CS 188 Fall 2011 — Introduction to Artificial Intelligence — Final Exam

INSTRUCTIONS

- You have 3 hours.
- The exam is closed book, closed notes except two pages of crib sheets.
- Please use non-programmable calculators only.
- 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.

Last Name	
First Name	
SID	
Login	
All the work on this exam is my own. **(please sign)**	

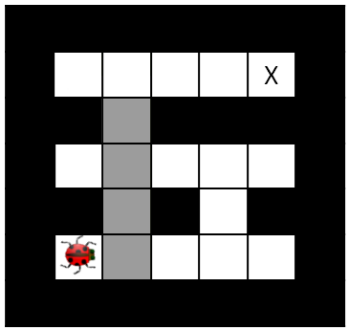
For staff use only

Q. 1	Q. 2	Q. 3	Q. 4	Q. 5	Q. 6	Q. 7	Q. 8	Total
/24	/12	/12	/8	/8	/14	/10	/12	/100

THIS PAGE INTENTIONALLY LEFT BLANK

1. (24 points) Question Menagerie

Parts (a) and (b) The hive of insects needs your help again. As before, you control an insect in a rectangular maze-like environment with dimensions $M \times N$, as shown to the right. At each time step, the insect can move into a free adjacent square or stay in its current location. All actions have cost 1.



In this particular case, the insect must pass through a series of partially flooded tunnels. Flooded squares are lightly shaded in the example map shown. The insect can hold its breath for A time steps in a row. Moving into a flooded square requires your insect to expend 1 unit of air, while moving into a free square refills its air supply.

(a) (2 pt) Give a minimal state space for this problem (i.e. do not include extra information). You should answer for a general instance of the problem, not the specific map shown.
A tuple of location coordinates $m \in \{1, \ldots, M\}$ and $n \in \{1, \ldots, N\}$ and the remaining air supply $a \in \{1, \ldots, A\}$.

(b) (1 pt) Give the size of your state space. $M \times N \times A$

Parts (c), (d), and (e) Consider a search problem where all edges have cost 1 and the optimal solution has cost C. Let h be a heuristic which is $\max\{h^* - k, 0\}$, where h^* is the actual cost to the closest goal and k is a nonnegative constant.

(c) (2 pt) Circle all of the following that are true (if any).

(i) h is admissible. h^* is admissible and $h \leq h^*$.

(ii) h is consistent. h^* is consistent and subtracting a constant from both sides does not change the underlying inequality.

(iii) A* tree search (no closed list) with h will be optimal. Tree search requires admissibility.

(iv) A* graph search (with closed list) with h will be optimal. Graph search requires admissibility and consistency.

(d) (1 pt) Which of the following is the most reasonable description of how much more work will be done (= how many more nodes will be expanded) with heuristic h compared to h^*, as a function of k?

(i) Constant in k

(ii) Linear in k

(iii) Exponential in k

(iv) Unbounded

At $k = 0$, only the d nodes on an optimal path to the closest goal are expanded for search depth (= optimal path length) d. At $k = \max(h)$, the problem reduces to uninformed search and BFS expands b^d nodes for branching factor b. In general, all nodes within distance k of the closest goal will have heuristic $h = 0$ and uninformed search may expand them. Note that search reduces to BFS since A^* with $h = 0$ is UCS and in this search problem all edges have cost 1 so path cost = path length.

Now consider the same search problem, but with a heuristic h' which is 0 at all states that lie along an optimal path to a goal and h^* elsewhere.

(e) **(2 pt)** Circle all of the following that are true (if any).

(i) h' is admissible.

(ii) h' is consistent.

(iii) A* tree search (no closed list) with h' will be optimal.

(iv) A* graph search (with closed list) with h' will be optimal.

h' is not consistent as needed for optimality of graph search. Consistency is violated by any edge connecting a state outside of the optimal path to the optimal path since h' drops faster than the reduction in the true cost.

(f) **(2 pt)** You are running minimax in a large game tree. However, you know that the minimax value is between $x - \epsilon$ and $x + \epsilon$, where x is a real number and ϵ is a small real number. You know nothing about the minimax values at the other nodes. Describe briefly but precisely how to modify the alpha-beta pruning algorithm to take advantage of this information. Initialize the alpha and beta values at the root to $\alpha = x - \epsilon$ and $\beta = x + \epsilon$ then run alpha-beta as normal. This will prune all of the game tree out of our known minimax bounds.

(g) **(2 pt)** An agent prefers chocolate ice cream over strawberry and prefers strawberry over vanilla. Moreover, the agent is indifferent between deterministically getting strawberry and a lottery with a 90% chance of chocolate and a 10% chance of vanilla. Which of the following utilities can capture the agent's preferences?

(i) Chocolate 2, Strawberry 1, Vanilla 0

(ii) Chocolate 10, Strawberry 9, Vanilla 0

(iii) Chocolate 21, Strawberry 19, Vanilla 1

(iv) No utility function can capture these preferences.

One can verify that (ii) captures the agent's preferences. Next, observe that (iii) is an affine transformation of the utilities of (ii) and so does not change the preference ordering.

(h) **(2 pt)** Fill in the joint probability table so that the binary variables X and Y are independent.

X	Y	$P(X,Y)$
$+$	$+$	$3/5$
$+$	$-$	$1/5$
$-$	$+$	$3/20$
$-$	$-$	$1/20$

Independence requires that $P(X,Y) = P(X)P(Y)$.

(i) **(1 pt)** Suppose you are sampling from a conditional distribution $P(B|A = +a)$ in a Bayes' net. Give the fraction of samples that will be *accepted* in rejection sampling. $P(A = +a)$ since samples that do not match the evidence are rejected.

(j) **(2 pt)** One could simplify the particle filtering algorithm by getting rid of the resampling phase and instead keeping weighted particles at all times, with the weight of a particle being the product of all observation probabilities $P(e_i|X_i)$ up to and including the current timestep. Circle all of the following that are true (if any).

(i) This will always work as well as standard particle filtering.

(ii) This will generally work less well than standard particle filtering because all the particles will cluster in the most likely part of the state space.

(iii) This will generally work less well than standard particle filtering because most particles will end up in low-likelihood parts of the state space.

(iv) This will generally work less well than standard particle filtering because the number of particles you have will decrease over time.

Without resampling particle filters tend to give degenerate estimates of the distribution as only a few particles in high-likelihood states will have a significant weight. Resampling helps by re-normalizing the distribution.

(k) **(2 pt)** Circle all settings in which particle filtering is preferable to exact HMM inference (if any).

(i) The state space is very small.

(ii) The state space is very large.

(iii) Speed is more important than accuracy.

(iv) Accuracy is more important than speed.

Exact HMM inference scales with the state space. For particle filtering, the number of particles determines the time complexity and quality of the approximation. Large (even continuous) state spaces can be appromixated by a much smaller number of particles.

(l) **(2 pt)** Circle all of the following statements that are true (if any) about the perceptron and MIRA (with capacity $C = \infty$).

(i) Immediately after updating on a missed example, perceptron will classify that example correctly.

(ii) Immediately after updating on a missed example, MIRA will classify that example correctly.

(iii) On a missed example, from the same starting weight vector, the perceptron might make an update with a larger step size than MIRA.

(iv) On a missed example, from the same starting weight vector, MIRA might make an update with a larger step size than the perceptron.

(v) Immediately after updating on a missed example, perceptron will have a lower training error rate.

(vi) Immediately after updating on a missed example, MIRA will have a lower training error rate.

i. The perceptron learning rule has a $0-1$ notion of error and not update according to the margin of misclassification. It is not guaranteed to correctly classify after a single update. ii. The MIRA update for a missed example optimizes the weights to correctly classify the mistake. There are no guarantees on the relative step sizes or single update training error.

Parts (m) and (n) You have classification data with classes $Y \in \{+1, -1\}$ and features $F_i \in \{+1, -1\}$ for $i \in \{1, \ldots, K\}$. In an attempt to turbocharge your classifier, you duplicate each feature, so now each example has $2K$ features, with $F_{K+i} = F_i$ for $i \in \{1, \ldots, K\}$. The following questions compare the *original* feature set with the *doubled* one. You may assume that in the case of ties, class $+1$ is always chosen. Assume that there are equal numbers of training examples in each class.

(m) **(2 pt)** For a Naïve Bayes classifier, circle all of the following that apply.

(i) The test accuracy could be higher with the original features.

(ii) The test accuracy could be higher with the doubled features.

(iii) The test accuracy will be the same with either feature set.

(iv) On a given training instance, the conditional probability $P(Y|F_1, \ldots)$ on a training instance could be more extreme (i.e. closer to 0 or 1) with the original features.

(v) On a given training instance, the conditional probability $P(Y|F_1, \ldots)$ on a training instance could be more extreme (i.e. closer to 0 or 1) with the doubled features.

(vi) On a given training instance, the conditional probability $P(Y|F_1, \ldots)$ on a training instance will be the same with either feature set.

Naïve Bayes makes the conditional independence assumption that all features are independent given the class label. Redundant features lead to "overconfidence" that may result in errors.

(n) **(1 pt)** For a perceptron classifier, circle all of the following that apply.

(i) The test accuracy could be higher with the original features.

(ii) The test accuracy could be higher with the doubled features.

(iii) The test accuracy will be the same with either feature set.

The perceptron classification rule is $h(x) = \text{sign} \sum_i w_i f_i(x)$ so doubling the features gives $h(x) = \text{sign} \sum_i 2w_i f_i(x) = \text{sign}\, 2 \sum_i w_i f_i(x) = \text{sign} \sum_i w_i f_i(x)$ so it is unchanged by feature duplication.

2. (12 points) Consistency and Harmony

(a) (2 pt) Consider a simple two-variable CSP, with variables A and B, both having domains $\{1, 2, 3, 4\}$. There is only one constraint, which is that A and B sum to 6. In the table below, circle the the values that remain in the domains after enforcing arc consistency.

Variable	Values remaining in domain
A	{ 1 , 2 , 3 , 4 }
B	{ 1 , 2 , 3 , 4 }

Now consider a general CSP C, with variables X_i having domains D_i. Assume that all constraints are between pairs of variables. C is not necessarily arc consistent, so we enforce arc consistency in the standard way, i.e. by running AC-3 (the arc consistency algorithm from class). For each variable X_i, let D_i^{AC} be its resulting arc consistent domain.

(b) (2 pt) Circle all of the following statements that are *guaranteed* to be true about the domains D_i^{AC} compared to D_i.

(i) For all i, $D_i^{AC} \subseteq D_i$.

(ii) If C is initially arc consistent, then for all i, $D_i^{AC} = D_i$.

(iii) If C is not initially arc consistent, then for at least one i, $D_i^{AC} \neq D_i$.

(c) (2 pt) Let n be the number of solutions to C and let n^{AC} be the number of solutions remaining after arc consistency has been enforced. Circle all of the following statements that *could* be true, if any.

(i) $n^{AC} = 0$ but no D_i^{AC} are empty.

(ii) $n^{AC} \geq 1$ but some D_i^{AC} is empty.

(iii) $n^{AC} \geq 1$ but a backtracking solver could still backtrack.

(iv) $n^{AC} > 1$

(v) $n^{AC} < n$

(vi) $n^{AC} = n$

(vii) $n^{AC} > n$

Arc consistency does not imply variables are jointly-consistent as in path consistency or other k consistencies so (i) and (iii) are possible. Multiple solutions could be possible no matter what the consistency so (iv) is true. The CSP has a certain number of solutions regardless of whether or not a particular particular assignment is arc consistent or not, so (vi) could be true.

Consider an alternate, more aggressive property that can hold for an arc $X \to Y$ of a CSP. We say that $X \to Y$ is *arc harmonic* if **for all** $x \in X$, **for all** $y \in Y$, (x, y) satisfies any constraint that exists between X and Y. Just as with arc consistency, we can make a single arc harmonic by deleting domain values. Also as with arc consistency, we can enforce arc harmony over an entire CSP by processing one non-harmonic arc at a time, making it harmonic.

(d) (2 pt) Consider again the simple two-variable CSP, with variables A and B, both having domains $\{1, 2, 3, 4\}$ with the constraint that A and B should sum to 6. Specify reduced (but non-empty) domains for which the CSP is arc harmonic by circling remaining values in the table below. Note that, unlike with arc consistency, there will in general be multiple ways to make arcs harmonic, but you only need to specify one possibility here.

Variable	Values remaining in domain
A	{ 1 , 2 , 3 , 4 }
B	{ 1 , 2 , 3 , 4 }

After we have enforced arc harmony in the general CSP C, let D_i^{AH} be the reduced domain of the variable X_i, and let n^{AH} be the number of possible solutions remaining for this reduced CSP.

(e) (2 pt) Circle all of the following statements that *could* be true, if any.

(i) $n^{AH} = 0$ but no D_i^{AH} are empty.

(ii) $n^{AH} \geq 1$ but some D_i^{AH} is empty.

(iii) $n^{AH} \geq 1$ but a backtracking solver could still backtrack in this reduced CSP.

(iv) $n^{AH} > 1$

(v) $n^{AH} < n$

(vi) $n^{AH} = n$

(vii) $n^{AH} > n$

Multiple solutions could be possible no matter what the consistency or harmony so (iv) is true. Consider a single variable with two values that meet all constraints for all neighboring variables' domains. (v) holds because harmony could over-aggressively reduce domains: a value need only satisfy a constraint with one other value in the related domain to be part of a solution, and need not satisfy the constraint for all values in the related domain. (vi) could be true since the original CSP might already be arc-harmonic.

Define S_i, the *solution domain* of a variable X_i, to be the set of values that X_i takes in some solution. That is, $x \in S_i$ if and only if for some solution to the CSP, $X_i = x$.

(f) (2 pt) Circle all of the following statements that are *guaranteed* to be true, if any.

(i) $S_i \subseteq D_i$

(ii) $D_i \subseteq S_i$

(iii) $S_i \subseteq D_i^{AC}$

(iv) $D_i^{AC} \subseteq S_i$

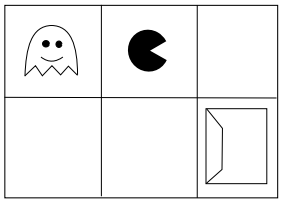
(v) $S_i \subseteq D_i^{AH}$

(vi) $D_i^{AH} \subseteq S_i$

Refer to reasoning in previous parts.

3. (12 points) Pac-Infiltration

A special operations Pacman must reach the exit (located on the bottom right square) in a ghost-filled maze, as shown to the right. Every time step, each agent (i.e., Pacman and each ghost) can move to any adjacent non-wall square or stay in place. Pacman moves first and the ghosts move collectively and simultaneously in response (so, consider the ghosts to be one agent). Pacman's goal is to exit as soon as possible. If Pacman and a ghost ever occupy the same square, either through Pacman or ghost motion, the game ends, and Pacman receives utility -1. If Pacman moves into the exit square, he receives utility $+1$ and the game ends. The ghosts are adversarial and seek to minimize Pacman's utility.



For parts (a) and (b), consider the 2x3 board with one ghost shown above. In this example, it is Pacman's move.

(a) (2 pt) What is the minimax value of this state?
+1 because Pacman is guaranteed a victory with his first-move advantage

(b) (2 pt) If we perform a depth-limited minimax search from this state, searching to depth $d = 1, 2, 3, 4$ and using zero as the evaluation function for non-terminal states, what minimax values will be computed? Reminder: each layer of depth includes one move from each agent.

d	Minimax value
1	0
2	1
3	1
4	1

For parts (c) and (d), consider a grid of size $N \times M$ containing K ghosts. Assume we perform a depth-limited minimax search, searching to depth d.

(c) (2 pt) How many states will be expanded (asymptotically) in terms of d using standard minimax? Your answer should use all the relevant variables needed and should not be specific to any particular map or configuration. $(5^{K+1})^d$. The branching factor for Pacman is 5 since he can move to any adjacent square (4 choices) or stay put (+1 choice). The branching factor for ghosts is 5^K since each of the K ghosts has the same 5 movement options. The number of nodes expanded is the branching factor raised to the depth d.

(d) (1 pt) For small boards, why is standard recursive minimax a poor choice for this problem? Your answer should be no more than one sentence!
For small boards the depth of the goal (exit with +1) will be small, and since the goal ends the game the search need not be continued and the further expansion done by standard minimax is a waste.

Let the current state be $s \in S$, specifying the positions of Pacman and the ghosts. Let the actions available to Pacman be $a \in A$, and the (collective) action available to the ghosts be $b \in B$ (i.e., each single action b moves each ghost once). Let $S_A(a, s)$ denote the result of applying action a to state s. Similarly, let $S_B(b, s)$ be the result of applying action b. Let $M(s)$ be the minimax value of a state s where Pacman is next to move.

(e) (2 pt) Write a Bellman-style equation that defines the minimax value $M(s)$ of a non-terminal state s in terms of other state values $M(s')$. Your equation should be general, not specific to the single state above. Do not worry about special casing for terminal states.

$$M(s) = \max_a \min_b M(S_B(b, S_A(a, s)))$$

since the minimax values are Pacman's best maximizing choice over the pursuing ghosts' minimizing choice given Pacman's action.

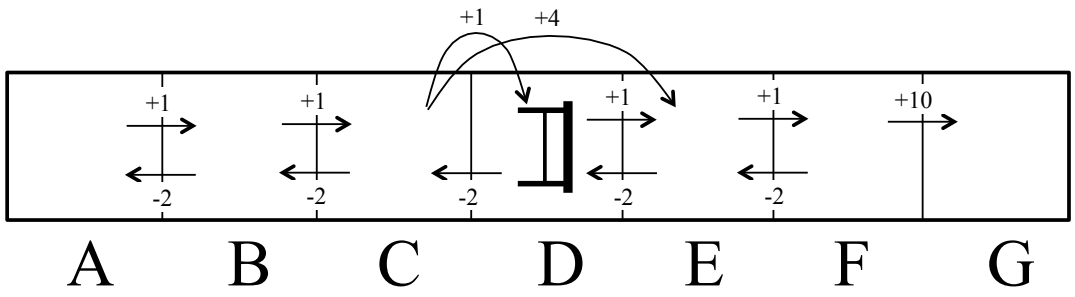
(f) (1 pt) If Pacman used your Bellman equation for an analogue of value iteration, what would the asymptotic time complexity of each round of value iteration be, in terms of the sizes of the sets defined above? (E.g., $|B||A|^2$).
$|A||B|$ since the max and min are done over every pacman-ghost action pair in $A \times B$.

The tables turn! Pacman is hiding while he waits for extraction. He cannot move, but the ghosts do not know where he is unless they collide with him. An optimal ghost mastermind is planning a search pattern for K ghosts that minimizes the worst-case time to find Pacman (i.e. guarantees that Pacman will be found in t steps for the smallest t possible).

(g) (2 pt) Specify a new state representation that the ghosts can use to plan their ghostly sweep. You should describe the elements of your state representation. As always, be compact in both state space size and answer size!
The ghost sweep is represented by the K ghost positions in $(MN)^K$ space and a boolean variable for each position to indicate whether or not it has been checked for Pacman in space 2^{MN}. The total state space size is the product $(MN)^K 2^{MN}$.

Consider an MDP modeling a hurdle race track, shown below. There is a single hurdle in square D. The terminal state is G. The agent can run *left* or *right*. If the agent is in square C, it cannot run *right*. Instead, it can *jump*, which either results in a fall to the hurdle square D, or a successful hurdle jump to square E. Rewards are shown below. Assume a discount of $\gamma = 1$.

Rewards:



Actions:

- *right*: Deterministically move to the right.
- *left*: Deterministically move to the left.
- *jump*: Stochastically jump to the right. This action is available for square C only.
  T(C, *jump*, E) = 0.5 (jump succeeds)
  T(C, *jump*, D) = 0.5 (jump fails)

Note: The agent cannot use *right* from square C.

(a) **(2 pt)** For the policy π of always moving forward (i.e., using actions *right* or *jump*), compute V^{π}(C).
$V^{\pi}i(C) = \frac{13+15}{2} = 14$. Without discounting ($\gamma = 1$), the value is the sum of rewards. The reward to go from C is $1+1+1+10 = 13$ if the jump fails or $4+1+10 = 15$ if the jump succeeds. The jump succeeds or fails with equality probability so the expected value is the average.

(b) **(3 pt)** Perform two iterations of value iteration and compute the following. Iteration 0 corresponds to the initialization of all values to 0.

V_2(B)	3.5
Q_2(B, *right*)	3.5
Q_2(B, *left*)	-1

	A	B	C	D	E	F	G
V_1	1	1	2.5	1	1	10	0
V_2	2	3.5	3	2	11	10	0

Remember that the V_{k+1} and Q_{k+1} updates use the values V_k and Q_k for the successor states. $V_2(B)$ is the maximum over the Qs for B. *Note:* Two iterations of value iteration computes the value for an $h = 2$ two step horizon episode. Thus $V_2(s)$ for any state s is equal to the maximum expected return for two time steps.

(c) (3 pt) Fill in the blank cells of the table below with the Q-values that result from applying the Q-learning update for the 4 transitions specified by the episode below. You may leave Q-values that are unaffected by the current update blank. Use a learning rate α of 0.5. Assume all Q-values are initialized to 0.

Episode

s	a	r	s	a	r	s	a	r	s	a	r	s
C	*jump*	+4	E	*right*	+1	F	*left*	-2	E	*right*	+1	F

	$Q(\text{C}, \textit{left})$	$Q(\text{C}, \textit{jump})$	$Q(\text{E}, \textit{left})$	$Q(\text{E}, \textit{right})$	$Q(\text{F}, \textit{left})$	$Q(\text{F}, \textit{right})$
Initial	0	0	0	0	0	0
Transition 1		2				
Transition 2				0.5		
Transition 3					-0.75	
Transition 4				0.75		

The Q-learning update is $Q'(s,a) = Q(s,a) + \alpha\left[r + \max_{a'} \gamma Q(s',a') - Q(s,a)\right]$.

$$2 = 0 + 0.5(4 + 0 - 0)$$
$$0.5 = 0 + 0.5(1 + 0 - 0)$$
$$-0.75 = 0 + 0.5(-2 + .0.5 - 0)$$
$$0.75 = 0.5 + 0.5(1 + 0 - 0.5)$$

A state-action is only updated when a transition is made from it. $Q(\text{C}, left)$, $Q(\text{E}, left)$, and $Q(\text{F}, right)$ state-actions are never experienced and so these values are never updated.

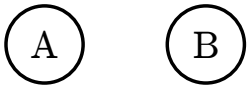
On transition 2, the Qs for F are still both 0, so the update increases the value by the reward +1 times the learning rate. On transition 3, the reward of -2 and $Q(\text{E}, right) = 0.5$ are included in the update. On transition 4, $Q(\text{F}, left)$ is now -0.75 but $Q(\text{F}, right)$ is still 0 so the next update to $Q(\text{E}, right)$ uses 0 in the max over the next state's action.

4. (8 points) A Game of d-Separation

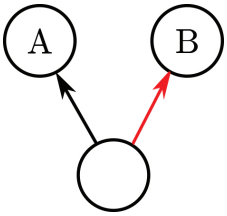
Consider the following two player game, which starts with a set of unconnected nodes that represent a Bayes net. Two nodes are labeled "A" and "B", and some are shaded. Play proceeds according to the following rules, with players taking alternating turns.

- On each turn, a player chooses any two nodes and draws a directed edge between them.
- No player may cause the graph to become cyclic or draw an edge that has already been drawn.
- Players attempt to keep A and B conditionally independent (i.e., d-separated) in the Bayes net given the shaded nodes.
- The first player to break the conditional independence between A and B loses.

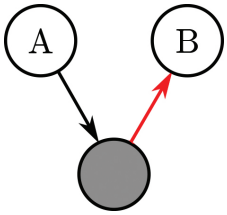
As an example, the game below starts with only two nodes, neither shaded. The first player has two options: draw an edge from A to B, or draw an edge from B to A. Either one breaks the independence of A and B, and so causes the first player to lose immediately.



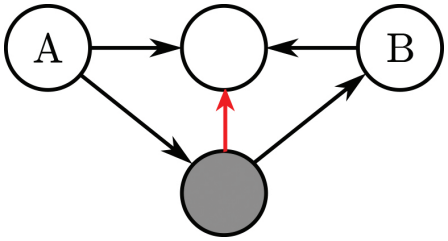
(a) (3 pt) For each of the following three positions, determine if the next player has a move which does not lose immediately. If there is at least one such a move, draw one. If every move loses immediately, draw an "X" over the entire position.



No move. Any edge to (blank) connects A and B.



The other direction activates the common effect and loses.

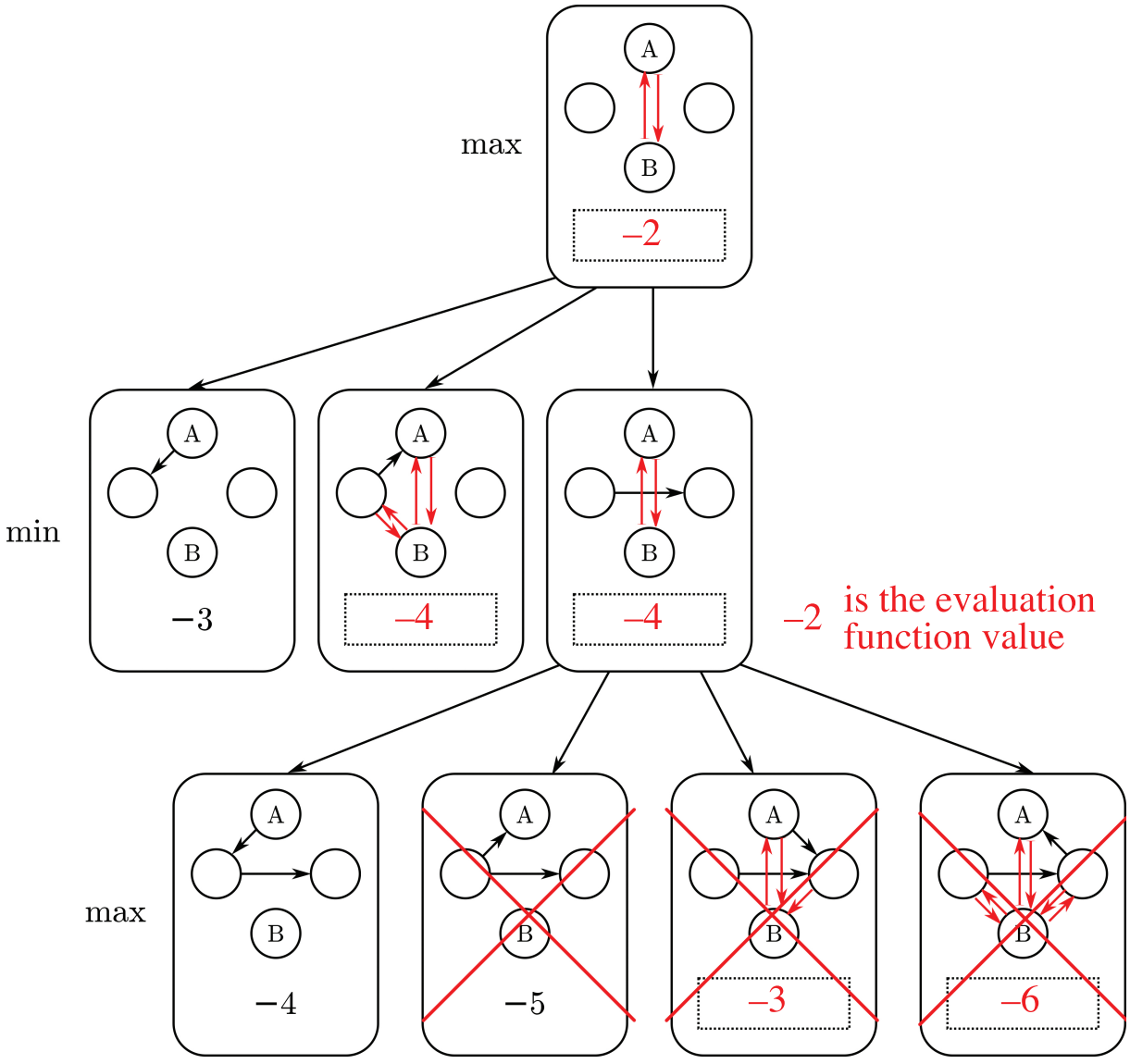


The other direction activates the common effect so that A,(observed),(unobserved),B is an active path.

As an evaluation function, take the negative of the number of valid moves (i.e., no duplicates or cycles) that break d-separation. A partial game tree is shown below.

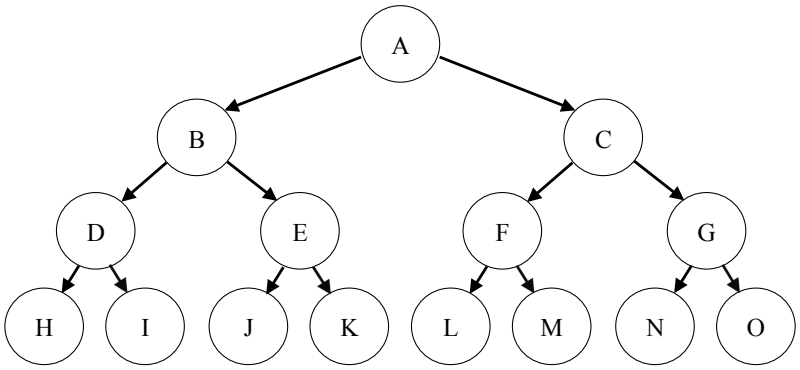
(b) **(3 pt)** Using minimax with this evaluation function, write the value of every node in the dotted boxes provided. Three values have been filled in for you.

(c) **(2 pt)** Draw an "X" over any nodes that alpha-beta search would prune when evaluating the root node, assuming it evaluates nodes from left to right.



5. (14 points) Occupy Cal

You are at Occupy Cal, and the leaders of the protest are deciding whether or not to march on California Hall. The decision is made centrally and communicated to the occupiers via the "human microphone"; that is, those who hear the information repeat it so that it propagates outward from the center. This scenario is modeled by the following Bayes net:



A	*P(A)*
+m	0.5
-m	0.5

π(X)	*X*	*P(X|π(X))*
+m	*+m*	0.9
+m	*-m*	0.1
-m	*+m*	0.1
-m	*-m*	0.9

Each random variable represents whether a given group of protestors hears instructions to march ($+m$) or not ($-m$). The decision is made at A, and both outcomes are equally likely. The protestors at each node relay what they hear to their two child nodes, but due to the noise, there is some chance that the information will be misheard. Each node except A takes the same value as its parent with probability 0.9, and the opposite value with probability 0.1, as in the conditional probability tables shown.

(a) (2 pt) Compute the probability that node A sent the order to march ($A = +m$) given that both B and C receive the order to march ($B = +m$, $C = +m$).

$p(A = +m \mid B = +m, C = +m) = \frac{p(A=+m,B=+m,C=+m)}{\sum_a p(A=a,B=+m,C=+m)} = \frac{p(A=+m)p(B=+m|A=+m)p(C=+m|A=+m)}{\sum_a p(A=a)p(B=+m|A=a)p(C=+m|A=a)} = \frac{p(A=+m)p(X=+m|\pi(X)=+m)^2}{\sum_a p(A=a)p(X=+m|\pi(X)=a)^2} = \frac{.5 \cdot .9^2}{.5 \cdot .9^2 + .5 \cdot .1^2)} = \frac{.9^2}{.9^2 + .1^2} \approx 0.988.$

Note that $P(A)$ is uniform. It can be pulled out of sums and cancelled in the numerator and denominator. For simplicity $P(A)$ terms are dropped.

For further simplification, note that the conditional distribution $P(X|\pi(X))$ is symmetric, so calculations can be simplified in terms of $p(X, \pi(X) same) = .9$ and $(p(X, \pi(X) different) = .1$.

(b) (2 pt) Compute the probability that D receives the order $+m$ given that A sent the order $+m$.

$$\begin{aligned} p(D=+m|A=+m) &= \frac{\sum_b p(D=+m, B=b, A=+m)}{\sum_d \sum_b p(D=+m, B=b, A=+m)} \\ &= \frac{\sum_b p(D=+m|B=b)p(B=b|A=+m)}{\sum_d \sum_b p(D=d|B=b)p(B=b|A=+m)} \\ &= \frac{.9^2 + .1^2}{.9^2 + .1^2 + 2(.9)(.1)} = 0.82 \end{aligned}$$

You are at node D, and you know what orders have been heard at node D. Given your orders, you may either decide to march (*march*) or stay put (*stay*). (Note that these actions are distinct from the orders $+m$ or $-m$ that you hear and pass on. The variables in the Bayes net and their conditional distributions still behave exactly as above.) If you decide to take the action corresponding to the decision that was actually made at A (not necessarily corresponding to your orders!), you receive a reward of $+1$, but if you take the opposite action, you receive a reward of -1.

(c) (2 pt) Given that you have received the order $+m$, what is the expected utility of your optimal action? (Hint: your answer to part (b) may come in handy.)

The maximum expected utility is 0.64. The expected utility of an action is $p(A = action|D = +m)(1) + p(A \neq action|D = +m)(-1)$, or the expected reward of matching plus the expected reward of not matching.

The conditional probability $p(D = +m|A = +m)$, the answer to part (b), is the probability that the order at D matches the order given at A. This follows from the symmetry of the distribution of A and the conditionals of the children (this can be verified by computing $p(A = +m|D = +m)$).

For marching, the expected utility is $p(D = +m|A = +m)(1) + p(D = -m|A = +m)(-1) = .82(1) + (1 - .82)(-1) = 0.64$. For standing, the expected utility is $p(D = +m|A = +m)(-1) + p(D = -m|A = +m)(1) = .82(-1) + (1 - .82)(1) = -0.64$.

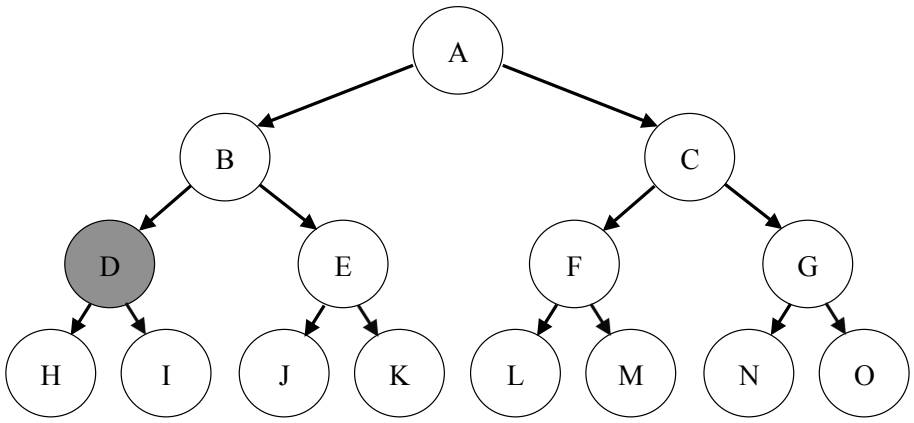
Now suppose that you can have your friends text you what orders they have received. (Hint: for the following two parts, you should not need to do much computation due to symmetry properties and intuition.)

(d) (2 pt) Compute the VPI of A given that $D = +m$. 0.36. The maximum expected utility knowing only $D = +m$ is 0.64 as seen above, whereas with perfect knowledge of A one can always act to achieve reward 1. The VPI is the difference in maximum expected utility $1 - 0.64 = 0.36$.

(e) (2 pt) Compute the VPI of F given that $D = +m$. Observing node F would change the posterior distribution of A so $P(A \mid D, F) \neq P(A \mid D)$, but the extra observation would not be sufficient to warrant a different action. Consequently, VPI$(F \mid D)$ is zero.

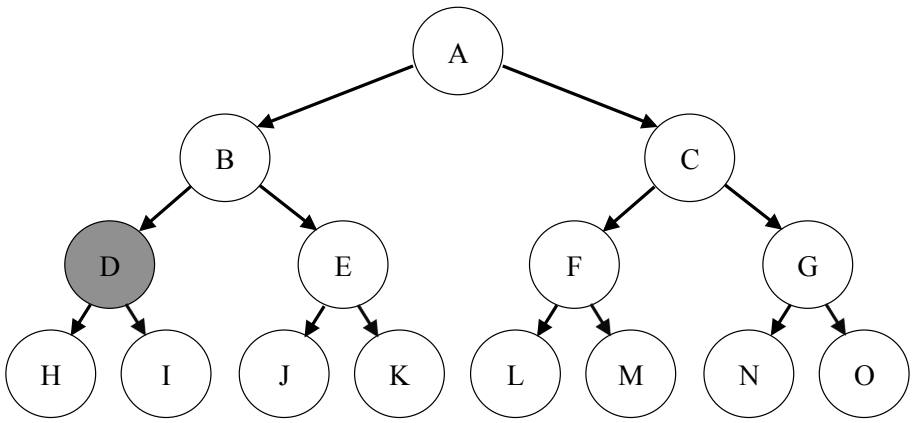
For the following parts, you should circle nodes in the accompanying diagrams that have the given properties. Use the quantities you have already computed and intuition to answer the following question parts; you should not need to do any computation.

(f) **(2 pt)** Circle the nodes for which knowing the value of that node changes your belief about the decision made at A given evidence at D (i.e. nodes X such that $P(A|X,D) \neq P(A|D)$).



Any node that is not conditionally independent of A given D will change the posterior belief about A. The only nodes that would not change the posterior are H and I.

(g) **(2 pt)** Circle the nodes which have nonzero VPI given evidence at D.



Circle A, B, C. If any of these nodes are observed, the optimal action is to follow the instruction at the newly observed node and ignore the information at D. The reason for doing so is that A, B or C are more accurate estimators of A than D. Since the extra information could change the action taken, the VPI is nonzero.

Observing any node at the same depth as D (E, F, G) or lower (H, I, J, K, L, M, N, O) would not warrant choosing a different action since these nodes are not more informative than D. Note that if we were given the choice of observing *two* additional nodes, for example E and F, than $\text{VPI}(E, F \mid D) > 0$ since the additional information from *two* nodes might change the action, for example if we get $D = +m, E = -m, F = -m$.

6. (10 points) HMMs and Particle Filtering

Consider a Markov Model with a binary state X (i.e., X_t is either 0 or 1). The transition probabilities are given as follows:

X_t	X_{t+1}	$P(X_{t+1} \mid X_t)$
0	0	0.9
0	1	0.1
1	0	0.5
1	1	0.5

(a) (2 pt) The prior belief distribution over the initial state X_0 is uniform, i.e., $P(X_0 = 0) = P(X_0 = 1) = 0.5$. After one timestep, what is the new belief distribution, $P(X_1)$?

X_1	$P(X_1)$
0	0.7
1	0.3

Since the prior of X_0 is uniform, the belief at the next step is the transition distribution from X_0:
$p(X_1 = 0) = p(X_0 = 0)p(X_1 = 0|X_0 = 0) + p(X_0 = 1)p(X_1 = 0|X_0 = 1) = .5(.9) + .5(.5) = .7.$
$p(X_1 = 1) = p(X_0 = 0)p(X_1 = 1|X_0 = 0) + p(X_0 = 1)p(X_1 = 1|X_0 = 1) = .5(.1) + .5(.5) = .3.$

Now, we incorporate sensor readings. The sensor model is parameterized by a number $\beta \in [0, 1]$:

X_t	E_t	$P(E_t \mid X_t)$
0	0	β
0	1	$(1 - \beta)$
1	0	$(1 - \beta)$
1	1	β

(b) (2 pt) At $t = 1$, we get the first sensor reading, $E_1 = 0$. Use your answer from part (a) to compute $P(X_1 = 0 \mid E_1 = 0)$. Leave your answer in terms of β.

$$p(X_1 = 0|E_1 = 0) = \frac{p(E_1 = 0|X_1 = 0)p(X_1 = 0)}{\sum_x p(E_1 = 0|X_1 = x)p(X_1 = x)}$$
$$= \frac{\beta(0.7)}{\beta(0.7) + (1 - \beta)(0.3)}$$

(c) (2 pt) For what range of values of β will a sensor reading $E_1 = 0$ increase our belief that $X_1 = 0$? That is, what is the range of β for which $P(X_1 = 0 \mid E_1 = 0) > P(X_1 = 0)$?
$\beta \in (0.5, 1]$. Intuitively, observing $E_1 = 0$ will only increase the belief that $X_1 = 0$ if $E_1 = 0$ is more likely under $X_1 = 0$ than not. Note that $\beta > 0.5$; $\beta = 0.5$ is uninformative since the conditional distribution is uniform. This can be verified algebraically by setting $p(X_1 = 0) = p(X_1 = 0|E_1 = 0)$ and solving for β.

(d) (2 pt) Unfortunately, the sensor breaks after just one reading, and we receive no further sensor information. Compute $P(X_\infty \mid E_1 = 0)$, the stationary distribution *very many* timesteps from now.

X_∞	$P(X_\infty \mid E_1 = 0)$
0	$\frac{5}{6}$
1	$\frac{1}{6}$

The stationary distribution π for transition matrix P satisfies $\pi = \pi P$. It is purely a function of the transition matrix and not the prior or past observations.

Determine π by setting up the matrix equation $\pi = \pi P$ with the additional equation $\pi_0 + \pi_1 = 1$ from the sum-to-one constraint of the probabilities and solving.

(e) (2 pt) How would your answer to part (d) change if we never received the sensor reading E_1, i.e. what is $P(X_\infty)$ given no sensor information?

X_∞	$P(X_\infty)$
0	$\frac{5}{6}$
1	$\frac{1}{6}$

The stationary distribution does not depend on past observations, so the distribution is unchanged whether E_1 is observed or not.

7. (12 points) ML

You are a Hollywood producer. You have a script in your hand and you want to make a movie. Before starting, however, you want to predict if the film you want to make will rake in huge profits, or utterly fail at the box office. You hire two critics A and B to read the script and rate it on a scale of 1 to 5 (assume only integer scores). Each critic reads it independently and announces their verdict. Of course, the critics might be biased and/or not perfect. For instance, for the past five movies you made, these are the critics' scores and the performance of the movie:

Movie Name	A	B	Profit?
Pellet Power	1	1	No
Ghosts!	3	2	Yes
Pac is Bac	4	5	No
Not a Pizza	3	4	Yes
Endless Maze	2	3	Yes

Table 1: Training Data

You decide to use machine learning to learn a classifier to predict if the script will lead to profits or not.

You first decide to use a perceptron to classify your data. This problem will use the multi-class formulation even though there are only two classes. Suppose you directly use the scores given above as features, together with a bias feature. That is $f_0 = 1$, $f_1 =$ score given by A and $f_2 =$ score given by B.

(a) (2 pt) You want to train the perceptron on the training data in Table 1. The initial weights are given below:

Profit	Weights
Yes	[-1, 0, 0]
No	[1, 0, 0]

(i) Which is the first training instance at which you update your weights?
Ghosts! The initial weights always predict *No* since the bias weight is +1 vs. −1 for *Yes* and the weights for the ratings are 0. The weights are updated on wrong classifications, so the prediction *Yes* for Ghosts! results in an update.

(ii) In the table below, write the updated weights after the first update.

Profit	Weights
Yes	[0, 3, 2]
No	[0, -3, -2]

Add the feature vector to the weights of the correct class and subtract it from the wrong class.

(b) (2 pt) More generally, irrespective of the training data, you want to know if your features are powerful enough to allow you to handle a range of scenarios. Some scenarios are given below. Circle those scenarios for which a perceptron using the features above can indeed perfectly classify the data.

(i) Your reviewers are awesome: if the total of their scores is more than 8, then the movie will definitely be a success and otherwise it will fail.

(ii) Your reviewers are art critics. Your movie will succeed if and only if each reviewer gives either a score of 2 or a score of 3.

(iii) Your reviewers have weird but different tastes. Your movie will succeed if and only if both reviewers agree.

For (i), set the *No* weights all to zero, set the bias weight for *Yes* to −8, and the weight for each reviewer to 1. The other scenarios are not linearly separable with these three features.

You decide to use a different set of features. Consider the following feature space:

$f_0 = 1$ (The bias feature)
$f_{1A} = 1$ if score given by A is 1, 0 otherwise
$f_{1B} = 1$ if score given by B is 1, 0 otherwise
$f_{2A} = 1$ if score given by A is 2, 0 otherwise
$f_{2B} = 1$ if score given by B is 2, 0 otherwise
...
$f_{5B} = 1$ if score given by B is 5, 0 otherwise

(c) (2 pt) Consider again the three scenarios in part (b). Using a perceptron with the new features, which of the three scenarios can be perfectly classified? Circle your answer(s) below:

(i) Your reviewers are awesome: if the total of their scores is more than 8, then the movie will definitely be a success, and otherwise it will fail.

(ii) Your reviewers are art critics. Your movie will succeed if and only if each reviewer gives either a score of 2 or a score of 3.

(iii) Your reviewers have weird but different tastes. Your movie will succeed if and only if both reviewers agree.

The features are indicators for the ratings of the reviewers, so any rule that is a linear combination of the ratings can be represented. (i) and (ii) can be represented by setting the *No* weights all to zero then setting the *Yes* weights for ratings that satisfy the rule to 1 and the bias to -1.5 to require both reviewers' ratings to be in the desired set.

You have just heard of naive Bayes and you want to use a naive Bayes classifier. You use the scores given by the reviewers as the features of the naive Bayes classifier, i.e., the random variables in your naive Bayes model are A and B, each with a domain of $\{1, 2, \ldots, 5\}$, and *Profit* with a domain of *Yes* and *No*.

(d) (2 pt) Draw the Bayes' net corresponding to the naive Bayes model.
Profit is the parent of both A and B. There are no other variables or edges.

(e) (2 pt) List the types of the conditional probability tables you need to estimate along with their sizes (e.g., $P(X \mid Y)$ has 24 entries).

Probability	Size
$P(Profit)$	2
$P(A\|Profit)$	10
$P(B\|Profit)$	10

The naive Bayes classifier requires a prior on the class and conditionals of the features given the class, as shown in the structure of the Bayes' net.

(f) (2 pt) Your nephew is taking the CS188 class at Berkeley. He claims that the naive Bayes classifier you just built is actually a linear classifier in the feature space used for part (c). In other words, the decision boundary of the naive Bayes classifier is a hyperplane in this feature space. For the positive class, what is the weight of the feature f_{3B} in terms of the parameters of the naive Bayes model? You can answer in symbols, but be precise. (Hint: Consider the log of the probability.)
$f_{3B} = \log p(B = 3|Yes)$. The probability under the naive Bayes model is the product of the class prior and the feature conditionals, where each probability is raised to the corresponding feature value. Taking the log of the naive Bayes probability transforms the expression to a sum of log probabilities times the features corresponding to the class prior and the conditionals of the observed features.