CS188 Fall 2018 Review: Final Preparation

1 . Bounded suboptimal search: weighted A*

In this class you met A*, an algorithm for informed search guaranteed to return an optimal solution when given an admissible heuristic. Often in practical applications it is too expensive to find an optimal solution, so instead we search for good suboptimal solutions.

Weighted A* is a variant of A* commonly used for suboptimal search. Weighted A* is exactly the same as A* but where the f-value is computed differently:

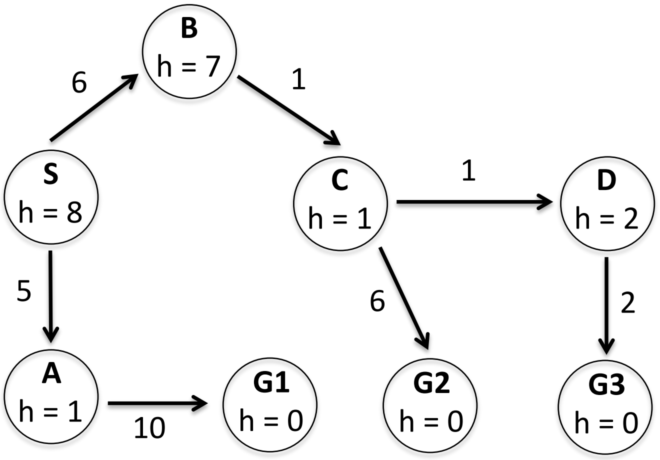
$$f(n) = g(n) + \varepsilon\, h(n)$$

where $\varepsilon \geq 1$ is a parameter given to the algorithm. In general, the larger the value of ε, the faster the search is, and the higher cost of the goal found.

Pseudocode for weighted A* tree search is given below. **NOTE:** The only differences from the A* tree search pseudocode presented in the lectures are: (1) *fringe* is assumed to be initialized with the start node before this function is called (this will be important later), and (2) now INSERT takes ε as a parameter so it can compute the correct f-value of the node.

```
function WEIGHTED-A*-TREE-SEARCH(problem, fringe, $\varepsilon$)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem, STATE[node]) then return node
        for child-node in child-nodes do
            fringe ← INSERT(child-node, fringe, $\varepsilon$)
```

(a) We'll first examine how weighted A* works on the following graph:



Execute weighted A* on the above graph with $\varepsilon = 2$, completing the following table. To save time, you can optionally just write the nodes added to the fringe, with their g and f values.

node	Goal?	*fringe*
-	-	$\{S : g = 0, f = 16\}$
S	No	$\{S \to A : g = 5, f = 7; S \to B : g = 6, f = 20\}$
$S \to A$	No	$\{S \to A \to G1 : g = 15, f = 15; S \to B : g = 6, f = 20\}$
$S \to A \to G1$	Yes	-

(b) After running weighted A* with weight $\varepsilon \geq 1$ a goal node G is found, of cost $g(G)$. Let C^* be the optimal solution cost, and suppose the heuristic is admissible. Select the strongest bound below that holds, and provide a proof.

● $g(G) \leq \varepsilon\, C^*$　○ $g(G) \leq C^* + \varepsilon$　○ $g(G) \leq C^* + 2\varepsilon$　○ $g(G) \leq 2^{\varepsilon}\, C^*$　○ $g(G) \leq \varepsilon^2\, C^*$

Proof: (Partial credit for reasonable proof sketches.)

When weighted A* terminates, an ancestor n of the optimal goal G^* is on the fringe. Since G was expanded before n, we have $f(G) \leq f(n)$. As a result:

$$g(G) = f(G) \leq f(n) = g(n) + \varepsilon h(n) \leq \varepsilon\,(g(n) + h(n)) \leq \varepsilon\, C^*$$

If you're confused about whether this all comes from, rembmer that $f(n) = g(n) + \epsilon h(n)$ comes from the problem statement and the inequality $g(n) + \epsilon h(n) \leq \epsilon(g(n) + h(n))$ is true by algebra.
Since we know that $g(n)$ is non-negative, it must be true that $g(n) + \epsilon h(n) <= \epsilon(g(n) + h(n))$. This is a common technique used when trying to prove/find a bound.

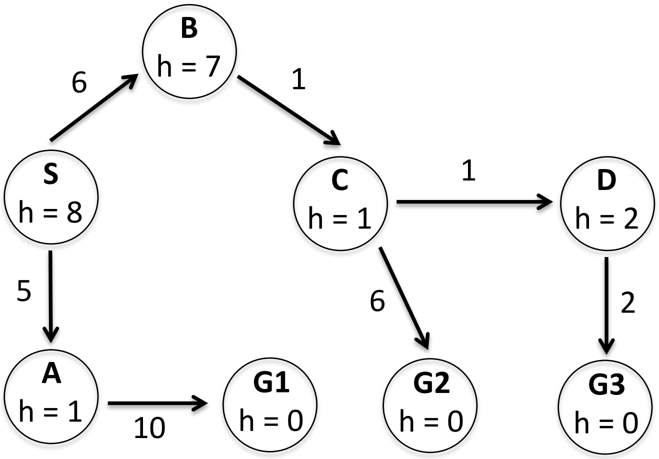
(c) Weighted A* includes a number of other algorithms as special cases. For each of the following, name the corresponding algorithm.

(i) $\varepsilon = 1$.
Algorithm: A*

(ii) $\varepsilon = 0$.
Algorithm: UCS

(iii) $\varepsilon \to \infty$ (i.e., as ε becomes arbitrarily large).
Algorithm: Greedy search

(d) Here is the same graph again:



(i) Execute weighted A* on the above graph with $\varepsilon = 1$, completing the following table as in part (a):

node	Goal?	*fringe*
-	-	$\{S : g = 0, f = 8\}$
S	No	$\{S \to A : g = 5, f = 6; S \to B : g = 6, f = 13\}$
$S \to A$	No	$\{S \to B : g = 6, f = 13; S \to A \to G1 : g = 15, f = 15\}$
$S \to B$	No	$\{S \to B \to C : g = 7, f = 8; S \to A \to G1 : g = 15, f = 15\}$
$S \to B \to C$	No	$\{S \to B \to C \to D : g = 8, f = 10; S \to B \to C \to G2 : g = 13, f = 13;$ $S \to A \to G1 : g = 15, f = 15\}$
$S \to B \to C \to D$	No	$\{S \to B \to C \to D \to G3 : g = 10, f = 10; S \to B \to C \to G2 : g = 13, f = 13;$ $S \to A \to G1 : g = 15, f = 15\}$
$S \to B \to C \to D \to G3$	Yes	-

(ii) You'll notice that weighted A* with $\varepsilon = 1$ repeats computations performed when run with $\varepsilon = 2$. Is there a way to reuse the computations from the $\varepsilon = 2$ search by starting the $\varepsilon = 1$ search with a different fringe? Let F denote the set that consists of both (i) all nodes the fringe the $\varepsilon = 2$ search ended with, and (ii) the goal node G it selected. Give a brief justification for your answer.

○ Use F as new starting fringe
○ Use F with goal G removed as new starting fringe
● Use F as new starting fringe, updating the f-values to account for the new ε
○ Use F with goal G removed as new starting fringe, updating the f-values to account for the new ε
○ Initialize the new starting fringe to all nodes visited in previous search
○ Initialize the new starting fringe to all nodes visited in previous search, updating the f-values to account for the new ε
○ It is not possible to reuse computations, initialize the new starting fringe as usual

Justification:

We have to include G in the fringe as it might still be optimal (e.g. if it is the only goal). We don't have to update the g-values, but we do have the update the f-values to reflect the new value of ε. With these modifications, it is valid to continue searching as the state of the fringe is as if A* with the new ε was run, but with some extraneous node expansions.

2 . Crossword Puzzles as CSPs

You are developing a program to automatically solve crossword puzzles, because you think a good income source for you might be to submit them to the New York Times ($200 for a weekday puzzle, $1000 for a Sunday).[1] For those unfamiliar with crossword puzzles, a crossword puzzle is a game in which one is given a grid of squares that must be filled in with intersecting words going from left to right and top to bottom. There are a given set of starting positions for words (in the grid below, the positions $1, 2, 3, 4$, and 5), where words must be placed going across (left to right) or down (top to bottom). At any position where words intersect, the letters in the intersecting words must match. Further, no two words in the puzzle can be identical. An example is the grid below, in which the down words (1, 2, and 3) are `DEN`, `ARE`, and `MAT`, while the across words (1, 4, and 5) are `DAM`, `ERA`, and `NET`.

Example Crossword Grid and Solution

^{1}D	^{2}A	^{3}M
^{4}E	R	A
^{5}N	E	T

A part of your plan to make crosswords, you decide you will create a program that uses the CSP solving techniques you have learned in CS 188, since you want to make yourself obsolete at your own job from the get-go. Your first task is to choose the representation of your problem. You start with a dictionary of all the words you could put in the crossword puzzle, where the dictionary is of size K and consists of the words $\{d_1, d_2, \ldots, d_K\}$. Assume that you are given a grid with N empty squares and M different entries for words (and there are 26 letters in the English language). In the example above, $N = 9$ and $M = 6$ (three words across and three words down).

You initially decide to use words as the variables in your CSP. Let D_1 denote the first down word, D_2 the second, D_3 the third, etc., and similarly let A_k denote the kth across word. For example, in the crossword above, $A_1 = \texttt{DAM}$, $D_1 = \texttt{DEN}$, $D_2 = \texttt{ARE}$, and so on. Let $D_1[i]$ denote the letter in the ith position of the word D_1.

(a) What is the size of the state space for this CSP?

Several answers are acceptable for this problem. The simplest is that the dictionary has size K and there are M words, giving state space size K^M. A slightly tighter bound is achieved by noting that once one word is placed, the next words must all be different, giving $K(K-1)(K-2)\cdots(K-M+1) = \frac{K!}{(K-M)!}$. Noticing that we are choosing M distinct words out of a possible K gives the state space bound $\binom{K}{M}$.

Several students tried to include N in their answers; since the letters have nothing to do with this formulation of the problem, this was incorrect. Many students also incorrectly had M^K.

(b) Precisely (i.e. use mathematical notation to) describe the constraints of the CSP when we use words as variables.

For every pair of across and down words D_k and A_l that intersect, we have the constraint that their letters are equal. Specifically, if they intersect in positions i and j, we have $D_k[i] = A_l[j]$.

We also have the pairwise constraints that none of the words are the same: for $k \neq k'$, $D_k \neq D_{k'}$ and $A_k \neq A_{k'}$, and for all k, k', we have $A_k \neq D_{k'}$.

In addition, each word must have the correct length. One possible formulation is that for all $L \in \mathbb{N}$, for all words D_k and A_l with length L in the puzzle, we have $\text{length}(D_k) = L$ and $\text{length}(A_l) = L$.

The biggest problem that students had was assuming that all crossword puzzles were contiguous squares (or rectangles) like the example. While that works for the above example, it will not work generally. Several students missed one or two of the above constraints, and all three were necessary for full credit. Minor mistakes included missing a few of the inequality constraints.

After defining your CSP, you decide to go ahead and make a small crossword using the grid below. Assume that you use the words on the right as your dictionary.

[1] `http://www.nytimes.com/2009/07/19/business/media/19askthetimes.html`

Crossword Grid

1	2	3	4	
5				■
6				■
7			■	■

Dictionary Words

`ARCS, BLAM, BEAR, BLOGS, LARD, LARP,`
`GAME, GAMUT, GRAMS, GPS, MDS, ORCS, WARBLER`

(c) Enforce all *unary* constraints by crossing out values in the table below.

D_1	ARCS	BLAM	BEAR	~~BLOGS~~	LARD	LARP	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
D_2	ARCS	BLAM	BEAR	~~BLOGS~~	LARD	LARP	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
D_3	ARCS	BLAM	BEAR	~~BLOGS~~	LARD	LARP	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
D_4	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	GPS	MDS	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
A_1	~~ARCS~~	~~BLAM~~	~~BEAR~~	BLOGS	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	GAMUT	GRAMS	~~ORCS~~	~~WARBLER~~
A_5	ARCS	BLAM	BEAR	~~BLOGS~~	LARD	LARP	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
A_6	ARCS	BLAM	BEAR	~~BLOGS~~	LARD	LARP	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
A_7	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	GPS	MDS	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~

(d) Assume that in backtracking search, we assign A_1 to be `GRAMS`. Enforce unary constraints, and in addition, cross out all the values eliminated by forward checking against A_1 as a result of this assignment.

D_1	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
D_2	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
D_3	ARCS	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
D_4	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	MDS	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
A_1	~~ARCS~~	~~BLAM~~	~~BEAR~~	BLOGS	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	GAMUT	$\boxed{\texttt{GRAMS}}$	~~ORCS~~	~~WARBLER~~
A_5	ARCS	BLAM	BEAR	~~BLOGS~~	LARD	LARP	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
A_6	ARCS	BLAM	BEAR	~~BLOGS~~	LARD	LARP	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
A_7	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	GPS	MDS	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~

(e) Now let's consider how much arc consistency can prune the domains for this problem, even when no assignments have been made yet. I.e., assume no variables have been assigned yet, enforce unary constraints first, and then enforce arc consistency by crossing out values in the table below.

D_1	~~ARCS~~	BLAM	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
D_2	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	LARD	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
D_3	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
D_4	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	GPS	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
A_1	~~ARCS~~	~~BLAM~~	~~BEAR~~	BLOGS	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
A_5	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	LARP	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
A_6	ARCS	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
A_7	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	MDS	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~

The common mistake in this question was to leave a few blocks of words that students thought could not be eliminated. Probably the most common was to allow both `LARD` and `LARP` for D_2 and A_5. This is incorrect; for D_2, no assignment of A_7 is consistent with `LARP`, and for A_5, no assignment of D_4 is consistent with `LARD`.

(f) How many solutions to the crossword puzzle are there? Fill them (or the single solution if there is only one) in below.

^{1}B	^{2}L	^{3}O	^{4}G	S
^{5}L	A	R	P	■
^{6}A	R	C	S	■
^{7}M	D	S	■	■

1	2	3	4	
5				■
6				■
7			■	■

1	2	3	4	
5				■
6				■
7			■	■

There is one solution (above)

Your friend suggests using letters as variables instead of words, thinking that sabotaging you will be funny. Starting from the top-left corner and going left-to-right then top-to-bottom, let X_1 be the first letter, X_2 be the second, X_3

the third, etc. In the very first example, $X_1 = \texttt{D}$, $X_2 = \texttt{A}$, and so on.

(g) What is the size of the state space for this formulation of the CSP?

26^N. There are 26 letters and N possible positions.

(h) Assume that in your implementation of backtracking search, you use the least constraining value heuristic. Assume that X_1 is the first variable you choose to instantiate. For the crossword puzzle used in parts (c)-(f), what letter(s) might your search assign to X_1?

We realized that this question was too vague to be answered correctly, so we gave everyone 2 points for the problem. The least constraining value heuristic, once a variable has been chosen, assigns the value that according to some metric (chosen by the implementer of the heuristic) leaves the domains of the remaining variables most open. How one eliminates values from the domains of other variables upon an assignment can impact the choice of the value as well (whether one uses arc consistency or forward checking).

We now sketch a solution to the problem assuming we use forward checking. Let $X_1, X_2, \ldots, X_5$ be the letters in the top row of the crossword and X_1, X_6, X_7, X_8 be the first column down. Upon assigning $X_1 = \texttt{G}$, the possible domains for the remaining letters are

$$X_2 \in \{\texttt{A}, \texttt{R}\}, X_3 \in \{\texttt{M}, \texttt{A}\}, X_4 \in \{\texttt{U}, \texttt{M}\}, X_5 \in \{\texttt{T}, \texttt{S}\}, X_6 \in \{\texttt{A}\}, X_7 \in \{\texttt{M}\}, X_8 \in \{\texttt{E}\}.$$

Upon assigning $X_1 = \texttt{B}$, the possible domains remaining are

$$X_2 \in \{\texttt{L}\}, X_3 \in \{\texttt{O}\}, X_4 \in \{\texttt{G}\}, X_5 \in \{\texttt{S}\}, X_6 \in \{\texttt{L}, \texttt{E}\}, X_7 \in \{\texttt{A}\}, X_8 \in \{\texttt{M}, \texttt{R}\}.$$

The remaining variables are unaffected since we are using only forward checking. Now, we see that with the assignment $X_1 = \texttt{G}$, the minimum size remaining for any domain is 1, while the sum of the sizes remaining domains is 11; for $X_1 = \texttt{B}$, the minimum size is 1, while the sum of the sizes remaining is 9. So depending on whether we use minimum domain or the sum of the sizes of the remaining domains, the correct solutions are G and B or only G, respectively.

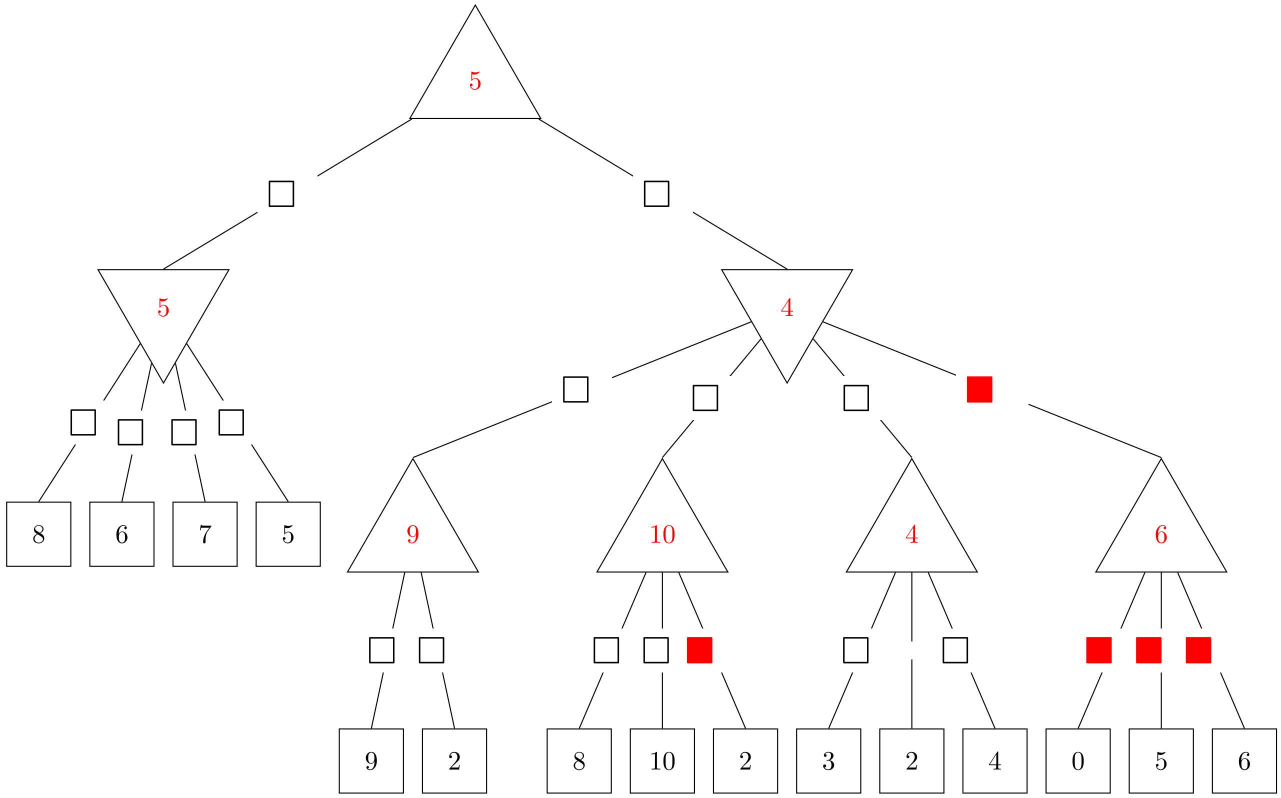
Any choice but $X_1 = \texttt{B}$ or $X_1 = \texttt{G}$ will eliminate all values for one of the other variables after forward checking.

3 . Game Trees

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

(a) Minimax

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



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

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

○ No nodes can be pruned

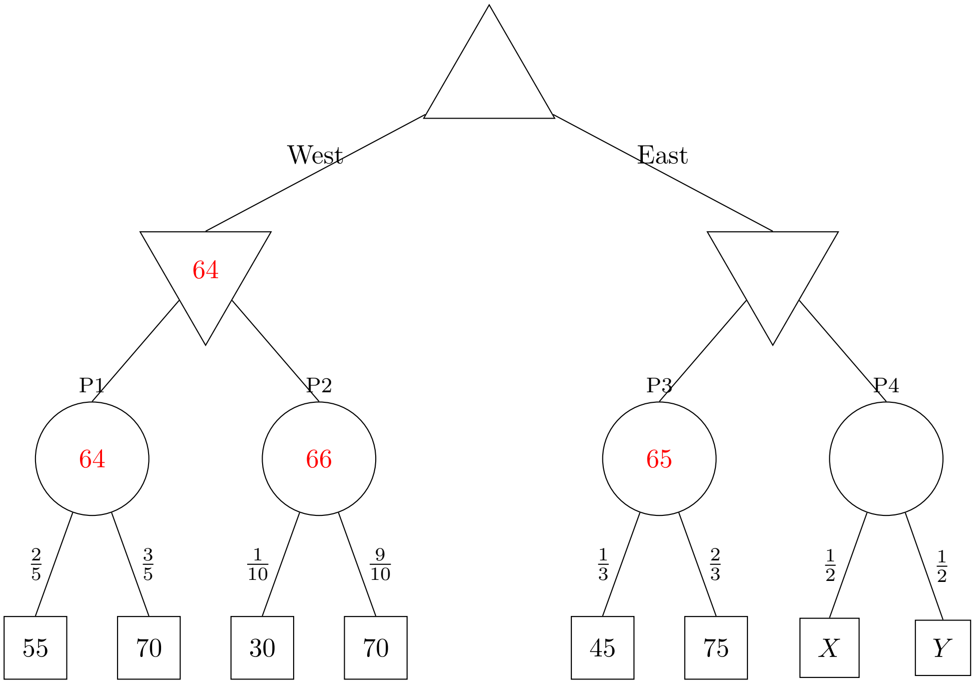
(b) Food Dimensions

The following questions are completely unrelated to the above parts.

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

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

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



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

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

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

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

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

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

4 . Discount MDPs

5		S			10
			0	0	

Consider the above gridworld. An agent is currently on grid cell S, and would like to collect the rewards that lie on both sides of it. If the agent is on a numbered square, its only available action is to Exit, and when it exits it gets reward equal to the number on the square. On any other (non-numbered) square, its available actions are to move East and West. Note that North and South are never available actions.

If the agent is in a square with an adjacent square downward, it does not always move successfully: when the agent is in one of these squares and takes a move action, it will only succeed with probability p. With probability $1 - p$, the move action will fail and the agent will instead move downwards. If the agent is not in a square with an adjacent space below, it will always move successfully.

For parts (a) and (b), we are using discount factor $\gamma \in [0, 1]$.

(a) Consider the policy π_{East}, which is to always move East (right) when possible, and to Exit when that is the only available action. For each non-numbered state x in the diagram below, fill in $V^{\pi_{\text{East}}}(x)$ in terms of γ and p.

5	$10\gamma^4p^2$	$10\gamma^3p^2$	$10\gamma^2p^2$	$10\gamma p$	10
			0	0	

(b) Consider the policy π_{West}, which is to always move West (left) when possible, and to Exit when that is the only available action. For each non-numbered state x in the diagram below, fill in $V^{\pi_{\text{West}}}(x)$ in terms of γ and p.

5	5γ	$5\gamma^2$	$5\gamma^3p$	$5\gamma^4p^2$	10
			0	0	

(c) For what range of values of p *in terms of* γ is it optimal for the agent to go West (left) from the start state (S)?

We want $5\gamma^2 \geq 10\gamma^3 p^2$, which we can solve to get:

Range: $p \in [0, \frac{1}{\sqrt{2\gamma}}]$

(d) For what range of values of p *in terms of* γ is π_{West} the optimal policy?

We need, for each of the four cells, to have the value of that cell under π_{West} to be at least as large as π_{East}. Intuitively, the farther east we are, the higher the value of moving east, and the lower the value of moving west (since the discount factor penalizes far-away rewards).
Thus, if moving west is the optimal policy, we want to focus our attention on the rightmost cell.
At the rightmost cell, in order for moving west to be optimal, then $V^{\pi_{\text{East}}}(s) \leq V^{\pi_{\text{West}}}(s)$, which is $10\gamma p \leq 5\gamma^4 p^2$, or $p \geq \frac{2}{\gamma^3}$.
However, since γ ranges from 0 to 1, the right side of this expression ranges from 2 to ∞, which means p (a probability, and thus bounded by 1) has no valid value.
Range: $\emptyset$

(e) For what range of values of p *in terms of* γ is π_{East} the optimal policy?

We follow the same logic as in the previous part. Specifically, we focus on the leftmost cell, where the condition for π_{East} to be the optimal policy is: $10\gamma^4 p^2 \geq 5\gamma$, which simplifies to $p \geq \frac{1}{\sqrt{2\gamma^3}}$. Combined with our bound on any probability being in the range $[0, 1]$, we get:

Range: $p \in \left[\frac{1}{\sqrt{2\gamma^3}}, 1\right]$, which could be an empty set depending on γ.

Recall that in approximate Q-learning, the Q-value is a weighted sum of features: $Q(s, a) = \sum_i w_i f_i(s, a)$. To derive a weight update equation, we first defined the loss function $L_2 = \frac{1}{2}(y - \sum_k w_k f_k(x))^2$ and found $dL_2/dw_m = -(y - \sum_k w_k f_k(x)) f_m(x)$. Our label y in this set up is $r + \gamma \max_a Q(s', a')$. Putting this all together, we derived the gradient descent update rule for w_m as $w_m \leftarrow w_m + \alpha\,(r + \gamma \max_a Q(s', a') - Q(s, a))\, f_m(s, a)$.

In the following question, you will derive the gradient descent update rule for w_m using a different loss function:

$$L_1 = \left| y - \sum_k w_k f_k(x) \right|$$

(f) Find dL_1/dw_m. Show work to have a chance at receiving partial credit. Ignore the non-differentiable point.

Note that the derivative of $|x|$ is -1 if $x < 0$ and 1 if $x > 0$. So for L_1, we have:

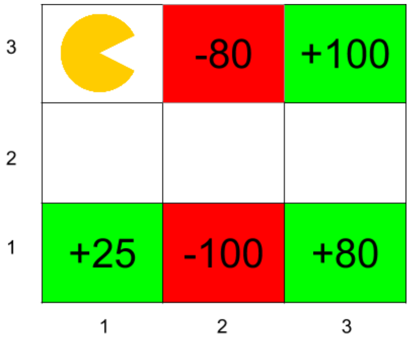
$$\frac{dL_1}{dw_m} = \begin{cases} -f_m(x) & y - \sum_k w_k f_k(x) > 0 \\ f_m(x) & y - \sum_k w_k f_k(x) < 0 \end{cases}$$

(g) Write the gradient descent update rule for w_m, using the L_1 loss function.

$$\begin{aligned} w_m &\leftarrow w_m - \alpha dL_1/dw_m \\ &\leftarrow \begin{cases} w_m + \alpha f_m(x) & y - \sum_k w_k f_k(x) > 0 \\ w_m - \alpha f_m(x) & y - \sum_k w_k f_k(x) < 0 \end{cases} \end{aligned}$$

5 . Q-Learning Strikes Back

Consider the grid-world given below and Pacman who is trying to learn the optimal policy. If an action results in landing into one of the shaded states the corresponding reward is awarded during that transition. All shaded states are terminal states, i.e., the MDP terminates once arrived in a shaded state. The other states have the *North, East, South, West* actions available, which deterministically move Pacman to the corresponding neighboring state (or have Pacman stay in place if the action tries to move out of the grad). Assume the discount factor $\gamma = 0.5$ and the Q-learning rate $\alpha = 0.5$ for all calculations. Pacman starts in state (1, 3).



(a) What is the value of the optimal value function V^* at the following states:

$V^*(3,2) =$ 100 $V^*(2,2) =$ 50 $V^*(1,3) =$ 12.5

The optimal values for the states can be found by computing the expected reward for the agent acting optimally from that state onwards. Note that you get a reward when you transition *into* the shaded states and not *out* of them. So for example the optimal path starting from (2,2) is to go to the +100 square which has a discounted reward of $0 + \gamma * 100 = 50$. For (1,3), going to either of +25 or +100 has the same discounted reward of 12.5.

(b) The agent starts from the top left corner and you are given the following episodes from runs of the agent through this grid-world. Each line in an Episode is a tuple containing (s, a, s', r).

Episode 1	Episode 2	Episode 3
(1,3), S, (1,2), 0	(1,3), S, (1,2), 0	(1,3), S, (1,2), 0
(1,2), E, (2,2), 0	(1,2), E, (2,2), 0	(1,2), E, (2,2), 0
(2,2), S, (2,1), -100	(2,2), E, (3,2), 0	(2,2), E, (3,2), 0
	(3,2), N, (3,3), +100	(3,2), S, (3,1), +80

Using Q-Learning updates, what are the following Q-values after the above three episodes:

Q((3,2),N) = 50 Q((1,2),S) = 0 $Q((2,2),E) =$ 12.5

Q-values obtained by Q-learning updates - $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(R(s,a,s') + \gamma \max_{a'} Q(s',a'))$.

(c) Consider a feature based representation of the Q-value function:

$$Q_f(s,a) = w_1 f_1(s) + w_2 f_2(s) + w_3 f_3(a)$$

$f_1(s)$: The x coordinate of the state $f_2(s)$: The y coordinate of the state

$$f_3(N) = 1,\ f_3(S) = 2,\ f_3(E) = 3,\ f_3(W) = 4$$

(i) Given that all w_i are initially 0, what are their values after the first episode:

$w_1 =$ ____-100____ $w_2 =$____-100____ $w_3 =$____-100____

Using the approximate Q-learning weight updates: $w_i \leftarrow w_i + \alpha[(R(s,a,s') + \gamma \max_{a'} Q(s',a')) - Q(s,a)]f_i(s,a)$. The only time the reward is non zero in the first episode is when it transitions into the -100 state.

(ii) Assume the weight vector w is equal to $(1, 1, 1)$. What is the action prescribed by the Q-function in state $(2, 2)$?

____West____

The action prescribed at (2,2) is $\max_a Q((2,2),a)$ where $Q(s,a)$ is computed using the feature representation. In this case, the Q-value for *West* is maximum $(2 + 2 + 4 = 8)$.

6 . Probability

(a) Consider the random variables A, B, and C. Circle all of the following equalities that are **always** true, if any.

1. $\mathbf{P}(A,B) = \mathbf{P}(A)\mathbf{P}(B) - \mathbf{P}(A|B)$

2. $\mathbf{P}(A,B) = \mathbf{P}(A)\mathbf{P}(B)$

3. $\mathbf{P}(A,B) = \mathbf{P}(A|B)\mathbf{P}(B) + \mathbf{P}(B|A)\mathbf{P}(A)$

4. $\boxed{\mathbf{P}(A) = \sum_{b \in B} \mathbf{P}(A|B=b)\mathbf{P}(B=b)}$

5. $\mathbf{P}(A,C) = \sum_{b \in B} \mathbf{P}(A|B=b)\mathbf{P}(C|B=b)\mathbf{P}(B=b)$

6. $\boxed{\mathbf{P}(A,B,C) = \mathbf{P}(C|A)\mathbf{P}(B|C,A)\mathbf{P}(A)}$

Now assume that A and B both can take on only the values true and false ($A \in \{\text{true}, \text{false}\}$ and $B \in \{\text{true}, \text{false}\}$). You are given the following quantities:

$$\begin{aligned}
\mathbf{P}(A = \text{true}) &= \tfrac{1}{2} \\
\mathbf{P}(B = \text{true} \mid A = \text{true}) &= 1 \\
\mathbf{P}(B = \text{true}) &= \tfrac{3}{4}
\end{aligned}$$

(b) What is $\mathbf{P}(B = \text{true} \mid A = \text{false})$?

Many people got lost trying to directly apply Bayes' rule. The simplest way to solve this is to realize that

$$\mathbf{P}(B = \text{true}) = \ \mathbf{P}(B = \text{true} \mid A = \text{true})\mathbf{P}(A = \text{true}) + \mathbf{P}(B = \text{true} \mid A = \text{false})\mathbf{P}(A = \text{false}).$$

Using this fact, you can solve for $\mathbf{P}(B = \text{true} \mid A = \text{false})$:

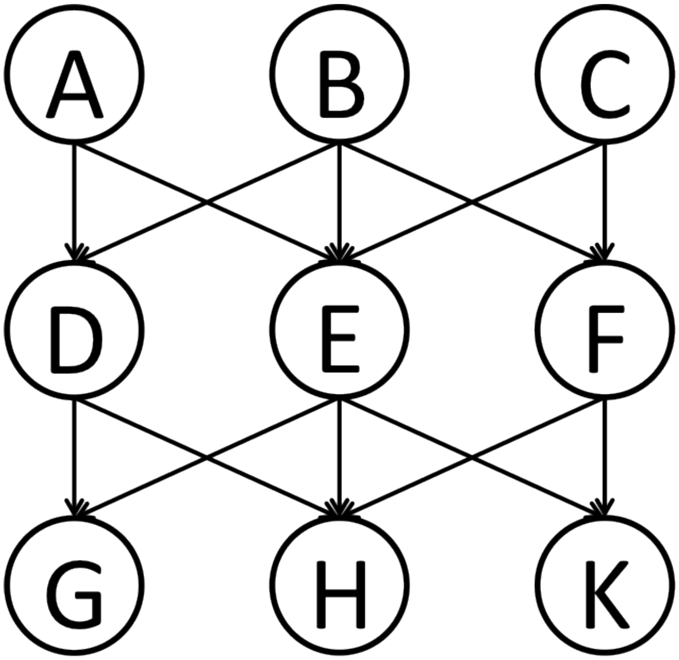
$$\begin{aligned}
(1)\left(\frac{1}{2}\right) + \mathbf{P}(B = \text{true} \mid A = \text{false})\left(\frac{1}{2}\right) &= \frac{3}{4} \\
\Longrightarrow \mathbf{P}(B = \text{true} \mid A = \text{false})\left(\frac{1}{2}\right) &= \frac{1}{4} \\
\Longrightarrow \mathbf{P}(B = \text{true} \mid A = \text{false}) &= \frac{1}{2}
\end{aligned}$$

Therefore $\mathbf{P}(B = \text{true} \mid A = \text{false}) = \frac{1}{2}$.

7 . Bayes’ Nets: Short Questions

(a) Bayes’ Nets: Conditional Independence

Based only on the structure of the (new) Bayes’ Net given below, circle whether the following conditional independence assertions are guaranteed to be true, guaranteed to be false, or cannot be determined by the structure alone. *Note: The ordering of the three answer columns might have been switched relative to previous exams!*



1	$A \perp\!\!\!\perp C$	Guaranteed false	Cannot be determined	[✓ Guaranteed true]
2	$A \perp\!\!\!\perp C \mid E$	Guaranteed false	[✓ Cannot be determined]	Guaranteed true
3	$A \perp\!\!\!\perp C \mid G$	Guaranteed false	[✓ Cannot be determined]	Guaranteed true
4	$A \perp\!\!\!\perp K$	Guaranteed false	[✓ Cannot be determined]	Guaranteed true
5	$A \perp\!\!\!\perp G \mid D, E, F$	Guaranteed false	Cannot be determined	[✓ Guaranteed true]
6	$A \perp\!\!\!\perp B \mid D, E, F$	Guaranteed false	[✓ Cannot be determined]	Guaranteed true
7	$A \perp\!\!\!\perp C \mid D, F, K$	Guaranteed false	[✓ Cannot be determined]	Guaranteed true
8	$A \perp\!\!\!\perp G \mid D$	Guaranteed false	[✓ Cannot be determined]	Guaranteed true

(b) Bayes' Nets: Elimination of a Single Variable

Assume we are running variable elimination, and we currently have the following three factors:

A	B	$f_1(A,B)$
$+a$	$+b$	0.1
$+a$	$-b$	0.5
$-a$	$+b$	0.2
$-a$	$-b$	0.5

A	C	D	$f_2(A,C,D)$
$+a$	$+c$	$+d$	0.2
$+a$	$+c$	$-d$	0.1
$+a$	$-c$	$+d$	0.5
$+a$	$-c$	$-d$	0.1
$-a$	$+c$	$+d$	0.5
$-a$	$+c$	$-d$	0.2
$-a$	$-c$	$+d$	0.5
$-a$	$-c$	$-d$	0.2

B	D	$f_3(B,D)$
$+b$	$+d$	0.2
$+b$	$-d$	0.2
$-b$	$+d$	0.5
$-b$	$-d$	0.1

The next step in the variable elimination is to eliminate B.

(i) Which factors will participate in the elimination process of B? f_1, f_3

(ii) Perform the join over the factors that participate in the elimination of B. Your answer should be a table similar to the tables above, it is your job to figure out which variables participate and what the numerical entries are.

A	B	D	$f_4'(A,B,D)$
$+a$	$+b$	$+d$	0.1*0.2 = 0.02
$+a$	$+b$	$-d$	0.1*0.2 = 0.02
$+a$	$-b$	$+d$	0.5*0.5 = 0.25
$+a$	$-b$	$-d$	0.5*0.1 = 0.05
$-a$	$+b$	$+d$	0.2*0.2 = 0.04
$-a$	$+b$	$-d$	0.2*0.2 = 0.04
$-a$	$-b$	$+d$	0.5*0.5 = 0.25
$-a$	$-b$	$-d$	0.5*0.1 = 0.05

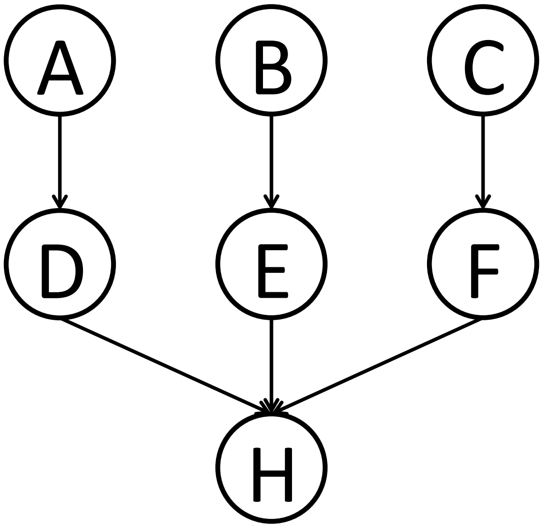
(iii) Perform the summation over B for the factor you obtained from the join. Your answer should be a table similar to the tables above, it is your job to figure out which variables participate and what the numerical entries are.

A	D	$f_4(A,D)$
$+a$	$+d$	0.02+ 0.25 = 0.27
$+a$	$-d$	0.02 + 0.05 = 0.07
$-a$	$+d$	0.04 + 0.25 = 0.29
$-a$	$-d$	0.04 + 0.05 = 0.09

(c) Elimination Sequence

For the Bayes' net shown below, consider the query $P(A|H=+h)$, and the variable elimination ordering B, E, C, F, D.

(i) In the table below fill in the factor generated at each step — we did the first row for you.



Variable Eliminated	Factor Generated	Current Factors
(no variable eliminated yet)	(no factor generated)	$P(A), P(B), P(C), P(D|A), P(E|B), P(F|C), P(+h|D,E,F)$
B	$f_1(E)$	$P(A), P(C), P(D|A), P(F|C), P(+h|D,E,F), f_1(E)$
E	$f_2(+h,D,F)$	$P(A), P(C), P(D|A), P(F|C), f_2(+h,D,F)$
C	$f_3(F)$	$P(A), P(D|A), f_2(+h,D,F), f_3(F)$
F	$f_4(+h,D)$	$P(A), P(D|A), f_4(+h,D)$
D	$f_5(+h,A)$	$P(A), f_5(+h,A)$

(ii) Which is the largest factor generated? Assuming all variables have binary-valued domains, how many entries does the corresponding table have? $f_2(+h,D,F)$, its table has $2^2 = 4$ entries

(d) Sampling

(i) Consider the query $P(A|-b,-c)$. After rejection sampling we end up with the following four samples: $(+a,-b,-c,+d), (+a,-b,-c,-d), (+a,-b,-c,-d), (-a,-b,-c,-d)$. What is the resulting estimate of $P(+a|-b,-c)$?

$\frac{3}{4}$.

(ii) Consider again the query $P(A|-b,-c)$. After likelihood weighting sampling we end up with the following four samples: $(+a,-b,-c,-d), (+a,-b,-c,-d), (-a,-b,-c,-d), (-a,-b,-c,+d)$, and respective weights: $0.1, 0.1, 0.3, 0.3$. What is the resulting estimate of $P(+a|-b,-c)$?

$\frac{0.1+0.1}{0.1+0.1+0.3+0.3} = \frac{0.2}{0.8} = \frac{1}{4}$

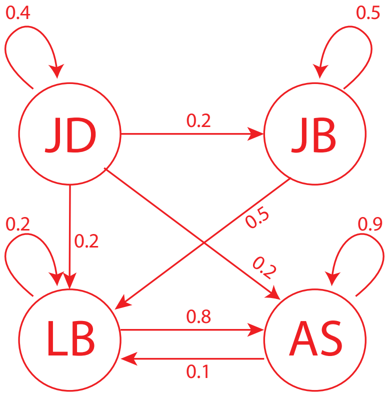
8 . HMM: Where is the key?

The cs188 staff have a key to the homework bin. It is the master key that unlocks the bins to many classes, so we take special care to protect it.

Every day John Duchi goes to the gym, and on the days he has the key, 60% of the time he forgets it next to the bench press. When that happens one of the other three GSIs, equally likely, always finds it since they work out right after. Jon Barron likes to hang out at Brewed Awakening and 50% of the time he is there with the key, he forgets the key at the coffee shop. Luckily Lubomir always shows up there and finds the key whenever Jon Barron forgets it. Lubomir has a hole in his pocket and ends up losing the key 80% of the time somewhere on Euclid street. However, Arjun takes the same path to Soda and always finds the key. Arjun has a 10% chance to lose the key somewhere in the AI lab next to the Willow Garage robot, but then Lubomir picks it up.

The GSIs lose the key at most once per day, around noon (after losing it they become extra careful for the rest of the day), and they always find it the same day in the early afternoon.

(a) Draw on the left the Markov chain capturing the location of the key and fill in the transition probability table on the right. In this table, the entry of row JD and column JD corresponds to $P(X_{t+1} = \text{JD}|X_t = \text{JD})$, the entry of row JD and column JB corresponds to $P(X_{t+1} = \text{JB}|X_t = \text{JD})$, and so forth.



	JD_{t+1}	JB_{t+1}	LB_{t+1}	AS_{t+1}
JD_t	0.4	0.2	0.2	0.2
JB_t	0	0.5	0.5	0
LB_t	0	0	0.2	0.8
AS_t	0	0	0.1	0.9

Monday early morning Prof. Abbeel handed the key to Jon Barron. (The initial state distribution assigns probability 1 to $X_0 = \text{JB}$ and probability 0 to all other states.)

(b) The homework is due Tuesday at midnight so the GSIs need the key to open the bin. What is the probability for each GSI to have the key at that time? Let X_0, X_{Mon} and X_{Tue} be random variables corresponding to who has the key when Prof. Abbeel hands it out, who has the key on Monday evening, and who has the key on Tuesday evening, respectively. Fill in the probabilities in the table below.

	$P(X_0)$	$P(X_{\text{Mon}})$	$P(X_{\text{Tue}})$
JD	0	0.0	$0*.4+.5*.0+.5*.0+0*.0 = .00$
JB	1	0.5	$0*.2+.5*.5+.5*.0+0*.0 = .25$
LB	0	0.5	$0*.2+.5*.5+.5*.2+0*.1 = .35$
AS	0	0.0	$0*.2+.5*.0+.5*.8+0*.9 = .40$

(c) The GSIs like their jobs so much that they decide to be professional GSIs permanently. They assign an extra credit homework (make computers truly understand natural language) due *at the end of time*. What is the probability that each GSI holds the key at a point infinitely far in the future. Hint:

$$P_\infty(x) = \sum_{x'} P(X_{\text{next day}} = x \mid X_{\text{current day}} = x')P_\infty(x')$$

The goal is to compute the stationary distribution. From the Markov chain it is obvious that $P_\infty(JD) = 0$ and $P_\infty(JB) = 0$. Let $x = P_\infty(LB)$ and $y = P_\infty(AS)$. Then the definition of stationarity implies

$$\begin{aligned} x &= 0.2x + 0.1y \\ y &= .8x + .9y \end{aligned}$$

Since we must have $x \geq 0$ and $y \geq 0$, we can choose any $x > 0$ and solve for y. For example, $x = 1$ yields $y = 8$, which normalized results in $P_\infty(LB) = 1/9$ and $P_\infty(AS) = 8/9$.

Every evening the GSI who has the key feels obliged to write a short anonymous report on their opinion about the state of AI. Arjun and John Duchi are optimistic that we are right around the corner of solving AI and have an 80% chance of writing an optimistic report, while Lubomir and Jon Barron have an 80% chance of writing a pessimistic report. The following are the titles of the first few reports:

Monday: Survey: Computers Become Progressively Less Intelligent (pessimistic)
Tuesday: How to Solve Computer Vision in Three Days (optimistic)

(d) In light of that new information, what is the probability distribution for the key on Tuesday midnight given that Jon Barron has it Monday morning? You may leave the result as a ratio or unnormalized.

We are trying to perform inference in an HMM, so we must simply perform the forward algorithm. The HMM described in our problem is

The calculations are as follows:

	$P(X_{\text{Mon}})$	$P(X_{\text{Mon}} \mid R_{Mon} = \text{pessim.})$	$P(X_{\text{Tue}} \mid R_{Mon} = \text{pessim.})$	$P(X_{\text{Tue}} \mid R_{Mon} = \text{pessim.}, R_{Tue} = \text{optim.})$
JD	0.0	$\propto 0.0 * 0.2 \propto 0.0 = 0.0$	0.00	$\propto 0.00 * 0.8 = 0.00/0.44$
JB	0.5	$\propto 0.5 * 0.8 \propto 0.4 = 0.5$	0.25	$\propto 0.25 * 0.2 = 0.05/0.44$
LB	0.5	$\propto 0.5 * 0.8 \propto 0.4 = 0.5$	0.35	$\propto 0.35 * 0.2 = 0.07/0.44$
AS	0.0	$\propto 0.0 * 0.2 \propto 0.0 = 0.0$	0.40	$\propto 0.40 * 0.8 = 0.32/0.44$

On Thursday afternoon Prof. Abbeel noticed a suspiciously familiar key on top of the Willow Garage robot's head. He thought to himself, "This can't possibly be the master key." (He was wrong!) Lubomir managed to snatch the key and distract him before he inquired more about it and is the key holder Thursday at midnight (i.e., $X_{\text{Thu}} = \text{LB}$). In addition, the Friday report is this:

Thursday: ??? (report unknown)
Friday: AI is a scam. I know it, you know it, it is time for the world to know it! (pessimistic)

(e) Given that new information, what is the probability distribution for the holder of the key on Friday at midnight?

In (the extension of) the HMM above, $R_{\text{Thu}} \perp\!\!\!\perp X_{\text{Fri}} \mid X_{\text{Thu}}$, so we compute

	$P(X_{\text{Thu}})$	$P(X_{\text{Fri}})$	$P(X_{\text{Fri}} \mid R_{Fri} = \text{pessim.})$
JD	0	0	$\propto 0.2 * 0.0 = 0.0$
JB	0	0	$\propto 0.8 * 0.0 = 0.0$
LB	1	0.2	$\propto 0.8 * 0.2 = 0.5$
AS	0	0.8	$\propto 0.2 * 0.8 = 0.5$

(f) Prof. Abbeel recalls that he saw Lubomir holding the same key on Tuesday night. Given this new information (in addition to the information in the previous part), what is the probability distribution for the holder of the key on Friday at midnight?

The answer does not change because $X_{\text{Tue}} \perp\!\!\!\perp X_{\text{Fri}} \mid X_{\text{Thu}}$

(g) Suppose in addition that we know that the titles of the reports for the rest of the week are:

Saturday: Befriend your PC now. Soon your life will depend on its wishes (optimistic)
Sunday: How we got tricked into studying AI and how to change field without raising suspicion (pessimistic)

Will that new information change our answer to (f)? Choose one of these options:

1. [✓ Yes, reports for Saturday and Sunday affect our prediction for the key holder on Friday.]
2. No, our prediction for Friday depends only on what happened in the past.

9 . Ghostbusters

Suppose Pacman gets a noisy observation of a ghost's location for T moves, and then may guess where the ghost is at timestep T to eat it. To model the problem, you use an HMM, where the ith hidden state is the location of the ghost at timestep i and the ith evidence variable is the noisy observation of the ghost's location at time step i. *Assume Pacman always acts rationally.*

(a) If Pacman guesses correctly, he gets to eat the ghost resulting in a utility of 20. Otherwise he gets a utility of 0. If he does not make any guess, he gets a utility of 0.

Which of the following algorithms could Pacman use to determine the ghost's most likely location at time T? (Don't worry about runtime.)

□ Viterbi
■ Forward algorithm for HMMs
■ Particle filtering with a lot of particles
■ Variable elimination on the Bayes Net representing the HMM
□ None of the above, Pacman should use ________

We want to find the ghost location X_T that maximizes $P(X_T|e_{1:T})$. This can be done by calculating $P(X_T|e_{1:T})$ using the forward algorithm or variable elimination, and can be estimated using particle filtering. However, it cannot be calculated using Viterbi (since that maximizes $P(X_1, \cdots X_T|e_{1:T})$).

(b) In the previous part, there was no penalty for guessing. Now, Pacman has to *pay* 10 *utility* in order to try to eat the ghost. Once he pays, he still gets 20 utility for correctly guessing and eating the ghost, and 0 utility for an incorrect guess. Pacman determines that the most likely ghost location at time T is (x, y), and the probability of that location is p.

What is the expected utility of guessing that the ghost is at (x, y), as a function of p? $20p - 10$

With probability p, Pacman is right and gets utility 20, and with probability $1 - p$ he is wrong and gets utility 0. He always pays 10 utility. So the expected utility becomes $20p + 0(1 - p) - 10$.

When should Pacman guess that the ghost is at (x, y)?

○ Never (he should not guess)
○ If $p <$ ________.
● If $p >$ 0.5.
○ Always

Not guessing has a utility of 0, so Pacman should guess when the expected utility of guessing is > 0, which is when $p > 0.5$.

(c) Now, in addition to the -10 utility for trying to eat the ghost, Pacman can also pay 5 utility to learn the exact location of the ghost. (So, if Pacman pays the 5 utility and eats the ghost, he pays 15 utility and gains 20 utility for a total of 5 utility.)

When should Pacman pay the 5 utility to find the exact ghost location?

○ Never
● If $p <$ 0.75.
○ If $p >$ ________.
○ Always

Paying 5 utility means that Pacman is guaranteed to eat the ghost, getting $20 - 10 - 5 = 5$ utility in total. He should choose this option when it is better than the other two options (not guessing, or guessing without the info). This happens when $5 > 0$ and $5 > 20p - 10$, and thus it would be when $p < 0.75$.

(d) Now, Pacman can try to eat one out of Blinky (B), Inky (I) and Clyde (C) (three of the ghosts). He has some preferences about which one to eat, but he's afraid that his preferences are not rational. Help him out by showing him a utility function that matches his listed preferences, or mark "Not possible" if no rational utility function will work. You may choose any real number for each utility value. **If "Not possible" is marked, we will ignore any written utility function.**

(i) The preferences are $B \prec I$ and $I \prec C$ and $[0.5, B; 0.5, C] \prec I$

$U(B)$	$U(I)$	$U(C)$
1	4	5

○ Not possible

(ii) The preferences are $I \prec B$ and $[0.5, B; 0.5, C] \prec C$ and $[0.5, B; 0.5, C] \prec [0.5, B; 0.5, I]$

$U(B)$	$U(I)$	$U(C)$

● Not possible

The second preference implies $B \prec C$, the third implies $C \prec I$, and so we have $I \prec B \prec C \prec I$, which is irrational and no utility function would work.

10 . Perceptrons

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

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

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

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

(b) A single perceptron can compute the XOR function.

○ True ● False

(c) A perceptron is guaranteed to learn a separating decision boundary for a separable dataset within a finite number of training steps.

● True ○ False

(d) Given a linearly separable dataset, the perceptron algorithm is guaranteed to find a max-margin separating hyperplane.

○ True ● False

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

□ The square of the difference between the correct digit and the digit predicted by your network

□ The probability of the correct digit under your network

■ The negative log-probability of the correct digit under your network

○ None of the above

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

11 . Naive Bayes: Pacman or Ghost?

You are standing by an exit as either Pacmen or ghosts come out of it. Every time someone comes out, you get two observations: a visual one and an auditory one, denoted by the random variables X_v and X_a, respectively. The visual observation informs you that the individual is either a Pacman ($X_v = 1$) or a ghost ($X_v = 0$). The auditory observation X_a is defined analogously. Your observations are a noisy measurement of the individual's true type, which is denoted by Y. After the indiviual comes out, you find out what they really are: either a Pacman ($Y = 1$) or a ghost ($Y = 0$). You have logged your observations and the true types of the first 20 individuals:

individual i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
first observation $X_v^{(i)}$	0	0	1	0	1	0	0	1	1	1	0	1	1	0	1	1	1	0	0	0
second observation $X_a^{(i)}$	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
individual's type $Y^{(i)}$	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0

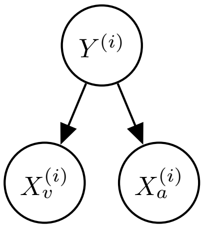
The superscript (i) denotes that the datum is the ith one. Now, the individual with $i = 20$ comes out, and you want to predict the individual's type $Y^{(20)}$ given that you observed $X_v^{(20)} = 1$ and $X_a^{(20)} = 1$.

(a) Assume that the types are independent, and that the observations are independent conditioned on the type. You can model this using naïve Bayes, with $X_v^{(i)}$ and $X_a^{(i)}$ as the features and $Y^{(i)}$ as the labels. Assume the probability distributions take on the following form:

$$P(X_v^{(i)} = x_v | Y^{(i)} = y) = \begin{cases} p_v & \text{if } x_v = y \\ 1 - p_v & \text{if } x_v \neq y \end{cases}$$

$$P(X_a^{(i)} = x_a | Y^{(i)} = y) = \begin{cases} p_a & \text{if } x_a = y \\ 1 - p_a & \text{if } x_a \neq y \end{cases}$$

$$P(Y^{(i)} = 1) = q$$



for $p_v, p_a, q \in [0, 1]$ and $i \in \mathbb{N}$.

(i) What's the maximum likelihood estimate of p_v, p_a and q?

$p_v = \underline{\frac{4}{5}}$ $\quad$ $p_a = \underline{\frac{3}{5}}$ $\quad$ $q = \underline{\frac{1}{2}}$

To estimate q, we count 10 $Y = 1$ and 10 $Y = 0$ in the data. For p_v, we have $p_v = 8/10$ cases where $X_v = 1$ given $Y = 1$ and $1 - p_v = 2/10$ cases where $X_v = 1$ given $Y = 0$. So $p_v = 4/5$. For p_a, we have $p_a = 2/10$ cases where $X_a = 1$ given $Y = 1$ and $1 - p_v = 0/10$ cases where $X_v = 1$ given $Y = 0$. The average of 2/10 and 1 is 3/5.

(ii) What is the probability that the next individual is Pacman given your observations? Express your answer in terms of the parameters p_v, p_a and q (you might not need all of them).

$$P(Y^{(20)} = 1 | X_v^{(20)} = 1, X_a^{(20)} = 1) = \underline{\frac{p_v p_a q}{p_v p_a q + (1 - p_v)(1 - p_a)(1 - q)}}$$

The joint distribution $P(Y = 1, X_v = 1, X_a = 1) = p_v p_a q$. For the denominator, we need to sum out over Y, that is, we need $P(Y = 1, X_v = 1, X_a = 1) + P(Y = 0, X_v = 1, X_a = 1)$.

Now, assume that you are given additional information: you are told that the individuals are actually coming out of a bus that just arrived, and each bus carries *exactly* 9 individuals. Unlike before, the types of every 9 consecutive individuals are *conditionally* independent given the bus type, which is denoted by Z. Only after all of the 9 individuals have walked out, you find out the bus type: one that carries mostly Pacmans ($Z = 1$) or one that carries mostly ghosts ($Z = 0$). Thus, you only know the bus type in which the first 18 individuals came in:

individual i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
first observation $X_v^{(i)}$	0	0	1	0	1	0	0	1	1	1	0	1	1	0	1	1	1	0	0	0
second observation $X_a^{(i)}$	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
individual's type $Y^{(i)}$	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0

bus j	0	1	
bus type $Z^{(j)}$	0	1	

(b) You can model this using a variant of naïve bayes, where now 9 consecutive labels $Y^{(i)}, \dots, Y^{(i+8)}$ are *conditionally* independent given the bus type $Z^{(j)}$, for bus j and individual $i = 9j$. Assume the probability distributions take on the following form:

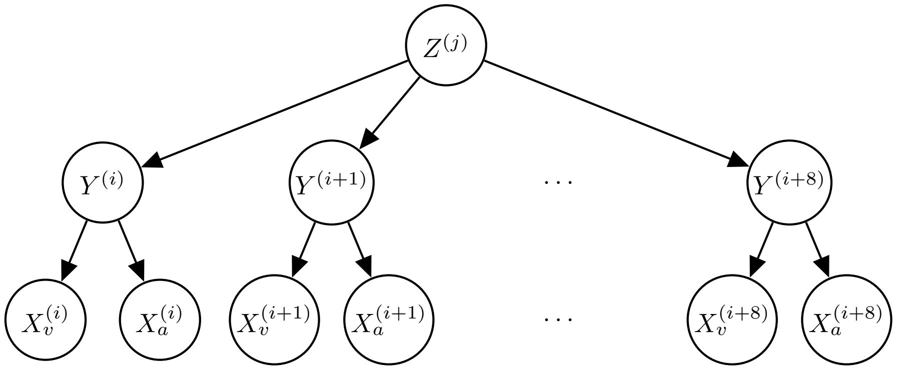
$$P(X_v^{(i)} = x_v | Y^{(i)} = y) = \begin{cases} p_v & \text{if } x_v = y \\ 1 - p_v & \text{if } x_v \neq y \end{cases}$$

$$P(X_a^{(i)} = x_a | Y^{(i)} = y) = \begin{cases} p_a & \text{if } x_a = y \\ 1 - p_a & \text{if } x_a \neq y \end{cases}$$

$$P(Y^{(i)} = 1 | Z^{(j)} = z) = \begin{cases} q_0 & \text{if } z = 0 \\ q_1 & \text{if } z = 1 \end{cases}$$

$$P(Z^{(j)} = 1) = r$$

for $p, q_0, q_1, r \in [0, 1]$ and $i, j \in \mathbb{N}$.



(i) What's the maximum likelihood estimate of q_0, q_1 and r?

$q_0 =$ $\frac{2}{9}$ $\qquad q_1 =$ $\frac{8}{9}$ $\qquad r =$ $\frac{1}{2}$

For r, we've seen one ghost bus and one pacman bus, so $r = 1/2$. For q_0, we're finding $P(Y = 1|Z = 0)$, which is $2/9$. For q_1, we're finding $P(Y = 1|Z = 1)$, which is $8/9$.

(ii) Compute the following joint probability. Simplify your answer as much as possible and express it in terms of the parameters p_v, p_a, q_0, q_1 and r (you might not need all of them).

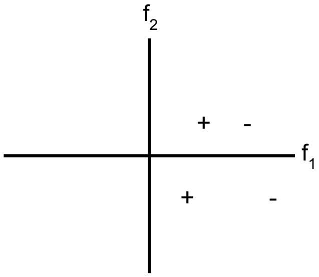
$P(Y^{(20)} = 1, X_v^{(20)} = 1, X_a^{(20)} = 1, Y^{(19)} = 1, Y^{(18)} = 1) =$ $\underline{\quad p_a p_v [q_0^3(1-r) + q_1^3 r] \quad}$

$$\begin{aligned}
&P(Y^{(20)} = 1, X_v^{(20)} = 1, X_a^{(20)} = 1, Y^{(19)} = 1, Y^{(18)} = 1) \\
&= \sum_z P(Y^{(20)} = 1 | Z^{(2)} = z) P(Z^{(2)} = z) P(X_v^{(20)} = 1 | Y^{(20)} = 1) P(X_a^{(20)} = 1 | Y^{(20)} = 1) \\
&\qquad P(Y^{(19)} = 1 | Z^{(2)} = z) P(Y^{(18)} = 1 | Z^{(2)} = z) \\
&= q_0(1-r) p_a p_v q_0 q_0 + q_1 r p_a p_v q_1 q_1 \\
&= p_a p_v [q_0^3(1-r) + q_1^3 r]
\end{aligned}$$

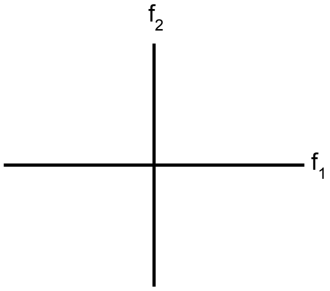
12 . Decision Trees and Other Classifiers

(a) Suppose you have a small training data set of four points in *distinct* locations, two from the "+" class and two from the "–" class. For each of the following conditions, draw a particular training data set (**of exactly four points**: +, +, −, and −) that satisfy the conditions. If this is impossible, mark "Not possible". *If "Not possible" is marked, we will ignore any data points.*

For example, if the conditions were "A depth-1 decision tree can perfectly classify the training data points," an acceptable answer would be the data points to the right.



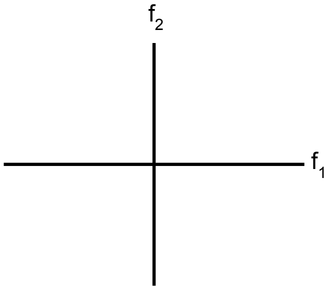
(i) A linear perceptron with a bias term can perfectly classify the training data points, but a linear perceptron without a bias term cannot.



○ Not possible

Any four points that are linearly separable, with the separating line clearly not passing through the origin

(ii) A depth-2 decision tree cannot classify the training data perfectly



● Not possible

Not possible, since the points must be in distinct locations.

(b) You are still trying to classify between "+" and "-", but your two features now can take on only three possible values, $\{-1, 0, 1\}$. You would like to use a Naive Bayes model with the following CPTs:

X	$P(X)$
-	0.4
+	0.6

X	F_1	$P(F_1\|X)$
-	-1	0.4
-	0	0.5
-	1	0.1
+	-1	0.7
+	0	0.1
+	1	0.2

X	F_2	$P(F_2\|X)$
-	-1	0.1
-	0	0.1
-	1	0.8
+	-1	0.6
+	0	0.1
+	1	0.3

(i) If you observe that $F_1 = -1$ and $F_2 = -1$, how will you classify X using Naive Bayes?

○ $X = -$ ● $X = +$

$P(F_1 = -1, F_2 = -1, X = +) = 0.7 * 0.6 * 0.6 > 0.4 * 0.1 * 0.4 = P(F_1 = -1, F_2 = -1, X = -)$

(ii) If you observe that $F_1 = 0$ and $F_2 = 0$, how will you classify X using Naive Bayes?

● $X = -$ ○ $X = +$

$P(F_1 = 0, F_2 = 0, X = +) = 0.1 * 0.1 * 0.6 < 0.5 * 0.1 * 0.4 = P(F_1 = 0, F_2 = 0, X = -)$

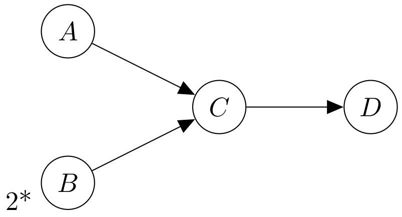
(iii) If you observe that $F_1 = 1$ and $F_2 = 1$, how will you classify X using Naive Bayes?

○ $X = -$ ● $X = +$

$P(F_1 = 1, F_2 = 1, X = +) = 0.2 * 0.3 * 0.6 > 0.8 * 0.1 * 0.4 = P(F_1 = 1, F_2 = 1, X = -)$

13 . Bayes' Net Sampling

Assume you are given the following Bayes' net and the corresponding distributions over the variables in the Bayes' net.



$P(A)$	
+a	0.1
-a	0.9

$P(B)$	
+b	.7
-b	.3

$P(C\|A, B)$			
+c	+a	+b	.25
-c	+a	+b	.75
+c	-a	+b	.6
-c	-a	+b	.4
+c	+a	-b	.5
-c	+a	-b	.5
+c	-a	-b	.2
-c	-a	-b	.8

$P(D\|C)$		
+d	+c	.5
-d	+c	.5
+d	-c	.8
-d	-c	.2

(a) Assume we receive evidence that $A = +a$. If we were to draw samples using rejection sampling, on expectation what percentage of the samples will be **rejected**?

Since $P(+a) = \frac{1}{10}$, we would expect that only 10% of the samples could be saved. Therefore, expected 90% of the samples will be rejected.

(b) Next, assume we observed both $A = +a$ and $D = +d$. What are the weights for the following samples under likelihood weighting sampling?

Sample	Weight
$(+a, -b, +c, +d)$	$P(+a) \cdot P(+d\| + c) = 0.1 * 0.5 = 0.05$
$(+a, -b, -c, +d)$	$P(+a) \cdot P(+d\| - c) = 0.1 * 0.8 = 0.08$
$(+a, +b, -c, +d)$	$P(+a) \cdot P(+d\| - c) = 0.1 * 0.8 = 0.08$

(c) Given the samples in the previous question, estimate $P(-b| + a, +d)$.

$$P(-b| + a, +d) = \frac{P(+a) \cdot P(+d| + c) + P(+a) \cdot P(+d| - c)}{P(+a) \cdot P(+d| + c) + 2 \cdot P(+a) \cdot P(+d| - c)} = \frac{0.05 + 0.08}{0.05 + 2 \cdot 0.08} = \frac{13}{21}$$

(d) Assume we need to (approximately) answer two different inference queries for this graph: $P(C| + a)$ and $P(C| + d)$. You are required to answer one query using likelihood weighting and one query using Gibbs sampling. In each case you can only collect a relatively small amount of samples, so for maximal accuracy you need to make sure you cleverly assign algorithm to query based on how well the algorithm fits the query. Which query would you answer with each algorithm?

Algorithm	Query
Likelihood Weighting	$P(C\| + a)$

Algorithm	Query
Gibbs Sampling	$P(C\| + d)$

Justify your answer:
You should use Gibbs sampling to find the query answer $P(C| + d)$. This is because likelihood weighting only takes upstream evidence into account when sampling. Therefore, Gibbs, which utilizes both upstream and downstream evidence, is more suited to the query $P(C| + d)$ which has downstream evidence.