CS 188 Spring 2019 — Introduction to Artificial Intelligence — Final Exam

- You have 170 minutes. The time will be projected at the front of the room. You may not leave during the last 10 minutes of the exam.
- Do NOT open exams until told to. Write your SIDs in the top right corner of every page.
- If you need to go to the bathroom, bring us your exam, phone, and SID. We will record the time.
- In the interest of fairness, we want everyone to have access to the same information. To that end, we will not be answering questions about the content. If a clarification is needed, it will be projected at the front of the room. **Make sure to periodically check the clarifications**.
- The exam is closed book, closed laptop, and closed notes except your three-page double-sided cheat sheet. Turn off and put away all electronics.
- We will give you two sheets of scratch paper. Please do not turn them in with your exam. Mark your answers ON THE EXAM IN THE DESIGNATED ANSWER AREAS. We will not grade anything on scratch paper.
- For multiple choice questions:
  - ☐ means mark ALL options that apply
  - ○ means mark ONE choice
  - When selecting an answer, please fill in the bubble or square COMPLETELY (● and ■)

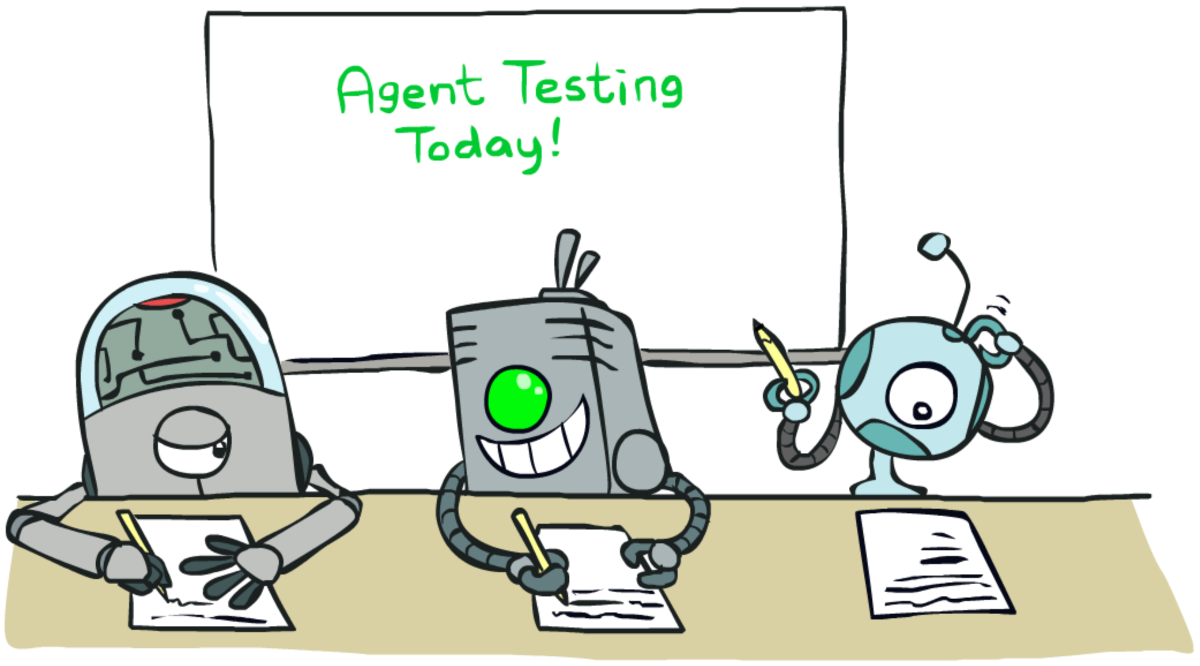
First name	
Last name	
SID	
Student to the right (SID and Name)	
Student to the left (SID and Name)	

Q1.	Agent Testing Today!	/1
Q2.	Search	/12
Q3.	Pacman's Treasure Hunt	/14
Q4.	Inexpensive Elimination	/9
Q5.	Sampling	/10
Q6.	HMM Smoothing	/11
Q7.	Partying Particle #No Filter(ing)	/8
Q8.	Double Decisions and VPI	/11
Q9.	Naively Fishing	/11
Q10.	Neural Networks and Decision Trees	/13
	Total	/100

THIS PAGE IS INTENTIONALLY LEFT BLANK

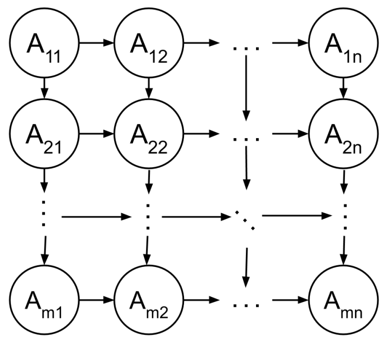
Q1. [1 pt] Agent Testing Today!

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. [12 pts] Search



(a) Consider the class of directed, $m \times n$ grid graphs as illustrated above. (We assume that $m, n > 2$.) Each edge has a cost of 1. The start state is A_{11} at top left and the goal state at A_{mn} is bottom right.

(i) [2 pts] If we run **Uniform-Cost** graph search (breaking ties randomly), what is the maximum possible size of the fringe? Write your answer in terms of m and n in big-O style, ignoring constants. For example, if you think the answer is $m^3n^3 + 2$, write m^3n^3.

$\boxed{\min\{m, n\}}$

All edge lengths are 1, so search expands out in a manner similar to BFS: the fringe is essentially the diagonal sweeping across the grid starting from 1,1. Its maximum size is $O(\min\{m, n\})$.

(ii) [2 pts] If we run **Depth-First** graph search with a stack, what is the maximum possible size of the stack?

$\boxed{m + n}$

Because all the arrows go right and down, the longest path in the graph is the one from 1,1 to m, n, which has length $O(m + n)$.

(b) Now answer the same questions for *undirected* $m \times n$ grid graphs (i.e., the links go in both directions between neighboring nodes).

(i) [2 pts] Maximum fringe size for **Uniform-Cost** graph search?

$\boxed{\min\{m, n\}}$

The nodes k steps from the start state will be exactly the same as in the directed graph, because following any left or up arrow will just get you to an already-visited state.

(ii) [2 pts] Maximum stack size for **Depth-First** graph search?

$\boxed{mn}$

Here, with ties broken randomly, DFS could follow a path that visits every node in the graph before reaching the goal—e.g., by travelling up and down the columns—hence the stack can grow to $O(mn)$.

(c) The following questions are concerned with an *undirected* grid graph G and with Manhattan distance $d((i, j), (k, l)) = |i - k| + |j - l|$. Now the start and goal locations can be anywhere in the graph.

(i) [1 pt] *True/False*: Let G^+ be a copy of G with n extra links added with edge cost 1 that connect arbitrary non-adjacent nodes. Then Manhattan distance is an admissible heuristic for finding shortest paths in G^+.

True [✓ False]

Manhattan is exact for G, and shortest paths in G^+ can be shorter than those in G—e.g., an added link might go straight to the goal, so Manhattan is not admissible.

(ii) [1 pt] *True/False*: Let G^- be a copy of G with n arbitrarily chosen links deleted, and let $h^+(s,t)$ be the exact cost of the shortest path from location s to location t in G^+. Then $h^+(\cdot, g)$ is an admissible heuristic for finding shortest paths in G^- when the goal is location g.

[✓ True] ○ False

The set of paths in G^- is a subset of those in G, whch is a subset of those in G^+, and hence no shorter path exists in G^- than the shortest path in G^+, so $h^+(\cdot, g)$ is admissible.

(iii) [2 pts] Suppose that K robots are at K different locations (x_k, y_k) on the complete grid G, and can move simultaneously. The goal is for them all to meet in one location as soon as possible, subject to the constraint that if two robots meet in the same location en route, they immediately settle down together and cannot move after that. Define d_X to be the maximum x separation, i.e., $|\max\{x_1, \ldots, x_K\} - \min\{x_1, \ldots, x_K\}|$, with d_Y defined similarly. Which of the following is the most accurate admissible heuristic for this problem? (Select one only.)

- ○ $\lceil d_X/2 \rceil + \lceil d_Y/2 \rceil$
- ○ $d_X + d_Y$
- ○ $\lceil d_X/2 \rceil + \lceil d_Y/2 \rceil + K/4$
- [✓ $\max\{\lceil d_X/2 \rceil, \lceil d_Y/2 \rceil, K/4\}$]
- ○ $\max\{\lceil d_X/2 \rceil + \lceil d_Y/2 \rceil, K/4\}$

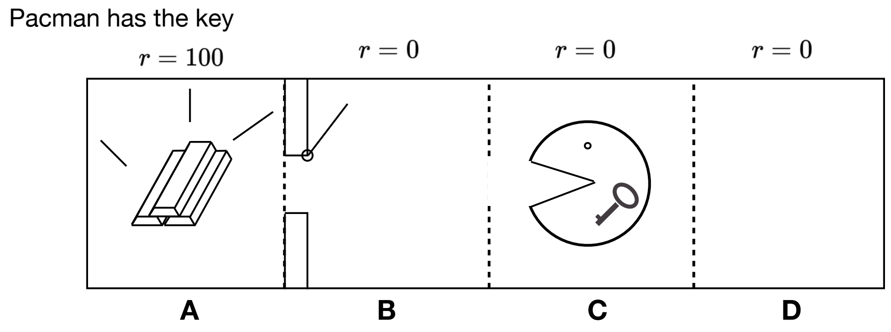
The no-meeting requirement means that at most 4 robots can arrive at the goal location in one time step, so $K/4$ is a lower bound. Some robot has to travel at least $\lceil d_X/2 \rceil$ steps, and some robot has to travel at least $\lceil d_Y/2 \rceil$ steps, so those are lower bounds; but it may not be the same robot. (Consider the K robots arrayed in a single vertical line and a single horizontal line, with the lines crossing in the middle.) So we cannot add the bounds.

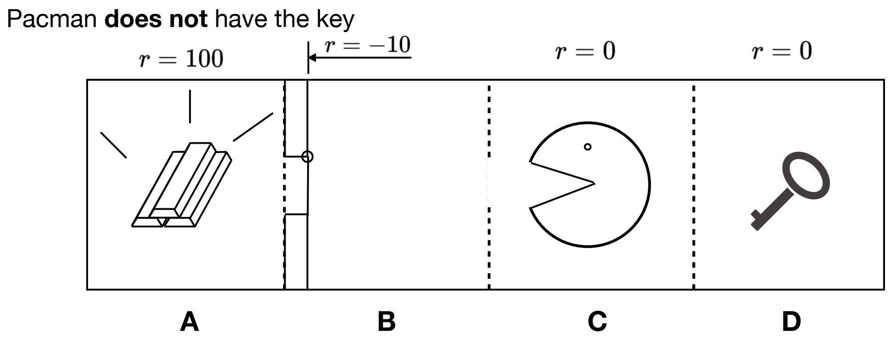
Q3. [14 pts] Pacman's Treasure Hunt

Pacman is hunting for gold in a linear grid-world with cells A, B, C, D. Cell A contains the gold but the entrance to A is locked. Pacman can pass through if he has a key. The possible states are as follows: X_k means that Pacman is in cell X and has the key; X_{-k} means that Pacman is in cell X and does not have the key. The initial state is always C_{-k}.

In each state Pacman has two possible actions, left and right. These actions are deterministic but do not change the state if Pacman tries to enter cell A without the key or run into a wall (left from cell A or right from cell D). The key is in cell D and entering cell D causes the key to be picked up instantly.

If Pacman tries to enter cell A without the key, he receives a reward of -10, i.e. $R(B_{-k}, left, B_{-k}) = -10$. The "exit" action from cell A receives a reward of 100. All other actions have 0 reward.





(a) [2 pts] Consider the discount factor $\gamma = 0.1$ and the following policy:

State	A_k	B_k	C_k	D_k	A_{-k}	B_{-k}	C_{-k}	D_{-k}
Action	exit	left	left	left	exit	left	right	right

Fill in $V^\pi(B_{-k})$ and $V^\pi(C_{-k})$ for this policy in the table below.

State	A_k	B_k	C_k	D_k	A_{-k}	B_{-k}	C_{-k}	D_{-k}
V	100	10	1	0.1	100	$-11.\bar{1}$	0.01	N/A

(b) [3 pts] Now, we will redefine the MDP so that Pacman has a probability $\beta \in [0, 1]$, on each attempt, of crashing through the gate even without the key. So our transition function from B will be modified as follows:

$$T(B_{-k}, left, A_{-k}) = \beta \text{ and } T(B_{-k}, left, B_{-k}) = 1 - \beta.$$

All other aspects remain the same. The immediate reward for attempting to go through the gate is -10 if Pacman fails to go through the gate, as before, and 0 if Pacman succeeds in crashing through gate. Which of the following are true? (Select one or more choices.)

[✓] For any fixed $\gamma < 1$, there is some value $\beta < 1$ such that trying to crash through the gate is better than fetching the key.

☐ For any fixed β, there is some value of $\gamma < 1$ such that trying to crash through the gate is better than fetching the key.

[✓] For $\beta = \frac{1}{2}$, there is some value of $\gamma < 1$ such that trying to crash through the gate is better than fetching the key.

☐ None of the above

(c) Thus far we've assumed knowledge of the transition function $T(s, a, s')$. Now let's assume we *do not.*

(i) [2 pts] Which of the following can be used to obtain a policy if we don't know the transition function?

☐ Value Iteration followed by Policy Extraction

[✓] Approximate Q-learning

☐ TD learning followed by Policy Extraction

[✓] Policy Iteration with a learned $T(s, a, s')$

(ii) [1 pt] Under which conditions would one benefit from using approximate Q-learning over vanilla Q-learning? (Select one only)

[✓] When the state space is very high-dimensional

○ When the transition function is known

○ When the transition function is unknown

○ When the discount factor is small

(iii) [4 pts] Suppose we choose to use Q-learning (in absence of the transition function) and we obtain the following observations:

s_t	a	s_{t+1}	reward
C_k	left	B_k	0
B_k	left	A_k	0
A_k	exit	terminal	100
B_{-k}	left	B_{-k}	-10

What values does the Q-function attain if we initialize the Q-values to 0 and replay the experience in the table **exactly two times**? Use a learning rate, α, of 0.5 and a discount factor, γ, of 0.1.

1. $Q(A_k, exit)$: ○ 100 [✓ 75] ○ 50 ○ 0
2. $Q(B_k, left)$: ○ 10 ○ 5 [✓ 2.5] ○ 0
3. $Q(C_k, left)$: ○ 10 ○ 5 ○ 2.5 [✓ 0]
4. $Q(B_{-k}, left)$: ○ -10 ○ 5 ○ -2.5 [✓ -7.5]

(d) Suppose we want to define the (deterministic) transition model using propositional logic instead of a table. States are defined using proposition symbols be A_t, B_t, C_t, D_t and K_t, where, e.g., A_t means that Pacman is in cell A at time t and K_t means that the Pacman has the key. The action symbols are $\texttt{Left}_t$, $\texttt{Right}_t$, $\texttt{Exit}_t$.

(i) [1 pt] Which of the following statements are correct formulations of the successor-state axiom for A_t?

○ $A_t \Leftrightarrow (A_{t-1} \Rightarrow (\texttt{Left}_{t-1} \wedge B_{t-1} \wedge K_{t-1})$

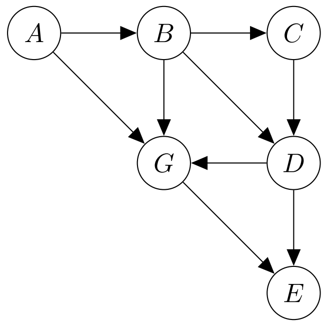
- ○ $A_t \Leftrightarrow (A_t \wedge \texttt{Left}_t) \vee (B_{t-1} \wedge \texttt{Left}_{t-1} \wedge K_{t-1})$
- [✓ $A_t \Leftrightarrow (A_{t-1} \wedge \texttt{Left}_{t-1}) \vee (B_{t-1} \wedge \texttt{Left}_{t-1} \wedge K_{t-1})$]
- ○ $A_t \Leftrightarrow (A_{t-1} \wedge \texttt{Left}_{t-1}) \vee (B_t \wedge \texttt{Left}_t \wedge K_t)$
- ○ $A_t \Leftrightarrow (A_{t-1} \wedge \texttt{Left}_{t-1}) \vee (B_{t-1} \wedge \texttt{Left}_{t-1} \wedge \neg K_{t-1})$

(ii) [1 pt] Which of the following statements are correct formulations of the successor-state axiom for K_t?

- ○ $K_t \Leftrightarrow K_{t-1} \wedge (C_{t-1} \vee \texttt{Right}_{t-1})$
- ○ $K_t \Leftrightarrow K_t \vee (C_t \wedge \texttt{Right}_t)$
- [✓ $K_t \Leftrightarrow K_{t-1} \vee (C_{t-1} \wedge \texttt{Right}_{t-1})$]
- ○ $K_t \Leftrightarrow K_{t-1} \vee D_{t-1}$

Q4. [9 pts] Inexpensive Elimination

In this problem, we will be using the Bayes Net below. Assume all random variables are binary-valued.



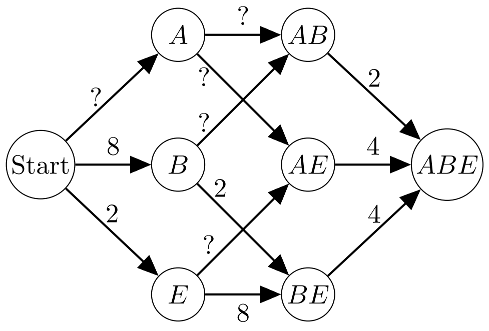
(a) [1 pt] Consider using Variable Elimination to get $P(C, G|D = d)$.

What is the factor generated if B is the first variable to be eliminated? Comma-separate all variables in the resulting *factor*, e.g., $f(A, B, C)$, without conditioned and unconditioned variables. Alphabetically order variables in your answer. E.g., $P(A)$ before $P(B)$, and $P(A|B)$ before $P(A|C)$.

$$f(\underline{\quad A, C, d, G \quad}) = \sum_b \underline{\quad P(b|A)P(C|b)P(d|b, C)P(G|A, b, d) \quad}$$

(b) Suppose we want to find the *optimal* ordering for variable elimination such that we have the *smallest sum of factor sizes*. Recall that all random variables in the graph are binary-valued (that means if there are two factors, one over two variables and another over three variables, the sum of factor sizes is 4+8=12).

In order to pick this ordering, we consider using A* Tree Search. Our state space graph consists of states which represent the variables we have eliminated so far, and does not take into account the order which they are eliminated. For example, eliminating B is a transition from the start state to state B, then eliminating A will result in state AB. Similarly, eliminating A is a transition from the start state to state A, then eliminating B will also result in state AB. An edge represents a step in variable elimination, and has weight equal to the size of the factor generated **after** eliminating the variable.



(i) [2 pts] *Yes/No*: As the graph is defined, we have assumed that different elimination orderings of the **same** subset of random variables will always produce the same **final set** of factors. Does this hold for all graphs?

[✓ Yes] No For any subset of variables, we will need to join the same factors eventually and marginalize over all of the variables in the subset

(ii) [4 pts] For this part, we consider possible heuristics $h(s)$, for a **generic** Bayes Net which has N variables left to eliminate at state s. Each remaining variable has domain size D.

Let the set $\mathbf{E}$ be the costs of edges from state s (e.g. for $s = A$, $\mathbf{E}$ is the costs of edges from A to AE, and A to AB). Which of the following would be admissible heuristics?

(We could reduce the number of heuristics.) This question takes a bit of thinking since each of the heuristics has to be considered separately.

2. $2 \min E$Consider a situation with just 1 more variable to eliminate. This would predict double the actual size.

3. $N \min E$ Consider a complete graph of 3 variables (triangle bayes net). First step, must generate D^2 factor. Next will generate D size factor. $N * D^2 > D^2 + D$

5. $\max (E)/N$ Part 3 is still a counterexample, since the table sizes increases exponentially

6. $N * D$ The minimum size of a factor eliminated is D, so $N * D$ would be a lower bound.

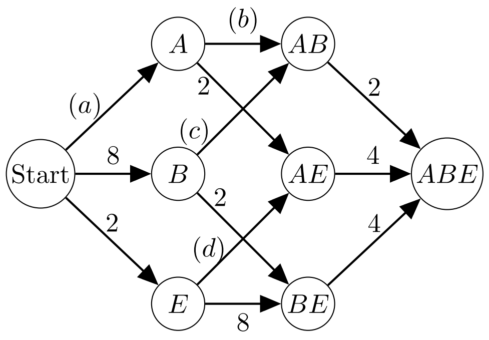
7. D^K This is a lower bound for the given state.

8. $N * D^K$ If we eliminate a variable, K could decrease, so this will not hold. (Think about case 2 explanation).

■ $\min \mathbf{E}$	☐ $\max \mathbf{E}$	■ $N * D$
☐ $2 \min \mathbf{E}$	☐ $\max (\mathbf{E})/N$	☐ None are admissible

(c) [2 pts] Now let's consider A* tree search on our Bayes Net for the query $P(C, G|D = d)$, where d is observed evidence.

Fill in the edge weights $(a) - (d)$ to complete the graph.



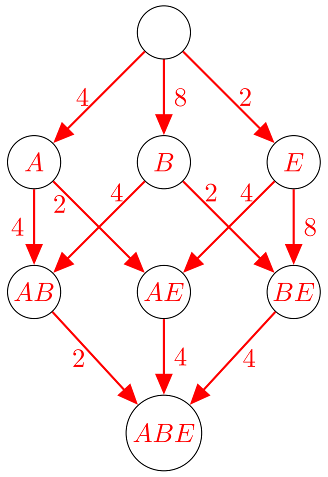
$(a) =$ 4

$(b) =$ 4

$(c) =$ 4

$(d) =$ 4

Solution: We create the following graph and label edge weights according to the size of the factor generated at each edge:



Q5. [10 pts] Sampling

Variables H, F, D, E and W denote the event of being health conscious, having free time, following a healthy diet, exercising and having a normal body weight, respectively. If an event does occur, we denote it with a $+$, otherwise $-$, e.g., $+e$ denotes an exercising and $-e$ denotes not exercising.

- A person is health conscious with probability 0.8.
- A person has free time with probability 0.4.
- If someone is health conscious, they will follow a healthy diet with probability 0.9.
- If someone is health conscious and has free time, then they will exercise with probability 0.9.
- If someone is health conscious, but does not have free time, they will exercise with probability 0.4.
- If someone is not health conscious, but they do have free time, they will exercise with probability 0.3.
- If someone is neither health conscious, nor they have free time, then they will exercise with probability 0.1.
- If someone follows both a healthy diet and exercises, they will have a normal body weight with probability 0.9.
- If someone only follows a healthy diet and does not exercise, or vice versa, they will have a normal body weight with probability 0.5.
- If someone neither exercises nor has a healthy diet, they will have a normal body weight with probability 0.2.

(a) [2 pts] Select the minimal set of edges that needs to be added to the following Bayesian network

$$\begin{tikzcd} W & D \arrow[l] & H \arrow[l] \arrow[dl] \\ & E \arrow[ul] & F \arrow[l] \end{tikzcd}$$

$D \to H$	[✓ $F \to E$]	$D \to E$	[✓ $D \to W$]
[✓ $H \to D$]	[✓ $D \to W$]	$F \to D$	$W \to D$
$H \to F$	$H \to W$	[✓ $E \to W$]	[✓ $E \to W$]

(b) Suppose we want to estimate the probability of a person being normal body weight given that they exercise (i.e. $P(+w|+e)$), and we want to use **likelihood weighting**.

(i) [1 pt] We observe the following sample: $(-w, -d, +e, +f, -h)$. What is our estimate of $P(+w|+e)$ given this one sample? Express your answer in decimal notation rounded to the second decimal point, or express it as a fraction simplified to the lowest terms.

$\boxed{0.00}$

The given sample does not have $W = +w$.

(ii) [2 pts] Now, suppose that we observe another sample: $(+w, +d, +e, +f, +h)$. What is our new estimate for $P(+w|+e)$? Express your answer in decimal notation rounded to the second decimal point, or express it as a fraction simplified to the lowest terms.

$\boxed{0.75 \text{ or } \frac{3}{4}}$

The likelihood weight for $(-w, -d, +e, +f, -h)$ is 0.3, while the likelihood weight for $(+w, +d, +e, +f, +h)$ is 0.9. The answer is then given by $\frac{0.9}{0.9+0.3} = 0.75$

(iii) [1 pt] *True/False*: After 10 iterations, both rejection sampling and likelihood weighting would typically compute equally accurate probability estimates. Each sample counts as an iteration, and rejecting a sample counts as an iteration.

○ True [✓ ● False]

In rejection sampling, we would most likely reject some samples, but we can use all our samples for likelihood weighting.

(c) Suppose we now want to use **Gibbs sampling** to estimate $P(+w|+e)$

(i) [2 pts] We start with the sample $(+w, +d, +e, +h, +f)$, and we want to resample the variable W. What is the probability of sampling $(-w, +d, +e, +h, +f)$? Express your answer in decimal notation rounded to the second decimal point, or express it as a fraction simplified to the lowest terms.

0.10

We would like to estimate $P(-w|+d, +e, +h, +f)$, which is equal to $P(-w|+d, +e)$ from conditional independence assumptions. Based on the given information at the start of question, we get $P(-w|+e, +d) = 0.1$.

(ii) [1 pt] Suppose we observe the following sequence of samples via Gibbs sampling:

$$(+w, +d, +e, +h, +f), (-w, +d, +e, +h, +f), (-w, -d, +e, +h, +f), (-w, -d, +e, -h, +f)$$

What is your estimate of $P(+w|+e)$ given these samples?

0.25

The evidence $+e$ is satisfied by all the samples, and the we have $W = +w$ in only one of them, so the answer is $\frac{1}{4}$, or 0.25.

(iii) [1 pt] While estimating $P(+w|+e)$, the following is a possible sequence of sample that can be obtained via Gibbs sampling:

$$(+w, +d, +e, +h, +f), (-w, +d, +e, +h, +f), (-w, -d, +e, +h, +f), (-w, -d, +e, -h, +f), (-w, -d, -e, -h, +f)$$

○ True [✓ False]

The evidence variable is kept fixed while using Gibbs sampling, which is not satisfied in the last element of the sequence above.

Q6. [11 pts] HMM Smoothing

Consider the HMM with state variables X_t and observation variables E_t. The joint distribution is given by

$$P(X_{1:T}, E_{1:T} = e_{1:T}) = P(X_1) \prod_{t=1}^{T-1} P(X_{t+1}|X_t) \prod_{t=1}^{T} P(E_t = e_t|X_t).$$

where $X_{1:T}$ means $X_1, ..., X_T$ and $E_{1:T} = e_{1:T}$ means $E_1 = e_1, ..., E_T = e_T$. We learned about how the forward algorithm can be used to solve the *filtering* problem, which calculates $P(X_t|E_{1:t} = e_{1:t})$. We will now focus on the *smoothing* problem, which calculates $P(X_t|E_{1:T} = e_{1:T})$, where $1 \leq t < T$, for obtaining a more informed estimate of the past state X_t given all observed evidence $E_{1:T}$.

Now define the following vectors of probabilities:

- $\alpha(X_t) \equiv P(E_{1:t} = e_{1:t}, X_t)$, the probability of seeing evidence $E_1 = e_1$ through $E_t = e_t$ and being in state X_t;
- $\beta(X_t) \equiv P(E_{t+1:T} = e_{t+1:T}|X_t)$, the probability of seeing evidence $E_{t+1} = e_{t+1}$ through $E_T = e_T$ having started in state X_t.

(a) [2 pts] Let us consider $\beta(X_{T-1})$. Which of the following are equivalent to $P(E_T = e_T|X_{T-1})$? (Select one or more.)

[✓] $\sum_{x_t} P(X_T = x_t|X_{T-1})P(E_T = e_T|X_T = x_t)$

[✓] $\sum_{x_t} P(X_T = x_t|X_{T-1})P(E_T = e_T|X_T = x_t, X_{T-1})$

☐ $P(X_T = x_t|X_{T-1})P(E_T = e_T|X_T = x_t)$

☐ $P(X_T = x_t|X_{T-1})P(E_T = e_T|X_T = x_t, X_{T-1})$

$$\begin{aligned} P(E_T = e_T|X_{T-1}) &= \sum_{x_t} P(E_T = e_T, X_T = x_t|X_{T-1}) \\ &= \sum_{x_t} P(X_T = x_t|X_{T-1})P(E_T = e_T|X_T = x_t, X_{T-1}) \\ &= \sum_{x_t} P(X_T = x_t|X_{T-1})P(E_T = e_T|X_{T-1}) \end{aligned}$$

1st line: marginalization

2nd line: chain rule

3rd line: E_T is independent of X_T given X_{T-1}

(b) [4 pts] In lecture we covered the *forward recursion* for filtering. An almost identical algorithm can be derived for computing the sequence $\alpha(X_1), \ldots, \alpha(X_T)$. For β, we need a *backward recursion*. What is the appropriate expression for $\beta(X_t)$ to implement such a recursion? The expression may have up to four parts, as follows:

$$P(E_{t+1} = e_{t+1}, ..., E_T = e_T|X_t) = \boxed{\quad\text{(i)}}\boxed{\quad\text{(ii)}}\boxed{\quad\text{(iii)}}\boxed{\quad\text{(iv)}}$$

For each blank (i) through (iv), mark the appropriate subexpression. If it is possible to write the expression for $\beta(X_t)$ without a particular subexpression, mark "None."

(i) [1 pt] ○ $\sum_{x_{t-1}}$ ○ $\sum_{x_t}$ [✓ $\sum_{x_{t+1}}$] ○ None

(ii) [1 pt] ○ $\alpha(X_{t-1} = x_{t-1})$ ○ $\alpha(X_t = x_t)$ ○ $\alpha(X_{t+1} = x_{t+1})$
○ $\beta(X_{t-1} = x_{t-1})$ [✓ $\beta(X_{t+1} = x_{t+1})$] ○ None

(iii) [1 pt] ○ $P(X_t = x_t|X_{t-1})$ ● $P(X_{t+1} = x_{t+1}|X_t)$
○ $P(X_t|X_{t-1} = x_{t-1})$ ○ $P(X_{t+1}|X_t = x_t)$ ○ None

(iv) [1 pt] ○ $P(E_{t-1} = e_{t-1}|X_{t-1})$ ○ $P(E_t = e_t|X_t)$ ○ $P(E_{t+1} = e_{t+1}|X_{t+1})$
○ $P(E_{t-1} = e_{t-1}|X_{t-1} = x_{t-1})$ ○ $P(E_t = e_t|X_t = x_t)$ ● $P(E_{t+1} = e_{t+1}|X_{t+1} = x_{t+1})$
○ None

$$
\begin{aligned}
&P(E_{t+1:T} = e_{t+1:T}|X_t)\\
&= \sum_{x_{t+1}} P(E_{t+1:T} = e_{t+1:T}, X_{t+1} = x_{t+1}|X_t)\\
&= \sum_{x_{t+1}} P(E_{t+1:T} = e_{t+1:T}|X_{t+1} = x_{t+1}, X_t)P(X_{t+1} = x_{t+1}|X_t)\\
&= \sum_{x_{t+1}} P(E_{t+1:T} = e_{t+1:T}|X_{t+1} = x_{t+1})P(X_{t+1} = x_{t+1}|X_t)\\
&= \sum_{x_{t+1}} P(E_{t+2:T} = e_{t+2:T}|X_{t+1} = x_{t+1})P(E_{t+1} = e_{t+1}|X_{t+1} = x_{t+1}, E_{t+2:T} = e_{t+2:T})P(X_{t+1} = x_{t+1}|X_t)\\
&= \sum_{x_{t+1}} P(E_{t+2:T} = e_{t+2:T}|X_{t+1} = x_{t+1})P(E_{t+1} = e_{t+1}|X_{t+1} = x_{t+1})P(X_{t+1} = x_{t+1}|X_t)\\
&= \sum_{x_{t+1}} \beta(X_{t+1} = x_{t+1}P(E_{t+1} = e_{t+1}|X_{t+1} = x_{t+1})P(X_{t+1} = x_{t+1}|X_t)
\end{aligned}
$$

First equals sign: marginalization

Second equals sign: chain rule

Third equals sign: $E_{t+1:T}$ is independent of X_t given X_{t+1}

Fourth equals sign: chain rule

Fifth equals sign: E_{t+1} is independent of $E_{t+2:T}$ given X_{t+1}

Sixth equals sign: definition of *beta*

Rearranging terms gives us the answer.

(c) [1 pt] If the number of values that each X_t can take on is K and the number of timesteps is T, what is the **total** computational complexity of calculating $\beta(X_t)$ **for all** t, where $1 \leq t \leq T$?

○ $O(K)$ ○ $O(T)$ ● $O(K^2T)$ ○ $O(KT^2)$ ○ $O(K^2T^2)$ ○ None

For each t, we have to sum over every value of X_{t+1} for every value of X_t.

(d) [2 pts] Which of the following expressions are equivalent to $P(X_t = x_t|E_{1:T} = e_{1:T})$? (Select one or more.)

☐ $\sum_{x'_t} \alpha(X_t = x'_t)\beta(X_t = x'_t)$

☐ $\alpha(X_t = x_t)\beta(X_t = x_t)$

■ $\frac{\alpha(X_t=x_t)\beta(X_t=x_t)}{\sum_{x'_t} \alpha(X_t=x'_t)\beta(X_t=x'_t)}$

■ $\frac{\alpha(X_t=x_t)\beta(X_t=x_t)}{\sum_{x'_T} \alpha(X_T=x'_T)}$

We observe that $P(E_{1:T} = e_{1:T}) = \sum_{x'_t} \alpha(X_t = x'_t)\beta(X_t = x'_t) = \sum_{x'_T} \alpha(X_T = x'_T)$. We also observe that $P(E_{1:T} = e_{1:T}, X_t = x_t) = \alpha(X_t = x_t)\beta(X_t = x_t)$. We can use Bayes rule to compute

$$P(X_t = x_t|E_{1:T} = e_{1:T}) = \frac{P(E_{1:T} = e_{1:T}, X_t = x_t)}{P(E_{1:T} = e_{1:T})}$$

(e) [2 pts] If the number of values that each X_t can take on is K and the number of timesteps is T, what is the *lowest* **total** computational complexity of calculating $P(X_t|E_{1:T} = e_{1:T})$ **for all** t, where $1 \le t \le T$?

○ $O(K)$ ○ $O(T)$ [✓ $O(K^2T)$] ○ $O(KT^2)$ ○ $O(K^2T^2)$ ○ None

It takes $O(K^2T)$ time to calculate all of the β's and similarly $O(K^2T)$ time to calculate all of the α's. Based on part (d), it is sufficient to perform a single forward pass to compute all the α's and a single backward pass to compute all the β's in order to calculate $P(X_t|E_{1:T} = e_{1:T})$ for all t, where $1 \le t \le T$. We cannot do better because we need to at least compute $P(E_{1:T} = e_{1:T})$ which requires at least $O(K^2T)$ time.

Q7. [8 pts] Partying Particle #No Filter(ing)

Algorithm 1 Particle Filtering

```
1: procedure PARTICLE FILTERING($T, N$)        ▷ $T$: number of time steps, $N$: number of sampled particles
2:     $x \leftarrow$ sample $N$ particles from initial state distribution $P(X_0)$        ▷ Initialize
3:     for $t \leftarrow 0$ to $T-1$ do        ▷ $X_t$: hidden state, $E_t$: observed evidence
4:         $x_i \leftarrow$ sample particle from $P(X_{t+1}|X_t = x_i)$ for $i = 1, \dots, N$        ▷ Time Elapse Update
5:         $w_i \leftarrow P(E_{t+1}|X_{t+1} = x_i)$ for $i = 1, \dots, N$        ▷ Evidence Update
6:         $x \leftarrow$ resample $N$ particles according to weights w        ▷ Particle Resampling
7:     end for
8:     return $x$
9: end procedure
```

Algorithm 1 outlines the particle filtering algorithm discussed in lecture. The variable x represents a list of N particles, while w is a list of N weights for those particles.

(a) Here, we consider the unweighted particles in x as approximating a distribution.

(i) [1 pt] After executing line 4, which distribution do the particles x represent?

[✓ $P(X_{t+1}|E_{1:t})$] ○ $P(X_{1:t+1}|E_{1:t})$ ○ $P(X_{t+1}|E_{1:t+1})$ ○ None

(ii) [1 pt] After executing line 6, which distribution do the particles x represent?

○ $P(X_{t+1}|X_t, E_{1:t+1})$ ○ $P(X_{1:t+1}|E_{1:t+1})$ [✓ $P(X_{t+1}|E_{1:t+1})$] ○ None

1. Before line 4, x estimated $P(X_t \mid E_{1:t})$. Line 4 is the time elapse update, sampling from the distribution $P(X_{t+1}|X_t = x)$. After the update, x estimates $P(X_{t+1} \mid E_{1:t})$. Note the second choice is estimating too many states, and the third choice depends on E_{t+1} that has yet to be incorporated.
2. (ii) After we sum by state, normalize, and then re-sample, our new states x approximate $P(X_{t+1}|E_{1:t+1})$ which is the distribution we want for HMM's.

(b) The particle filtering algorithm should return a sample-based approximation to the true posterior distribution $P(X_T \mid E_{1:T})$. The algorithm is **consistent** if and only if the approximation converges to the true distribution as $N \to \infty$. In this question, we present several modifications to Algorithm 1. For each modification, indicate if the algorithm is still **consistent** or **not consistent**, and if it is **consistent**, indicate whether you expect it to be **more accurate** in general in terms of its estimate of $P(X_T \mid E_{1:T})$ (i.e., you would expect the estimated distribution to be closer to the true one) or **less accurate**. Assume unlimited computational resources and arbitrary precision arithmetic.

(i) [2 pts] We modify line 6 to sample 1 or $2N-1$ particles with equal probability $p = 0.5$ for each time step (as opposed to a fixed number of particles N). You can assume that $P(E_{t+1}|X_{t+1}) > 0$ for all observations and states. This algorithm is:

○ Consistent and More Accurate
○ Consistent and Less Accurate
[✓ Not Consistent]

This algorithm is not consistent because we will sample only one particle, with probability tending to one. Every time we end up sampling 1 particle, our algorithm can no longer represent the posterior distribution with arbitrary precision even as $N \to \infty$.

(ii) [1 pt] Replace lines 4–6 as follows:
4′: Compute a tabular representation of $P(X_t = s|E_{1:t})$ based on the proportion of particles in state s.
5′: Use the forward algorithm to calculate $P(X_{t+1}|E_{1:t+1})$ exactly from the tabular representation.
6′: Set x to be a sample of N particles from $P(X_{t+1}|E_{1:t+1})$.
This algorithm is:

[✓ Consistent and More Accurate] ○ Not consistent
○ Consistent and Less Accurate

This algorithm is consistent. It is more accurate since the particle filtering algorithm is a consistent approximator while the forward algorithm computes the exact distribution. The forward algorithm might be computationally intractable for large HMMs (a key reason to use particle filtering), but we are instructed to ignore resource limitations.

(iii) [1 pt] At the start of the algorithm, we initialize each entry in w to 1s. Keep line 4, but replace lines 5 and 6 with the following multiplicative update:

```
5′: For $i = 1, \dots, N$ do
6′      $w_i \leftarrow w_i * P(E_{t+1}|X_{t+1} = x_i)$.
```

Finally, **only** at the end of the T iterations, we resample x according to the cumulative weights w just like in line 6, producing a list of particle positions. This algorithm is:

○ Consistent and More Accurate ○ Not Consistent
[✓ Consistent and Less Accurate]

This algorithm is consistent. Indeed, it is equivalent to likelihood weighting, which is consistent. One can also see this result by comparison to the particle filtering algorithm. The time elapse updates are unmodified. The observation updates are weighted by $P(E_t|X_t)$ as in particle filtering. The only difference is that normalization and resampling happens at the end of the loop rather than as part of each observation update. It is less accurate since in practice, the error for fixed N grows exponentially with T, but for fixed T it still converges as N goes to ∞.

(c) [2 pts] Suppose that instead of particle filtering we run the following algorithm on the Bayes net with T time steps corresponding to the HMM:

1. Fix all the evidence variables $E_{1:T}$ and initialize each X_t to a random value x_t.
2. For $i = 1, \ldots, N$ do
   - Choose a variable X_t uniformly at random from $X_1, \ldots, X_T$.
   - Resample X_t according to the distribution $P(X_t | X_{t-1} = x_{t-1}, X_{t+1} = x_{t+1}, E_t = e_t)$.
   - Record the value of X_T as a sample.

Finally, estimate $P(X_T = s | E_{1:T} = e_{1:T})$ by the proportion of samples with $X_T = s$. This algorithm is:

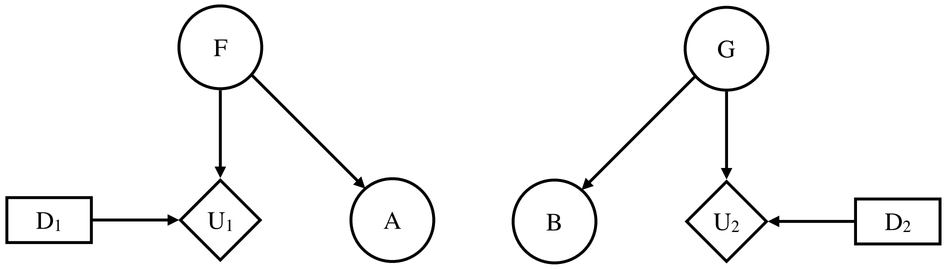
[✓ Consistent] Not Consistent

This is just Gibbs sampling. It will converge to the true posterior distribution $P(X_{1:T} \mid E_{1:T})$, even though the sample only changes when X_T is sampled.

Q8. [11 pts] Double Decisions and VPI

In both parts of this problem, you are given a decision network with **two** decision nodes: D_1 and D_2. Your total utility is the sum of two subparts: $U_{\text{total}} = U_1 + U_2$, where U_1 only depends on decision D_1 and U_2 only depends on decision D_2.

(a) Consider the following decision network:



For each subpart below, select all comparison relations that are true or **could** be true.

(i) [1 pt]

$VPI(\{A,B\})$ ☐ $>$ [✓ $=$] ☐ $<$ $VPI(A) + VPI(B)$

(ii) [1 pt]

$VPI(\{A,F\})$ ☐ $>$ [✓ $=$] [✓ $<$] $VPI(A) + VPI(F)$

(i) Information about variable A only affects the decision D_1 and utility U_1. Information about variable B only affects the decision D_2 and utility U_2. Therefore, the VPI of the two random variables is additive.

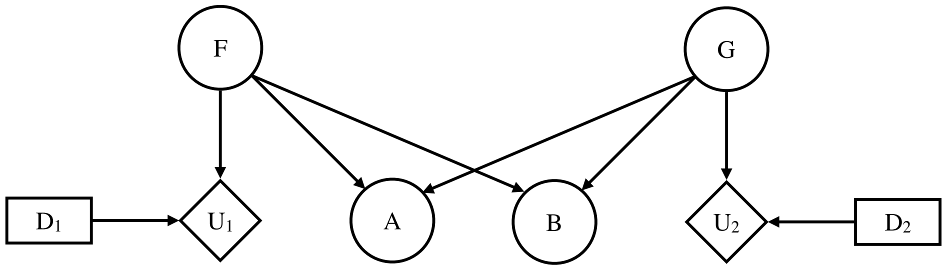
(ii) The node U_1 is d-separated from A, conditioned on F. Moreover, neither A nor F have any affect on utility U_2. This implies that $VPI(\{A,F\}) = VPI(F)$. Since $VPI(F)$ is non-negative, $VPI(\{A,F\}) \leq VPI(A) + VPI(F)$ since $VPI(A) \geq 0$.

(iii) [1 pt]

$VPI(\{B,G\})$ ☐ $>$ [✓ $=$] ☐ $<$ $VPI(G)$

(iii) The node U_2 is d-separated from B, conditioned on G. Moreover, neither B nor G have any affect on utility U_1. This implies that $VPI(\{B,G\}) = VPI(G)$.

(b) Now consider the following decision network:



For each subpart below, select all comparison relations that are true or **could** be true.

(i) [2 pts]

$VPI(\{F, G\})$ □ $>$ ■ $=$ □ $<$ $VPI(F) + VPI(G)$

(ii) [2 pts]

$VPI(\{A, F\})$ ■ $>$ ■ $=$ ■ $<$ $VPI(A) + VPI(F)$

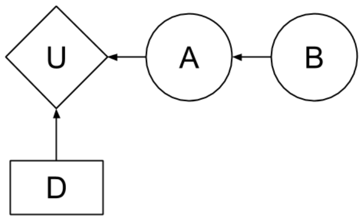
(i) The node F is d-separated from U_2, meaning that the utility U_2 is independent of F. Likewise, the utility U_1 does not depend on G. Therefore (absent any information about A or B), F only influences utility U_1 and G only influences utility U_2, so their VPI is additive.

(ii) Suppose that $G \perp\!\!\!\perp A$ and $G \perp\!\!\!\perp F|A$ (recall that in a Bayes net, there may be additional independence relations other than those implied by the graph structure). In that case, random variable A does not impact utility U_2 at all, and we would have $VPI(\{A, F\}) \leq VPI(A) + VPI(F)$ just like in part (a)(ii) above. Now suppose instead that $G \perp\!\!\!\perp A$ but that F and G are not independent conditioned on A. In that case, A and F together provide information that affects utility U_2, but not separately. We could then have $VPI(\{A, F\}) > VPI(A) + VPI(F)$.

(iii) [2 pts]

$VPI(\{B, G\})$ ■ $>$ ■ $=$ □ $<$ $VPI(G)$

(iii) $VPI(\{B, G\}) = VPI(B|G) + VPI(G)$. Unlike in part (a)(iii) above, $VPI(B|G)$ can be positive because B is no longer d-separated from utility node U_1 (conditioned on G). Since $VPI(B|G) \geq 0$, $VPI(\{B, G\}) \geq VPI(G)$



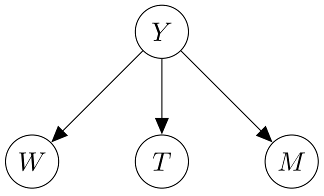
(c) [2 pts] Select all statements that are true. For the first two statements, consider the decision network above.

■ If A, B are independent, then $VPI(B) = 0$
□ If A, B are **guaranteed to be dependent**, then $VPI(B) > 0$
■ In general, Gibbs Sampling can estimate the probabilities used when calculating VPI
□ In general, Particle Filtering can estimate the probabilities used when calculating VPI

Since A, B are independent, then $VPI(B) = 0$. For the second part, even though there is a guaranteed dependence relationship, the VPI may still be zero if decision don't change even after observing evidence (not change in probability, or one decision is always much better than the other). The third selection is true, as you can use gibbs sampling to sample from a bayes net. On the other hand, particle filtering is for HMMs, which you cannot use.

Q9. [11 pts] Naively Fishing

Pacman has developed a hobby of fishing. Over the years, he has learned that a day can be considered fit or unfit for fishing Y which results in three features: whether or not Ms. Pacman can show up M, the temperature of the day T, and how high the water level is W. Pacman models it as the following Naive Bayes classification problem, shown below:



$$\begin{tikzcd} & Y \arrow[dl] \arrow[d] \arrow[dr] & \\ W & T & M \end{tikzcd}$$

(a) We wish to calculate the probability a day is fit for fishing given features of the day. Consider the conditional probability tables that Pacman has estimated over the years:

Y	P(Y)
yes	0.1
no	0.9

M	Y	P(M|Y)
yes	*yes*	0.5
no	*yes*	0.5
yes	*no*	0.2
no	*no*	0.8

W	Y	P(W|Y)
high	*yes*	0.1
low	*yes*	0.9
high	*no*	0.5
low	*no*	0.5

T	Y	P(T|Y)
cold	*yes*	0.2
warm	*yes*	0.2
hot	*yes*	0.5
cold	*no*	0.1
warm	*no*	0.2
hot	*no*	0.6

(i) [1 pt] Using the method of Naive Bayes, what are these conditional probabilities, calculated from the conditional probability tables above? Fill in your final, decimal answer in the boxes below.

$P(Y = \text{yes}|M = \text{yes}, T = \text{cold}, W = \text{high}) =$ $\boxed{0.1}$

$P(Y = \text{no}|M = \text{yes}, T = \text{cold}, W = \text{high}) =$ $\boxed{0.9}$

(ii) [1 pt] Using the method of Naive Bayes, do we predict that the day is fit for fishing if Ms. Pacman is available, the weather is cold, and the water level is high?

○ Fit for fishing [✓ Not fit for fishing]

(b) Assume for this problem we do not have estimates for the conditional probability tables, and that Pacman is still using a Naive Bayes model. Write down an expression for each of the following queries. Express your solution using the conditional probabilities $P(M|T)$, $P(T|Y)$, $P(W|Y)$, and $P(Y)$ from the Naive Bayes model.

(i) [1 pt] Pacman now wishes to find the probability of Ms. Pacman being available or not given that the temperature is hot and the water level is low. Select all expressions that are equal to $P(M|T, W)$.

[✓ $\frac{\sum_f P(f)P(M|f)P(T|f)P(W|f)}{\sum_f P(f)P(W|f)P(T|f)}$] ☐ $\sum_f P(M|f)$ ☐ $\sum_f \frac{P(M|f)P(T|f)P(W|f)}{P(T|f)P(W|f)}$ ☐ None of the above

(ii) [2 pts] Pacman now wants to now choose the class that gives the maximum $P(\text{features}|\text{class})$, that is, choosing the class that maximizes the probability of seeing the features. Write an expression that is equal to $P(M, T, W|Y)$.

$P(M, T, W|Y) =$ $\boxed{}$ $P(M|Y)P(T|Y)P(W|Y)$

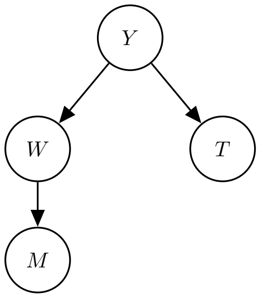
(iii) [2 pts] Assume that Pacman is equally likely to go fishing as he is to not, i.e. $P(Y = yes) = P(Y = no)$. Which method would give the correct Naive Bayes classification of whether a day is a good day for fishing if Pacman observes values for M, T, and W?

[✓ $\arg\max_y P(M, T, W|Y = y)$] [✓ $\arg\max_y P(Y = y|M, T, W)$] ☐ None of the above

(c) Assume Pacman now has the underlying Bayes Net model, and the conditional probability tables from the previous parts do not apply. Recall that predictions are made under the Naive Bayes classification using the conditional probability, $P(Y|W,T,M)$, and the Naive Bayes assumption that features are independent given the class. We wish to explore if the Naive Bayes model is guaranteed to be able to able to represent the true distribution.

For each of the following true distributions, select 1) if the Naive Bayes modeling assumption holds and 2) if the Naive Bayes model is guaranteed to be able to represent the **true conditional probability**, $P(Y|W,T,M)$.

(i) [2 pts]



Does the Naive Bayes modeling assumption hold here?

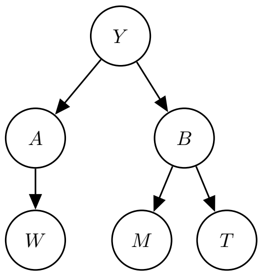
Yes [✓ No]

Can the Naive Bayes model represent the true conditional distribution $P(Y|W,T,M)$?

[✓ Yes] No

The Naive Bayes assumption does not hold as W is not independent of M given the class Y. However, based on the true distribution model, we can conclude that $P(Y|W,M,T) = P(Y|W,T)$. In other words, M is not required to model the conditional distribution. If we set $P(M|Y) = \textit{constant}$ (same for both values of Y), the new Bayes Net model will also have $P(Y|W,M,T) = P(Y|W,T)$.

(ii) [2 pts]



Does the Naive Bayes modeling assumption hold here?

Yes [✓ No]

Can the Naive Bayes model represent the true conditional distribution $P(Y|W,T,M)$?

Yes [✓ No]

The Naive Bayes assumption does not hold as M is not independent of T given the class Y. As for why the Naive Bayes can't represent the true distribution... it's annoying to show so I'm not gonna :P.

Q10. [13 pts] Neural Networks and Decision Trees

(a) Given the Boolean function represented by the truth table on the right, indicate which of the models can perfectly represent this function for some choice of parameters. Where no constraints are stated on the architecture (e.g. the number of neurons, activation function), answer yes if there exists *some* architecture which could represent the function.

(i) [2 pts] $f(x, y) = x \oplus y$ can be modelled by:

☐ A neural network with a single layer (no hidden layer)

🟥 A neural network with two layers (a single hidden layer)

☐ A decision tree of depth one

🟥 A decision tree of depth two

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

A perceptron (no hidden layer neural network) $g(x, y) = \alpha(\beta(x, y))$ where $\beta(x, y) = ax + by + c$ and α is a (monotonic) activation function cannot correctly classify the XOR function. Note $\beta(1, 1) = a + b + c$ and $\beta(0, 0) = c$. WLOG, suppose $a+b+c \geq c$ (we can always flip the signs of all coefficients and reflect g to make this true). Suppose for contradiction that $(0, 0)$ and $(1, 1)$ are both correctly classified, then $\alpha(a+b+c) = 0 = \alpha(a)$. Since activation function α must be monotonic, it follows that all points $\alpha(x)$ for $x \in [c, a+b+c]$ must also be 0. Note that one of $\alpha(1, 0) = a + c$ or $\alpha(0, 1) = b + c$ must lie between $[c, a + b + c]$. Thus at least one of $(1, 0)$ or $(0, 1)$ is misclassifed, giving a contradiction.

By contrast, a neural network with a hidden layer has sufficient capacity to represent XOR. Indeed, the universal approximation theorem shows that a single hidden layer network can represent *any* function given enough neurons. For XOR, a small network suffices: $g(x, y) = \text{sgn}(B \cdot \text{sgn}(A \begin{pmatrix} x \\ y \end{pmatrix}))$ where $\text{sgn}(x)$ is -1 when $x < 0$, 0 when $x = 0$ and $+1$ when $x > 0$ and $A = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$, $B = \begin{pmatrix} 1 & 1 \end{pmatrix}$.

A decision tree of depth one can only split on a single variable. Since XOR depends on the values of both variables, no tree of depth one can represent it.

A decision tree of depth two can represent any two-variable boolean function, including XOR.

(ii) [2 pts] $f(x, y) = \neg(x \vee y)$ can be modelled by:

🟥 A neural network with a single layer (no hidden layer)

🟥 A neural network with two layers (a single hidden layer)

☐ A decision tree of depth one

🟥 A decision tree of depth two

x	y	$\neg(x \vee y)$
0	0	1
0	1	0
1	0	0
1	1	0

A single layer neural network $g(x, y) = \text{sgn}(-x - y + 1)$ classifies $f(x, y)$ correctly. A two layer neural network can also classify it (we can always make the second layer an identity).

A decision tree of depth one cannot represent f since it depends on two variables. A decision tree of depth two can represent it (and any other two-variable boolean function).

(b) Ada is training a single layer (no hidden layer) neural network to predict a one-dimensional value from a one-dimensional input:

$$f(x) = g(Wx + b)$$

where $g(y) = \text{relu}(y) = \max(0, y)$. She initializes her weight and bias parameters to be:

$$W = -1, \quad b = 0.$$

(i) [1 pt] The derivative of the ReLU function takes the form:

$$\text{relu}'(y) = \begin{cases} s & y > u, \\ t & y < u. \end{cases}$$

Select the appropriate choice of s, t and u below.

s is equal to:	t is equal to:	u is equal to:
○ $s = 0$.	[✓ $t = 0$.]	○ $u = -1$.
[✓ $s = 1$.]	○ $t = 1$.	[✓ $u = 0$.]
○ $s = y$.	○ $t = y$.	○ $u = 1$.

ReLU is a piecewise lienar function. At $y \geq 0$, $\text{relu}(y) = y$, which has derivative $\frac{dy}{dy} = 1$. At $y < 0$, $\text{relu}(y) = 0$, which has derivative $\frac{d0}{dy} = 0$. (Note that $y = 0$ has different derivatives taking the limit from the left and right, but this was not required to answer the question.) So:

$$\text{relu}'(y) = \begin{cases} 1 & y > 0, \\ 0 & y < 0. \end{cases}$$

(ii) [2 pts] Compute the following partial derivatives of f with respect to the weight W assuming the same parameters W and b as above.

$\left.\frac{\partial f}{\partial W}\right|_{x=-1} =$ ___-1___ $\left.\frac{\partial f}{\partial W}\right|_{x=1} =$ ___0___

Note $\frac{\partial f}{\partial W} = \text{relu}'(Wx + b) \cdot x$ by the chain rule. At $x = -1$, $Wx + b = 1$ and so $\text{relu}'(Wx + b) = 1$. Thus $\left.\frac{\partial f}{\partial W}\right|_{x=-1} = -1$. At $x = 1$ then $Wx + b = -1$ and $\text{relu}'(Wx + b) = 0$ so $\left.\frac{\partial f}{\partial W}\right|_{x=1} = 0$.

(iii) [2 pts] For inputs $x > 0$, what range of values will $\frac{\partial f}{\partial W}$ take on for the current values of W and b?

___0___ $\leq \frac{\partial f}{\partial W} \leq$ ___0___

Note with $W = -1$ and $b = 0$, $Wx + b < 0$ for all $x > 0$. Thus $\text{relu}'(Wx + b) = 0$ for all $x > 0$. Thus $\frac{\partial f}{\partial W} = 0$ for all $x > 0$.

(iv) [1 pt] Suppose we now use gradient descent to train W and b to minimize a squared loss. Assume the training data consists only of inputs $x > 0$. For the given weight initialization, which of the following activation functions will result in the loss decreasing over time? You may find your answer to (b)(iii) helpful.

☐ Rectified Linear Unit: $g(y) = \text{relu}(y)$.

[✓ Hyperbolic Tangent: $g(y) = \tanh y$.]

[✓ Sigmoid Function: $g(y) = \sigma(y)$.]

When using a ReLU, (b)(iii) shows that there will never be a gradient through W for data points $x > 0$. By similar reasoning there is also never a gradient through b. Accordingly the parameters will never update, and so the loss will stay constant over time. This problem is known as a "dead ReLU", and is a common problem in training neural networks.

Other activation functions such as tanh and σ do not suffer from this problem: although the gradient tends towards zero for extreme values, it is never exactly zero.

(c) [1 pt] Suppose you have found a learning rate α_b that achieves low loss when using batch gradient descent. A learning rate α_s for stochastic gradient descent that achieves similarly low loss would be expected to be:

○ Higher than for batch gradient descent: $\alpha_s > \alpha_b$

[✓ Lower than for batch gradient descent: $\alpha_s < \alpha_b$]

The stochastic gradient estimator is higher variance than the batch gradient, which takes the average gradient across a batch of data points. Accordingly a lower learning rate should be used for stochastic than batch gradient descent.

(d) Your friend Alex is training a deep neural network, AlexNet, to classify images. He observes that the training loss is low, but that loss on a held-out validation dataset is high. Validation loss can be improved by:

(i) [1 pt]

[✓ Decreasing the number of layers.]

○ Increasing the number of layers.

Decreasing the number of layers increases model bias but decreases variance, which has a regularizing effect.

(ii) [1 pt]

[✓ Decreasing the size of each hidden layer.]

○ Increasing the size of each hidden layer.

Decreasing the number of parameters at each layer increases model bias but decreases variance, which has a regularizing effect.

THIS PAGE IS INTENTIONALLY LEFT BLANK