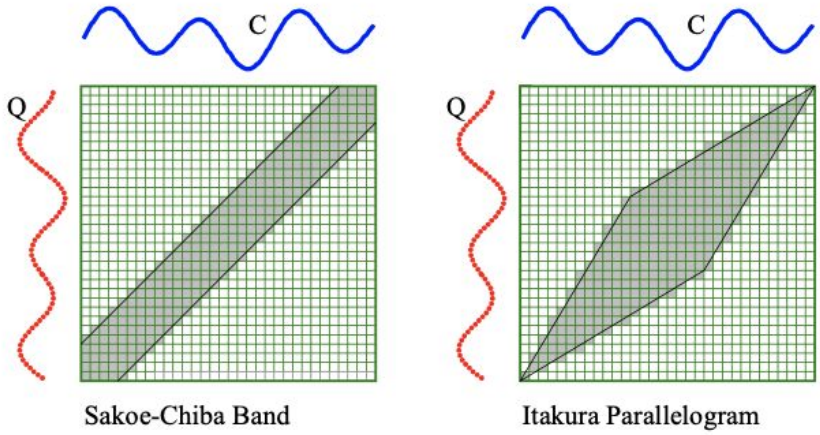


Figure 6: Different kinds of bands used in DTW

Speech generally tends to have the most variability in its middle and very little in the beginning or the end. For example, the word `mama` can have two sample pronunciations: `"maama"` and `"mamma"`. Consequently, choosing a good shape for the bands for the speech recognition task may help improve the classification accuracy.

Q9.C.1. Which of the two bands is more apt for the speech recognition task? **(1.5 points)**

- ◯ Sakoe-Chiba Band
- ◯ Itakura Parallelogram

10. Planning Under Uncertainty

(12 points)

Recommender systems are algorithms that are aimed at suggesting relevant items to users (e.g., the Youtube recommender suggesting videos to watch next). Typical recommender systems view the problem of generating recommendations as a prediction problem. However, we can also view it as a sequential decision problem. In many domains, user choices are sequential in nature. For example, we may buy a book written by the author of another book we recently liked. Thus, the recommendation is a sequential process at each point of which the system decides on some recommendations to issue. The system should also take into account the utility of a particular recommendation. For example, we might want to recommend a product whose immediate reward is lower but leads to more profitable rewards in the future. We know that a Markov Decision Process (MDP) is a stochastic decision model of sequential decisions, and the optimal policy generated for an MDP model automatically takes these considerations into account. With these views in mind, we can construct an MDP-based recommender system.

MDP-based recommender system

Let us examine an approach for the construction of an MDP-based recommender system for a website selling books. The system makes one book recommendation at a time, and the goal is to maximize book sales. Consider the following books for sale on the website.

Label	Title	Author	Genre	Price in USD
B1	If Tomorrow Comes	Sidney Sheldon	Thriller	10
B2	Tell Me Your Dreams	Sidney Sheldon	Thriller	12
B3	When Breath Becomes Air	Paul Kalanithi	Memoir	15

Table 10.1: Books for sale

Defining the MDP

Let's assume that we already have the book purchase history data and use it to define and initialize the MDP model.

States: The states of our MDP model represent the relevant information that we have about the user. This information corresponds to previous choices made by users in the form of a set of ordered sequences of purchases. Let's consider sequences of at most `3` books purchased to denote a state. For example, `<B1, B2, B3>` is used to denote the state in which the user's last `3` purchased books were `B1`, `B2`, and `B3`. Sequences with `L < 3` books are transformed into a set in which the last $3-L$ books are marked as missing `(M)`. Table 10.2 shows the states identified for our MDP model. State `S1: <M, M, M>` denotes the start state of a new user.

State S0: <NP> is a terminal state which denotes that a current user of the system purchases no more books.

Label	Sequence
S0	<NP>
S1	<M, M, M>
S2	<M, M, B1>
S3	<M, M, B3>
S4	<M, B1, B2>
S5	<B1, B2, B3>
S6	<M, B3, B1>
S7	<B3, B1, B2>

Table 10.2: States

Actions: The actions of our MDP correspond to a recommendation of a book. The set of possible actions is {B1, B2, B3}.
Rewards: The rewards in our MDP encode the utilities of selling books. For example, the reward for state <B1, B2, B3> is the income from the sale of book B3 ($15) - the last item in the purchase sequence. Table 10.3 shows the states and their corresponding rewards.

State	Reward
S0	0
S1	0
S2	10
S3	15
S4	12
S5	15
S6	10
S7	12

Table 10.3: States and Rewards

Transition function: The transition function *(T)* assigns a probability distribution to every (state, action) pair. Thus, `T(s, a, s')` is the probability of making a transition from state `s` to state `s'` when action `a` is taken. The transition function for our MDP model from a state `<x1, x2, x3>` to state `<x2, x3, x">` is the probability that the user will select book `x"` given that book `x'` is recommended. That is, `T(<x1, x2, x3>, x', <x2, x3, x">)`. Figure 10.1 shows the MDP graph with states, actions, and transition function values.

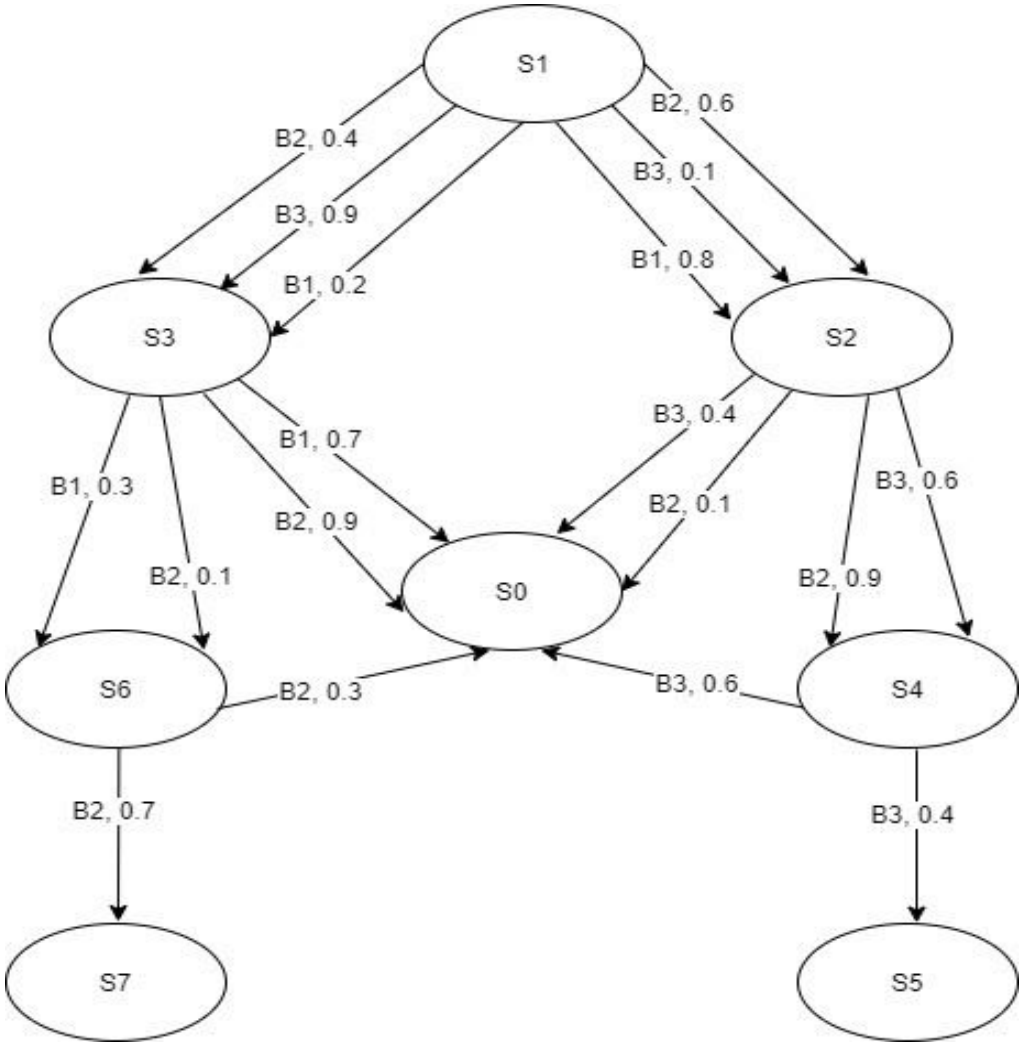


Figure 10.1: MDP Graph

States with no outgoing arrows are terminal states whereas the other states are non-terminal states. An arrow between two states represents action and transition function value. For example, on recommending book `B3` to a user in state `S1`, there is `0.9` probability that the user will buy `B3` and enter state `S3` and there is `0.1` probability that the user will instead buy book `B1` and enter state `S2`.

Solving the MDP

Now that we have formulated the MDP model for our book recommender system, we need to solve it to determine the best book to recommend for every possible state.

As a warm-up activity, perform one iteration of the value iteration with a discount factor of `1` to find the utility of the state `S1`. Consider the rewards shown in Table 10.3 as the initial utilities (i.e., values of iteration `0`).

Q10.a. What is the utility of state `S1` after one iteration of value iteration? **(1.5 points)**

_____

Let's proceed to find the optimal book recommendation policy using the modified policy iteration algorithm.

Iteration 1:

An iteration of the modified policy iteration algorithm has two steps.

Step 1. Policy evaluation

In the modified policy iteration algorithm, we estimate the utilities of the current policy by performing some simplified value iteration steps instead of performing the exact evaluation.

Q10.b. (4.5 points)
The initial policy for all the non-terminal states is given in the below table. Similar to the warm-up activity, consider rewards given in Table 10.3 as the initial utilities of states. Perform `2` steps of simplified value iteration with a discount factor of `1`. Report the estimated utilities in the below table.

State	Initial Policy	Utility
S1	B2	
S2	B3	
S3	B1	
S4	B3	
S6	B2	

Step 2. Policy Improvement

In this step, we calculate a new improved policy based on the utilities obtained in the previous step.

Q10.c. Fill the below table with the new policy obtained from the utilities calculated in **Q10.b. (2.5 points)**

State	Policy
S1	
S2	
S3	
S4	
S6	

Iteration 2:

Repeat the same steps performed in the previous iteration.

Q10.d. Evaluate the policy obtained in the previous iteration (**Q10.c.**). Perform 2 steps of simplified value iteration with a discount factor of 1 and with rewards defined in Table 10.3 as the initial utilities. Report the calculated utilities in the below table. **(2 points)**

State	Utility
S1	
S2	26.2
S3	
S4	
S6	

Q10.e. Calculate a new policy based on the utilities obtained in Question **Q10.d.** and report them in the below table. **(1.5 points)**

State	Policy
S1	
S2	B2
S3	
S4	
S6	

If you are interested in learning more details, check out the below paper after you have finished the question.

https://arxiv.org/ftp/arxiv/papers/1301/1301.0600.pdf

---

11. Extra Credit

(Up to 2 points)

Survey! We will be awarding extra credit to the entire class based on survey completion rates. <u>The credits will only be awarded if the survey submission rate exceeds 80%.</u>

The survey link will be released later next week on Canvas & Piazza.

The completion rate is defined as follows:

(number of students who completed the survey) / (number of students enrolled in the class)

The completion rate (up to 1.0) will be multiplied by 2 and added to the score of your final exam.

Please consider taking the survey! As we have said over the course of the semester, this course is constantly evolving and we are looking at ways of making it better. We take feedback very seriously and incorporate it into our teaching methods wherever possible.

Checklist

And now mark the checklist below making sure you have taken care of each of the points mentioned:

- ☐ I have read the pinned Piazza post with the title 'Final Exam Clarifications Thread', and I am familiar with all of the clarifications made by the Teaching staff.
- ☐ All answers with more than 6 digits after the decimal point have been rounded to 6 decimal places (unless otherwise instructed).
- ☐ **All pages** are being uploaded in the correct order that they were presented to me, and none of the pages are missing/removed.
- ☐ Any extra pages **(including blanks)** are only attached at the END of this exam, after this page with clear pointers to wherever the actual answer is in the PDF (reference properly).
- ☐ I am submitting only one PDF and nothing else (no docx, doc, etc.).
- ☐ The PDF I am submitting is not blank (unless I want it to be).
- ☐ **I will go over the uploaded pictures on Gradescope and make sure that all the answers are clearly visible.**
- ☐ I will submit a copy of the PDF to Canvas for backup purposes.
- ☐ **I acknowledge that dull / illegible / uneven scans and submission that don't follow the above guidelines will not be graded.**