CS 188 Fall 2017 — Introduction to Artificial Intelligence — Final Exam V1

- You have approximately 170 minutes.
- The exam is closed book, closed calculator, and closed notes except your three-page crib sheet.
- Mark your answers ON THE EXAM ITSELF. If you are not sure of your answer you may wish to provide a *brief* explanation. All short answer sections can be successfully answered in a few sentences AT MOST.
- For multiple choice questions:
  - □ means mark **all options** that apply
  - ○ means mark a **single choice**
  - When selecting an answer, please fill in the bubble or square **completely** (● and ■)

First name	
Last name	
SID	
Student to your right	
Student to your left	

Your Discussion/Exam Prep* TA (fill all that apply):

□ Brijen (Tu)	□ Aaron (W)	□ Aarash (W)	□ Shea* (W)
□ Peter (Tu)	□ Mitchell (W)	□ Daniel (W)	□ Daniel* (W)
□ David (Tu)	□ Abhishek (W)	□ Yuchen* (Tu)	
□ Nipun (Tu)	□ Caryn (W)	□ Andy* (Tu)	
□ Wenjing (Tu)	□ Anwar (W)	□ Nikita* (Tu)	

For staff use only:

Q1.	Agent Testing Today!	/1
Q2.	Potpourri	/14
Q3.	Search	/9
Q4.	CSPs	/8
Q5.	Game Trees	/9
Q6.	Something Fishy	/10
Q7.	Policy Evaluation	/8
Q8.	Bayes Nets: Inference	/8
Q9.	Decision Networks and VPI	/9
Q10.	Neural Networks: Representation	/15
Q11.	Backpropagation	/9
	Total	/100

THIS PAGE IS INTENTIONALLY LEFT BLANK

Q1. [1 pt] Agent Testing Today!

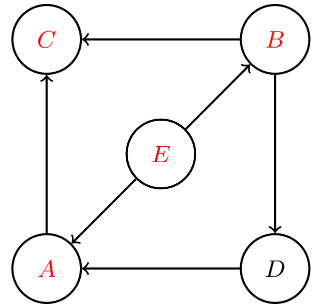
It's testing time! Not only for you, but for our CS188 robots as well! Circle your favorite robot below.

Any answer was acceptable.

Q2. [14 pts] Potpourri

(a) [1 pt] Fill in the unlabelled nodes in the Bayes Net below with the variables $\{A, B, C, E\}$ such that the following independence assertions are true:

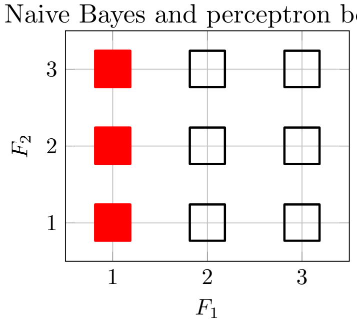
1. $A \perp\!\!\!\perp B \mid E, D$
2. $E \perp\!\!\!\perp D \mid B$
3. $E \perp\!\!\!\perp C \mid A, B$
4. $C \perp\!\!\!\perp D \mid A, B$



(b) [4 pts] For each of the 4 plots below, create a classification dataset which can or cannot be classified correctly by Naive Bayes and perceptron, as specified. Each dataset should consist of nine points represented by the boxes, shading the box ■ for positive class or leaving it blank □ for negative class. Mark *Not Possible* if no such dataset is possible.

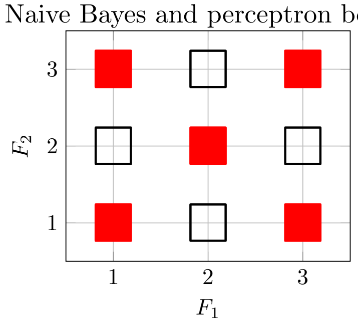
For *can* be classified by Naive Bayes, there should be some probability distributions $P(Y)$ and $P(F_1|Y), P(F_2|Y)$ for the class Y and features F_1, F_2 that can correctly classify the data according to the Naive Bayes rule, and for *cannot* there should be no such distribution. For perceptron, assume that there is a bias feature in addition to F_1 and F_2.

Naive Bayes and perceptron both **can** classify:



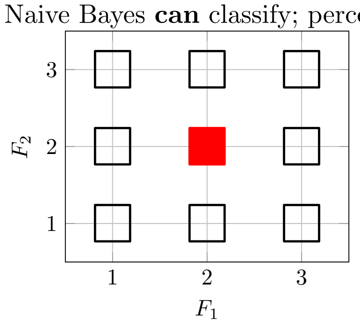
○ Not Possible

Naive Bayes and perceptron both **cannot** classify:



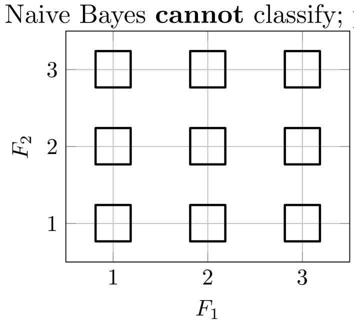
○ Not Possible

Naive Bayes **can** classify; perceptron **cannot** classify:



○ Not Possible

Naive Bayes **cannot** classify; perceptron **can** classify:

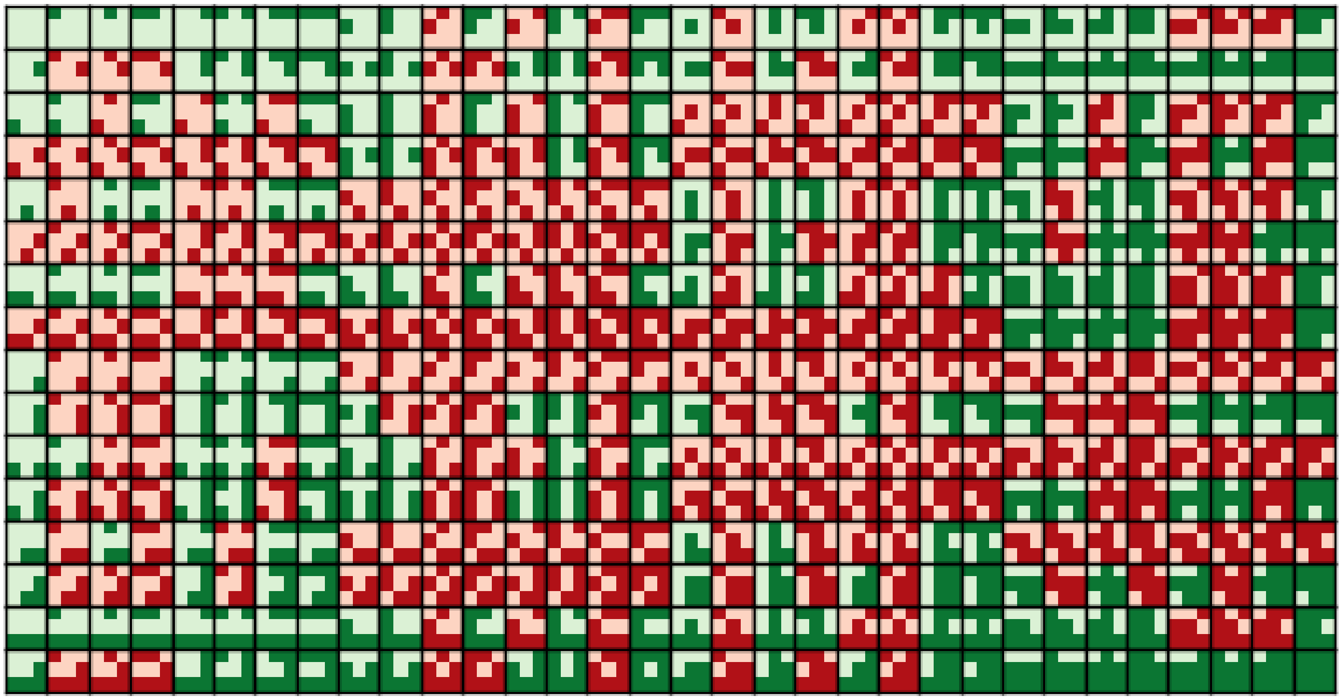


● Not Possible

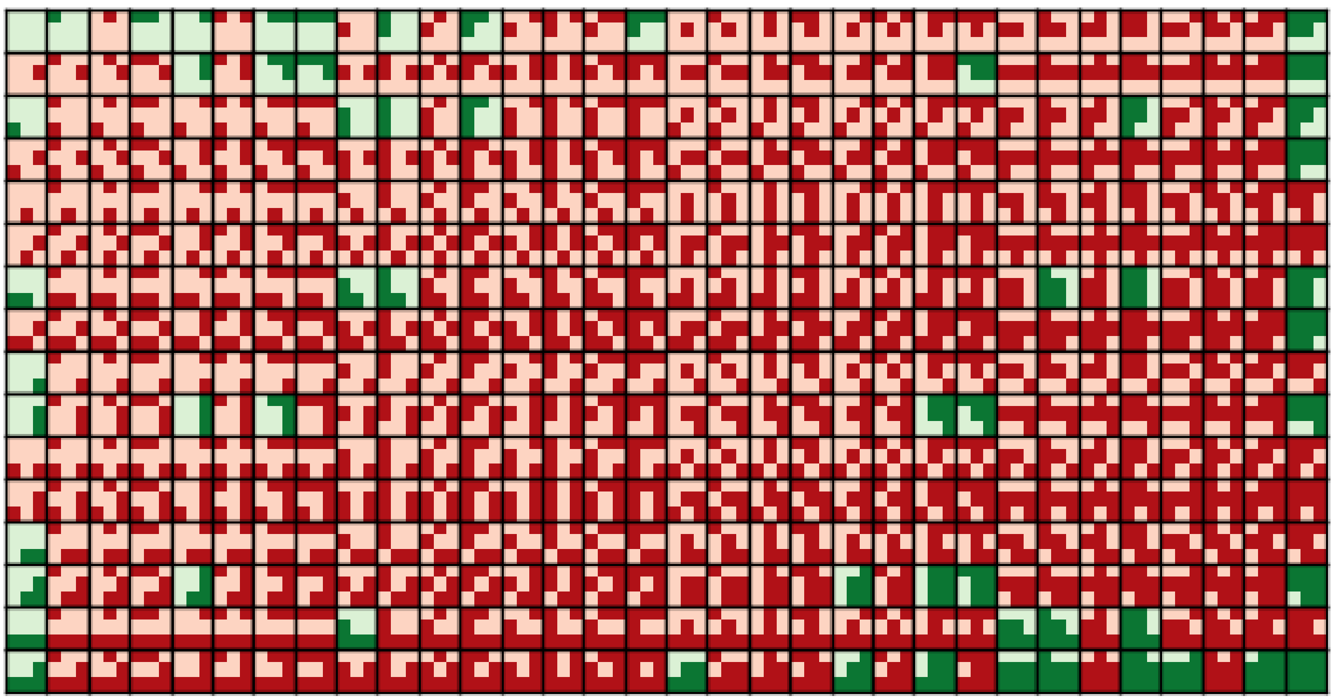
Many solutions were accepted for all except the bottom right. Naive Bayes can correctly classify any linearly separable dataset (as well as other datasets), and so it can classify every dataset that perceptron can. The full set of datasets which can be classified correctly are shown in the figures below, with classifiable datasets in green and unclassifiable in red:

Naive Bayes (classifiable datasets in green, unclassifiable in red):

SID: ________



Perceptron (classifiable datasets in green, unclassifiable in red):



(c) [1 pt] Consider a multi-class perceptron for classes A, B, and C with current weight vectors:

$w_A = (1, -4, 7)$, $w_B = (2, -3, 6)$, $w_C = (7, 9, -2)$

A new training sample is now considered, which has feature vector $f(x) = (-2, 1, 3)$ and label $y^* = B$. What are the resulting weight vectors after the perceptron has seen this example and updated the weights?

$w_A =$ (3, -5, 4) $\qquad w_B =$ (0, -2, 9) $\qquad w_C =$ (7, 9, -2)

(d) [1 pt] A single perceptron can compute the XOR function.

○ True ● False

(e) [1 pt] A perceptron is guaranteed to learn a separating decision boundary for a separable dataset within a finite number of training steps.

● True ○ False

(f) [1 pt] Given a linearly separable dataset, the perceptron algorithm is guaranteed to find a max-margin separating hyperplane.

○ True ● False

(g) [1 pt] You would like to train a neural network to classify digits. Your network takes as input an image and outputs probabilities for each of the 10 classes, 0-9. The network's prediction is the class that it assigns the highest probability to. From the following functions, select all that would be suitable loss functions to minimize using gradient descent:

☐ The square of the difference between the correct digit and the digit predicted by your network
☐ The probability of the correct digit under your network
■ The negative log-probability of the correct digit under your network
○ None of the above

- Option 1 is incorrect because it is non-differentiable. The correct digit and your model's predicted digit are both integers, and the square of their difference takes on values from the set $\{0^2, 1^2, \ldots, 9^2\}$. Losses that can be used with gradient descent must take on values from a continuous range and have well-defined gradients.
- Option 2 is not a loss because you would like to *maximize* the probability of the correct digit under your model, not minimize it.
- Option 3 is a common loss used for classification tasks. When the probabilities produced by a neural network come from a softmax layer, this loss is often combined with the softmax computation into a single entity known as the "softmax loss" or "softmax cross-entropy loss".

(h) [1 pt] From the list below, mark all triples that are **inactive**. A shaded circle means that node is conditioned on.

☐ ○ → ○ → ○ ☐ ○ ← ○ → ○ ■ ○ → ○ ← ○
■ ○ → ● → ○ ■ ○ ← ● → ○ ☐ ○ → ● ← ○

(i) [2 pts]

A	

Consider the gridworld above. At each timestep the agent will have two available actions from the set $\{North, South, East, West\}$. Actions that would move the agent into the wall may never be chosen, and allowed actions always succeed. The agent receives a reward of +8 every time it enters the square marked A. Let the discount factor be $\gamma = \frac{1}{2}$.

At each cell in the following tables, fill in the value of that state after iteration k of Value Iteration.

$k = 0$

0	0
0	0

$k = 1$

0	8
8	0

$k = 2$

4	8
8	4

$k = 3$

4	10
10	4

(j) [1 pt] Consider an HMM with T timesteps, hidden state variables $X_1, \ldots X_T$, and observed variables $E_1, \ldots E_T$. Let S be the number of possible states for each hidden state variable X. We want to compute (with the forward algorithm) or estimate (with particle filtering) $P(X_T \mid E_1 = e_1, \ldots E_T = e_T)$. How many particles, in terms of S and T, would it take for particle filtering to have the same time complexity as the forward algorithm? You can assume that, in particle filtering, each sampling step can be done in constant time for a single particle (though this is not necessarily the case in reality):

Particles = S^2

Q3. [9 pts] Search

Suppose we have a connected graph with N nodes, where N is finite but large. Assume that every node in the graph has exactly D neighbors. **All edges are undirected**. We have exactly one start node, S, and exactly one goal node, G.

Suppose we know that the shortest path in the graph from S to G has length L. That is, it takes at least L edge-traversals to get from S to G or from G to S (and perhaps there are other, longer paths).

We'll consider various algorithms for searching for paths from S to G.

(a) [2 pts] Uninformed Search
Using the information above, give the tightest possible bounds, using big $\mathcal{O}$ notation, on **both the absolute best case and the absolute worst case number of node expansions** for each algorithm. Your answer should be a function in terms of variables from the set $\{N, D, L\}$. You may not need to use every variable.

(i) [1 pt] DFS **Graph** Search

Best case: $\mathcal{O}(L)$. If we are lucky, DFS could send us directly on the shortest path to the goal without expanding anything else. Worst case: $\mathcal{O}(N)$. Worst Case is we expand every node in the graph before expanding G; because this is graph search, we can't expand anything more than once.

(ii) [1 pt] BFS **Tree** Search

Best case: $\mathcal{O}(D^{L-1})$ Worst case: $\mathcal{O}(D^L)$
In the best case, G is the first node expanded at depth L of the tree (expanded immediately after all nodes of depth $L-1$ are expanded). The structure of the graph gives that there are no more than D^{L-1} nodes of depth $L-1$, and since we can ignore this one extra node at depth L in the asymptotic bound, we have $\mathcal{O}(D^{L-1})$. In the worst case, BFS needs to expand all paths with depth $\leq L$ (i.e. G is the last node of depth L expanded), and so needs to expand $\mathcal{O}(D^L)$ nodes.

(b) [2 pts] Bidirectional Search
Notice that because the graph is undirected, finding a path from S to G is equivalent to finding a path from G to S, since reversing a path gives us a path from the other direction of the same length.

This fact inspired **bidirectional search**. As the name implies, bidirectional search consists of two simultaneous searches which both use the same algorithm; one from S towards G, and another from G towards S. When these searches meet in the middle, they can construct a path from S to G.

More concretely, in bidirectional search:

- We start Search 1 from S and Search 2 from G.
- The searches take turns popping nodes off of their separate fringes. First Search 1 expands a node, then Search 2 expands a node, then Search 1 again, etc.
- This continues until one of the searches expands some node X which the other search has also expanded.
- At that point, Search 1 knows a path from S to X, and Search 2 knows a path from G to X, which provides us with a path from X to G. We concatenate those two paths and return our path from S to G.

Don't stress about further implementation details here!

Repeat part (a) with the bidirectional versions of the algorithms from before. Give the tightest possible bounds, using big $\mathcal{O}$ notation, on both the absolute best and worst case number of node expansions by the bidirectional search algorithm. Your bound should still be a function of variables from the set $\{N, D, L\}$.

(i) [1 pt] Bidirectional DFS **Graph** Search

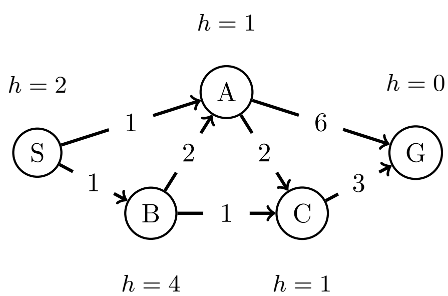
Best case: $\mathcal{O}(L)$. Bidirectional Search does not meaningfully change the number of nodes visited for DFS. If we are lucky, Bidi-DFS could send us directly on the shortest path in both directions without expanding anything else. Worst case: $\mathcal{O}(N)$. Worst Case is our two searches expands every node in the graph before meeting at some X; because this is graph search, we can't expand anything more than once.

(ii) [1 pt] Bidirectional BFS **Tree** Search

Best case: $\mathcal{O}(D^{\frac{L}{2}-1})$. Bidirectional Search improves BFS. Each search will expand half of the optimal path to the goal before meeting in the middle, at some node at depth $L/2$ for both searches. In the best case, this node is the first one expanded at that depth for both searches, so the number of node expansions is $\mathcal{O}(D^{\frac{L}{2}-1})$ for the same reason as in part a(ii). Worst case: $\mathcal{O}(D^{\frac{L}{2}})$. In the worst case the searches both need to expand at depths up to and including $D^{\frac{L}{2}}$.

In parts (c)-(e) below, consider the following graph, with start state S and goal state G. Edge costs are labeled on the edges, and heuristic values are given by the h values next to each state.

In the search procedures below, break any ties alphabetically, so that if nodes on your fringe are tied in values, the state that comes first alphabetically is expanded first.



(c) [1 pt] Greedy Graph Search

What is the path returned by greedy graph search, using the given heuristic?

- ● $S \rightarrow A \rightarrow G$
- ○ $S \rightarrow A \rightarrow C \rightarrow G$
- ○ $S \rightarrow B \rightarrow A \rightarrow C \rightarrow G$
- ○ $S \rightarrow B \rightarrow A \rightarrow G$
- ○ $S \rightarrow B \rightarrow C \rightarrow G$

(d) A* Graph Search

(i) [1 pt] List the nodes in the order they are expanded by A* graph search:

Order: $\boxed{S, A, C, B, G}$

(ii) [1 pt] What is the path returned by A* graph search?

- ○ $S \rightarrow A \rightarrow G$
- ● $S \rightarrow A \rightarrow C \rightarrow G$
- ○ $S \rightarrow B \rightarrow A \rightarrow C \rightarrow G$
- ○ $S \rightarrow B \rightarrow A \rightarrow G$
- ○ $S \rightarrow B \rightarrow C \rightarrow G$

(e) Heuristic Properties

(i) [1 pt] Is this heuristic *admissible*? If so, mark *Already admissible*. If not, find a minimal set of nodes that would need to have their values changed to make the heuristic admissible, and mark them below.

- ● Already admissible
- ☐ Change $h(S)$ ☐ Change $h(A)$ ☐ Change $h(B)$
- ☐ Change $h(C)$ ☐ Change $h(D)$ ☐ Change $h(G)$

(ii) [1 pt] Is this heuristic *consistent*? If so, mark *Already consistent*. If not, find the minimal set of nodes that would need to have their values changed to make the heuristic consistent, and mark them below.

- ○ Already consistent
- ☐ Change $h(S)$ ☐ Change $h(A)$ ■ Change $h(B)$
- ☐ Change $h(C)$ ☐ Change $h(D)$ ☐ Change $h(G)$

SID: ________

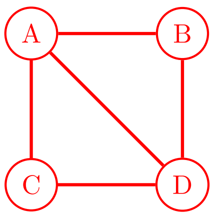
Q4. [8 pts] CSPs

Four people, A, B, C, and D, are all looking to rent space in an apartment building. There are three floors in the building, 1, 2, and 3 (where 1 is the lowest floor and 3 is the highest). Each person must be assigned to some floor, but it's ok if more than one person is living on a floor. We have the following constraints on assignments:

- A and B must not live together on the same floor.
- If A and C live on *the same* floor, they must both be living on floor 2.
- If A and C live on *different* floors, one of them must be living on floor 3.
- D must not live on the same floor as anyone else.
- D must live on a higher floor than C.

We will formulate this as a CSP, where each person has a variable and the variable values are floors.

(a) [1 pt] Draw the edges for the constraint graph representing this problem. Use binary constraints only. You do not need to label the edges.



(b) [2 pts] Suppose we have assigned C = 2. Apply forward checking to the CSP, filling in the boxes next to the values for each variable that are eliminated:

A	■ 1	□ 2	□ 3
B	□ 1	□ 2	□ 3
C		□ 2	
D	■ 1	■ 2	□ 3

(c) [3 pts] Starting from the original CSP with full domains (i.e. without assigning any variables or doing the forward checking in the previous part), enforce arc consistency for the entire CSP graph, filling in the boxes next to the values that are eliminated for each variable:

A	■ 1	□ 2	□ 3
B	□ 1	□ 2	□ 3
C	□ 1	□ 2	■ 3
D	■ 1	□ 2	□ 3

(d) [2 pts] Suppose that we were running local search with the min-conflicts algorithm for this CSP, and currently have the following variable assignments.

A	3
B	1
C	2
D	3

Which variable would be reassigned, and which value would it be reassigned to? Assume that any ties are broken alphabetically for variables and in numerical order for values.

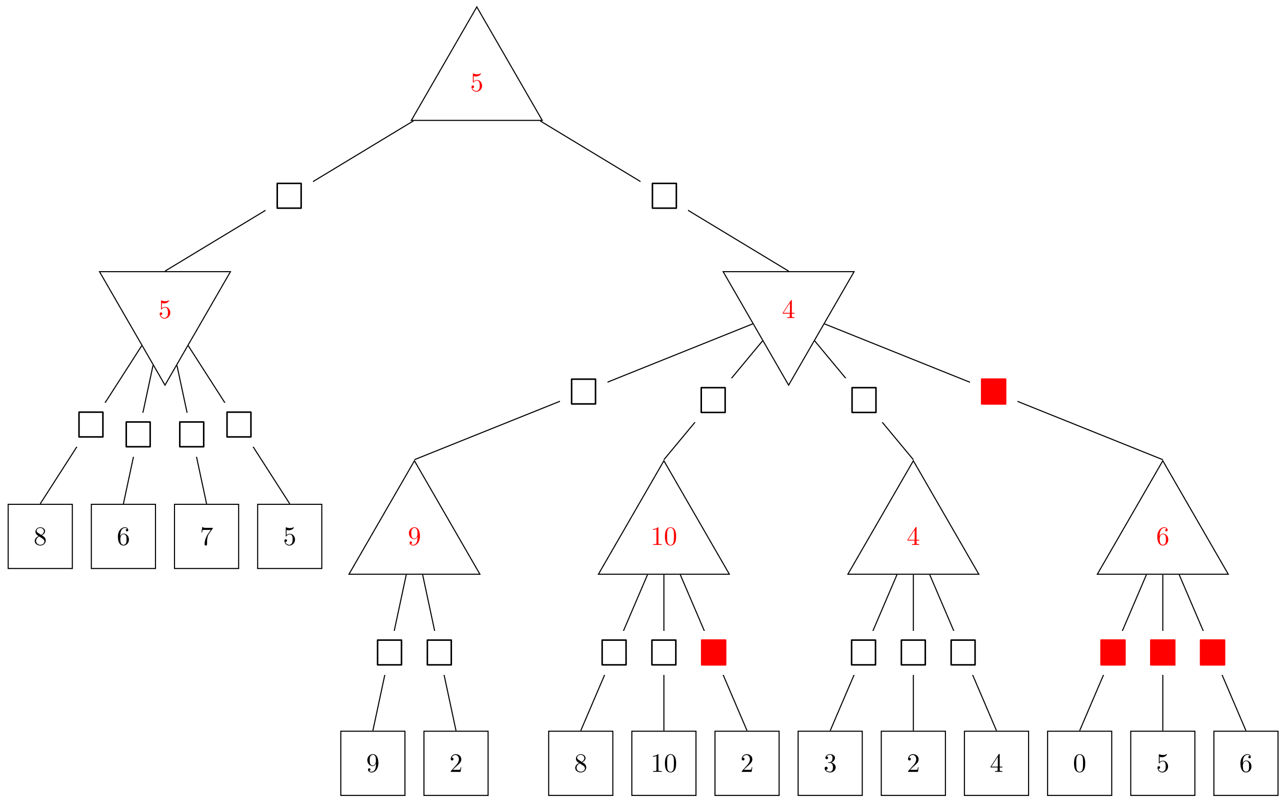
The variable ● A ○ B ○ C ○ D will be assigned the new value ○ 1 ● 2 ○ 3

Q5. [9 pts] Game Trees

The following problems are to test your knowledge of Game Trees.

(a) Minimax

The first part is based upon the following tree. Upward triangle nodes are maximizer nodes and downward are minimizers. (small squares on edges will be used to mark pruned nodes in part (ii))



(i) [1 pt] Complete the game tree shown above by filling in values on the maximizer and minimizer nodes.

(ii) [3 pts] Indicate which nodes can be pruned by marking the edge above each node that can be pruned (you do not need to mark any edges below pruned nodes). In the case of ties, please prune any nodes that could not affect the root node's value. Fill in the bubble below if no nodes can be pruned.

○ No nodes can be pruned

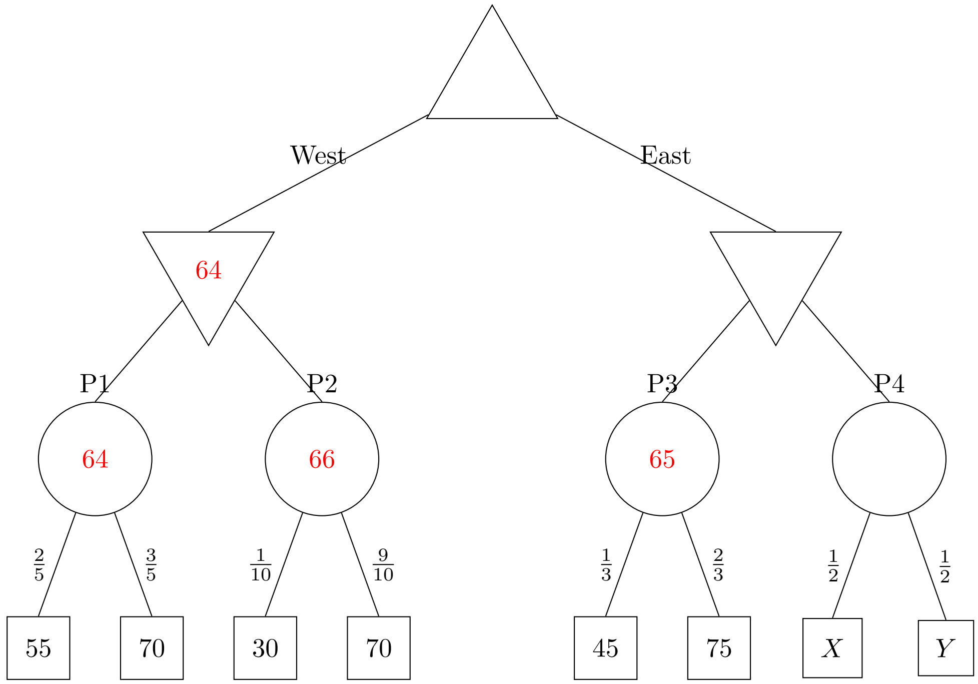
(b) Food Dimensions

The following questions are completely unrelated to the above parts.

Pacman is playing a tricky game. There are 4 portals to food dimensions. But, these portals are guarded by a ghost. Furthermore, neither Pacman nor the ghost know for sure how many pellets are behind each portal, though they know what options and probabilities there are for all but the last portal.

Pacman moves first, either moving West or East. After which, the ghost can block **1** of the portals available.

You have the following gametree. The maximizer node is Pacman. The minimizer nodes are ghosts and the portals are chance nodes with the probabilities indicated on the edges to the food. In the event of a tie, the left action is taken. Assume Pacman and the ghosts play optimally.



(i) [1 pt] Fill in values for the nodes that do not depend on X and Y.

(ii) [4 pts] What conditions must X and Y satisfy for Pacman to move East? What about to definitely reach the P4? Keep in mind that X and Y denote numbers of food pellets and must be **whole numbers**: $X, Y \in \{0, 1, 2, 3, \ldots\}$.

To move East: $\boxed{X + Y > 128}$

To reach P4: $\boxed{X + Y = 129}$

The first thing to note is that, to pick A over B, $value(A) > value(B)$.
Also, the expected value of the parent node of X and Y is $\frac{X+Y}{2}$.
$\implies min(65, \frac{X+Y}{2}) > 64$
$\implies \frac{X+Y}{2} > 64$
So, $X + Y > 128 \implies value(A) > value(B)$

To ensure reaching X or Y, apart from the above, we also have $\frac{X+Y}{2} < 65$
$\implies 128 < X + Y < 130$
So, $X, Y \in \mathbb{N} \implies X + Y = 129$

Q6. [10 pts] Something Fishy

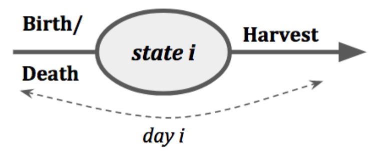
In this problem, we will consider the task of managing a fishery for an infinite number of days. (Fisheries farm fish, continually harvesting and selling them.) Imagine that our fishery has a very large, enclosed pool where we keep our fish.

Harvest (11pm): Before we go home each day at 11pm, we have the option to harvest some (possibly all) of the fish, thus removing those fish from the pool and earning us some profit, x dollars for x fish.

Birth/death (midnight): At midnight each day, some fish are born and some die, so the number of fish in the pool changes. An ecologist has analyzed the ecological dynamics of the fish population. They say that if at midnight there are x fish in the pool, then after midnight there will be exactly $f(x)$ fish in the pool, where f is a function they have provided to us. (We will pretend it is possible to have fractional fish.)

To ensure you properly maximize your profit while managing the fishery, you choose to model it using a Markov decision problem.

For this problem we will define States and Actions as follows:
State: the number of fish in the pool that day (before harvesting)
Action: the number of fish you harvest that day



(a) [2 pts] How will you define the transition and reward functions?

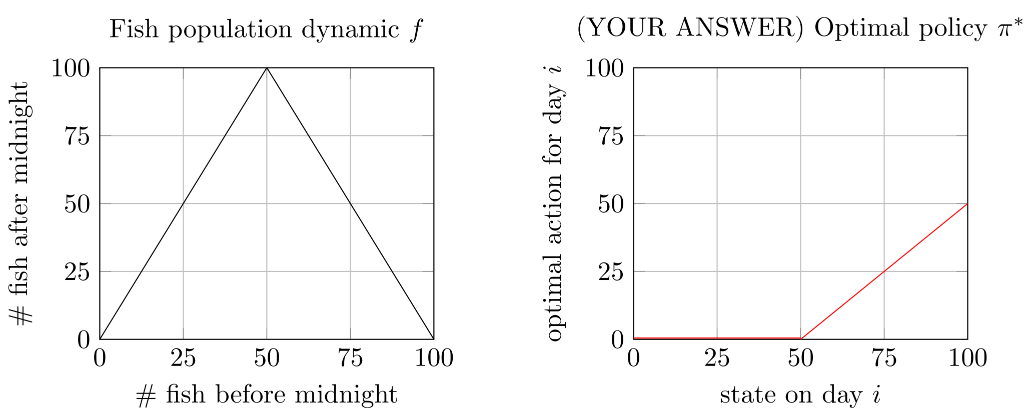
$T(s, a, s') = 1$ if $f(\max(s - a, 0)) = s'$ else 0

$R(s, a) = \min(a, s)$

Note that taking the maximum with 0 in T and taking the minimum with s in R were not required for full credit.

(b) [4 pts] Suppose the discount rate is $\gamma = 0.99$ and f is as below. Graph the optimal policy π^*.

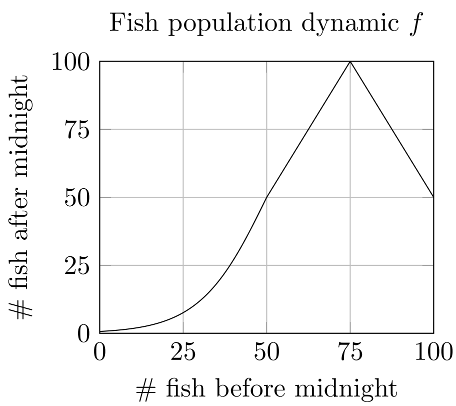
Only answers which depict the piece-wise function that is $\pi^* = 0$ on $s \in [0, 50]$ and $\pi^* = s - 50$ on $s \in [50, 100]$ were accepted.

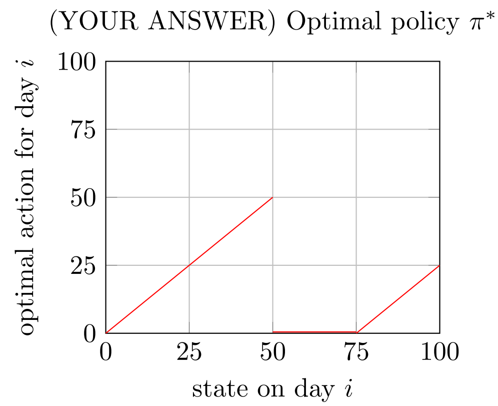


(c) [4 pts] Suppose the discount rate is $\gamma = 0.99$ and f is as below. Graph the optimal policy π^*.

There are three graded components to this answer: the $\pi^* = s$ harvest-all region in $[0, 50]$, the $\pi^* = 0$ grow-to-optimal region in $[50, 75]$, and the $\pi^* = s - 75$ harvest-to-optimal region in $[75, 100]$. The first component was worth most of the points, as the other two components are difficult to come up with. Additionally, the other two components did not need to border at exactly 75. (Note: This answer was verified by running value iteration on a computer.)

SID: ________





Q7. [8 pts] Policy Evaluation

In this question, you will be working in an MDP with states S, actions A, discount factor γ, transition function T, and reward function R.

We have some fixed policy $\pi : S \to A$, which returns an action $a = \pi(s)$ for each state $s \in S$. We want to learn the Q function $Q^\pi(s, a)$ for this policy: the expected discounted reward from taking action a in state s and then continuing to act according to π: $Q^\pi(s, a) = \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma Q^\pi(s', \pi(s'))]$. The policy π will not change while running any of the algorithms below.

(a) [1 pt] Can we guarantee anything about how the values Q^π compare to the values Q^* for an optimal policy π^*?

- ● $Q^\pi(s, a) \leq Q^*(s, a)$ for all s, a
- ○ $Q^\pi(s, a) = Q^*(s, a)$ for all s, a
- ○ $Q^\pi(s, a) \geq Q^*(s, a)$ for all s, a
- ○ None of the above are guaranteed

(b) Suppose T and R are *unknown*. You will develop sample-based methods to estimate Q^π. You obtain a series of *samples* $(s_1, a_1, r_1), (s_2, a_2, r_2), \dots (s_T, a_T, r_T)$ from acting according to this policy (where $a_t = \pi(s_t)$, for all t).

(i) [4 pts] Recall the update equation for the Temporal Difference algorithm, performed on each sample in sequence:

$$V(s_t) \leftarrow (1 - \alpha)V(s_t) + \alpha(r_t + \gamma V(s_{t+1}))$$

which approximates the expected discounted reward $V^\pi(s)$ for following policy π from each state s, for a learning rate α.

Fill in the blank below to create a similar update equation which will approximate Q^π using the samples.

You can use any of the terms $Q, s_t, s_{t+1}, a_t, a_{t+1}, r_t, r_{t+1}, \gamma, \alpha, \pi$ in your equation, as well as $\sum$ and max with any index variables (i.e. you could write $\max_a$, or $\sum_a$ and then use a somewhere else), but no other terms.

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha\left[r_t + \gamma Q(s_{t+1}, a_{t+1})\right]$$

(ii) [2 pts] Now, we will approximate Q^π using a linear function: $Q(s, a) = \mathbf{w}^\top \mathbf{f}(s, a)$ for a weight vector $\mathbf{w}$ and feature function $\mathbf{f}(s, a)$.

To decouple this part from the previous part, use Q_{samp} for the value in the blank in part (i) (i.e. $Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha Q_{samp}$).

Which of the following is the correct sample-based update for $\mathbf{w}$?

- ○ $\mathbf{w} \leftarrow \mathbf{w} + \alpha[Q(s_t, a_t) - Q_{samp}]$
- ○ $\mathbf{w} \leftarrow \mathbf{w} - \alpha[Q(s_t, a_t) - Q_{samp}]$
- ○ $\mathbf{w} \leftarrow \mathbf{w} + \alpha[Q(s_t, a_t) - Q_{samp}]\mathbf{f}(s_t, a_t)$

- [x] $\mathbf{w} \leftarrow \mathbf{w} - \alpha[Q(s_t, a_t) - Q_{samp}]\mathbf{f}(s_t, a_t)$
- [ ] $\mathbf{w} \leftarrow \mathbf{w} + \alpha[Q(s_t, a_t) - Q_{samp}]\mathbf{w}$
- [ ] $\mathbf{w} \leftarrow \mathbf{w} - \alpha[Q(s_t, a_t) - Q_{samp}]\mathbf{w}$

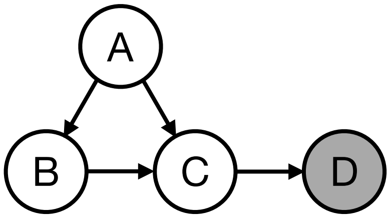
(iii) [1 pt] The algorithms in the previous parts (part i and ii) are:

- [ ] model-based
- [x] model-free

SID: _________

Q8. [8 pts] Bayes Nets: Inference

Consider the following Bayes Net, where we have observed that $D = +d$.



$P(A)$	
$+a$	0.5
$-a$	0.5

$P(B \mid A)$		
$+a$	$+b$	0.5
$+a$	$-b$	0.5
$-a$	$+b$	0.2
$-a$	$-b$	0.8

$P(C \mid A, B)$			
$+a$	$+b$	$+c$	0.8
$+a$	$+b$	$-c$	0.2
$+a$	$-b$	$+c$	0.6
$+a$	$-b$	$-c$	0.4
$-a$	$+b$	$+c$	0.2
$-a$	$+b$	$-c$	0.8
$-a$	$-b$	$+c$	0.1
$-a$	$-b$	$-c$	0.9

$P(D \mid C)$		
$+c$	$+d$	0.4
$+c$	$-d$	0.6
$-c$	$+d$	0.2
$-c$	$-d$	0.8

(a) [1 pt] Below is a list of samples that were collected using prior sampling. Mark the samples that would be **rejected** by rejection sampling.

■ $+a \quad -b \quad +c \quad -d$
□ $+a \quad -b \quad +c \quad +d$
■ $-a \quad +b \quad -c \quad -d$
□ $+a \quad +b \quad +c \quad +d$

(b) [3 pts] To decouple from the previous part, you now receive a *new* set of samples shown below:

$+a \quad +b \quad +c \quad +d$
$-a \quad -b \quad -c \quad +d$
$+a \quad +b \quad +c \quad +d$
$+a \quad -b \quad -c \quad +d$
$-a \quad -b \quad -c \quad +d$

For this part, express your answers as exact decimals or fractions **simplified to lowest terms**.

Estimate the probability $P(+a \mid +d)$ if these new samples were collected using...

(i) [1 pt] ... rejection sampling: $\boxed{\frac{3}{5}}$

(ii) [2 pts] ... likelihood weighting: $\boxed{\frac{0.4 + 0.4 + 0.2}{0.4 + 0.2 + 0.4 + 0.2 + 0.2} = \frac{10}{14} = \frac{5}{7}}$

(c) [4 pts] Instead of sampling, we now wish to use **variable elimination** to calculate $P(+a \mid +d)$. We start with the factorized representation of the joint probability:

$$P(A, B, C, +d) = P(A)P(B \mid A)P(C \mid A, B)P(+d \mid C)$$

(i) [1 pt] We begin by eliminating the variable B, which creates a new factor f_1. Complete the expression for the factor f_1 in terms of other factors.

$f_1(\boxed{A, C}) = \boxed{\sum_b P(b \mid A)P(C \mid A, b)}$

(ii) [1 pt] After eliminating B to create a factor f_1, we next eliminate C to create a factor f_2. What are the remaining factors after both B and C are eliminated?

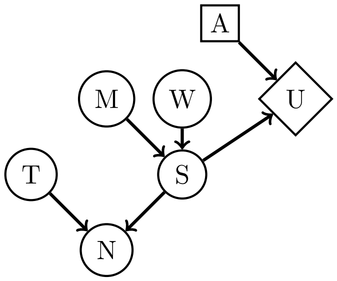
■ $p(A)$ □ $p(B \mid A)$ □ $p(C \mid A, B)$ □ $p(+d \mid C)$ □ f_1 ■ f_2

(iii) [2 pts] After eliminating both B and C, we are now ready to calculate $P(+a \mid +d)$. Write an expression for $P(+a \mid +d)$ in terms of the remaining factors.

$P(+a \mid +d) = \boxed{\frac{P(+a)f_2(+a, +d)}{\sum_a P(a)f_2(a, +d)}}$

Q9. [9 pts] Decision Networks and VPI

(a) Consider the decision network structure given below:



Mark all of the following statements that **could possibly be true**, for some probability distributions for $P(M), P(W), P(T), P(S|M,W)$, and $P(N|T,S)$ and some utility function $U(S,A)$:

(i) [1.5 pts]

☐ VPI(T) < 0 ■ VPI(T) $= 0$ ☐ VPI(T) > 0 ■ VPI(T) $=$ VPI(N)

VPI can never be negative. VPI(T) = 0 must be true since T is independent of S. VPI(N) could also be zero if N and S are independent.

(ii) [1.5 pts]

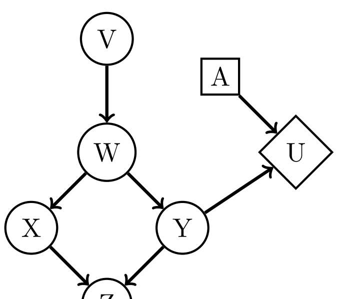
☐ VPI($T|N$) < 0 ■ VPI($T|N$) $= 0$ ■ VPI($T|N$) > 0 ■ VPI($T|N$) $=$ VPI($T|S$)

VPI can never be negative. $VPI(T|N) = 0$ if N is conditionally independent of S given N, but will usually be positive. $VPI(T|S) = 0$, and as we've seen $VPI(T|N)$ could also be zero.

(iii) [1.5 pts]

■ VPI(M) $>$ VPI(W) ☐ VPI(M) $>$ VPI(S) ■ VPI(M) $<$ VPI(S) ☐ VPI($M|S$) $>$ VPI(S)

(b) Consider the decision network structure given below.



Mark all of the following statements that are **guaranteed to be true**, regardless of the probability distributions for any of the chance nodes and regardless of the utility function.

(i) [1.5 pts]

☐ VPI(Y) $= 0$ Observing Y could increase MEU

☐ VPI(X) $= 0$ Y can depend on X because of the path through W

☐ VPI(Z) $=$ VPI(W, Z) Consider a case where Y is independent of Z but not independent of W. Then $VPI(Z) = 0 < VPI(W, Z)$

■ VPI(Y) $=$ VPI(Y, X) After Y is revealed, X will add no more information about Y.

(ii) [1.5 pts]

■ VPI(X) $\leq$ VPI(W) $VPI(W \mid X) + VPI(X) = VPI(X, W) = VPI(X \mid W) + VPI(W)$. We know $VPI(X \mid W) = 0$, since X is conditionally independent of Y, given W. So $VPI(W \mid X) + VPI(X) = VPI(W)$. Since VPI is non-negative, $VPI(W \mid X) \geq 0$, so $VPI(X) \leq VPI(W)$.

■ VPI(V) $\leq$ VPI(W) Since the only path from V to Y is through W, revealing V cannot give more information about Y than revealing W.

☐ VPI($V \mid W$) = VPI(V) $VPI(V \mid W) = 0$ by conditional independence, but $VPI(V)$ is not necessarily 0

☐ VPI($W \mid V$) = VPI(W) Consider a case where W is a deterministic function of V and Y is a deterministic function of W, then $VPI(W \mid V) = 0 \neq VPI(W)$

(iii) [1.5 pts]

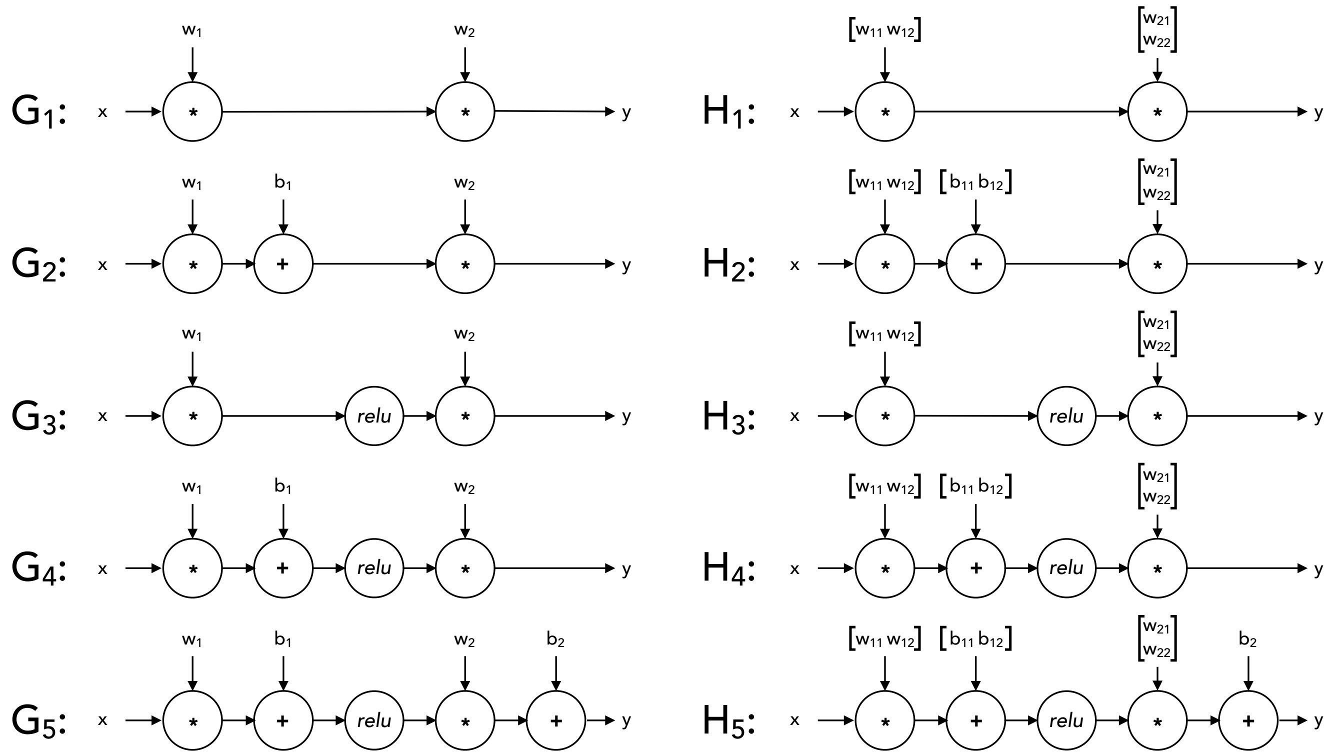
■ VPI($X \mid W$) = 0 X is independent of Y given W

☐ VPI($Z \mid W$) = 0 Y could depend on Z, given W

■ VPI(X, W) = VPI(V, W) Both are equal to VPI(W), since both X and V are conditionally independent of Y given W.

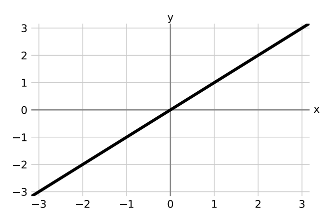
☐ VPI(W, Y) = VPI(W) + VPI(Y) VPI(W,Y) = VPI(Y), and we can have $VPI(W) > 0$

Q10. [15 pts] Neural Networks: Representation

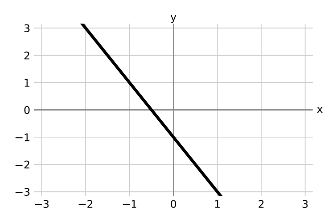


For each of the piecewise-linear functions below, mark all networks from the list above that can represent the function **exactly** on the range $x \in (-\infty, \infty)$. In the networks above, $relu$ denotes the element-wise ReLU nonlinearity: $relu(z) = max(0, z)$. The networks G_i use 1-dimensional layers, while the networks H_i have some 2-dimensional intermediate layers.

(a) [5 pts]



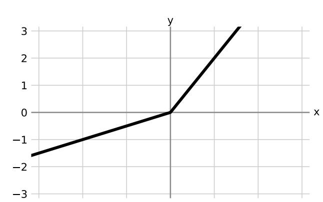
- [x] G_1
- [x] G_2
- [ ] G_3
- [ ] G_4
- [ ] G_5
- [x] H_1
- [x] H_2
- [x] H_3
- [x] H_4
- [x] H_5
- ○ None of the above



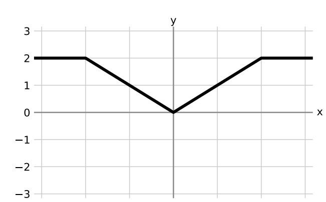
- [ ] G_1
- [x] G_2
- [ ] G_3
- [ ] G_4
- [ ] G_5
- [ ] H_1
- [x] H_2
- [ ] H_3
- [x] H_4
- [x] H_5
- ○ None of the above

The networks G_3, G_4, G_5 include a ReLU nonlinearity on a scalar quantity, so it is impossible for their output to represent a non-horizontal straight line. On the other hand, H_3, H_4, H_5 have a 2-dimensional hidden layer, which allows two ReLU elements facing in opposite directions to be added together to form a straight line. The second subpart requires a bias term because the line does not pass through the origin.

(b) [5 pts]



- [ ] G_1
- [ ] G_2
- [ ] G_3
- [ ] G_4
- [ ] G_5
- [ ] H_1
- [ ] H_2
- [x] H_3
- [x] H_4
- [x] H_5
- ○ None of the above



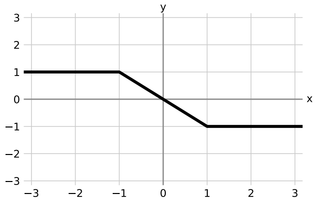
- [ ] G_1
- [ ] G_2
- [ ] G_3
- [ ] G_4
- [ ] G_5
- [ ] H_1
- [ ] H_2
- [ ] H_3
- [ ] H_4
- [ ] H_5
- ● None of the above

These functions include multiple non-horizontal linear regions, so they cannot be represented by any of the networks G_i which apply ReLU no more than once to a scalar quantity.

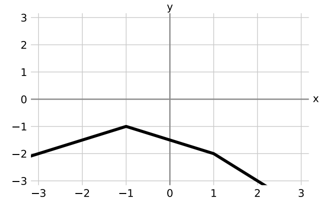
The first subpart can be represented by any of the networks with 2-dimensional ReLU nodes. The point of nonlinearity occurs at the origin, so nonzero bias terms are not required.

The second subpart has 3 points where the slope changes, but the networks H_i only have a single 2-dimensional ReLU node. Each application of ReLU to one element can only introduce a change of slope for a single value of x.

(c) [5 pts]



- [ ] G_1 [ ] H_1
- [ ] G_2 [ ] H_2
- [ ] G_3 [ ] H_3
- [ ] G_4 [ ] H_4
- [ ] G_5 [x] H_5
- ○ None of the above



- [ ] G_1 [ ] H_1
- [ ] G_2 [ ] H_2
- [ ] G_3 [ ] H_3
- [ ] G_4 [x] H_4
- [ ] G_5 [x] H_5
- ○ None of the above

Both functions have two points where the slope changes, so none of the networks $G_i; H_1, H_2$ can represent them.

An output bias term is required for the first subpart because one of the flat regions must be generated by the flat part of a ReLU function, but neither one of them is at $y = 0$.

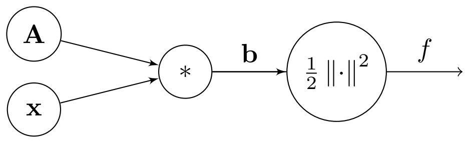
The second subpart doesn't require a bias term at the output: it can be represented as $-relu(\frac{-x+1}{2}) - relu(x+1)$. Note how if the segment at $x > 2$ were to be extended to cross the x axis, it would cross exactly at $x = -1$, the location of the other slope change. A similar statement is true for the segment at $x < -1$.

Q11. [9 pts] Backpropagation

In this question we will perform the backward pass algorithm on the formula

$$f = \frac{1}{2} \|\mathbf{A}\mathbf{x}\|^2$$

Here, $\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$, $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $\mathbf{b} = \mathbf{A}\mathbf{x} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 \\ A_{21}x_1 + A_{22}x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$, and $f = \frac{1}{2}\|\mathbf{b}\|^2 = \frac{1}{2}\left(b_1^2 + b_2^2\right)$ is a scalar.



(a) [1 pt] Calculate the following partial derivatives of f.

(i) [1 pt] Find $\frac{\partial f}{\partial \mathbf{b}} = \begin{bmatrix} \frac{\partial f}{\partial b_1} \\ \frac{\partial f}{\partial b_2} \end{bmatrix}$.

○ $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ ● $\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$ ○ $\begin{bmatrix} b_2 \\ b_1 \end{bmatrix}$ ○ $\begin{bmatrix} f(b_1) \\ f(b_2) \end{bmatrix}$ ○ $\begin{bmatrix} A_{11} \\ A_{22} \end{bmatrix}$ ○ $\begin{bmatrix} b_1 + b_2 \\ b_1 - b_2 \end{bmatrix}$

(b) [3 pts] Calculate the following partial derivatives of b_1.

(i) [1 pt] $\left(\frac{\partial b_1}{\partial A_{11}}, \frac{\partial b_1}{\partial A_{12}}\right)$

○ (A_{11}, A_{12}) ○ $(0, 0)$ ○ (x_2, x_1) ○ $(A_{11}x_1, A_{12}x_2)$ ● (x_1, x_2)

(ii) [1 pt] $\left(\frac{\partial b_1}{\partial A_{21}}, \frac{\partial b_1}{\partial A_{22}}\right)$

○ (A_{21}, A_{22}) ○ (x_1, x_2) ○ $(1, 1)$ ● $(0, 0)$ ○ $(A_{21}x_1, A_{22}x_2)$

(iii) [1 pt] $\left(\frac{\partial b_1}{\partial x_1}, \frac{\partial b_1}{\partial x_2}\right)$

● (A_{11}, A_{12}) ○ (A_{21}, A_{22}) ○ $(0, 0)$ ○ (b_1, b_2) ○ $(A_{21}x_1, A_{22}x_2)$

(c) [3 pts] Calculate the following partial derivatives of f.

(i) [1 pt] $\left(\frac{\partial f}{\partial A_{11}}, \frac{\partial f}{\partial A_{12}}\right)$

○ (A_{11}, A_{12}) ○ $(A_{11}b_1, A_{12}b_2)$ ○ $(A_{11}x_1, A_{12}x_2)$
● (x_1b_1, x_2b_1) ○ (x_1b_2, x_2b_2) ○ (x_1b_1, x_2b_2)

(ii) [1 pt] $\left(\frac{\partial f}{\partial A_{21}}, \frac{\partial f}{\partial A_{22}}\right)$

○ (A_{21}, A_{22}) ○ $(A_{21}b_1, A_{22}b_2)$ ○ $(A_{21}x_1, A_{22}x_2)$
○ (x_1b_1, x_2b_1) ● (x_1b_2, x_2b_2) ○ (x_1b_1, x_2b_2)

(iii) [1 pt] $\left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right)$

○ $(A_{11}b_1 + A_{12}b_2, A_{21}b_1 + A_{22}b_2)$ ● $(A_{11}b_1 + A_{21}b_2, A_{12}b_1 + A_{22}b_2)$
○ $(A_{11}b_1 + A_{12}b_1, A_{21}b_2 + A_{22}b_2)$ ○ $(A_{11}b_1 + A_{21}b_1, A_{12}b_2 + A_{22}b_2)$

(d) [2 pts] Now we consider the general case where $\mathbf{A}$ is an $n \times d$ matrix, and $\mathbf{x}$ is a $d \times 1$ vector. As before, $f = \frac{1}{2}\|\mathbf{A}\mathbf{x}\|^2$.

(i) [1 pt] Find $\frac{\partial f}{\partial \mathbf{A}}$ in terms of $\mathbf{A}$ and $\mathbf{x}$ only.

○ $\mathbf{x}^\top\mathbf{A}^\top\mathbf{A}\mathbf{x}$ ● $\mathbf{A}\mathbf{x}\mathbf{x}^\top$ ○ $\mathbf{A}\left(\mathbf{A}^\top\mathbf{A}\right)^{-1}$ ○ $\mathbf{A}\mathbf{A}^\top\mathbf{A}\mathbf{x}$ ○ $\mathbf{A}$

(ii) [1 pt] Find $\frac{\partial f}{\partial \mathbf{x}}$ in terms of $\mathbf{A}$ and $\mathbf{x}$ only.

○ $\mathbf{x}$ ○ $\left(\mathbf{A}^\top\mathbf{A}\right)^{-1}\mathbf{x}$ ○ $\mathbf{x}\mathbf{x}^\top\mathbf{x}$ ○ $\mathbf{x}^\top\mathbf{A}^\top\mathbf{A}\mathbf{x}$ ● $\mathbf{A}^\top\mathbf{A}\mathbf{x}$

THIS PAGE IS INTENTIONALLY LEFT BLANK