CS 188 Spring 2023 Introduction to Artificial Intelligence Final

- You have 170 minutes.
- The exam is closed book, no calculator, and closed notes, other than four double-sided cheat sheets that you may reference.
- For multiple choice questions,
  - ☐ means mark **all options** that apply
  - ◯ means mark a **single choice**

First name	
Last name	
SID	
Name of person to the right	
Name of person to the left	
Discussion TAs (or None)	

Honor code: "As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others."

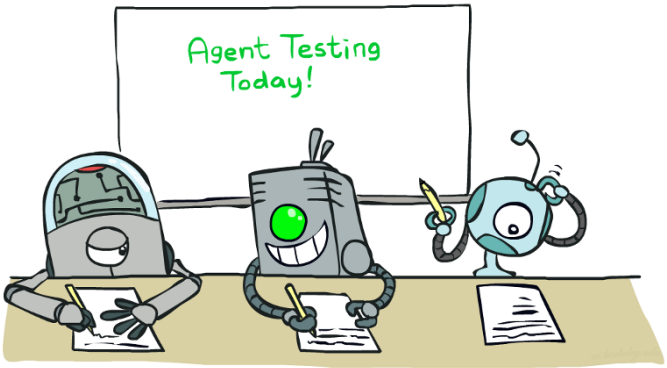
By signing below, I affirm that all work on this exam is my own work, and honestly reflects my own understanding of the course material. I have not referenced any outside materials (other than my cheat sheets), nor collaborated with any other human being on this exam. I understand that if the exam proctor catches me cheating on the exam, that I may face the penalty of an automatic "F" grade in this class and a referral to the Center for Student Conduct.

Signature: _____

Point Distribution

Q1.	Search and Slime	12
Q2.	Searching for a Path with a CSP	11
Q3.	A Multiplayer MDP	9
Q4.	Slightly Different MDPs	13
Q5.	Bayes' Nets	8
Q6.	Dynamic Bayes' Nets: Book Club	10
Q7.	Inequalities of VPI	7
Q8.	Inverted Naive Bayes	13
Q9.	Higher-Dimensional Perceptrons	8
Q10.	Q-Networks	9
	Total	100

It's testing time for our CS188 robots!
Circle your favorite robot below. (ungraded, just for fun)



Q1. [12 pts] Search and Slime

Slowmo the snail is trying to find a path to a goal in an $N \times M$ grid maze. The available actions are to move one square up, down, left, right, and each action costs 1. Moves into walls are impossible. The goal test is a black-box function (the snail does not know how the function works) that returns true if its state argument is a goal state, and false otherwise.

(a) [1 pt] Assume that there is a reachable goal state. Which algorithms are **complete** for this problem? Select all that apply.

☐ Breadth-first tree search ☐ Uniform-cost tree search ☐ A* graph search with $h = 0$
☐ Depth-first tree search ☐ A* tree search with $h = 0$ ◯ None of the above

(b) [1 pt] Assume that there is a reachable goal state. Which algorithms are **optimal** for this problem? Select all that apply.

☐ Breadth-first tree search ☐ Uniform-cost tree search ☐ A* graph search with $h = 0$
☐ Depth-first tree search ☐ A* tree search with $h = 0$ ◯ None of the above

(c) [2 pts] Assume that there *may or may not be* a reachable goal state. Which algorithms are **complete** for this problem? Select all that apply.

☐ Breadth-first tree search ☐ Uniform-cost tree search ☐ A* graph search with $h = 0$
☐ Depth-first tree search ☐ A* tree search with $h = 0$ ◯ None of the above

For the rest of the question, consider a modified version of the search problem: Slowmo the snail hates entering squares that are already covered in his own slime trail. This means that an action is impossible if it would enter a square that Slowmo has previously entered. There *may or may not be* a reachable goal state in this problem.

(d) [2 pts] What is the size of the state space (in terms of N and M) required to correctly reflect this modified problem?

(e) [2 pts] In this modified problem, which algorithms are **complete**? Select all that apply.

☐ Breadth-first tree search ☐ Uniform-cost tree search ☐ A* graph search with $h = 0$
☐ Depth-first tree search ☐ A* tree search with $h = 0$ ◯ None of the above

(f) [2 pts] In this modified problem, which algorithms are **optimal**? Select all that apply.

☐ Breadth-first tree search ☐ Uniform-cost tree search ☐ A* graph search with $h = 0$
☐ Depth-first tree search ☐ A* tree search with $h = 0$ ◯ None of the above

(g) [2 pts] Consider an arbitrary search problem. We want to modify this search problem so that the agent is not allowed to move into a state more than once.

Which of the following changes to the original problem, each considered in isolation, correctly represents this modification? Select all that apply.

☐ Change the successor function to disallow actions that move into a previously-visited state.

☐ Add a list of visited states in the state space, and also change the successor function to disallow actions that move into a previously-visited state.

☐ Change the goal test to return false if a state has been visited more than once.

☐ Add a list of visited states in the state space, and also change the goal test to return false if a state has been visited more than once.

◯ None of the above

Q2. [11 pts] Searching for a Path with a CSP

We have a graph with directed edges between nodes. We want to find a simple path (i.e. no node is visited twice) between node s and node g.

We decide to use a CSP to solve this problem. The CSP has one variable for each edge in the graph: the variable X_{vw} represents whether the edge between node v and node w is part of the path. Each variable has domain {true, false}.

In each of the next three parts, a constraint is described; select the logical expression that represents the constraint. Notation:

- $N(v)$ is the set of all neighbors of node v. (Recall: a node w is a neighbor of v iff there is an edge from v to w).
- The function `ExactlyOne()` returns an expression that is true iff exactly one of its inputs is true.
- $\bigwedge_{x_i \in S} x_i$ refers to the conjunction of all x_i in the set S.
- $\bigvee_{x_i \in S} x_i$ refers to the disjunction of all x_i in the set S.

(a) [2 pts] The starting node s has a single outgoing edge in the path and no incoming edge in the path.

- ◯ $\texttt{ExactlyOne}(\{X_{sv} \mid v \in N(s)\}) \wedge \bigwedge_{v \in N(s)} \neg X_{vs}$
- ◯ $\texttt{ExactlyOne}(\{X_{sv} \mid v \in N(s)\}) \wedge \bigwedge_{v \in N(s)} X_{vs}$
- ◯ $\texttt{ExactlyOne}(\{X_{vs} \mid v \in N(s)\}) \wedge \bigwedge_{v \in N(s)} \neg X_{sv}$
- ◯ $\texttt{ExactlyOne}(\{X_{vs} \mid v \in N(s)\}) \wedge \bigwedge_{v \in N(s)} X_{sv}$

(b) [2 pts] The goal node g has a single incoming edge in the path and no outgoing edge in the path.

- ◯ $\texttt{ExactlyOne}(\{X_{gv} \mid v \in N(s)\}) \wedge \bigwedge_{v \in N(g)} \neg X_{vg}$
- ◯ $\texttt{ExactlyOne}(\{X_{gv} \mid v \in N(s)\}) \wedge \bigwedge_{v \in N(g)} X_{vg}$
- ◯ $\texttt{ExactlyOne}(\{X_{vg} \mid v \in N(s)\}) \wedge \bigwedge_{v \in N(g)} \neg X_{gv}$
- ◯ $\texttt{ExactlyOne}(\{X_{vg} \mid v \in N(s)\}) \wedge \bigwedge_{v \in N(g)} X_{gv}$

(c) [2 pts] If node v has an incoming edge in the path, then it must have exactly one outgoing edge in the path.

- ◯ $\bigwedge_{w \in N(v)} X_{wv} \iff \texttt{ExactlyOne}(\{X_{vw} \mid w \in N(v)\})$
- ◯ $\bigwedge_{w \in N(v)} X_{wv} \implies \texttt{ExactlyOne}(\{X_{vw} \mid w \in N(v)\})$
- ◯ $\bigvee_{w \in N(v)} X_{wv} \iff \texttt{ExactlyOne}(\{X_{vw} \mid w \in N(v)\})$
- ◯ $\bigvee_{w \in N(v)} X_{wv} \implies \texttt{ExactlyOne}(\{X_{vw} \mid w \in N(v)\})$

(d) [2 pts] Suppose the graph has N nodes and M edges. How many different assignments to the variables exist in this CSP? Your answer should be an expression, possibly in terms of N and M.

Suppose we set up our CSP on a graph with nodes s, a, b, c, d, e, g. We want to run a hill climbing algorithm to maximize the number of satisfied constraints. Sideways steps (variable changes that don't change the number of satisfied constraints) are allowed. All variables in the CSP initially set to false.

(e) [2 pts] Assuming that each of the following variables exists in the CSP, select all variables that might be set to true in the first step of running hill-climbing.

☐ X_{sa} ☐ X_{sb} ☐ X_{cs} ☐ X_{ab} ☐ X_{de} ☐ X_{eg} ◯ None of the above

(f) [1 pt] Will our hill-climbing algorithm always find a solution to the CSP if one exists?

- ◯ Yes, the algorithm will incrementally add edges to the path until the path connects s and g.
- ◯ Yes, the algorithm will eventually search over all combinations of assignments to the variables until a satisfying one is found.
- ◯ No, the search will sometimes find a solution to the CSP that is not a valid path.
- ◯ No, the search will sometimes fail to satisfy all of the constraints in the CSP.

Q3. [9 pts] A Multiplayer MDP

Alice and Bob are playing a game on a 2-by-2 grid, shown below. To start the game, a puck is placed on square A.

At each time step, the available actions are:

- Up, Left, Down, Right: These actions move the puck deterministically. Actions that move the puck off the grid are disallowed. These actions give both players a reward of 0.
- Exit: This action ends the game. Note that the Exit action can be taken no matter where the puck is. This action gives Alice and Bob rewards depending on the puck's final location, as shown below.

Each player tries to maximize their own reward, and does not care about what rewards the other player gets.

X	Y
W	Z

Figure 1: The grid that the puck moves on.

Puck location when "exit" is taken	Alice's reward	Bob's reward
W	−1	+3
X	0	0
Y	+1	+2
Z	−1	+2

Figure 2: Exit rewards for Alice and Bob.

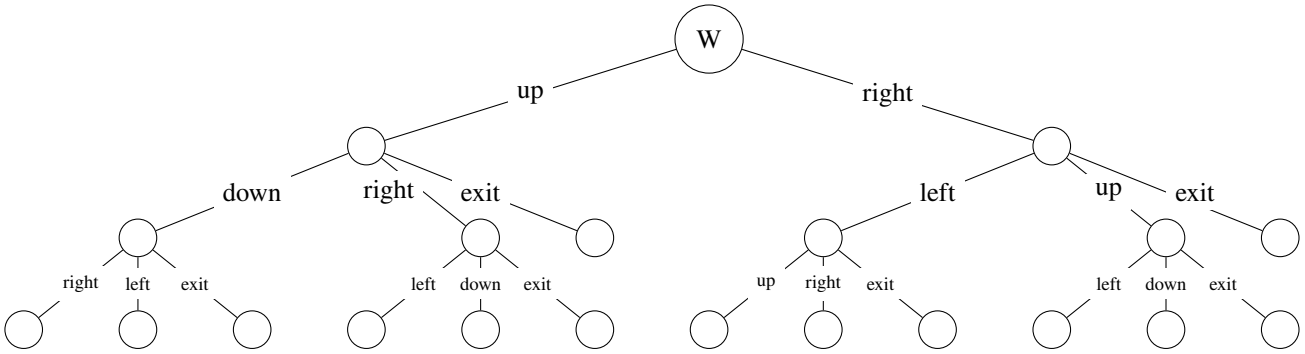
(a) [2 pts] Suppose Bob is the only player taking actions in this game. Bob models this game as an MDP with a discount factor $0 < \gamma \leq 1$.

For which of the following states does Bob's optimal policy depend on the value of γ? Select all that apply.

☐ W ☐ X ☐ Y ☐ Z ○ None of the above.

For the rest of the question, Alice and Bob alternate taking actions. Alice goes first.

Alice models this game with the following game tree, and wants to run depth-limited search with limit 3 turns. She uses an evaluation of 0 for any leaf node that is not a terminal state. (Note: Circles do not necessarily represent chance nodes.)



(b) [1 pt] This is a zero-sum game.

○ True ○ False

(c) [1 pt] What is Alice's value of the root node of the tree?

(d) [1 pt] What is Bob's value of the root node of the tree?

(e) [2 pts] Assuming that Alice and Bob play optimally, what are possible sequences of actions that they might play? Select all that apply.

- ☐ up, right, exit
- ☐ up, right, down
- ☐ right, left, right
- ☐ right, up, exit
- ☐ right, exit
- ◯ None of the above.

Alice instead decides to model the game as an MDP. Assumptions:

- $\gamma = 0.5$
- Alice knows Bob's policy is π.
- $D(s, a)$ represents the new state you move into when you start at state s and take action a.

(f) [2 pts] Fill in the blanks to derive a modified Bellman equation that Alice can use to compute the values of states. Let $s' = D(s, a)$ and $s'' = D(s', \pi(s'))$.

$$V(s) = \max_a R(s, a, s') + \textbf{(i)}\ \textbf{(ii)} + \textbf{(iii)}\ \textbf{(iv)}$$

(i)	◯ 1	◯ 0.5	◯ 0.25
(ii)	◯ 0	◯ $R(s, \pi(s), s')$	◯ $R(s', \pi(s'), s'')$
(iii)	◯ 1	◯ 0.5	◯ 0.25
(iv)	◯ $V(s)$	◯ $V(s')$	◯ $V(s'')$

Q4. [13 pts] Slightly Different MDPs

Each subpart of this question is independent.

For all MDPs in this question, you may assume that:

- The maximum reward for any single action is some fixed number $R > 0$, and the minimum reward is 0.
- The discount factor satisfies $0 < \gamma \leq 1$.
- There are a finite number of possible states and actions.

(a) [2 pts] Which statements are always true? Select all that apply.

- ☐ $\sum_{s \in S} T(s, a, s') = 1$.
- ☐ $\sum_{s' \in S} T(s, a, s') = 1$.
- ☐ $\sum_{a \in A} T(s, a, s') \leq 1$.
- ☐ For all state-action pairs (s, a), there exists some s', such that $T(s, a, s') = 1$.
- ◯ None of the above

(b) [2 pts] Which statements are always true? Select all that apply.

- ☐ Every MDP has a unique set of optimal values for each state.
- ☐ Every MDP has a unique optimal policy.
- ☐ If we change the discount factor γ of an MDP, the original optimal policy will remain optimal.
- ☐ If we scale the reward function $r(s, a)$ of an MDP by a constant multiplier $\alpha > 0$, the original optimal policy will remain optimal.
- ◯ None of the above

(c) [2 pts] Which statements are true? Select all that apply.

- ☐ Policy iteration is guaranteed to converge to a policy whose values are the same as the values computed by value iteration in the limit.
- ☐ Policy iteration usually converges to exact values faster than value iteration.
- ☐ Temporal difference (TD) learning is off-policy.
- ☐ An agent is performing Q-learning, using a table representation of Q (with all Q-values initialized to 0). If this agent takes only optimal actions during learning, this agent will eventually learn the optimal policy.
- ◯ None of the above

(d) [2 pts] We modify the reward function of an MDP by adding or subtracting at most ϵ from each single-step reward. (Assume $\epsilon > 0$.)

We fix a policy π, and compute $V_\pi(s)$, the values of all states s in the original MDP under policy π.

Then, we compute $V'_\pi(s)$, the values of all states in the modified MDP under the same policy π.

What is the maximum possible difference $\left|V'_\pi(s) - V_\pi(s)\right|$ for any state s?

- ◯ ϵ
- ◯ $\gamma\epsilon$
- ◯ $\sum_{n=0}^{\infty} \epsilon(\gamma)^n = \epsilon/(1-\gamma)$
- ◯ ϵR
- ◯ None of the above

(e) [3 pts] We modify the reward function of an MDP by adding exactly C to each single-step reward. (Assume $C > 0$.) Let V and V' be the optimal value functions for the original and modified MDPs. Select all true statements.

- ☐ For all states s, $V'(s) - V(s) = \sum_{n=0}^{\infty} C(\gamma)^n = C/(1-\gamma)$.
- ☐ For all states s, $V'(s) - V(s) = C$.
- ☐ For some MDPs, the difference $V'(s) - V(s)$ may vary depending on s.
- ◯ None of the above

(f) [2 pts] We notice that an MDP's state transition probability $T(s, a, s')$ does **not** depend on the action a.

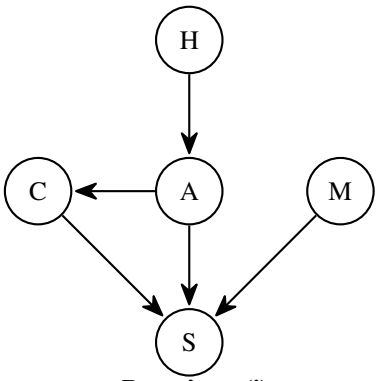
Can we derive an optimal policy for this MDP without computing exact or approximate values (or Q-values) for each state?

- ◯ Yes, because the optimal policy for this MDP can be derived directly from the reward function.
- ◯ Yes, because policy iteration gives an optimal policy and does not require computing values (or Q-values).
- ◯ No, because we need a set of optimal values (or Q-values) to do policy extraction.
- ◯ No, because there is no optimal policy for such an MDP.
- ◯ None of the above

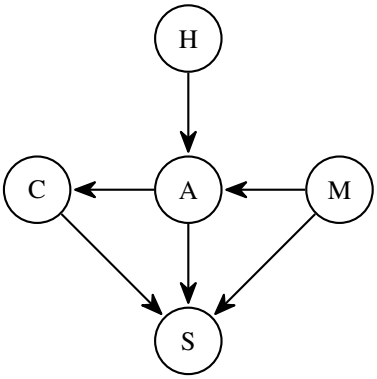
Q5. [8 pts] Bayes' Nets

The head of Pac-school is selecting the speaker (S) for the commencement ceremony, which depends on the student's major (M), academic performance (A), and confidence (C). The student's hard work (H) affects their academic performance (A), and academic performance (A) can affect confidence (C).

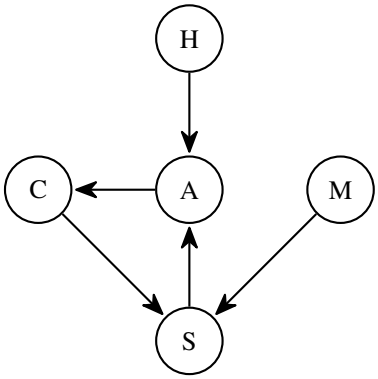
(a) [1 pt] Select all of the well-formed Bayes' nets that can represent a scenario *consistent with* the assertions given above (even if the Bayes' net is not the most efficient representation).



Bayes' net (i)



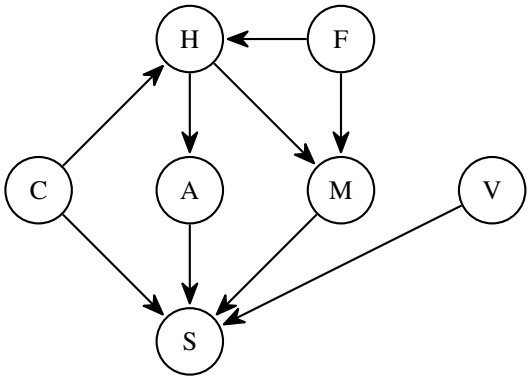
Bayes' net (ii)



Bayes' net (iii)

☐ Bayes' net (i)
☐ Bayes' net (ii)
☐ Bayes' net (iii)
◯ None of the above

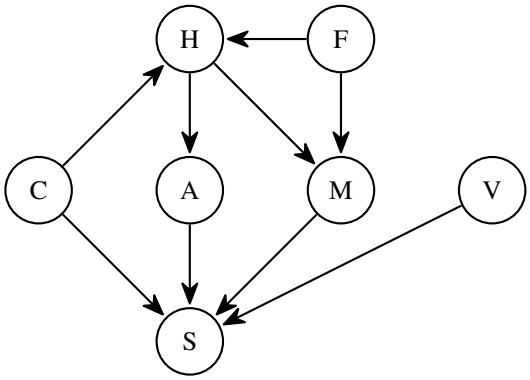
For the rest of the question, consider this Bayes net with added nodes Fearlessness (F) and Voice (V).



(b) [2 pts] Select all the conditional independence expressions that follow from the two standard axioms (1) a variable is independent of its non-descendants given its parents, and (2) a variable is independent of everything else given its Markov blanket.

☐ $F \perp\!\!\!\perp C$
☐ $M \perp\!\!\!\perp C \mid H, F$
☐ $H \perp\!\!\!\perp S \mid C, A, M$
☐ $F \perp\!\!\!\perp A \mid H, M$
◯ None of the above

The Bayes' net, repeated for your convenience:



Assume all the variables have a domain size of 2. Consider the query $P(S|+h,+v,+f)$ in the Bayes net from the previous part.

(c) [1 pt] Suppose the query is answered by variable elimination, using the variable ordering C, A, M. What is the size (number of entries) of the largest factor *generated* during variable elimination? (Ignore the sizes of the initial factors.)

◯ 2^1 ◯ 2^3 ◯ 2^5
◯ 2^2 ◯ 2^4 ◯ 2^6

(d) [2 pts] If we use Gibbs sampling to answer this query, what other variables need to be referenced when sampling M?

☐ H ☐ F ◯ None of the above
☐ A ☐ M
☐ S ☐ C

Now suppose the CPT for variable A is as follows, and assume there are no zeroes in any other CPT.

A	H	$P(A\|H)$
$+a$	$+h$	1
$-a$	$+h$	0
$+a$	$-h$	0
$-a$	$-h$	1

Hint: Recall that one requirement for Gibbs sampling to converge is that, in the limit of infinitely many samples, every state with non-zero probability is sampled infinitely often, regardless of the initial state.

(e) [1 pt] Will Gibbs sampling converge to the correct answer for the query $P(S \mid +f)$?

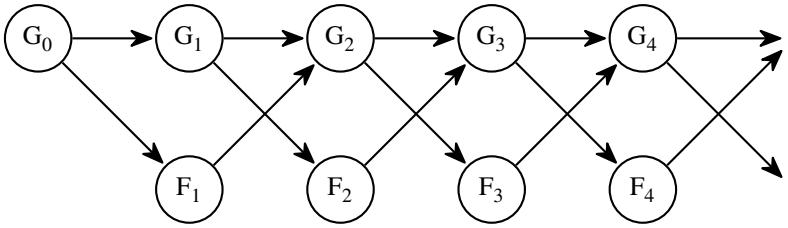
◯ Yes, because convergence properties of Gibbs sampling depend only on the structure of Bayes net, not on the values in the CPTs.
◯ Yes, but it will converge very slowly because of the 0s and 1s in the CPT.
◯ No, because some states with non-zero probability are unreachable from other such states.
◯ No, because Gibbs sampling always fails with deterministic CPTs like this one.
◯ None of the above

(f) [1 pt] Will Gibbs sampling converge to the correct answer for the query $P(S \mid +h,+v,+f)$?

◯ Yes, because convergence properties of Gibbs sampling depend only on the structure of Bayes net, not on the values in the CPTs.
◯ Yes, but it will converge very slowly because of the 0s and 1s in the CPT.
◯ No, because some states with non-zero probability are unreachable from other such states.
◯ No, because Gibbs sampling always fails with deterministic CPTs like this one.
◯ None of the above

Q6. [10 pts] Dynamic Bayes' Nets: Book Club

Each week t ($t \geq 0$), the members of a book club decide on the genre of the book G_t to read that week: romance ($+g_t$) or science fiction ($-g_t$). Each week's choice is influenced by the genre chosen in the previous week. In addition, after week t ($t \geq 0$), they collect feedback (F_{t+1}) from the members, which can be overall positive ($+f_{t+1}$) or negative ($-f_{t+1}$), and that influences the choice at week $t+2$. The situation is shown in the Bayes' net below:



(a) [1 pt] Recall that the joint distribution up to time T for the "standard" HMM model with state X_t and evidence E_t is given by $P(X_{0:T}, E_{1:T}) = P(X_0) \prod_{t=1}^{T} P(X_t \mid X_{t-1}) P(E_t \mid X_t)$.

Write an expression for the joint distribution in the model shown above.

$P(F_{1:T}, G_{0:T}) =$

(b) [2 pts] Which of the following Markov assumptions are implied by the network structure, assuming $t \geq 2$?

- ☐ $P(G_t \mid G_{0:t-1}, F_{1:t-1}) = P(G_t \mid F_{t-1})$
- ☐ $P(G_t \mid G_{0:t-1}, F_{1:t-1}) = P(G_t \mid G_{t-1})$
- ☐ $P(G_t, F_t \mid G_{0:t-1}, F_{1:t-1}) = P(G_t, F_t \mid G_{t-1})$
- ☐ $P(G_t, F_t \mid G_{0:t-1}, F_{1:t-1}) = P(G_t, F_t \mid G_{t-1}, F_{t-1})$
- ☐ $P(F_t \mid G_{0:t-1}, F_{1:t-1}) = P(F_t \mid F_{t-1})$
- ☐ $P(F_t \mid G_{0:t-1}, F_{1:t-1}) = P(F_t \mid G_{t-1}, F_{t-2})$
- ○ None of the above

The conditional probability tables for G_t and F_t are shown below, where a, b, c, d, p, q are constants between 0 and 1.

F_t	G_{t-1}	$P(F_t \mid G_{t-1})$
+	+	p
−	+	$1-p$
+	−	q
−	−	$1-q$

G_t	F_{t-1}	G_{t-1}	$P(G_t \mid F_{t-1}, G_{t-1})$
+	+	+	a
−	+	+	$1-a$
+	+	−	b
−	+	−	$1-b$
+	−	+	c
−	−	+	$1-c$
+	−	−	d
−	−	−	$1-d$

Let's consider this model as a Markov chain with state variables (F_t, G_t).

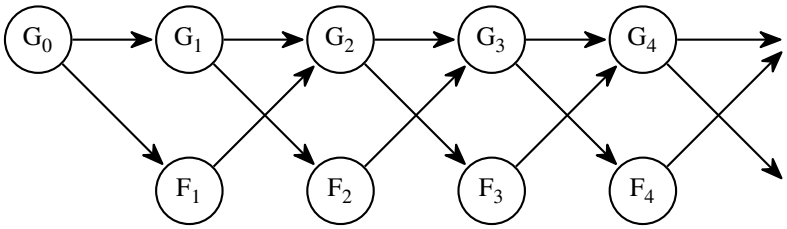
(c) [2 pts] Write out the transition probabilities below.

Your answer should be an expression, possibly in terms of a, b, c, d, p, and q.

$x_1 = P(+f_t, +g_t \mid +f_{t-1}, +g_{t-1}) =$ $\quad x_2 = P(+f_t, +g_t \mid +f_{t-1}, -g_{t-1}) =$

$x_3 = P(+f_t, +g_t \mid -f_{t-1}, +g_{t-1}) =$ $\quad x_4 = P(+f_t, +g_t \mid -f_{t-1}, -g_{t-1}) =$

The Bayes' net, repeated for your convenience:



(d) [2 pts] As t goes to infinity, the stationary distribution of this Markov chain is shown below, where y_1, y_2, y_3, y_4 are constants between 0 and 1.

$$y_1 = P(+f_\infty, +g_\infty)$$
$$y_2 = P(+f_\infty, -g_\infty)$$
$$y_3 = P(-f_\infty, +g_\infty)$$
$$y_4 = P(-f_\infty, -g_\infty)$$

Write an equation that must be true if this is a stationary distribution.

Your answer should be an equation, possibly in terms of x_1, x_2, x_3, x_4 (from the previous part), y_1, y_2, y_3, y_4.

(e) [3 pts] Select all true statements.

☐ There are values of a, b, c, d such that the feedback model defined by p, q is irrelevant to the stationary distribution of the genre G.

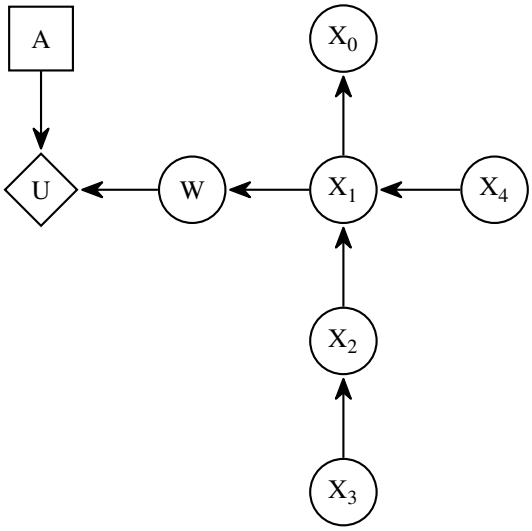
☐ Even if $P(F_t, G_t)$ reaches a unique stationary distribution, it is possible that the marginal probabilities $P(F_t)$ and $P(G_t)$ do not reach unique stationary distributions.

☐ For any values of a, b, c, d, p, q, there is at least one stationary distribution.

◯ None of the above

Q7. [7 pts] Inequalities of VPI

Consider the decision network shown below.



Choose the single equality or inequality symbol that results in the strongest assertion that is necessarily true in this network. (For example, if $a = b$ and $a \geq b$ are both necessarily true, choose $a = b$ because it is a strictly stronger assertion than $a \geq b$.)

(a) [1 pt] VPI(X_1, X_2, X_3) _____ VPI(X_1)

○ = ○ > ○ ≥ ○ < ○ ≤ ○ Not enough information

(b) [1 pt] VPI(X_1) _____ VPI(W)

○ = ○ > ○ ≥ ○ < ○ ≤ ○ Not enough information

(c) [1 pt] VPI(X_1, X_2) _____ VPI(X_2) + VPI(X_1)

○ = ○ > ○ ≥ ○ < ○ ≤ ○ Not enough information

(d) [1 pt] VPI(X_3, X_4) _____ VPI(X_3) + VPI(X_4)

○ = ○ > ○ ≥ ○ < ○ ≤ ○ Not enough information

(e) [1 pt] VPI($X_1, X_2|X_3$) _____ VPI($X_2|X_3$)

○ = ○ > ○ ≥ ○ < ○ ≤ ○ Not enough information

(f) [2 pts] Under which of the following circumstances would VPI(**X**) be 0 for *every* set of random variables **X**? Select all that apply.

☐ U is a deterministic function of A and W.

☐ U is independent of A given W.

☐ U is independent of W given A.

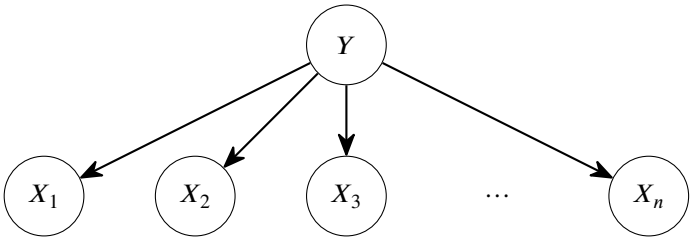
☐ The best action choice, given W, is the same for each value of W.

☐ The expected utility of the best action choice, given W, is the same for each value of W.

○ None of the above

Q8. [13 pts] Inverted Naive Bayes

Consider a standard naive Bayes model with $n > 10$ Boolean features $X_1, \ldots, X_n$ and a Boolean class variable Y. In this question, Boolean variables have values 0 or 1.



(a) [1 pt] How many parameters do we need to learn in this model (not counting parameters that can be derived by the sum-to-1 rule)?

Example of sum-to-1 rule: Given $P(Y = 0)$, we can compute $P(Y = 1)$ since we know that these two values sum to 1. Therefore, $P(Y = 0)$ and $P(Y = 1)$ only count as one parameter we need to learn.

Your answer should be an expression, possibly in terms of n.

(b) [2 pts] We observe one training example with features $x_1, \ldots, x_n$ and class y. Fill in the blanks to derive the likelihood of this training example.

$$L = P(x_1, \ldots, x_n, y) = P(Y = 1)^{\textbf{(i)}}\ P(Y = 0)^{\textbf{(ii)}} \prod_{i=1}^{n} P(X_i = 1|y)^{\textbf{(iii)}}\ P(X_i = 0|y)^{\textbf{(iv)}}$$

(i)	◯	x	◯	$1 - x$	◯	y	◯	$1 - y$
(ii)	◯	x	◯	$1 - x$	◯	y	◯	$1 - y$
(iii)	◯	x	◯	$1 - x$	◯	y	◯	$1 - y$
(iv)	◯	x	◯	$1 - x$	◯	y	◯	$1 - y$

(c) [1 pt] Suppose that this training example was missing a feature value, so we only observed $x_2, \ldots, x_n$ and class y.

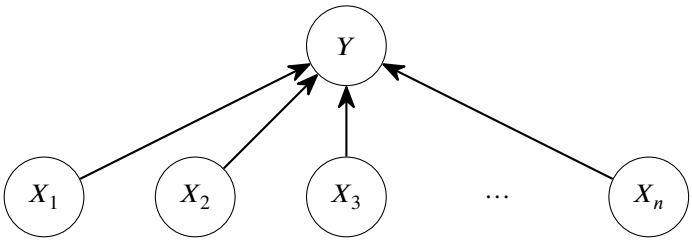
How is the likelihood with a missing feature $L' = P(x_2, \ldots, x_n, y)$, related to the original likelihood L?

◯ L' is equal to L, but with any terms involving $P(X_1|y)$ dropped.

◯ L' is equal to L, with an extra term related to the prior probabilities $P(X_1)$, and any terms involving $P(X_1|y)$ dropped.

◯ The correct expression for L' cannot be determined from L.

Suppose we "invert" the naive Bayes model so that the arrows point from the feature variables to the class variable:



(d) [1 pt] How many parameters do we need to learn in this model (not counting parameters that can be derived by the sum-to-1 rule)?

Your answer should be an expression, possibly in terms of n.

For the rest of this question, Boolean variables are represented as positive or negative (e.g. $+y$ and $-y$).

Suppose you have all the parameters from the original naive Bayes model. Using any relevant parameters from the original model, derive the following parameters in the inverted naive Bayes' model, so that the two models represent the same joint distribution.

Your answers should be expressions, possibly in terms of $P(+y)$, $P(-y)$, $P(+x_i|+y)$, $P(-x_i|+y)$, $P(+x_i|-y)$, and $P(-x_i|-y)$, for $1 \leq i \leq n$. Hint: You can also use summations $\sum$ and products $\prod$.

(e) [2 pts] $P(+x_i) =$

(f) [2 pts] $P(+y|+x_1, +x_2, \ldots, +x_n) \propto$

(g) [3 pts] Select all true statements.

☐ For any set of parameters in the original naive Bayes model, there is some set of parameters in the inverted naive Bayes model that captures the same joint distribution.

☐ For any set of parameters in the inverted naive Bayes model, there is some set of parameters in the original naive Bayes model that captures the same joint distribution.

☐ The inverted naive Bayes model will typically require far more training data than the original naive Bayes model to achieve the same level of test accuracy.

○ None of the above

(h) [1 pt] What learning behavior should we expect to see with this inverted naive Bayes' model?

○ High training accuracy, high test accuracy

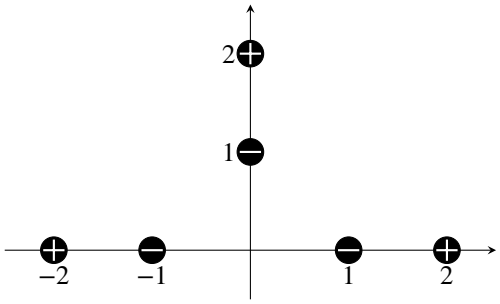
○ High training accuracy, low test accuracy

○ Low training accuracy, high test accuracy

○ Low training accuracy, low test accuracy

Q9. [8 pts] Higher-Dimensional Perceptrons

Consider a dataset with 6 points on a 2D coordinate grid Each point belongs to one of two classes. Points $[-1, 0], [1, 0], [0, 1]$ belong to the negative class. Points $[-2, 0], [2, 0], [0, 2]$ belong to the positive class.



(a) [1 pt] Suppose we run the perceptron algorithm with the initial weight vector set to $[0, 5]$.

What is the updated weight vector after processing the data point $[0, 1]$?

(b) [1 pt] How many iterations of the perceptron algorithm will run before the algorithm converges? Processing one data point counts as one iteration. If the algorithm never converges, write ∞.

In the next few subparts, we'll consider *transforming* the data points by applying some modification to each of the data points. Then, we pass these modified data points into the perceptron algorithm.

For example, consider the transformation $[x, y] \rightarrow [x, y, x^2, 1]$. In this transformation, we add two extra dimensions: one whose value is always the square of the first coordinate, and one whose value is always the constant 1. For example, the point at $[2, 0]$ is transformed into a point at $[2, 0, 4, 1]$ in 4-dimensional space.

(c) [2 pts] Which of the following data transformations will cause the perceptron algorithm to converge, when run on the transformed data? Select all that apply.

- ☐ $[x, y] \rightarrow [y, x]$
- ☐ $[x, y] \rightarrow [x, y, 1]$
- ☐ $[x, y] \rightarrow [x, y, x^2, 1]$
- ☐ $[x, y] \rightarrow [x, y, x^2 + y^2]$
- ◯ None of the above.

(d) [2 pts] Suppose we transform $[x, y]$ to $[x, y, x^2 + y^2, 1]$, and pass the transformed data points into the perceptron.

Write one possible weight vector that the perceptron algorithm may converge to.

(e) [2 pts] Construct another transformation (not equal to the ones above) that will allow the perceptron algorithm to converge.

Hint: The transformation $[x, y] \rightarrow [x, y, x^2 + y^2, 1]$ allows the perceptron algorithm to converge.

Fill in the blank: $[x, y] \rightarrow [x, y, \_\_\_, 1]$.

Exam continues on next page.

Q10. [9 pts] Q-Networks

Consider running Q-learning on the following Pacman problem: the maze is an x-by-x square, and each position can contain a food pellet or no food pellet. There are no ghosts or walls. Pacman's only actions are $\{\text{up}, \text{down}, \text{left}, \text{right}\}$.

(a) [2 pts] How many Q-values do we need to learn for this problem?

Your answer should be an expression, possibly in terms of x.

To learn every Q-value, we could run standard Q-learning, but we decide to try a different approach:

Suppose somebody tells us N exact Q-values for N different state-action pairs: the exact Q-value for the state-action pair (s_i, a_i) is q_i, for $1 \leq i \leq N$. (N is less than the total number of state-action pairs.)

We decide to use these exact Q-values to train a neural network, so that we can estimate other Q-values we don't know. To train this neural network, we need to apply gradient descent to minimize the following loss function:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} (f(\theta, s_i, a_i) - q_i)^2$$

θ represents the weights of the neural network. $f(\theta, s_i, a_i)$ represents running the neural network with weights θ on state-action pair (s_i, a_i).

(b) [1 pt] What is the gradient $\frac{\partial L}{\partial \theta}$?

Your answer should be an expression, possibly in terms of N, $\frac{\partial f}{\partial \theta}$, and q_i.

$\frac{\partial L}{\partial \theta} =$

(c) [1 pt] After running t iterations of gradient descent, our current weights are θ_t. The learning rate is α.

What are the weights on the next iteration, θ_{t+1}?

Your answer should be an expression, possibly in terms of θ_t, α, and $\frac{\partial L}{\partial \theta}$.

$\theta_{t+1} =$

(d) [2 pts] Eventually, gradient descent converges to the weights θ^*.

We use the neural network with weights θ^* to compute Q-values, and extract a policy out of these Q-values:

$$\pi(s) = \arg\max_a f(\theta^*, s, a)$$

Is π the optimal policy for this problem?

◯ Yes ◯ No ◯ Not enough information

Instead of the Pacman problem, consider a different problem where the action space is continuous. In other words, there are infinitely many actions available from a given state.

(e) [3 pts] Can we still use the strategy from the previous subparts (without any modifications) to obtain a policy π?

◯ Yes ◯ No

Briefly explain why or why not.