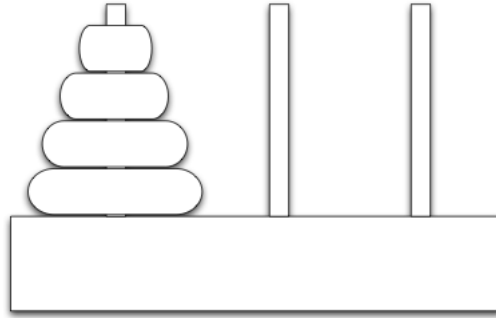


# CS 188 Fall 2018 Section 1: Search

## 1 Towers of Hanoi



The Towers of Hanoi is a famous problem for studying recursion in computer science and recurrence equations in discrete mathematics. We start with  $N$  discs of varying sizes on a peg (stacked in order according to size), and two empty pegs. We are allowed to move a disc from one peg to another, but we are never allowed to move a larger disc on top of a smaller disc. The goal is to move all the discs to the rightmost peg (see figure).

In this problem, we will formulate the Towers of Hanoi as a search problem.

(a) Propose a state representation for the problem

One possible state representation would be to store three lists, corresponding to which discs are on which peg. If we assume that the  $N$  discs are numbered in order of increasing size  $1, \dots, n$ , then we can represent each peg as an ordered list of integers corresponding to which discs are on that peg.

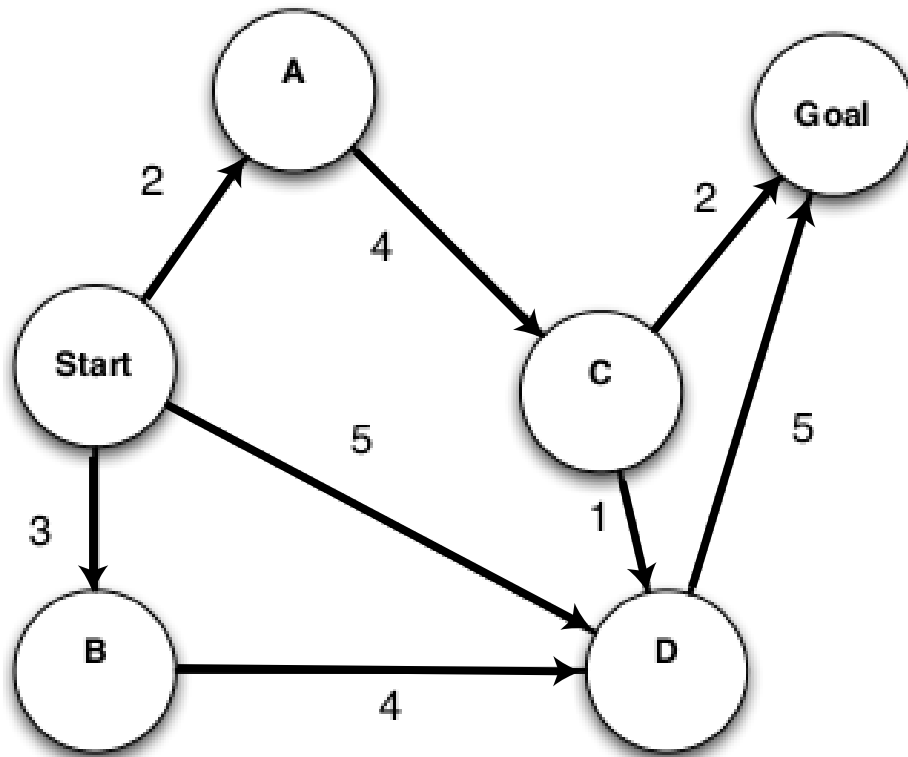
(b) What is the start state?  $([1, \dots, n], [], [])$

(c) From a given state, what actions are legal?

We can pop the first integer from any list (i.e., peg) and push it onto the front of another list (peg), so long as it is smaller than the integer currently at the front of the list being pushed to (i.e., peg being moved to).

(d) What is the goal test? Is the state the same as  $([], [], [1, \dots, n])$ ?

## 2 Search algorithms in action



For each of the following search strategies, work out the path returned by the search on the graph shown above. In all cases, assume ties resolve in such a way that states with earlier alphabetical order are expanded first. The start and goal state are S and G, respectively.

a) Depth-first search.

Path Returned: Start-A-C-D-Goal

b) Breadth-first search.

Path Returned: Start-D-Goal

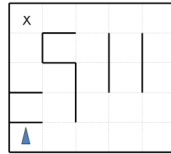
c) Uniform cost search.

Path Returned: Start-A-C-Goal

# CS188 Fall 2018 Section 2: Graph Search + CSPs

## 1 Search and Heuristics

Imagine a car-like agent wishes to exit a maze like the one shown below:



The agent is directional and at all times faces some direction  $d \in (N, S, E, W)$ . With a single action, the agent can *either* move forward at an adjustable velocity  $v$  *or* turn. The turning actions are *left* and *right*, which change the agent's direction by 90 degrees. Turning is only permitted when the velocity is zero (and leaves it at zero). The moving actions are *fast* and *slow*. *Fast* increments the velocity by 1 and *slow* decrements the velocity by 1; in both cases the agent then moves a number of squares equal to its NEW adjusted velocity. Any action that would result in a collision with a wall crashes the agent and is illegal. Any action that would reduce  $v$  below 0 or above a maximum speed  $V_{\max}$  is also illegal. The agent's goal is to find a plan which parks it (stationary) on the exit square using as few actions (time steps) as possible.

As an example: if the agent shown were initially stationary, it might first turn to the east using (*right*), then move one square east using *fast*, then two more squares east using *fast* again. The agent will of course have to *slow* to turn.

1. If the grid is  $M$  by  $N$ , what is the size of the state space? Justify your answer. You should assume that all configurations are reachable from the start state.

The size of the state space is  $4MN(V_{\max} + 1)$ . The state representation is (direction facing,  $x, y$ , speed). Note that the speed can take any value in  $\{0, \dots, V_{\max}\}$ .

2. Is the Manhattan distance from the agent's location to the exit's location admissible? Why or why not?

No, Manhattan distance is not an admissible heuristic. The agent can move at an average speed of greater than 1 (by first speeding up to  $V_{\max}$  and then slowing down to 0 as it reaches the goal), and so can reach the goal in less time steps than there are squares between it and the goal. A specific example: the target is 6 squares away, and the agent's velocity is already 4. By taking only 4 *slow* actions, it reaches the goal with a velocity of 0.

3. State and justify a non-trivial admissible heuristic for this problem which is not the Manhattan distance to the exit.

There are many answers to this question. Here are a few, in order of weakest to strongest:

- (a) The number of turns required for the agent to face the goal.
- (b) Consider a relaxation of the problem where there are no walls, the agent can turn and change speed arbitrarily. In this relaxed problem, the agent would move with  $V_{\max}$ , and then suddenly stop at the goal, thus taking  $d_{\text{manhattan}}/V_{\max}$  time.
- (c) We can improve the above relaxation by accounting for the deceleration dynamics. In this case the agent will have to slow down to 0 when it is about to reach the goal. Note that this heuristic will always return a greater value than the previous one, but is still not an overestimate of the true cost to reach the goal. We can say that this heuristic *dominates* the previous one.

4. If we used an inadmissible heuristic in A\* graph search, would the search be complete? Would it be optimal?

If the heuristic function is bounded, then A\* graph search would visit all the nodes eventually, and would find a path to the goal state if there exists one. An inadmissible heuristic does not guarantee optimality as it can make the good optimal goal look as though it is very far off, and take you to a suboptimal goal.

5. If we used an *admissible* heuristic in A\* graph search, is it guaranteed to return an optimal solution? What if the heuristic was consistent?

Admissible heuristics do not necessarily guarantee optimality; they are only guaranteed to return an optimal solution if they are consistent as well.

6. Give a general advantage that an inadmissible heuristic might have over an admissible one.

The time to solve an A\* search problem is a function of two factors: the number of nodes expanded, and the time spent per node. An inadmissible heuristic may be faster to compute, leading to a solution that is obtained faster due to less time spent per node. It can also be a closer estimate to the actual cost function (even though at times it will overestimate!), thus expanding less nodes. We lose the guarantee of optimality by using an inadmissible heuristic. But sometimes we may be okay with finding a suboptimal solution to a search problem.

## 2 Course Scheduling

You are in charge of scheduling for computer science classes that meet Mondays, Wednesdays and Fridays. There are 5 classes that meet on these days and 3 professors who will be teaching these classes. You are constrained by the fact that each professor can only teach one class at a time.

The classes are:

1. Class 1 - Intro to Programming: meets from 8:00-9:00am
2. Class 2 - Intro to Artificial Intelligence: meets from 8:30-9:30am
3. Class 3 - Natural Language Processing: meets from 9:00-10:00am
4. Class 4 - Computer Vision: meets from 9:00-10:00am
5. Class 5 - Machine Learning: meets from 10:30-11:30am

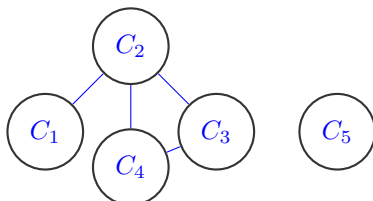
The professors are:

1. Professor A, who is qualified to teach Classes 1, 2, and 5.
2. Professor B, who is qualified to teach Classes 3, 4, and 5.
3. Professor C, who is qualified to teach Classes 1, 3, and 4.

1. Formulate this problem as a CSP problem in which there is one variable per class, stating the domains (after enforcing unary constraints), and binary constraints. Constraints should be specified formally and precisely, but may be implicit rather than explicit.

| Variables | Domains (or unary constraints) | Binary Constraints |
|-----------|--------------------------------|--------------------|
| $C_1$     | $\{A, C\}$                     | $C_1 \neq C_2$     |
| $C_2$     | $\{A\}$                        | $C_2 \neq C_3$     |
| $C_3$     | $\{B, C\}$                     | $C_2 \neq C_4$     |
| $C_4$     | $\{B, C\}$                     | $C_3 \neq C_4$     |
| $C_5$     | $\{A, B\}$                     |                    |

2. Draw the constraint graph associated with your CSP.

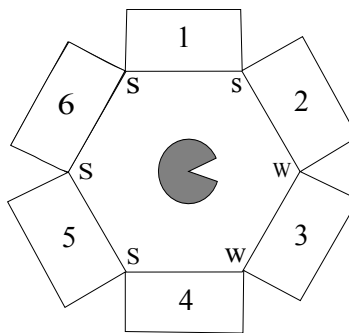


### 3 (Optional) CSPs: Trapped Pacman

Pacman is trapped! He is surrounded by mysterious corridors, each of which leads to either a pit (P), a ghost (G), or an exit (E). In order to escape, he needs to figure out which corridors, if any, lead to an exit and freedom, rather than the certain doom of a pit or a ghost.

The one sign of what lies behind the corridors is the wind: a pit produces a strong breeze (S) and an exit produces a weak breeze (W), while a ghost doesn't produce any breeze at all. Unfortunately, Pacman cannot measure the strength of the breeze at a specific corridor. Instead, he can stand *between* two adjacent corridors and feel the max of the two breezes. For example, if he stands between a pit and an exit he will sense a strong (S) breeze, while if he stands between an exit and a ghost, he will sense a weak (W) breeze. The measurements for all intersections are shown in the figure below.

Also, while the total number of exits might be zero, one, or more, Pacman knows that two neighboring squares will *not* both be exits.



Pacman models this problem using variables  $X_i$  for each corridor  $i$  and domains P, G, and E.

1. State the binary and/or unary constraints for this CSP (either implicitly or explicitly).

Binary:

$X_1 = P$  or  $X_2 = P$ ,     $X_2 = E$  or  $X_3 = E$ ,  
 $X_3 = E$  or  $X_4 = E$ ,     $X_4 = P$  or  $X_5 = P$ ,  
 $X_5 = P$  or  $X_6 = P$ ,     $X_1 = P$  or  $X_6 = P$ ,  
 $\forall i, j, Adj(i, j) \text{ and } \neg(X_i = E \text{ and } X_j = E)$

Unary:

$X_2 \neq P$ ,  
 $X_3 \neq P$ ,  
 $X_4 \neq P$

2. Cross out the values from the domains of the variables that will be deleted in enforcing arc consistency.

|       |   |   |   |
|-------|---|---|---|
| $X_1$ | P |   |   |
| $X_2$ |   | G | E |
| $X_3$ |   | G | E |
| $X_4$ |   | G | E |
| $X_5$ | P |   |   |
| $X_6$ | P | G | E |

3. According to MRV, which variable or variables could the solver assign first?

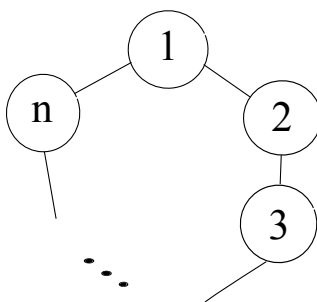
$X_1$  or  $X_5$  (tie breaking)

4. Assume that Pacman knows that  $X_6 = G$ . List all the solutions of this CSP or write *none* if no solutions exist.

(P,E,G,E,P,G)

(P,G,E,G,P,G)

The CSP described above has a circular structure with 6 variables. Now consider a CSP forming a circular structure that has  $n$  variables ( $n > 2$ ), as shown below. Also assume that the domain of each variable has cardinality  $d$ .



5. Explain precisely how to solve this general class of circle-structured CSPs efficiently (i.e. in time linear in the number of variables), using methods covered in class. Your answer should be at most two sentences.

We fix  $X_j$  for some  $j$  and assign it a value from its domain (i.e. use cutset conditioning on one variable). The rest of the CSP now forms a tree structure, which can be efficiently solved without backtracking by enforcing arc consistency. We try all possible values for our selected variable  $X_j$  until we find a solution.

6. 2 If standard backtracking search were run on a circle-structured graph, enforcing arc consistency at every step, what, if anything, can be said about the worst-case backtracking behavior (e.g. number of times the search could backtrack)?

A tree structured CSP can be solved without any backtracking. Thus, the above circle-structured CSP can be solved after backtracking at most  $d$  times, since we might have to try up to  $d$  values for  $X_j$  before finding a solution.

# CS188 Fall 2018 Section 3: CSPs + Games

## 1 CSP: Air Traffic Control

We have five planes: A, B, C, D, and E and two runways: international and domestic. We would like to schedule a time slot and runway for each aircraft to **either** land or take off. We have four time slots:  $\{1, 2, 3, 4\}$  for each runway, during which we can schedule a landing or take off of a plane. We must find an assignment that meets the following constraints:

- Plane B has lost an engine and must land in time slot 1.
- Plane D can only arrive at the airport to land during or after time slot 3.
- Plane A is running low on fuel but can last until at most time slot 2.
- Plane D must land before plane C takes off, because some passengers must transfer from D to C.
- No two aircrafts can reserve the same time slot for the same runway.

a) Complete the formulation of this problem as a CSP in terms of variables, domains, and constraints (both unary and binary). Constraints should be expressed implicitly using mathematical or logical notation rather than with words.

**Variables:** A, B, C, D, E for each plane.

**Domains:** a tuple  $(runway\ type, time\ slot)$  for runway type  $\in \{international, domestic\}$  and time slot  $\in \{1, 2, 3, 4\}$ .

**Constraints:**

$$\begin{array}{ll} B[1] = 1 & A[1] \leq 2 \\ D[1] \geq 3 & D[1] < C[1] \\ & A \neq B \neq C \neq D \neq E \end{array}$$

b) For the following subparts, we add the following two constraints:

- Planes A, B, and C cater to international flights and can only use the international runway.
- Planes D and E cater to domestic flights and can only use the domestic runway.

i With the addition of the two constraints above, we completely reformulate the CSP. You are given the variables and domains of the new formulation. Complete the constraint graph for this problem given the original constraints and the two added ones.



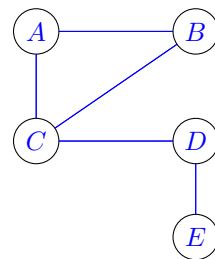
**Variables:** A, B, C, D, E for each plane.

**Constraint Graph:**

**Domains:**  $\{1, 2, 3, 4\}$

**Explanation of Constraint Graph:**

We can now encode the runway information into the identity of the variable, since each runway has more than enough time slots for the planes it serves. We represent the non-colliding time slot constraint as a binary constraint between the planes that use the same runways.



- ii What are the domains of the variables after enforcing arc-consistency? Begin by enforcing unary constraints. (Cross out values that are no longer in the domain.)

|   |              |              |              |              |
|---|--------------|--------------|--------------|--------------|
| A | <del>1</del> | 2            | <del>3</del> | <del>4</del> |
| B | 1            | <del>2</del> | <del>3</del> | <del>4</del> |
| C | <del>1</del> | <del>2</del> | <del>3</del> | 4            |
| D | <del>1</del> | <del>2</del> | 3            | <del>4</del> |
| E | 1            | 2            | <del>3</del> | 4            |

- iii Arc-consistency can be rather expensive to enforce, and we believe that we can obtain faster solutions using only **forward-checking** on our variable assignments. Using the Minimum Remaining Values heuristic, perform backtracking search on the graph, breaking ties by picking lower values and characters first. List the  $(variable, assignment)$  pairs in the order they occur (including the assignments that are reverted upon reaching a dead end). Enforce unary constraints before starting the search.

(You don't have to use this table, it won't be graded.)

|   |   |   |   |   |
|---|---|---|---|---|
| A | 1 | 2 | 3 | 4 |
| B | 1 | 2 | 3 | 4 |
| C | 1 | 2 | 3 | 4 |
| D | 1 | 2 | 3 | 4 |
| E | 1 | 2 | 3 | 4 |

**Answer:** (B, 1), (A, 2), (C, 3), (C, 4), (D, 3), (E, 1)

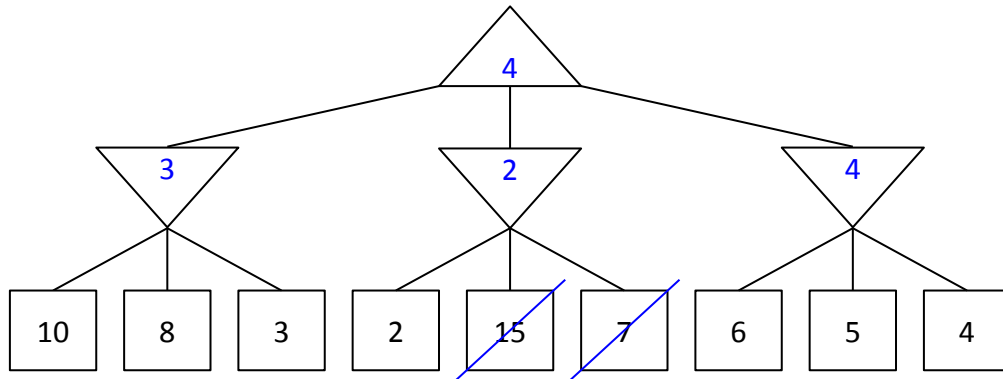
- c) Suppose we have just one runway and  $n$  planes, where no two planes can use the runway at once. We are assured that the constraint graph will always be tree-structured and that a solution exists. What is the runtime complexity in terms of the number of planes,  $n$ , of a CSP solver that runs arc-consistency and then assigns variables in a topological ordering?

$O(n^3)$ . Modified AC-3 for tree-structured CSPs runs arc-consistency backwards and then assigns variables in forward topological (linearized) ordering so we that we don't have to backtrack. The runtime complexity of modified AC-3 for tree-structured CSPs is  $O(nd^2)$ , but note that the domain of each variable must have a domain of size at least  $n$  since a solution exists

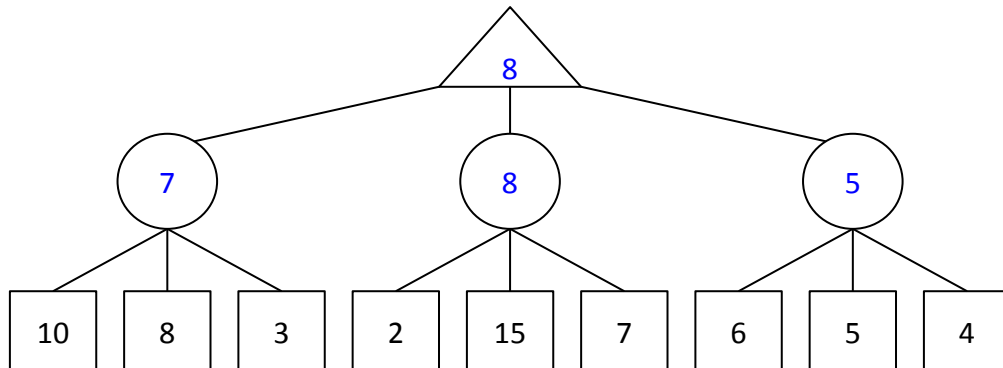
## 2 Games

1. Consider the zero-sum game tree shown below. Triangles that point up, such as at the top node (root), represent choices for the maximizing player; triangles that point down represent choices for the minimizing

player. Assuming both players act optimally, fill in the minimax value of each node.



2. Which nodes can be pruned from the game tree above through alpha-beta pruning? If no nodes can be pruned, explain why not. Assume the search goes from left to right; when choosing which child to visit first, choose the left-most unvisited child.
  
3. (optional) Again, consider the same zero-sum game tree, except that now, instead of a minimizing player, we have a chance node that will select one of the three values uniformly at random. Fill in the expectimax value of each node. The game tree is redrawn below for your convenience.

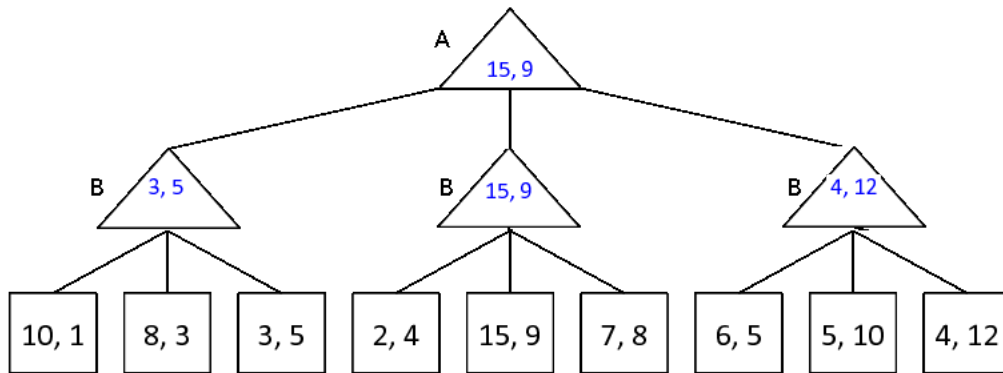


4. (optional) Which nodes can be pruned from the game tree above through alpha-beta pruning? If no nodes can be pruned, explain why not. **No nodes can be pruned. There will always be the possibility that some leaf further down the branch will have a very high value, which increases the overall average value.**

### 3 (Optional) Nonzero-sum Games

1. Let's look at a non-zero-sum version of a game. In this formulation, player A's utility will be represented as the first of the two leaf numbers, and player B's utility will be represented as the second of the two leaf

numbers. Fill in this non-zero game tree assuming each player is acting optimally.



2. Which nodes can be pruned from the game tree above through alpha-beta pruning? If no nodes can be pruned, explain why not. **No nodes can be pruned. Because this game is non-zero-sum, there can exist a leaf node anywhere in the tree that is good for both player A and player B.**

# CS188 Fall 2018 Section 4: Games and MDPs

## 1 Utilities

1. Consider a utility function of  $U(x) = 2x$ . What is the utility for each of the following outcomes?

(a) 3

$$U(3) = 2(3) = 6$$

(b)  $L(\frac{2}{3}, 3; \frac{1}{3}, 6)$

$$U(L(\frac{2}{3}, 3; \frac{1}{3}, 6)) = \frac{2}{3}U(3) + \frac{1}{3}U(6) = 8$$

(c) -2

$$U(-2) = 2(-2) = -4$$

(d)  $L(0.5, 2; 0.5, L(0.5, 4; 0.5, 6))$   $U(L(0.5, 2; 0.5, L(0.5, 4; 0.5, 6))) = 0.5U(2) + 0.5(0.5U(4) + 0.5U(6))$   
 $= 2 + 0.5(4 + 6) = 7$

2. Consider a utility function of  $U(x) = x^2$ . What is the utility for each of the following outcomes?

(a) 3

$$U(3) = 3^2 = 9$$

(b)  $L(\frac{2}{3}, 3; \frac{1}{3}, 6)$

$$U(L(\frac{2}{3}, 3; \frac{1}{3}, 6)) = \frac{2}{3}U(3) + \frac{1}{3}U(6) = 6 + 12 = 18$$

(c) -2

$$U(-2) = (-2)^2 = 4$$

(d)  $L(0.5, 2; 0.5, L(0.5, 4; 0.5, 6))$

$$U(L(0.5, 2; 0.5, L(0.5, 4; 0.5, 6))) = 0.5U(2) + 0.5(0.5U(4) + 0.5U(6)) = 2 + 0.5(8 + 18) = 15$$

3. What is the expected monetary value (EMV) of the lottery  $L(\frac{2}{3}, \$3; \frac{1}{3}, \$6)$ ?

$$\frac{2}{3} \cdot \$3 + \frac{1}{3} \cdot \$6 = \$4$$

4. For each of the following types of utility function, state how the utility of the lottery  $U(L)$  compares to the utility of the amount of money equal to the EMV of the lottery,  $U(EMV(L))$ . Write  $<$ ,  $>$ ,  $=$ , or  $?$  for can't tell.

(a)  $U$  is an arbitrary function.

$$U(L) \text{ ? } U(EMV(L))$$

(b)  $U$  is monotonically increasing and its rate of increase is increasing (its second derivative is positive).

$$U(L) > U(EMV(L)).$$

As an example, consider  $U = x^2$  from Q2. Then  $U(L) = 18$  and  $U(EMV(L)) = 4^2 = 16$ .

(c)  $U$  is monotonically increasing and linear (its second derivative is zero).

$$U(L) = U(EMV(L))$$

- (d)  $U$  is monotonically increasing and its rate of increase is decreasing (its second derivative is negative).  
 $U(L) < U(EMV(L))$ .  
 Consider  $U = \sqrt{x}$ . Then  $U(L) = \frac{2}{3} \cdot \sqrt{3} + \frac{1}{3} \cdot \sqrt{6} \approx 1.97$ , and  $U(EMV(L)) = \sqrt{4} = 2$ .

## 2 MDPs: Micro-Blackjack

In micro-blackjack, you repeatedly draw a card (with replacement) that is equally likely to be a 2, 3, or 4. You can either Draw or Stop if the total score of the cards you have drawn is less than 6. If your total score is 6 or higher, the game ends, and you receive a utility of 0. When you Stop, your utility is equal to your total score (up to 5), and the game ends. When you Draw, you receive no utility. There is no discount ( $\gamma = 1$ ). Let's formulate this problem as an MDP with the following states: 0, 2, 3, 4, 5 and a *Done* state, for when the game ends.

1. What is the transition function and the reward function for this MDP? The transition function is

$$\begin{aligned}
 T(s, \text{Stop}, \text{Done}) &= 1 \\
 T(0, \text{Draw}, s') &= 1/3 \text{ for } s' \in \{2, 3, 4\} \\
 T(2, \text{Draw}, s') &= 1/3 \text{ for } s' \in \{4, 5, \text{Done}\} \\
 T(3, \text{Draw}, s') &= \begin{cases} 1/3 \text{ if } s' = 5 \\ 2/3 \text{ if } s' = \text{Done} \end{cases} \\
 T(4, \text{Draw}, \text{Done}) &= 1 \\
 T(5, \text{Draw}, \text{Done}) &= 1 \\
 T(s, a, s') &= 0 \text{ otherwise}
 \end{aligned}$$

The reward function is

$$\begin{aligned}
 R(s, \text{Stop}, \text{Done}) &= s, s \leq 5 \\
 R(s, a, s') &= 0 \text{ otherwise}
 \end{aligned}$$

2. Fill in the following table of value iteration values for the first 4 iterations.

| States | 0    | 2 | 3 | 4 | 5 |
|--------|------|---|---|---|---|
| $V_0$  | 0    | 0 | 0 | 0 | 0 |
| $V_1$  | 0    | 2 | 3 | 4 | 5 |
| $V_2$  | 3    | 3 | 3 | 4 | 5 |
| $V_3$  | 10/3 | 3 | 3 | 4 | 5 |
| $V_4$  | 10/3 | 3 | 3 | 4 | 5 |

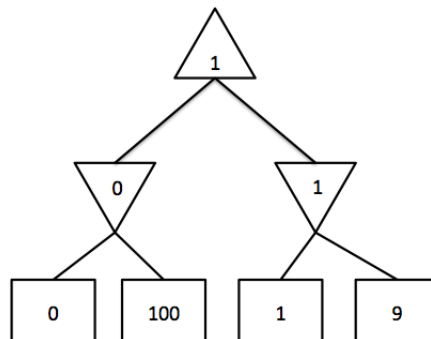
3. You should have noticed that value iteration converged above. What is the optimal policy for the MDP?

| States  | 0    | 2    | 3    | 4    | 5    |
|---------|------|------|------|------|------|
| $\pi^*$ | Draw | Draw | Stop | Stop | Stop |

### 3 (Optional) Minimax and Expectimax

In this problem, you will investigate the relationship between expectimax trees and minimax trees for zero-sum two player games. Imagine you have a game which alternates between player 1 (max) and player 2. The game begins in state  $s_0$ , with player 1 to move. Player 1 can either choose a move using minimax search, or expectimax search, where player 2's nodes are chance rather than min nodes.

1. Draw a (small) game tree in which the root node has a larger value if expectimax search is used than if minimax is used, or argue why it is not possible.



We can see here that the above game tree has a root value of 1 for the minimax strategy. If we instead switch to expectimax and replace the min nodes with chance nodes, the root of the tree takes on a value of 50 and the optimal action changes for MAX.

2. Draw a (small) game tree in which the root node has a larger value if minimax search is used than if expectimax is used, or argue why it is not possible.

Optimal play for MIN, by definition, means the best moves for MIN to obtain the lowest value possible. Random play includes moves that are not optimal. Assuming there are no ties (no two leaves have the same value), expectimax will always average in suboptimal moves. Averaging a suboptimal move (for MIN) against an optimal move (for MIN) will always increase the expected outcome.

With this in mind, we can see how there is no game tree where the value of the root for expectimax is lower than the value of the root for minimax. One is optimal play – the other is suboptimal play averaged with optimal play, which by definition leads to a higher value for MIN.

3. Under what assumptions about player 2 should player 1 use minimax search rather than expectimax search to select a move?

Player 1 should use minimax search if he/she expects player 2 to move optimally.

4. Under what assumptions about player 2 should player 1 use expectimax search rather than minimax search?

If player 1 expects player 2 to move randomly, he/she should use expectimax search. This will optimize for the maximum expected value.

5. Imagine that player 1 wishes to act optimally (rationally), and player 1 knows that player 2 also intends to act optimally. However, player 1 also knows that player 2 (mistakenly) believes that player 1 is moving uniformly at random rather than optimally. Explain how player 1 should use this knowledge to select a move. Your answer should be a precise algorithm involving a game tree search, and should include a sketch of an appropriate game tree with player 1's move at the root. Be clear what type of nodes are at each ply and whose turn each ply represents.

Use two games trees:

Game tree 1: max is replaced by a chance node. Solve this tree to find the policy of MIN.

Game tree 2: the original tree, but MIN doesn't have any choices now, instead is constrained to follow the policy found from Game Tree 1.

# CS188 Fall 2018 Section 5: MDP + RL

## 1 MDPs: Micro-Blackjack

In micro-blackjack, you repeatedly draw a card (with replacement) that is equally likely to be a 2, 3, or 4. You can either Draw or Stop if the total score of the cards you have drawn is less than 6. If your total score is 6 or higher, the game ends, and you receive a utility of 0. When you Stop, your utility is equal to your total score (up to 5), and the game ends. When you Draw, you receive no utility. There is no discount ( $\gamma = 1$ ). Let's formulate this problem as an MDP with the following states: 0, 2, 3, 4, 5 and a *Done* state, for when the game ends.

1. What is the transition function and the reward function for this MDP?

The transition function is

$$\begin{aligned}T(s, \text{Stop}, \text{Done}) &= 1 \\T(0, \text{Draw}, s') &= 1/3 \text{ for } s' \in \{2, 3, 4\} \\T(2, \text{Draw}, s') &= 1/3 \text{ for } s' \in \{4, 5, \text{Done}\} \\T(3, \text{Draw}, s') &= \begin{cases} 1/3 & \text{if } s' = 5 \\ 2/3 & \text{if } s' = \text{Done} \end{cases} \\T(4, \text{Draw}, \text{Done}) &= 1 \\T(5, \text{Draw}, \text{Done}) &= 1 \\T(s, a, s') &= 0 \text{ otherwise}\end{aligned}$$

The reward function is

$$\begin{aligned}R(s, \text{Stop}, \text{Done}) &= s, s \leq 5 \\R(s, a, s') &= 0 \text{ otherwise}\end{aligned}$$

2. Perform one iteration of policy iteration for one step of this MDP, starting from the fixed policy below:

| States      | 0    | 2    | 3    | 4    | 5    |
|-------------|------|------|------|------|------|
| $\pi_i$     | Draw | Stop | Draw | Stop | Draw |
| $V^{\pi_i}$ | 2    | 2    | 0    | 4    | 0    |
| $\pi_{i+1}$ | Draw | Stop | Stop | Stop | Stop |

## 2 Learning in Gridworld

Consider the example gridworld that we looked at in lecture. We would like to use TD learning and q-learning to find the values of these states.



|   |   |   |
|---|---|---|
|   | A |   |
| B | C | D |
|   | E |   |

Suppose that we have the following observed transitions:

(B, East, C, 2), (C, South, E, 4), (C, East, A, 6), (B, East, C, 2)

The initial value of each state is 0. Assume that  $\gamma = 1$  and  $\alpha = 0.5$ .

1. What are the learned values from TD learning after all four observations?

$$V(B) = 3.5$$

$$V(C) = 4$$

All other states have a value of 0.

2. What are the learned Q-values from Q-learning after all four observations?

$$Q(B, East) = 3$$

$$Q(C, South) = 2$$

$$Q(C, East) = 3$$

All other q-states have a value of 0.

### 3 Pacman with Feature-Based Q-Learning

We would like to use a Q-learning agent for Pacman, but the state size for a large grid is too massive to hold in memory. To solve this, we will switch to feature-based representation of Pacman's state.

1. Say our two minimal features are the number of ghosts within 1 step of Pacman ( $F_g$ ) and the number of food pellets within 1 step of Pacman ( $F_p$ ). You'll notice that these features depend only on the state, not the actions you take. Keep that in mind as you answer the next couple of questions. For this pacman board:



Extract the two features (calculate their values).

$$f_g = 2, f_p = 1$$

2. With Q Learning, we train off of a few episodes, so our weights begin to take on values. Right now  $w_g = 100$  and  $w_p = -10$ . Calculate the Q value for the state above.

First of all, the Q value will not depend on what action is taken, because the features we extract do not depend on the action, only the state.

$$Q(s, a) = w_g * f_g + w_p * f_p = 100 * 2 + -10 * 1 = 190$$

3. We receive an episode, so now we need to update our values. An episode consists of a start state  $s$ , an action  $a$ , an end state  $s'$ , and a reward  $r$ . The start state of the episode is the state above (where you already calculated the feature values and the expected Q value). The next state has feature values  $F_g = 0$  and  $F_p = 2$  and the reward is 50. Assuming a discount of  $\gamma = 0.5$ , calculate the new estimate of the Q value for  $s$  based on this episode.

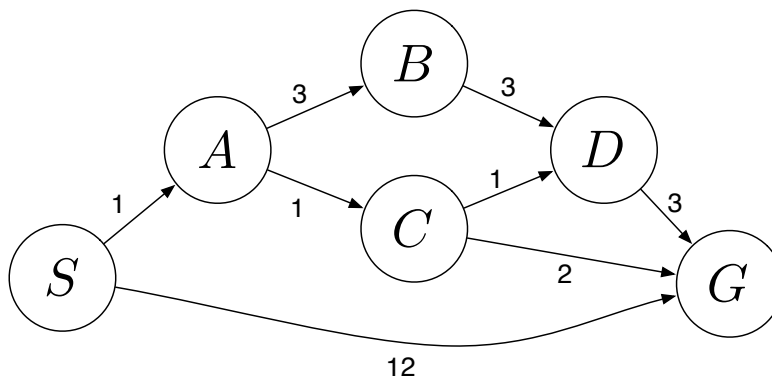
$$\begin{aligned} Q_{new}(s, a) &= R(s, a, s') + \gamma * \max_{a'} Q(s', a') \\ &= 50 + 0.5 * (100 * 0 + -10 * 2) \\ &= 40 \end{aligned}$$

4. With this new estimate and a learning rate ( $\alpha$ ) of 0.5, update the weights for each feature.

$$\begin{aligned} w_g &= w_g + \alpha * (Q_{new}(s, a) - Q(s, a)) * f_g(s, a) = 100 + 0.5 * (40 - 190) * 2 = -50 \\ w_p &= w_p + \alpha * (Q_{new}(s, a) - Q(s, a)) * f_p(s, a) = -10 + 0.5 * (40 - 190) * 1 = -85 \end{aligned}$$

Note that now the weight on ghosts is negative, which makes sense (ghosts should indeed be avoided). Although the weight on food pellets is now also negative, the difference between the two weights is now much lower.

## 1 . Search



Answer the following questions about the search problem shown above. Assume that ties are broken alphabetically. (For example, a partial plan  $S \rightarrow X \rightarrow A$  would be expanded before  $S \rightarrow X \rightarrow B$ ; similarly,  $S \rightarrow A \rightarrow Z$  would be expanded before  $S \rightarrow B \rightarrow A$ .) For the questions that ask for a path, please give your answers in the form ‘ $S - A - D - G$ .’

- (a) What path would breadth-first graph search return for this search problem?

$S - G$

- (b) What path would uniform cost graph search return for this search problem?

$S - A - C - G$

- (c) What path would depth-first graph search return for this search problem?

$S - A - B - D - G$

- (d) What path would A\* graph search, using a consistent heuristic, return for this search problem?

$S - A - C - G$

- (e) Consider the heuristics for this problem shown in the table below.

| State | $h_1$ | $h_2$ |
|-------|-------|-------|
| $S$   | 5     | 4     |
| $A$   | 3     | 2     |
| $B$   | 6     | 6     |
| $C$   | 2     | 1     |
| $D$   | 3     | 3     |
| $G$   | 0     | 0     |

- (i) Is  $h_1$  admissible? **Yes** No

- (ii) Is  $h_1$  consistent? **Yes** No

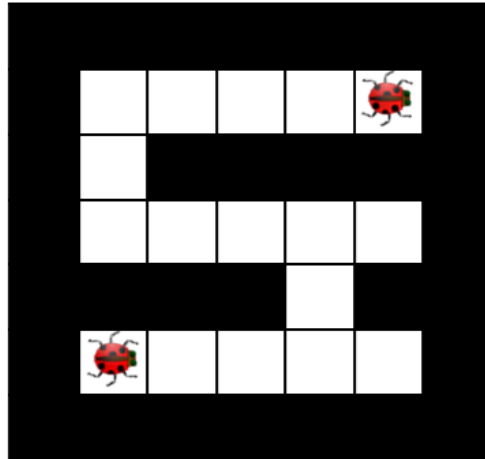
(iii) Is  $h_2$  admissible? ☒ **Yes** ☐ **No**

(iv) Is  $h_2$  consistent? ☐ **Yes** ☒ **No**

## 2 . Hive Minds: Redux

Let's revisit our bug friends. To recap, you control one or more insects in a rectangular maze-like environment with dimensions  $M \times N$ , as shown in the figures below. At each time step, an insect can move North, East, South, or West (but not diagonally) into an adjacent square if that square is currently free, or the insect may stay in its current location. Squares may be blocked by walls (as denoted by the black squares), but the map is known.

For the following questions, you should answer for a general instance of the problem, not simply for the example maps shown.



You now control a pair of long lost bug friends. You know the maze, but you do not have any information about which square each bug starts in. You want to help the bugs reunite. You must pose a search problem whose solution is an all-purpose sequence of actions such that, after executing those actions, both bugs will be on the same square, regardless of their initial positions. Any square will do, as the bugs have no goal in mind other than to see each other once again. Both bugs execute the actions mindlessly and do not know whether their moves succeed; if they use an action which would move them in a blocked direction, they will stay where they are. Unlike the flea in the previous question, bugs *cannot* jump onto walls. Both bugs can move in each time step. Every time step that passes has a cost of one.

(a) Give a *minimal* state representation for the above search problem.

A list of boolean variables, one for each position in the maze, indicating whether the position could contain a bug. You don't keep track of each bug separately because you don't know where each one starts; therefore, you need the same set of actions for each bug to ensure that they meet.

(b) Give the size of the state space for this search problem.

$$2^{MN}$$

(c) Give a nontrivial admissible heuristic for this search problem.

$h_{\text{friends}}$  = the maximum Manhattan distance of all possible pairs of points the bugs can be in.

### 3 . CSPs: Time Management

Two of our TAs, Arjun and Dave, are making their schedules for a busy morning. There are five tasks to be carried out:

- (F) Pick up food for the group's research seminar, which, sadly, takes one precious hour.
- (H) Prepare homework questions, which takes 2 consecutive hours.
- (P) Prepare the PR2 (robot that Pieter uses for research) for a group of preschoolers' visit, which takes one hour.
- (S) Lead the research seminar, which takes one hour.
- (T) Teach the preschoolers about the PR2 robot, which takes 2 consecutive hours.

The schedule consists of one-hour slots: 8am-9am, 9am-10am, 10am-11am, 11am-12pm. The requirements for the schedule are as follows:

1. In any given time slot each TA can do at most one task (F, H, P, S, T).
2. The PR2 preparation (P) should happen before teaching the preschoolers (T).
3. The food should be picked up (F) before the seminar (S).
4. The seminar (S) should be finished by 10am.
5. Arjun is going to deal with food pick up (F) since he has a car.
6. The TA not leading the seminar (S) should still attend, and hence cannot perform another task (F, T, P, H) during the seminar.
7. The seminar (S) leader does not teach the preschoolers (T).
8. The TA who teaches the preschoolers (T) must also prepare the PR2 robot (P).
9. Preparing homework questions (H) takes 2 consecutive hours, and hence should start at or before 10am.
10. Teaching the preschoolers (T) takes 2 consecutive hours, and hence should start at or before 10am.

To formalize this problem as a CSP, use the variables F, H, P, S and T. The values they take on indicate the TA responsible for it, and the starting time slot during which the task is carried out (for a task that spans 2 hours, the variable represents the starting time, but keep in mind that the TA will be occupied for the next hour also - make sure you enforce constraint (a)!). Hence there are eight possible values for each variable, which we will denote by A8, A9, A10, A11, D8, D9, D10, D11, where the letter corresponds to the TA and the number corresponds to the time slot. For example, assigning the value of A8 to a variables means that this task is carried about by Arjun from 8am to 9am.

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

8<sup>5</sup>.

(b) Which of the statements above include unary constraints?

(d), (e), (i), (j). (i) and (j) are both unary constraints, and binary constraints in a single sentence.

(c) In the table below, enforce all unary constraints by crossing out values in the table on the left below. If you made a mistake, cross out the whole table and use the right one.

|   |    |    |                |                |               |               |                |                |
|---|----|----|----------------|----------------|---------------|---------------|----------------|----------------|
| F | A8 | A9 | A10            | A11            | <del>D8</del> | <del>D9</del> | <del>D10</del> | <del>D11</del> |
| H | A8 | A9 | A10            | <del>A11</del> | D8            | D9            | D10            | <del>D11</del> |
| P | A8 | A9 | A10            | A11            | D8            | D9            | D10            | D11            |
| S | A8 | A9 | <del>A10</del> | <del>A11</del> | D8            | D9            | <del>D10</del> | <del>D11</del> |
| T | A8 | A9 | A10            | <del>A11</del> | D8            | D9            | D10            | <del>D11</del> |

- (d) Start from the table above, select the variable S and assign the value A9 to it. Perform forward checking by crossing out values in the table below. Again the table on the right is for you to use in case you believe you made a mistake.

|   |    |    |     |     |    |    |     |     |
|---|----|----|-----|-----|----|----|-----|-----|
| F | A8 | A9 | A10 | A11 | D8 | D9 | D10 | D11 |
| H | A8 | A9 | A10 | A11 | D8 | D9 | D10 | D11 |
| P | A8 | A9 | A10 | A11 | D8 | D9 | D10 | D11 |
| S | A8 | A9 | A10 | A11 | D8 | D9 | D10 | D11 |
| T | A8 | A9 | A10 | A11 | D8 | D9 | D10 | D11 |

- (e) Based on the result of (d), what variable will we choose to assign next based on the MRV heuristic (breaking ties alphabetically)? Assign the first possible value to this variable, and perform forward checking by crossing out values in the table below. Again the table on the right is for you to use in case you believe you made a mistake.

Variable F is selected and gets assigned value A8.

|   |    |    |     |     |    |    |     |     |
|---|----|----|-----|-----|----|----|-----|-----|
| F | A8 | A9 | A10 | A11 | D8 | D9 | D10 | D11 |
| H | A8 | A9 | A10 | A11 | D8 | D9 | D10 | D11 |
| P | A8 | A9 | A10 | A11 | D8 | D9 | D10 | D11 |
| S | A8 | A9 | A10 | A11 | D8 | D9 | D10 | D11 |
| T | A8 | A9 | A10 | A11 | D8 | D9 | D10 | D11 |

Have we arrived at a dead end (i.e., has any of the domains become empty)?

No.

- (f) We return to the result from enforcing just the unary constraints, which we did in (c). Select the variable S and assign the value A9. Enforce arc consistency by crossing out values in the table below.

|   |    |    |     |     |    |    |     |     |
|---|----|----|-----|-----|----|----|-----|-----|
| F | A8 | A9 | A10 | A11 | D8 | D9 | D10 | D11 |
| H | A8 | A9 | A10 | A11 | D8 | D9 | D10 | D11 |
| P | A8 | A9 | A10 | A11 | D8 | D9 | D10 | D11 |
| S | A8 | A9 | A10 | A11 | D8 | D9 | D10 | D11 |
| T | A8 | A9 | A10 | A11 | D8 | D9 | D10 | D11 |

- (g) Compare your answers to (d) and to (f). Does arc consistency remove more values or less values than forward checking does? Explain why.

Arc consistency removes more values. It's because AC checks consistency between any pair of variables, while FC only checks the relationship between pairs of assigned and unassigned variables.

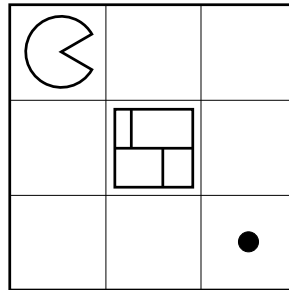
- (h) Check your answer to (f). Without backtracking, does any solution exist along this path? Provide the solution(s) or state that there is none.

AC along this path gives 1 solution: F: A8 H: A10 P: D8 S: A9 T: D10

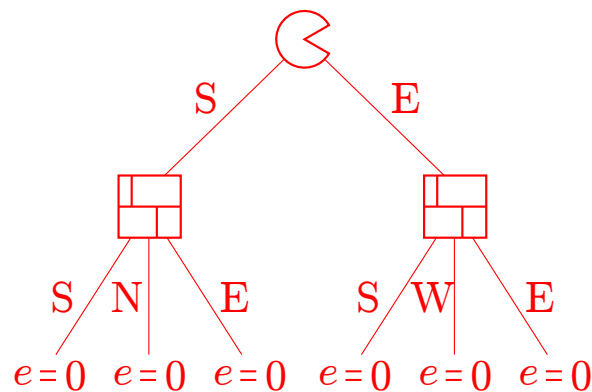
## 4 . Surrealist Pacman

In the game of Surrealist Pacman, Pacman  $\ominus$  plays against a moving wall  $\boxplus$ . On Pacman's turn, Pacman must move in one of the four cardinal directions, and must move into an unoccupied square. On the wall's turn, the wall must move in one of the four cardinal directions, and must move into an unoccupied square. The wall cannot move into a dot-containing square. Staying still is not allowed by either player. Pacman's score is always equal to the number of dots he has eaten.

The first game begins in the configuration shown below. Pacman moves first.



- (a) Draw a game tree with one move for each player. Nodes in the tree represent game states (location of all agents and walls). Edges in the tree connect successor states to their parent states. Draw only the legal moves.



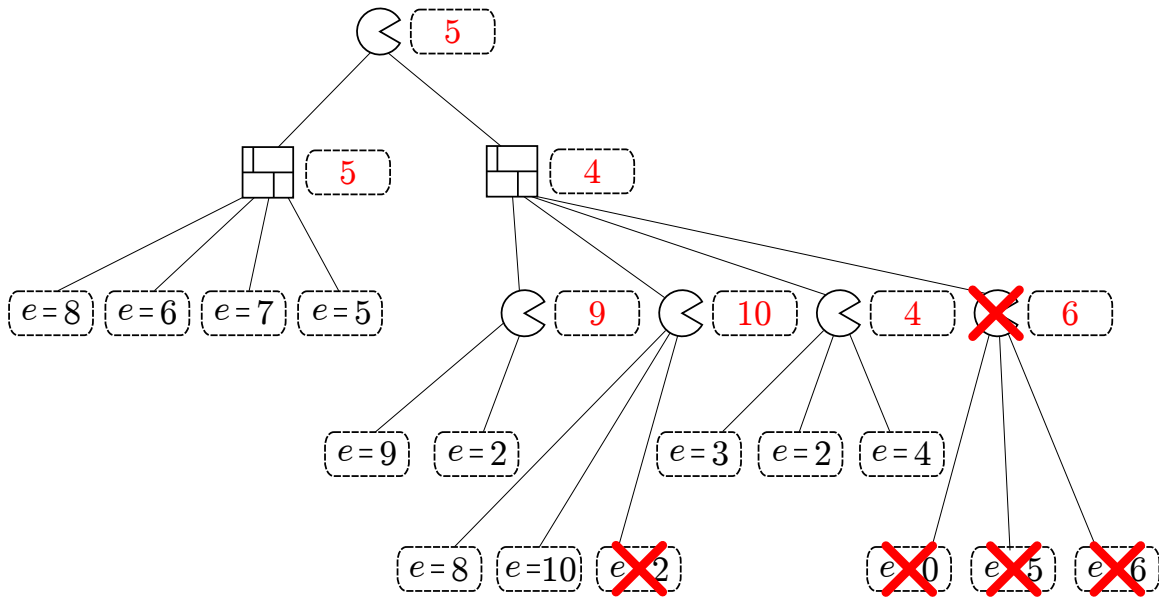
- (b) According to the depth-limited game tree you drew above what is the value of the game? Use Pacman's score as your evaluation function.
0. All leaves have value 0.
- (c) If we were to consider a game tree with ten moves for each player (rather than just one), what would be the value of the game as computed by minimax?
1. Pacman can force a win in ten moves.



A second game is played on a more complicated board. A partial game tree is drawn, and leaf nodes have been scored using an (unknown) evaluation function  $e$ .

(d) In the dashed boxes, fill in the values of all internal nodes using the minimax algorithm.

(e) Cross off any nodes that are not evaluated when using alpha-beta pruning (assuming the standard left-to-right traversal of the tree).



Running alpha-beta pruning on the game tree.

Root:  $\alpha = -\infty, \beta = \infty$

- - Left wall:  $\alpha = -\infty, \beta = \infty$

- - - - Leaf node:  $e = 8$ . Propagate  $e = 8$  back to parent.

- - Left wall: Current value is 8.  $\alpha = -\infty, \beta = 8$ . Max doesn't have a best value, so continue exploring.

- - - - Leaf node:  $e = 6$ . Propagate  $e = 6$  back to parent.

- - Left wall: Current value is 6.  $\alpha = -\infty, \beta = 6$ . Max doesn't have a best value, so continue exploring.

- - - - Leaf node:  $e = 7$ . Propagate  $e = 7$  back to parent.

- - Left wall: No update. Current value is 6.  $\alpha = -\infty, \beta = 6$ . Max doesn't have a best value, so continue exploring.

- - - - Leaf node:  $e = 5$ . Propagate  $e = 5$  back to parent.

- - Left wall: Current value is 5. We're done here, so propagate 5 to root.

Root: Current value is 5.  $\alpha = 5, \beta = \infty$ . Explore right.

- - Right wall:  $\alpha = 5, \beta = \infty$ .

- - - - 1st Pac:  $\alpha = 5, \beta = \infty$

- - - - - Leaf node:  $e = 9$ . Propagate  $e = 9$  back to parent.

- - - - 1st Pac: Current value is 9.  $\alpha = 9, \beta = \infty$  MIN doesn't have a best value, so continue exploring.

- - - - - Leaf node:  $e = 2$ . Propagate  $e = 2$  back to parent.

- - - - 1st Pac: No change. Current value is 9. Propagate 9 to parent.

- - Right wall: Current value is now 9.  $\alpha = 5, \beta = 9$ . MIN wants anything less than 9 at this point, but it's still possible for MAX to get more than 5. Continue exploring.

- - - - 2nd Pac:  $\alpha = 5, \beta = 9$

- - - - - Leaf node:  $e = 8$ . Propagate  $e = 8$  back to parent.

- - - - 2nd Pac: Current value is now 8.  $\alpha = 8, \beta = 9$ . Again, still possible for both players to benefit (Imagine value = 8.5). Continue exploring.

- - - - - Leaf node:  $e = 10$ . Propagate  $e = 10$  back to parent.

- - - - 2nd Pac: Current value is now 10. So now, we know that  $v > \beta$ , which means that one of the players is going to be unhappy. MAX wants something more than 10, but MIN is only satisfied with something less than 9, so we don't have to keep exploring.

- - - - *PRUNE*  $e = 2$ .

- - - - 2nd Pac: returns value of 10 to parent.

- - Left Wall: No change in value, current value is still 9.  $\alpha = 5, \beta = 9$ . Again, still possible for both players to benefit, so continue exploring.

- - - - 3rd Pac:  $\alpha = 5, \beta = 9$

- - - - - Leaf node:  $e = 3$ . Propagate  $e = 3$  back to parent.

- - - - 3rd Pac: Current value is 3.  $\alpha = 5, \beta = 9$ . Continue exploring.

- - - - - Leaf node:  $e = 2$ . Propagate  $e = 2$  back to parent.

- - - - 3rd Pac: No change in value. Current value is 3.  $\alpha = 5, \beta = 9$ . Continue exploring.

- - - - - Leaf node:  $e = 4$ . Propagate  $e = 4$  back to parent.

- - - - 3rd Pac: Current value is 4. We're done, so return value of 4 to parent.

- - Left Wall: Current value becomes 4. At this point, we know that MIN wants anything that is less than or equal to 4. However, MAX is only satisfied with something that is 5 or greater. Hence, we don't need to explore the rest of the children of this node since MAX will never let the game get down to this branch.- -

*Prune rest*

(Filling values returned by alpha-beta or not crossing off children of a crossed off node were not penalized.)

Suppose that this evaluation function has a special property: it is known to give the correct minimax value of any internal node to within 2, and the correct minimax values of the leaf nodes exactly. That is, if  $v$  is the true minimax value of a particular node, and  $e$  is the value of the evaluation function applied to that node,  $e - 2 \leq v \leq e + 2$ , and  $v = e$  if the node is a dashed box in the tree below.

Using this special property, you can modify the alpha-beta pruning algorithm to prune more nodes.

- (f) Standard alpha-beta pseudocode is given below (only the max-value recursion). Fill in the boxes on the right to replace the corresponding boxes on the left so that the pseudocode prunes as many nodes as possible, taking account of this special property of the evaluation function.

```

function MAX-VALUE(node,  $\alpha$ ,  $\beta$ )
   $e \leftarrow$  EVALUATIONFUNCTION(node)
  if node is leaf then
    return  $e$ 
  end if (1)
   $v \leftarrow -\infty$ 
  for child  $\leftarrow$  CHILDREN(node) do

$v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\textit{child}, \alpha, \beta))$

 (2)
    if  $v \geq \beta$  then
      return  $v$ 
    end if
     $\alpha \leftarrow \text{MAX}(\alpha, v)$ 
  end for
  return  $v$ 
end function

```

Fill in these boxes:

(1)

**if**  $e - 2 \geq \beta$  **then**  
     **return**  $e - 2$   
**end if**

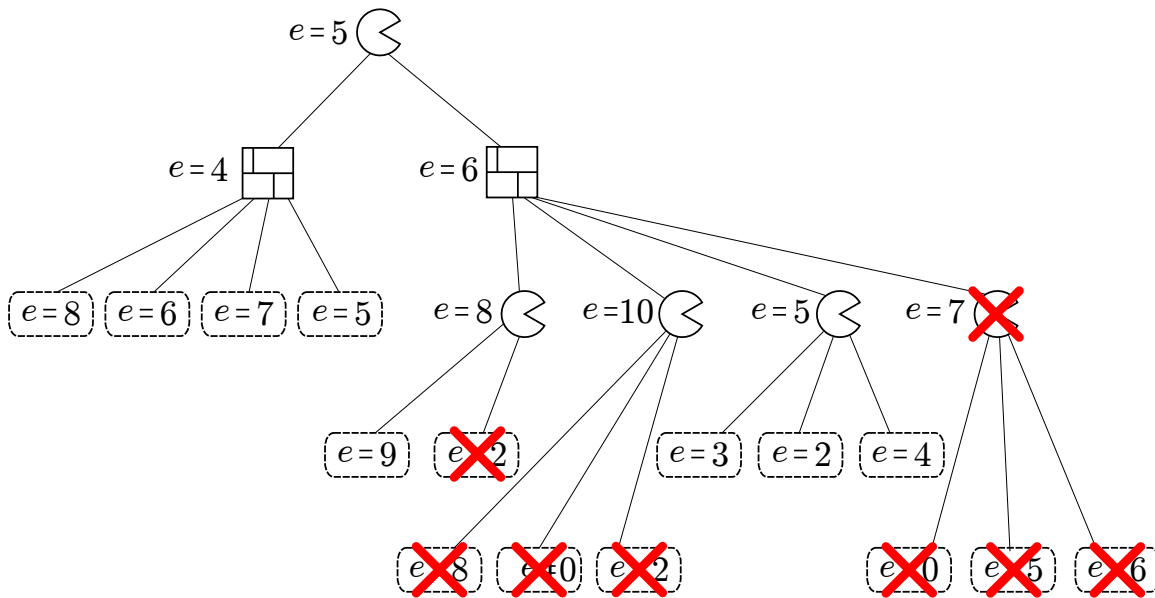
(2)

$v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\textit{child},$   
 $\text{MAX}(\alpha, e - 2), \text{MIN}(\beta, e + 2)))$

(Variations are possible.)

The same game tree is shown below, with the evaluation function applied to *internal* as well as leaf nodes.

- (g) In the game tree below cross off any nodes that can be pruned assuming the special property holds true. If not sure you correctly formalized into pseudo-code your intuition on how to exploit the special property for improved pruning, make sure to annotate your pruned nodes with a brief explanation of why each of them was pruned.



Running pruning on game tree in detail. W1 and W2 refer to the left and right wall nodes respectively. P1, P2, P3, P4 refer to Pacman nodes on the third level, left to right.

Root: Evaluation function returns 5. Range of value:  $[3, 7]$ . Set  $\alpha : 3, \beta : 7$  and explore(W1,  $\alpha, \beta$ ).

- W1: Evaluation function returns 4. Range of value:  $[2, 6]$ . Set  $\alpha : 3, \beta : 6$ .
- - Leaf node:  $e = 8$ . Send 8 back to W1.
- W1:  $v = 8$ .  $v < \alpha$ ? No. This means that MAX might still prefer this path.
- - Leaf node:  $e = 6$ . Send 6 back to W1.
- W1:  $6 < 8$  so  $v = 6$ .  $v < \alpha$ ? No. Continue exploring.
- - Leaf node:  $e = 7$ . Send 7 back to W1.
- W1:  $7 > 6$ , so no change,  $v = 6$ .  $v < \alpha$ ? No. Continue exploring.
- - Leaf node:  $e = 5$ . Send 5 back to W1.
- W1:  $5 < 6$ , so  $v = 5$ . Done. Send 5 back to root.

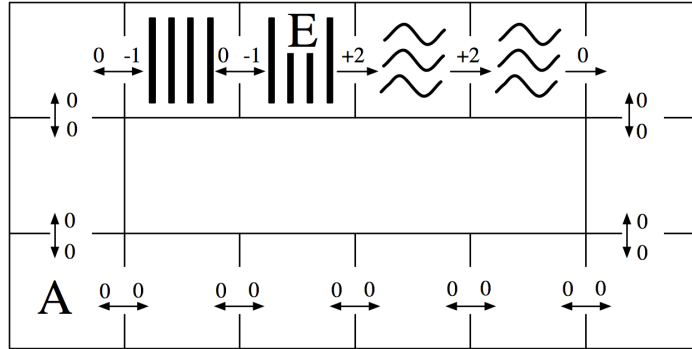
Root: Gets value 5, so  $v = 5$ .  $\alpha : \max(3, 5) = 5, \beta : 7$ . Still exploring right branch since MAX could get a value  $5 \leq v \leq 7$ . Explore(W2,  $\alpha, \beta$ ).

- W2: Evaluation function returns 6. Range of values  $[4, 8]$ .  $\alpha : 5, \beta : 7$ , explore(P1,  $\alpha, \beta$ ).
- - P1: Evaluation function returns 8. Range:  $[6, 10]$ ,  $\alpha : 6, \beta : 7$ .
- - - Leaf node:  $e = 9$ . Send  $e = 9$  back to P1.
- - P1:  $v = 9$ .  $v > \beta$ ? *Yes!*. We can prune here since P1 wants any value  $> 9$ . However, at the root we know that the maximum value that MAX can get is 7. Hence, there is no way the game can get down to P1. (Meaning that the value at the root can not be 9).
- - - Leaf node: Prune  $e = 2$ . Return 9 to W2.
- W2:  $v = 9$ .  $\alpha, \beta$  don't change:  $\alpha : 5, \beta : 7$ . Explore(P2,  $\alpha, \beta$ ).
- - P2: Evaluation function returns 10. Range  $[8, 12]$ ,  $\alpha : 8, \beta : 7$ . Notice that the best value for MIN that can be achieved at P2 is 8. However, the best value for MIN at the root is 7. Hence, there's no way the game can get down to P2. (Meaning the value of the root can not be 8). *Prune all of P2's children!*. We can return 8 to W2 since we know that there is some other path through W2 that yields a reward  $\leq 7$ .
- W2:  $v : \min(8, 9) = 8, \alpha : 5, \beta : 7$ .  $v < \alpha$ ? No! Explore(P3,  $\alpha, \beta$ ).
- - P3: Evaluation function returns 5. Range:  $[3, 7]$ ,  $\alpha : 5, \beta : 7$ .
- - - Leaf node:  $e = 5$ . Send 3 back to P3.
- - P3:  $v = 3, v > \beta$ ? No! Meaning MIN might still prefer this branch.  $\alpha : 5, \beta : 7$ .
- - - Leaf node:  $e = 2$ . Send 2 back to P3.
- - P3:  $v = 3, v > \beta$ ? No!  $\alpha : 5, \beta : 7$ .
- - - Leaf node:  $e = 4$ . Send 4 back to P3.
- - P3:  $v = 4$ . Done. Return 4 to W2.
- W2:  $v : \min(8, 4) = 4, \alpha : 5, \beta : 7$ .  $v < \alpha$ ? *Yes!* Since MAX can guarantee a value of 5 and MIN will only accept something  $< 4$ , don't need to explore any further. *Prune P4 and all its children*. Return 4 to root.

Root: *Done*.

## 5 . MDPs: Grid-World Water Park

Consider the MDP drawn below. The state space consists of all squares in a grid-world water park. There is a single waterslide that is composed of two ladder squares and two slide squares (marked with vertical bars and squiggly lines respectively). An agent in this water park can move from any square to any neighboring square, unless the current square is a slide in which case it must move forward one square along the slide. The actions are denoted by arrows between squares on the map and all deterministically move the agent in the given direction. The agent cannot stand still: it must move on each time step. Rewards are also shown below: the agent feels great pleasure as it slides down the water slide (+2), a certain amount of discomfort as it climbs the rungs of the ladder (-1), and receives rewards of 0 otherwise. The time horizon is infinite; this MDP goes on forever.



(a) How many (deterministic) policies  $\pi$  are possible for this MDP?

$2^{11}$

(b) Fill in the blank cells of this table with values that are correct for the corresponding function, discount, and state. *Hint: You should not need to do substantial calculation here.*

|                            | $\gamma$ | $s = A$  | $s = E$  |
|----------------------------|----------|----------|----------|
| $V_3^*(s)$                 | 1.0      | 0        | 4        |
| $V_{10}^*(s)$              | 1.0      | 2        | 4        |
| $V_{10}^*(s)$              | 0.1      | 0        | 2.2      |
| $Q_1^*(s, \text{west})$    | 1.0      | —        | 0        |
| $Q_{10}^*(s, \text{west})$ | 1.0      | —        | 3        |
| $V^*(s)$                   | 1.0      | $\infty$ | $\infty$ |
| $V^*(s)$                   | 0.1      | 0        | 2.2      |

$V_{10}^*(A), \gamma = 1$ : In 10 time steps with no discounting, the rewards don't decay, so the optimal strategy is to climb the two stairs (-1 reward each), and then slide down the two slide squares (+2 rewards each). You only have time to do this once. Summing this up, we get  $-1 - 1 + 2 + 2 = 2$ .

$V_{10}^*(E), \gamma = 1$ : No discounting, so optimal strategy is sliding down the slide. That's all you have time for. Sum of rewards =  $2 + 2 = 4$ .

$V_{10}^*(A), \gamma = 0.1$ . The discount rate is 0.1, meaning that rewards 1 step further into the future are discounted by a factor of 0.1. Let's assume from A, we went for the slide. Then, we would have to take the actions  $A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow F, F \rightarrow G$ . We get the first -1 reward from  $C \rightarrow D$ , discounted by  $\gamma^2$  since it is two actions in the future.  $D \rightarrow E$  is discounted by  $\gamma^3$ ,  $E \rightarrow F$  by  $\gamma^4$ , and  $F \rightarrow G$  by  $\gamma^5$ . Since  $\gamma$  is low, the positive rewards you get from the slide have less of an effect as the larger negative rewards you get from climbing up. Hence, the sum of rewards of taking the slide path would be negative; the optimal value is 0.

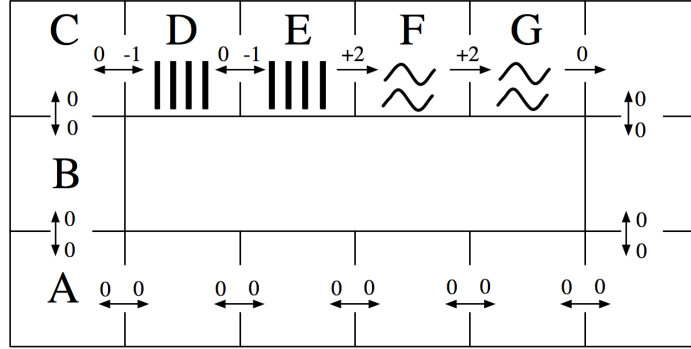
$V_{10}^*(E), \gamma = 0.1$ . Now, you don't have to do the work of climbing up the stairs, and you just take the slide down. Sum of rewards would be 2 (for  $E \rightarrow F$ ) + 0.2 (for  $F \rightarrow G$ , discounted by 0.1) = 2.2.

$Q_{10}^*(E, west), \gamma = 1$ . Remember that a Q-state (s,a) is when you start from state  $s$  and are committed to taking  $a$ . Hence, from E, you take the action West and land in D, using up one time step and getting an immediate reward of 0. From D, the optimal strategy is to climb back up the higher flight of stairs and then slide down the slide. Hence, the rewards would be  $-1(D \rightarrow E) + 2(E \rightarrow F) + 2(F \rightarrow G) = 3$ .

$V^*(s), \gamma = 1$ . Infinite game with no discount? Have fun sliding down the slide to your content from anywhere.

$V^*(s), \gamma = 0.1$ . Same reasoning apply to both A and E from  $V_{10}^*(s)$ . With discounting, the stairs are more costly to climb than the reward you get from sliding down the water slide. Hence, at A, you wouldn't want to head to the slide. From E, since you are already at the top of the slide, you should just slide down.

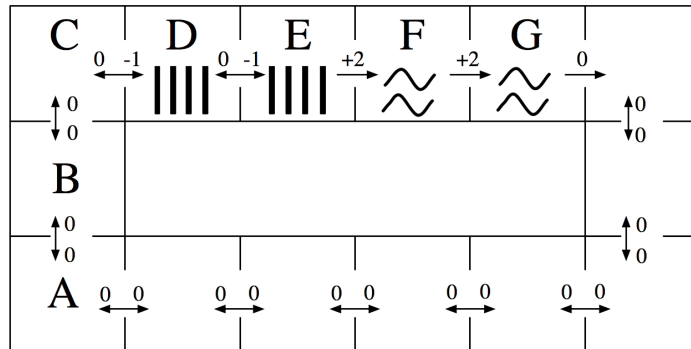
Use this labeling of the state space to complete the remaining subproblems:



- (c) Fill in the blank cells of this table with the Q-values that result from applying the Q-update for the transition specified on each row. You may leave Q-values that are unaffected by the current update blank. Use discount  $\gamma = 1.0$  and learning rate  $\alpha = 0.5$ . Assume all Q-values are initialized to 0. (Note: the specified transitions would not arise from a single episode.)

|                                                          | $Q(D, \text{west})$ | $Q(D, \text{east})$ | $Q(E, \text{west})$ | $Q(E, \text{east})$ |
|----------------------------------------------------------|---------------------|---------------------|---------------------|---------------------|
| Initial:                                                 | 0                   | 0                   | 0                   | 0                   |
| Transition 1: $(s = D, a = \text{east}, r = -1, s' = E)$ |                     | -0.5                |                     |                     |
| Transition 2: $(s = E, a = \text{east}, r = +2, s' = F)$ |                     |                     |                     | 1.0                 |
| Transition 3: $(s = E, a = \text{west}, r = 0, s' = D)$  |                     |                     |                     |                     |
| Transition 4: $(s = D, a = \text{east}, r = -1, s' = E)$ |                     | -0.25               |                     |                     |

The agent is still at the water park MDP, but now we're going to use function approximation to represent Q-values. Recall that a policy  $\pi$  is *greedy* with respect to a set of Q-values as long as  $\forall a, s \ Q(s, \pi(s)) \geq Q(s, a)$  (so ties may be broken in any way).



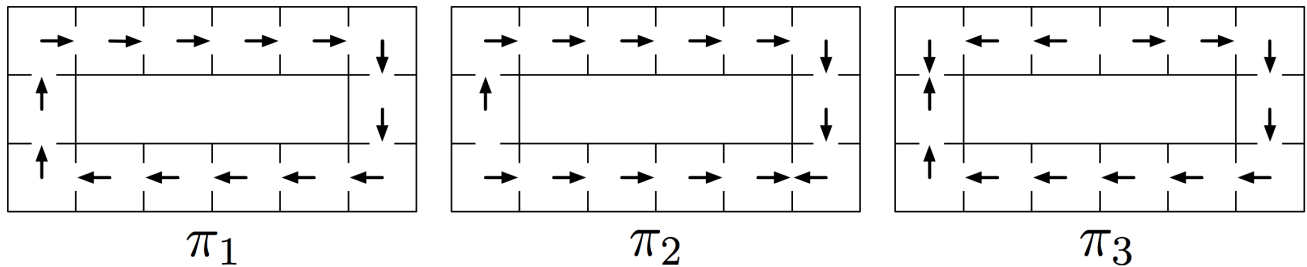
For the next subproblem, consider the following feature functions:

$$f(s, a) = \begin{cases} 1 & \text{if } a = \text{east,} \\ 0 & \text{otherwise.} \end{cases}$$

$$f'(s, a) = \begin{cases} 1 & \text{if } (a = \text{east}) \wedge \text{isSlide}(s), \\ 0 & \text{otherwise.} \end{cases}$$

(Note:  $\text{isSlide}(s)$  is true iff the state  $s$  is a slide square, i.e. either  $F$  or  $G$ .)

Also consider the following policies:



- (d) Which are greedy policies with respect to the Q-value approximation function obtained by running the single Q-update for the transition  $(s = F, a = \text{east}, r = +2, s' = G)$  while using the specified feature function? You may assume that all feature weights are zero before the update. Use discount  $\gamma = 1.0$  and learning rate  $\alpha = 1.0$ . Circle all that apply.

|      |                                                                                |                                                                                |                                                                                |
|------|--------------------------------------------------------------------------------|--------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| $f$  | $\pi_1$                                                                        | <span style="border: 1px solid black; padding: 2px;"><math>\pi_2</math></span> | $\pi_3$                                                                        |
| $f'$ | <span style="border: 1px solid black; padding: 2px;"><math>\pi_1</math></span> | <span style="border: 1px solid black; padding: 2px;"><math>\pi_2</math></span> | <span style="border: 1px solid black; padding: 2px;"><math>\pi_3</math></span> |

You see the sample  $(F, \text{east}, G, +2)$ . Use approximate Q-Learning to update the weights.

You should get that the new weights are both going to be positive since the sample reward was positive and the feature value was on for both  $f(F, \text{east})$  [since you took action east] and  $f'(F, \text{east})$  [since you took action east, and you were on the water slide].

Now, with your new weights, you need to see which greedy policy can be possible.

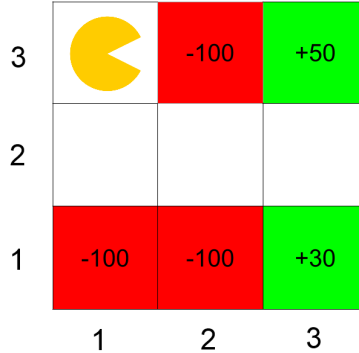
For  $f$ , going East is preferred if possible (when you calculate the Q-value, any Q-state with action east has a positive value, anything else has a value of 0. Hence, throw out  $\pi_1$  and  $\pi_3$ , since some arrows go west.

For  $f'$ , going East is preferred *if* you are on the slide (otherwise, everything else is just 0). All three policies contain the fact that you move east from F and G, so all policies are good.



## 6 . Deep inside $Q$ -learning

Consider the grid-world given below and an agent who is trying to learn the optimal policy. Rewards are only awarded for taking the *Exit* action from one of the shaded states. Taking this action moves the agent to the Done state, and the MDP terminates. Assume  $\gamma = 1$  and  $\alpha = 0.5$  for all calculations. All equations need to explicitly mention  $\gamma$  and  $\alpha$  if necessary.



- (a) 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           | Episode 4           | Episode 5           |
|---------------------|----------------------|---------------------|---------------------|---------------------|
| (1,3), S, (1,2), 0  | (1,3), S, (1,2), 0   | (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  | (1,2), E, (2,2), 0  | (1,2), E, (2,2), 0  |
| (2,2), E, (3,2), 0  | (2,2), S, (2,1), 0   | (2,2), E, (3,2), 0  | (2,2), E, (3,2), 0  | (2,2), E, (3,2), 0  |
| (3,2), N, (3,3), 0  | (2,1), Exit, D, -100 | (3,2), S, (3,1), 0  | (3,2), N, (3,3), 0  | (3,2), S, (3,1), 0  |
| (3,3), Exit, D, +50 |                      | (3,1), Exit, D, +30 | (3,3), Exit, D, +50 | (3,1), Exit, D, +30 |

Fill in the following Q-values obtained from direct evaluation from the samples:

$$Q((3,2), N) = \underline{50} \quad Q((3,2), S) = \underline{30} \quad Q((2,2), E) = \underline{40}$$

Direct evaluation is just averaging the discounted reward after performing action  $a$  in state  $s$ .

- (b) Q-learning is an online algorithm to learn optimal Q-values in an MDP with unknown rewards and transition function. The update equation is:

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a'))$$

where  $\gamma$  is the discount factor,  $\alpha$  is the learning rate and the sequence of observations are  $(\dots, s_t, a_t, s_{t+1}, r_t, \dots)$ . Given the episodes in (a), fill in the time at which the following Q values first become non-zero. Your answer should be of the form **(episode#, iter#)** where **iter#** is the Q-learning update iteration in that episode. If the specified Q value never becomes non-zero, write *never*.

$$Q((1,2), E) = \underline{\text{Never}} \quad Q((2,2), E) = \underline{(5,3)} \quad Q((3,2), S) = \underline{(5,4)}$$

This question was intended to demonstrate the way in which Q-values propagate through the state space. Q-learning is run in the following order - observations in ep 1 then observations in ep 2 and so on.

- (c) In Q-learning, we look at a window of  $(s_t, a_t, s_{t+1}, r_t)$  to update our Q-values. One can think of using an update rule that uses a larger window to update these values. Give an update rule for  $Q(s_t, a_t)$  given the window  $(s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2})$ .

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma r_{t+1} + \gamma^2 \max_{a'} Q(s_{t+2}, a'))$$

(Sample of the expected discounted reward using  $r_{t+1}$ )

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma((1 - \alpha)Q(s_{t+1}, a_{t+1}) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+2}, a'))))$$

(Nested Q-learning update)

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max((1 - \alpha)Q(s_{t+1}, a_{t+1}) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+2}, a')), \max_{a'} Q(s_{t+1}, a')))$$

(Max of normal Q-learning update and one step look-ahead update)

# CS188 Fall 2018 Section 6: Probability + Bayes' Nets

## 1 Probability

Use the probability table to calculate the following values:

| $X_1$ | $X_2$ | $X_3$ | $P(X_1, X_2, X_3)$ |
|-------|-------|-------|--------------------|
| 0     | 0     | 0     | 0.05               |
| 1     | 0     | 0     | 0.1                |
| 0     | 1     | 0     | 0.4                |
| 1     | 1     | 0     | 0.1                |
| 0     | 0     | 1     | 0.1                |
| 1     | 0     | 1     | 0.05               |
| 0     | 1     | 1     | 0.2                |
| 1     | 1     | 1     | 0.0                |

1.  $P(X_1 = 1, X_2 = 0) = 0.15$

2.  $P(X_3 = 0) = 0.65$

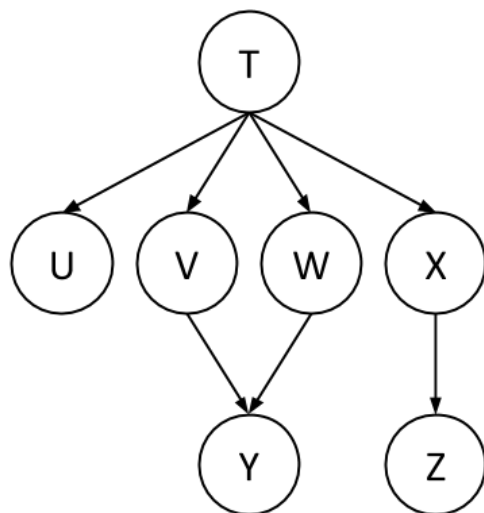
3.  $P(X_2 = 1|X_3 = 1) = 0.2/0.35$

4.  $P(X_1 = 0|X_2 = 1, X_3 = 1) = 1$

5.  $P(X_1 = 0, X_2 = 1|X_3 = 1) = 0.2/0.35$

## 2 D-Separation

Indicate whether each of the following conditional independence relationships is guaranteed to be true in the Bayes Net below. If the independence relationship does not hold, identify all active (d-connected) paths in the graph.



1.  $U \perp\!\!\!\perp X$

Not guaranteed, path U-T-X is active

2.  $U \perp\!\!\!\perp X|T$

Guaranteed

3.  $V \perp\!\!\!\perp W|Y$

Not guaranteed, paths V-T-W and V-Y-W are both active

4.  $V \perp\!\!\!\perp W|T$

Guaranteed

5.  $T \perp\!\!\!\perp Y|V$

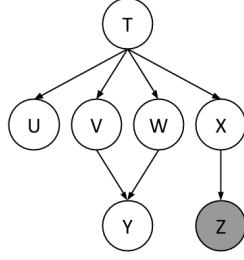
Not guaranteed, path T-W-Y is active

6.  $Y \perp\!\!\!\perp Z|W$

Not guaranteed, path Y-V-T-X-Z is active

7.  $Y \perp\!\!\!\perp Z|T$

Guaranteed, with T being observed, there are no active paths from Y to Z



### 3 Variable Elimination

Using the same Bayes Net (shown below), we want to compute  $P(Y \mid +z)$ . All variables have binary domains. Assume we run variable elimination to compute the answer to this query, with the following variable elimination ordering:  $X, T, U, V, W$ .

Complete the following description of the factors generated in this process:

After inserting evidence, we have the following factors to start out with:

$$P(T), P(U|T), P(V|T), P(W|T), P(X|T), P(Y|V, W), P(+z|X)$$

(a) When eliminating  $X$  we generate a new factor  $f_1$  as follows, which leaves us with the factors:

$$f_1(+z|T) = \sum_x P(x|T)P(+z|x) \quad P(T), P(U|T), P(V|T), P(W|T), P(Y|V, W), f_1(+z|T)$$

(b) When eliminating  $T$  we generate a new factor  $f_2$  as follows, which leaves us with the factors:

$$f_2(U, V, W, +z) = \sum_t P(t)P(U|t)P(V|t)P(W|t)f_1(+z|t) \quad P(Y|V, W), f_2(U, V, W, +z)$$

(c) When eliminating  $U$  we generate a new factor  $f_3$  as follows, which leaves us with the factors:

$$f_3(V, W, +z) = \sum_u f_2(u, V, W, +z) \quad P(Y|V, W), f_3(V, W, +z)$$

(d) When eliminating  $V$  we generate a new factor  $f_4$  as follows, which leaves us with the factors:

$$f_4(W, Y, +z) = \sum_v f_3(v, W, +z)P(Y|v, W) \quad f_4(W, Y, +z)$$

(e) When eliminating  $W$  we generate a new factor  $f_5$  as follows, which leaves us with the factors:

$$f_5(Y, +z) = \sum_w f_4(w, Y, +z) \quad f_5(Y, +z)$$

(f) How would you obtain  $P(Y \mid +z)$  from the factors left above:

Simply renormalize  $f_5(Y, +z)$  to obtain  $P(Y \mid +z)$ . Concretely,

$$P(y \mid +z) = \frac{f_5(y, +z)}{\sum_{y'} f_5(y', +z)}$$

(g) What is the size of the largest factor that gets generated during the above process?

$f_2(U, V, W, +z)$ . This contains 3 unconditioned variables, so it will have  $2^3 = 8$  probability entries ( $U, V, W$  are binary variables, and we only need to store the probability for  $+z$  for each possible setting of these variables).

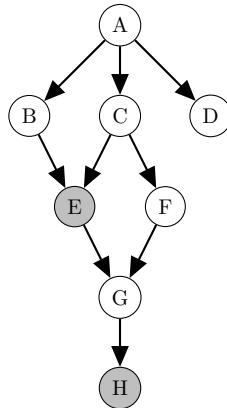
(m) Does there exist a better elimination ordering (one which generates smaller largest factors)?

Yes. One such ordering is  $X, U, T, V, W$ . All factors generated with this ordering contain at most 2 unconditioned variables, so the tables will have at most  $2^2 = 4$  probability entries (as all variables are binary).

# CS188 Fall 2018 Section 7: Bayes Nets and Decision Nets

## 1 Bayes' Nets: Inference

Assume we are given the following Bayes' net, and would like to perform inference to obtain  $P(B, D \mid E = e, H = h)$ .



1. What is the number of rows in the largest factor generated by *inference by enumeration*, for this query  $P(B, D \mid E = e, H = h)$ ? Assume all the variables are binary.

☐  $2^2$       ☐  $2^3$       ☒  $2^6$       ☐  $2^8$   
☐ None of the above.

Since the inference by enumeration first joins all the factors in the Bayes' net, that factor will contain six (unobserved) variables. The question assumes all variables are binary, so the answer is  $2^6$ .

2. Mark all of the following variable elimination orderings that are optimal for calculating the answer for the query  $P(B, D \mid E = e, H = h)$ . Optimality is measured by the sum of the sizes of the factors that are generated. Assume all the variables are binary.

☐  $C, A, F, G$       ☐  $F, G, C, A$       ☐  $A, C, F, G$       ☒  $G, F, C, A$   
☐ None of the above.

The sum of the sizes of factors that are generated for the variable elimination ordering  $G, F, C, A$  is  $2^1 + 2^1 + 2^2 + 2^2$  rows, which is smaller than for any of the other variable elimination orderings. The ordering  $F, G, C, A$  is close but the sum of the sizes of factors is slightly bigger, with  $2^2 + 2^1 + 2^2 + 2^2$  rows.

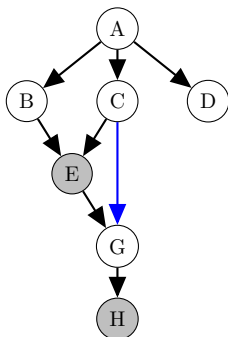
3. Suppose we decide to perform variable elimination to calculate the query  $P(B, D \mid E = e, H = h)$ , and choose to eliminate  $F$  first.

- (a) When  $F$  is eliminated, what intermediate factor is generated and how is it calculated? Make sure it is clear which variable(s) come before the conditioning bar and which variable(s) come after.

$$f_1(\underline{G \mid C, e}) = \sum_f \underline{P(f \mid C)P(G \mid f, e)}$$

This follows from the first step of variable elimination, which is to join all factors containing  $F$ , and then marginalize over  $F$  to obtain the intermediate factor  $f_1$ .

- (b) Now consider the set of distributions that can be represented by the remaining factors *after F is eliminated*. Draw the minimal number of directed edges on the following Bayes' Net structure, so that it can represent any distribution in this set. If no additional directed edges are needed, please fill in that option below.

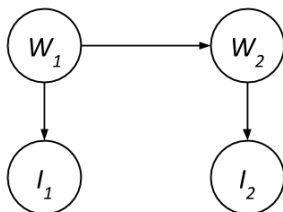


☐ No additional directed edges needed

An additional edge from C to G is necessary, because the intermediate factor is of the form  $f_1(G|C)$ . Without this edge from C to G, the Bayes' net would not be able to express the dependence of G on C. (Note that adding an edge from G to C is not allowed, since that would introduce a cycle.)

## 2 Sampling and Dynamic Bayes Nets

We would like to analyze people's ice cream eating habits on sunny and rainy days. Suppose we consider the weather, along with a person's ice-cream eating, over the span of two days. We'll have four random variables:  $W_1$  and  $W_2$  stand for the weather on days 1 and 2, which can either be rainy R or sunny S, and the variables  $I_1$  and  $I_2$  represent whether or not the person ate ice cream on days 1 and 2, and take values T (for truly eating ice cream) or F. We can model this as the following Bayes Net with these probabilities.



| $W_1$ | $P(W_1)$ |
|-------|----------|
| S     | 0.6      |
| R     | 0.4      |

| $W_1$ | $W_2$ | $P(W_2 W_1)$ |
|-------|-------|--------------|
| S     | S     | 0.7          |
| S     | R     | 0.3          |
| R     | S     | 0.5          |
| R     | R     | 0.5          |

| $W$ | $I$ | $P(I W)$ |
|-----|-----|----------|
| S   | T   | 0.9      |
| S   | F   | 0.1      |
| R   | T   | 0.2      |
| R   | F   | 0.8      |

Suppose we produce the following samples of  $(W_1, I_1, W_2, I_2)$  from the ice-cream model:

~~R, F, R, F~~   ~~R, F, R, F~~   ~~S, F, S, T~~   ~~S, T, S, T~~   S, T, R, F  
~~R, F, R, T~~   ~~S, T, S, T~~   ~~S, T, S, T~~   S, T, R, F   ~~R, F, S, T~~

- What is  $\hat{P}(W_2 = R)$ , the probability that sampling assigns to the event  $W_2 = R$ ?  
 Number of samples in which  $W_2 = R$ : 5. Total number of samples: 10. Answer  $5/10 = 0.5$ .
- Cross off samples above which are rejected by rejection sampling if we're computing  $P(W_2|I_1 = T, I_2 = F)$ .

Rejection sampling seems to be wasting a lot of effort, so we decide to switch to likelihood weighting. Assume we generate the following six samples given the evidence  $I_1 = T$  and  $I_2 = F$ :

$$(W_1, I_1, W_2, I_2) = \left\{ (S, T, R, F), (R, T, R, F), (S, T, R, F), (S, T, S, F), (S, T, S, F), (R, T, S, F) \right\}$$



3. What is the weight of the first sample (S, T, R, F) above?

The weight given to a sample in likelihood weighting is

$$\prod_{\text{Evidence variables } e} \Pr(e|\text{Parents}(e)).$$

In this case, the evidence is  $I_1 = \text{T}, I_2 = \text{F}$ . The weight of the first sample is therefore

$$w = \Pr(I_1 = \text{T}|W_1 = \text{S}) \cdot \Pr(I_2 = \text{F}|W_2 = \text{R}) = 0.9 \cdot 0.8 = 0.72$$

4. Use likelihood weighting to estimate  $P(W_2|I_1 = \text{T}, I_2 = \text{F})$ .

The sample weights are given by

| $(W_1, I_1, W_2, I_2)$ | $w$  | $(W_1, I_1, W_2, I_2)$ | $w$  |
|------------------------|------|------------------------|------|
| S, T, R, F             | 0.72 | S, T, S, F             | 0.09 |
| R, T, R, F             | 0.16 | S, T, S, F             | 0.09 |
| S, T, R, F             | 0.72 | R, T, S, F             | 0.02 |

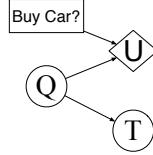
To compute the probabilities, we thus normalize the weights and find

$$\hat{P}(W_2 = \text{R}|I_1 = \text{T}, I_2 = \text{F}) = \frac{0.72 + 0.16 + 0.72}{0.72 + 0.16 + 0.72 + 0.09 + 0.09 + 0.02} = 0.889$$

$$\hat{P}(W_2 = \text{S}|I_1 = \text{T}, I_2 = \text{F}) = 1 - 0.889 = 0.111.$$

### 3 Decision Networks and VPI

A used car buyer can decide to carry out various tests with various costs (e.g., kick the tires, take the car to a qualified mechanic) and then, depending on the outcome of the tests, decide which car to buy. We will assume that the buyer is deciding whether to buy car  $c$  and that there is time to carry out at most one test which costs \$50 and which can help to figure out the quality of the car. A car can be in good shape (of good quality  $Q = +q$ ) or in bad shape (of bad quality  $Q = \neg q$ ), and the test might help to indicate what shape the car is in. There are only two outcomes for the test  $T$ : pass ( $T = \text{pass}$ ) or fail ( $T = \text{fail}$ ). Car  $c$  costs \$1,500, and its market value is \$2,000 if it is in good shape; if not, \$700 in repairs will be needed to make it in good shape. The buyers estimate is that  $c$  has 70% chance of being in good shape. The Decision Network is shown below.



1. Calculate the expected net gain from buying car  $c$ , given no test.

$$\begin{aligned}
 EU(\text{buy}) &= P(Q = +q) \cdot U(+q, \text{buy}) + P(Q = \neg q) \cdot U(\neg q, \text{buy}) \\
 &= .7 \cdot 500 + 0.3 \cdot -200 = 290
 \end{aligned}$$

2. Tests can be described by the probability that the car will pass or fail the test given that the car is in good or bad shape. We have the following information:

$$P(T = \text{pass} | Q = +q) = 0.9$$

$$P(T = \text{pass} | Q = \neg q) = 0.2$$

Calculate the probability that the car will pass (or fail) its test, and then the probability that it is in good (or bad) shape given each possible test outcome.

$$\begin{aligned}
 P(T = \text{pass}) &= \sum_q P(T = \text{pass}, Q = q) \\
 &= P(T = \text{pass} | Q = +q)P(Q = +q) + P(T = \text{pass} | Q = \neg q)P(Q = \neg q) \\
 &= 0.69 \\
 P(T = \text{fail}) &= 0.31 \\
 P(Q = +q | T = \text{pass}) &= \frac{P(T = \text{pass} | Q = +q)P(Q = +q)}{P(T = \text{pass})} \\
 &= \frac{0.9 \cdot 0.7}{0.69} = \frac{21}{23} \approx 0.91 \\
 P(Q = +q | T = \text{fail}) &= \frac{P(T = \text{fail} | Q = +q)P(Q = +q)}{P(T = \text{fail})} \\
 &= \frac{0.1 \cdot 0.7}{0.31} = \frac{7}{31} \approx 0.22
 \end{aligned}$$

3. Calculate the optimal decisions given either a pass or a fail, and their expected utilities.

$$\begin{aligned}
 EU(\text{buy} | T = \text{pass}) &= P(Q = +q | T = \text{pass})U(+q, \text{buy}) + P(Q = \neg q | T = \text{pass})U(\neg q, \text{buy}) \\
 &\approx 0.91 \cdot 500 + 0.09 \cdot (-200) \approx 437
 \end{aligned}$$

$$\begin{aligned}
 EU(\text{buy} | T = \text{fail}) &= P(Q = +q | T = \text{fail})U(+q, \text{buy}) + P(Q = \neg q | T = \text{fail})U(\neg q, \text{buy}) \\
 &\approx 0.22 \cdot 500 + 0.78 \cdot (-200) = -46
 \end{aligned}$$

$$EU(\neg \text{buy} | T = \text{pass}) = 0$$

$$EU(\neg \text{buy} | T = \text{fail}) = 0$$

Therefore:  $MEU(T = \text{pass}) = 437$  (with buy) and  $MEU(T = \text{fail}) = 0$  (using  $\neg\text{buy}$ )

4. Calculate the value of (perfect) information of the test. Should the buyer pay for a test?

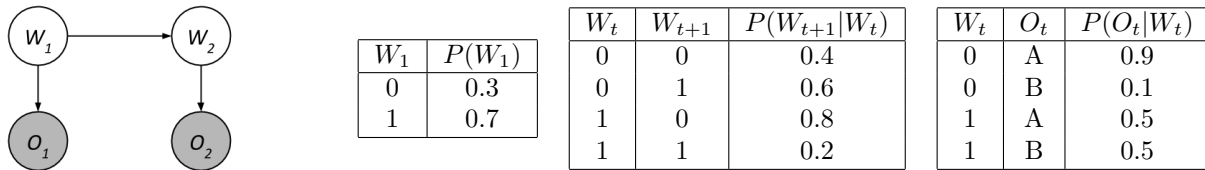
$$\begin{aligned} VPI(T) &= \left( \sum_t P(T = t) MEU(T = t) \right) - MEU(\phi) \\ &= 0.69 \cdot 437 + 0.31 \cdot 0 - 290 \approx 11.53 \end{aligned}$$

You shouldn't pay for it, since the cost is \$50.

# CS188 Fall 2018 Section 8: HMMs + Particle Filtering

## 1 HMMs

Consider the following Hidden Markov Model.



Suppose that we observe  $O_1 = A$  and  $O_2 = B$ .

Using the forward algorithm, compute the probability distribution  $P(W_2|O_1 = A, O_2 = B)$  one step at a time.

1. Compute  $P(W_1, O_1 = A)$ .

$$P(W_1, O_1 = A) = P(W_1)P(O_1 = A|W_1)$$

$$P(W_1 = 0, O_1 = A) = (0.3)(0.9) = 0.27$$

$$P(W_1 = 1, O_1 = A) = (0.7)(0.5) = 0.35$$

2. Using the previous calculation, compute  $P(W_2, O_1 = A)$ .

$$P(W_2, O_1 = A) = \sum_{x_1} P(x_1, O_1 = A)P(W_2|x_1)$$

$$P(W_2 = 0, O_1 = A) = (0.27)(0.4) + (0.35)(0.8) = 0.388$$

$$P(W_2 = 1, O_1 = A) = (0.27)(0.6) + (0.35)(0.2) = 0.232$$

3. Using the previous calculation, compute  $P(W_2, O_1 = A, O_2 = B)$ .

$$P(W_2, O_1 = A, O_2 = B) = P(W_2, O_1 = A)P(O_2 = B|W_2)$$

$$P(W_2 = 0, O_1 = A, O_2 = B) = (0.388)(0.1) = 0.0388$$

$$P(W_2 = 1, O_1 = A, O_2 = B) = (0.232)(0.5) = 0.116$$

4. Finally, compute  $P(W_2|O_1 = A, O_2 = B)$ .

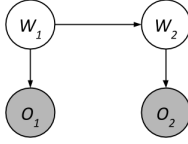
Renormalizing the distribution above, we have

$$P(W_2 = 0|O_1 = A, O_2 = B) = 0.0388/(0.0388 + 0.116) \approx 0.25$$

$$P(W_2 = 1|O_1 = A, O_2 = B) = 0.116/(0.0388 + 0.116) \approx 0.75$$

## 2 Particle Filtering

Let's use Particle Filtering to estimate the distribution of  $P(W_2|O_1 = A, O_2 = B)$ . Here's the HMM again:



| $W_1$ | $P(W_1)$ |
|-------|----------|
| 0     | 0.3      |
| 1     | 0.7      |

| $W_t$ | $W_{t+1}$ | $P(W_{t+1} W_t)$ |
|-------|-----------|------------------|
| 0     | 0         | 0.4              |
| 0     | 1         | 0.6              |
| 1     | 0         | 0.8              |
| 1     | 1         | 0.2              |

| $W_t$ | $O_t$ | $P(O_t W_t)$ |
|-------|-------|--------------|
| 0     | A     | 0.9          |
| 0     | B     | 0.1          |
| 1     | A     | 0.5          |
| 1     | B     | 0.5          |

We start with two particles representing our distribution for  $W_1$ .

$P_1 : W_1 = 0$

$P_2 : W_1 = 1$

Use the following random numbers to run particle filtering:

[0.22, 0.05, 0.33, 0.20, 0.84, 0.54, 0.79, 0.66, 0.14, 0.96]

1. **Observe:** Compute the weight of the two particles after evidence  $O_1 = A$ .

$$w(P_1) = P(O_t = A|W_t = 0) = 0.9$$

$$w(P_2) = P(O_t = A|W_t = 1) = 0.5$$

2. **Resample:** Using the random numbers, resample  $P_1$  and  $P_2$  based on the weights.

We now sample from the weighted distribution we found above. After normalizing the weights, we find that  $P_1$  maps to range [0, 0.643), and  $P_2$  maps to range [0.643, 1). Using the first two random samples, we find:

$$P_1 = \text{sample}(\text{weights}, 0.22) = 0$$

$$P_2 = \text{sample}(\text{weights}, 0.05) = 0$$

3. **Elapse Time:** Now let's compute the elapse time particle update. Sample  $P_1$  and  $P_2$  from applying the time update.

$$P_1 = \text{sample}(P(W_{t+1}|W_t = 0), 0.33) = 0$$

$$P_2 = \text{sample}(P(W_{t+1}|W_t = 0), 0.20) = 0$$

4. **Observe:** Compute the weight of the two particles after evidence  $O_2 = B$ .

$$w(P_1) = P(O_t = B|W_t = 0) = 0.1$$

$$w(P_2) = P(O_t = B|W_t = 0) = 0.1$$

5. **Resample:** Using the random numbers, resample  $P_1$  and  $P_2$  based on the weights.

Because both of our particles have  $X = 0$ , resampling will still leave us with two particles with  $X = 0$ .

$$P_1 = 0$$

$$P_2 = 0$$

6. What is our estimated distribution for  $P(W_2|O_1 = A, O_2 = B)$ ?

$$P(W_2 = 0|O_1 = A, O_2 = B) = 2/2 = 1$$

$$P(W_2 = 1|O_1 = A, O_2 = B) = 0/2 = 0$$

### 3 HMMs (Optional)

Consider a process where there are transitions among a finite set of states  $s_1, \dots, s_k$  over time steps  $i = 1, \dots, N$ . Let the random variables  $X_1, \dots, X_N$  represent the state of the system at each time step and be generated as follows:

- Sample the initial state  $s$  from an initial distribution  $P_1(X_1)$ , and set  $i = 1$
- Repeat the following:
  1. Sample a duration  $d$  from a duration distribution  $P_D$  over the integers  $\{1, \dots, M\}$ , where  $M$  is the maximum duration.
  2. Remain in the current state  $s$  for the next  $d$  time steps, i.e., set
 
$$x_i = x_{i+1} = \dots = x_{i+d-1} = s \quad (1)$$
  3. Sample a successor state  $s'$  from a transition distribution  $P_T(X_t|X_{t-1} = s)$  over the other states  $s' \neq s$  (so there are no self transitions)
  4. Assign  $i = i + d$  and  $s = s'$ .

This process continues indefinitely, but we only observe the first  $N$  time steps.

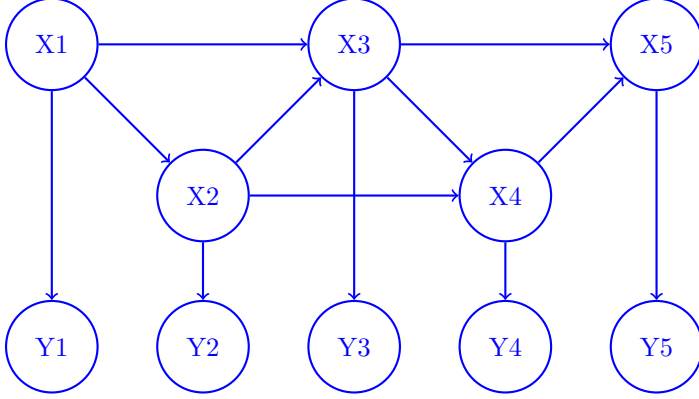
(a) Assuming that all three states  $s_1, s_2, s_3$  are different, what is the probability of the sample sequence  $s_1, s_1, s_2, s_2, s_2, s_3, s_3$ ? Write an algebraic expression. Assume  $M \geq 3$ .

$$p_1(s_1)p_D(2)p_T(s_2|s_1)p_D(3)p(s_3|s_2)(1 - p_D(1)) \quad (2)$$

At each time step  $i$  we observe a noisy version of the state  $X_i$  that we denote  $Y_i$  and is produced via a conditional distribution  $P_E(Y_i|X_i)$ .

(b) Only in this subquestion assume that  $N > M$ . Let  $X_1, \dots, X_N$  and  $Y_1, \dots, Y_N$  random variables defined as above. What is the maximum index  $i \leq N - 1$  so that  $X_1 \perp\!\!\!\perp X_N | X_i, X_{i+1}, \dots, X_{N-1}$  is guaranteed?  
 $i = N - M$

(c) Only in this subquestion, assume the max duration  $M = 2$ , and  $P_D$  uniform over  $\{1, 2\}$  and each  $x_i$  is in an alphabet  $\{a, b\}$ . For  $(X_1, X_2, X_3, X_4, X_5, Y_1, Y_2, Y_3, Y_4, Y_5)$  draw a Bayes Net over these 10 random variables with the property that removing any of the edges would yield a Bayes net inconsistent with the given distribution.



(d) In this part we will explore how to write the described process as an HMM with an extended state space. Write the states  $z = (s, t)$  where  $s$  is a state of the original system and  $t$  represents the time elapsed in that state. For example, the state sequence  $s_1, s_1, s_1, s_2, s_3, s_3$  would be represented as  $(s_1, 1), (s_1, 2), (s_1, 3), (s_2, 1), (s_3, 1), (s_3, 2)$ .

Answer all of the following in terms of the parameters  $P_1(X_1), P_D(d), P_T(X_{j+1}|X_j), P_E(Y_i|X_i), k$  (total number of possible states),  $N$  and  $M$  (max duration).

- What is  $P(Z_1)$ ?

$$P(x_1, t) = \begin{cases} P_1(x_1) & \text{if } t = 1 \\ 0 & \text{o.w.} \end{cases} \quad (3)$$

- What is  $P(Z_{i+1}|Z_i)$ ? Hint: You will need to break this into cases where the transition function will behave differently.

$$P(X_{i+1}, t_{i+1}|X_i, t_i) = \begin{cases} P_D(d \geq t_i + 1 | d \geq t_i) & \text{when } X_{i+1} = X_i \text{ and } t_{i+1} = t_i + 1 \text{ and } t_{i+1} \leq M \\ P_T(X_{i+1}|X_i)P_D(d = t_i | d \geq t_i) & \text{when } X_{i+1} \neq X_i \text{ and } t_{i+1} = 1 \\ 0 & \text{o.w.} \end{cases}$$

Where  $P_D(d \geq t_i + 1 | d \geq t_i) = P_D(d \geq t_i + 1) / P_D(d \geq t_i)$ .

Being in  $X_i, t_i$ , we know that  $d$  was drawn  $d \geq t_i$ . Conditioning on this fact, we have two choices, if  $d > t_i$  then the next state is  $X_{i+1} = X_i$ , and if  $d = t_i$  then  $X_{i+1} \neq X_i$  drawn from the transition distribution and  $t_{i+1} = 1$ .  
(4)

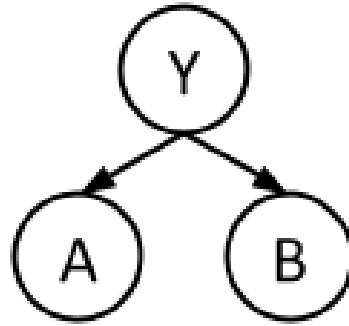
- What is  $P(Y_i|Z_i)$ ?  
 $p(Y_i|X_i, t_i) = P_E(Y_i|X_i)$

# CS188 Fall 2018 Section 9: Machine Learning

## 1 Naive Bayes

In this question, we will train a Naive Bayes classifier to predict class labels  $Y$  as a function of input features  $A$  and  $B$ .  $Y$ ,  $A$ , and  $B$  are all binary variables, with domains 0 and 1. We are given 10 training points from which we will estimate our distribution.

|     |   |   |   |   |   |   |   |   |   |   |
|-----|---|---|---|---|---|---|---|---|---|---|
| $A$ | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| $B$ | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| $Y$ | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |



1. What are the maximum likelihood estimates for the tables  $P(Y)$ ,  $P(A|Y)$ , and  $P(B|Y)$ ?

| $Y$ | $P(Y)$ | $A$ | $Y$ | $P(A Y)$ | $B$ | $Y$ | $P(B Y)$ |
|-----|--------|-----|-----|----------|-----|-----|----------|
| 0   | $3/5$  | 0   | 0   | $1/6$    | 0   | 0   | $1/3$    |
| 1   | $2/5$  | 1   | 0   | $5/6$    | 1   | 0   | $2/3$    |
|     |        | 0   | 1   | $1/4$    | 0   | 1   | $1/4$    |
|     |        | 1   | 1   | $3/4$    | 1   | 1   | $3/4$    |

2. Consider a new data point ( $A = 1$ ,  $B = 1$ ). What label would this classifier assign to this sample?

$$P(Y = 0, A = 1, B = 1) = P(Y = 0)P(A = 1|Y = 0)P(B = 1|Y = 0) \quad (1)$$

$$= (3/5)(5/6)(2/3) \quad (2)$$

$$= 1/3 \quad (3)$$

$$P(Y = 1, A = 1, B = 1) = P(Y = 1)P(A = 1|Y = 1)P(B = 1|Y = 1) \quad (4)$$

$$= (2/5)(3/4)(3/4) \quad (5)$$

$$= 9/40 \quad (6)$$

$$(7)$$

Our classifier will predict label 0.

3. Let's use Laplace Smoothing to smooth out our distribution. Compute the new distribution for  $P(A|Y)$  given Laplace Smoothing with  $k = 2$ .

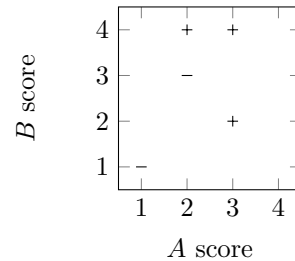
| $A$ | $Y$ | $P(A Y)$ |
|-----|-----|----------|
| 0   | 0   | $3/10$   |
| 1   | 0   | $7/10$   |
| 0   | 1   | $3/8$    |
| 1   | 1   | $5/8$    |



## 2 Perceptron

You want to predict if movies will be profitable based on their screenplays. You hire two critics A and B to read a script you have and rate it on a scale of 1 to 4. The critics are not perfect; here are five data points including the critics' scores and the performance of the movie:

| # | Movie Name   | A | B | Profit? |
|---|--------------|---|---|---------|
| 1 | Pellet Power | 1 | 1 | -       |
| 2 | Ghosts!      | 3 | 2 | +       |
| 3 | Pac is Bac   | 2 | 4 | +       |
| 4 | Not a Pizza  | 3 | 4 | +       |
| 5 | Endless Maze | 2 | 3 | -       |



1. First, you would like to examine the linear separability of the data. Plot the data on the 2D plane above; label profitable movies with + and non-profitable movies with - and determine if the data are linearly separable. **The data are linearly separable.**
2. Now you decide to use a perceptron to classify your data. Suppose you directly use the scores given above as features, together with a bias feature. That is  $f_0 = 1$ ,  $f_1 = \text{score given by A}$  and  $f_2 = \text{score given by B}$ .

Run one pass through the data with the perceptron algorithm, filling out the table below. Go through the data points in order, e.g. using data point #1 at step 1.

| step | Weights      | Score                                     | Correct? |
|------|--------------|-------------------------------------------|----------|
| 1    | $[-1, 0, 0]$ | $-1 \cdot 1 + 0 \cdot 1 + 0 \cdot 1 = -1$ | yes      |
| 2    | $[-1, 0, 0]$ | $-1 \cdot 1 + 0 \cdot 3 + 0 \cdot 2 = -1$ | no       |
| 3    | $[0, 3, 2]$  | $0 \cdot 1 + 3 \cdot 2 + 2 \cdot 4 = 14$  | yes      |
| 4    | $[0, 3, 2]$  | $0 \cdot 1 + 3 \cdot 3 + 2 \cdot 4 = 17$  | yes      |
| 5    | $[0, 3, 2]$  | $0 \cdot 1 + 3 \cdot 2 + 2 \cdot 3 = 12$  | no       |

Final weights:  $[-1, 1, -1]$

3. Have weights been learned that separate the data? **With the current weights, points will be classified as positive if  $-1 \cdot 1 + 1 \cdot A + -1 \cdot B \geq 0$ , or  $A - B \geq 1$ . So we will have incorrect predictions for data points 3:**

$$-1 \cdot 1 + 1 \cdot 2 + -1 \cdot 4 = -3 < 0$$

and 4:

$$-1 \cdot 1 + 1 \cdot 3 + -1 \cdot 4 = -2 < 0$$

Note that although point 2 has  $w \cdot f = 0$ , it will be classified as positive (since we classify as positive if  $w \cdot f \geq 0$ ).

4. 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. Circle the scenarios for which a perceptron using the features above can indeed perfectly classify movies which are profitable according to the given rules:
  - (a) Your reviewers are awesome: if the total of their scores is more than 8, then the movie will definitely be profitable, and otherwise it won't be. **Can classify (consider weights  $[-8, 1, 1]$ )**
  - (b) Your reviewers are art critics. Your movie will be profitable if and only if each reviewer gives either a score of 2 or a score of 3. **Cannot classify**
  - (c) Your reviewers have weird but different tastes. Your movie will be profitable if and only if both reviewers agree. **Cannot classify**

### 3 Maximum Likelihood

A Geometric distribution is a probability distribution of the number  $X$  of Bernoulli trials needed to get one success. It depends on a parameter  $p$ , which is the probability of success for each individual Bernoulli trial. Think of it as the number of times you must flip a coin before flipping heads. The probability is given as follows:

$$P(X = k) = p(1 - p)^{k-1} \quad (8)$$

$p$  is the parameter we wish to estimate.

We observe the following samples from a Geometric distribution:  $x_1 = 5, x_2 = 8, x_3 = 3, x_4 = 5, x_5 = 7$ . What is the maximum likelihood estimate for  $p$ ?

$$L(p) = P(X = x_1)P(X = x_2)P(X = x_3)P(X = x_4)P(X = x_5) \quad (9)$$

$$= P(X = 5)P(X = 8)P(X = 3)P(X = 5)P(X = 7) \quad (10)$$

$$= p^5(1 - p)^{23} \quad (11)$$

$$\log(L(p)) = 5 \log(p) + 23 \log(1 - p) \quad (12)$$

$$(13)$$

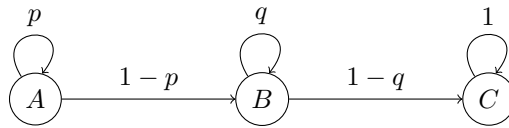
We must maximize the log-likelihood of  $p$ , so we will take the derivative, and set it to 0.

$$0 = \frac{5}{p} - \frac{23}{1 - p} \quad (14)$$

$$p = 5/28 \quad (15)$$

# 1 . A Not So Random Walk

Pacman is trying to predict the position of a ghost, which he knows has the following transition graph:



Here,  $0 < p < 1$  and  $0 < q < 1$  are arbitrary probabilities. It is known that the ghost always starts in state  $A$ . For this problem, we consider time to begin at 0. For example, at time 0, the ghost is in  $A$  with probability 1, and at time 1, the ghost is in  $A$  with probability  $p$  or in  $B$  with probability  $1 - p$ .

In all of the following questions, you may assume that  $n$  is large enough so that the given event occurs with non-zero probability.

- (i) Suppose  $p \neq q$ . What is the probability that the ghost is in  $A$  at time  $n$ ?

For the ghost to be in  $A$  at time  $n$ , it must have stayed in  $A$  for  $n$  steps, which occurs with probability

$$p^n.$$

- (ii) Suppose  $p \neq q$ . What is the probability that the ghost first reaches  $B$  at time  $n$ ?

For the ghost to first reach  $B$  at time  $n$ , it must have stayed in  $A$  for  $n - 1$  steps, then transitioned to  $B$ . This occurs with probability

$$f_{(i)}(n - 1) \cdot (1 - p) = p^{n-1}(1 - p).$$

- (iii) Suppose  $p \neq q$ . What is the probability that the ghost is in  $B$  at time  $n$ ?

For the ghost to be in  $B$  at time  $n$ , it must have first reached  $B$  at time  $i$  for some  $1 \leq i \leq n$ , then stayed there for  $n - i$  steps. Summing over all values of  $i$  gives

$$\sum_{i=1}^n f_{(ii)}(i) \cdot q^{n-i} = \sum_{i=1}^n p^{i-1}(1 - p)q^{n-i} = \frac{(1 - p)q^n}{p} \sum_{i=1}^n \left(\frac{p}{q}\right)^i = \frac{(1 - p)q^n}{p} \cdot \frac{p}{q} \cdot \frac{1 - \left(\frac{p}{q}\right)^n}{1 - \frac{p}{q}} = (1 - p) \frac{q^n - p^n}{q - p}.$$

- (iv) Suppose  $p \neq q$ . What is the probability that the ghost first reaches  $C$  at time  $n$ ?

For the ghost to first reach  $C$  at time  $n$ , it must have been in  $B$  at time  $n - 1$ , then transitioned to  $C$ . This occurs with probability

$$f_{(iii)}(n - 1) \cdot (1 - q) = (1 - p) \frac{q^{n-1} - p^{n-1}}{q - p} (1 - q).$$

- (v) Suppose  $p \neq q$ . What is the probability that the ghost is in  $C$  at time  $n$ ?

For the ghost to be in  $C$  at time  $n$ , it must not be in  $A$  or  $B$  at time  $n$ . This occurs with probability

$$1 - f_{(i)}(n) - f_{(iii)}(n) = 1 - p^n - (1 - p) \frac{q^n - p^n}{q - p}.$$

Alternatively, for the ghost to be in  $C$  at time  $n$ , it must have first reached  $C$  at time  $i$  for some  $2 \leq i \leq n$ , then stayed there for  $n-i$  steps. Note that we can equivalently range over  $1 \leq i \leq n$  for computational convenience, since  $f_{(\text{iv})}(1) = 0$ . Summing over all values of  $i$  gives

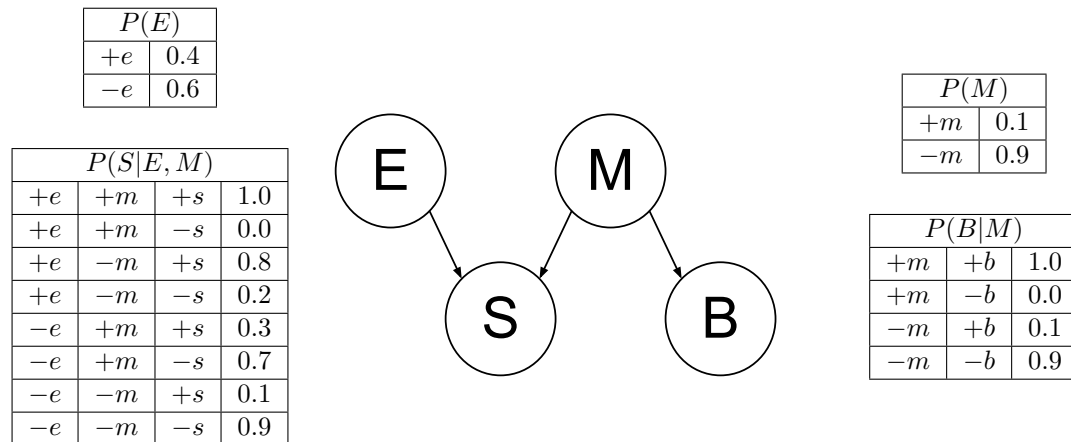
$$\begin{aligned} \sum_{i=2}^n f_{(\text{iv})}(i) \cdot 1^{n-i} &= \sum_{i=1}^n f_{(\text{iv})}(i) \cdot 1^{n-i} = \sum_{i=1}^n (1-p) \frac{q^{i-1} - p^{i-1}}{q-p} (1-q) \\ &= \frac{(1-p)(1-q)}{q-p} \left( \frac{1-q^n}{1-q} - \frac{1-p^n}{1-p} \right) = \frac{(1-p)(1-q^n) - (1-q)(1-p^n)}{q-p}, \end{aligned}$$

which is equivalent to the previous expression.

2 . December 21, 2012

A smell of sulphur ( $S$ ) can be caused either by rotten eggs ( $E$ ) or as a sign of the doom brought by the Mayan Apocalypse ( $M$ ). The Mayan Apocalypse also causes the oceans to boil ( $B$ ). The Bayesian network and corresponding conditional probability tables for this situation are shown below. For each part, you should give either a numerical answer (e.g. 0.81) or an arithmetic expression in terms of numbers from the tables below (e.g.  $0.9 \cdot 0.9$ ).

Note: be careful of doing unnecessary computation here.



(a) Compute the following entry from the joint distribution:

$$P(-e, -s, -m, -b) = P(-e)P(-m)P(-s|-e, -m)P(-b|-m) = (0.6)(0.9)(0.9)(0.9) = 0.4374$$

by expanding the joint according to the chain rule of conditional probability.

(b) What is the probability that the oceans boil?

$$P(+b) = P(+b|+m)P(+m) + P(+b|-m)P(-m) = (1.0)(0.1) + (0.1)(0.9) = 0.19$$

by marginalizing out  $m$  according to the law of total probability.

(c) What is the probability that the Mayan Apocalypse is occurring, given that the oceans are boiling?

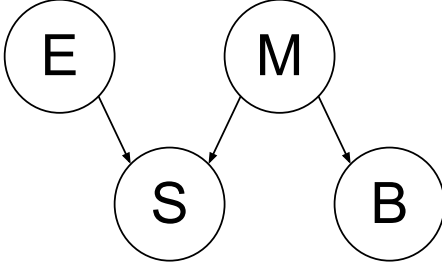
$$P(+m|+b) = \frac{P(+b|+m)P(+m)}{P(+b)} = \frac{(1.0)(0.1)}{0.19} \approx .5263$$

by the definition of conditional probability.

The figures and table below are identical to the ones on the previous page and are repeated here for your convenience.

|        |     |
|--------|-----|
| $P(E)$ |     |
| $+e$   | 0.4 |
| $-e$   | 0.6 |

|             |      |      |     |
|-------------|------|------|-----|
| $P(S E, M)$ |      |      |     |
| $+e$        | $+m$ | $+s$ | 1.0 |
| $+e$        | $+m$ | $-s$ | 0.0 |
| $+e$        | $-m$ | $+s$ | 0.8 |
| $+e$        | $-m$ | $-s$ | 0.2 |
| $-e$        | $+m$ | $+s$ | 0.3 |
| $-e$        | $+m$ | $-s$ | 0.7 |
| $-e$        | $-m$ | $+s$ | 0.1 |
| $-e$        | $-m$ | $-s$ | 0.9 |



|        |     |
|--------|-----|
| $P(M)$ |     |
| $+m$   | 0.1 |
| $-m$   | 0.9 |

|          |      |     |
|----------|------|-----|
| $P(B M)$ |      |     |
| $+m$     | $+b$ | 1.0 |
| $+m$     | $-b$ | 0.0 |
| $-m$     | $+b$ | 0.1 |
| $-m$     | $-b$ | 0.9 |

- (d) What is the probability that the Mayan Apocalypse is occurring, given that there is a smell of sulphur, the oceans are boiling, and there are rotten eggs?

$$P(+m | +s, +b, +e) =$$

$$\begin{aligned}
 \frac{P(+m, +s, +b, +e)}{\sum_m P(m, +s, +b, +e)} &= \frac{P(+e)P(+m)P(+s | +e, +m)P(+b | +m)}{\sum_m P(+e)P(m)P(+s | +e, m)P(+b | m)} \\
 &= \frac{(0.4)(0.1)(1.0)(1.0)}{(0.4)(0.1)(1.0)(1.0) + (0.4)(0.9)(0.8)(0.1)} \\
 &= \frac{0.04}{0.04 + 0.0288} \approx .5814
 \end{aligned}$$

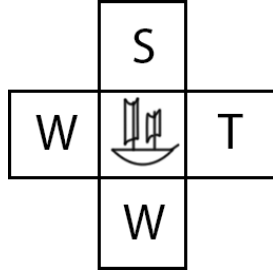
- (e) What is the probability that rotten eggs are present, given that the Mayan Apocalypse is occurring?

$$P(+e | +m) = P(+e) = 0.4$$

The first equality holds true as we have  $E \perp\!\!\!\perp M$  ( $E$  is independent of  $M$ ), which can be inferred from the graph of the Bayes' net.

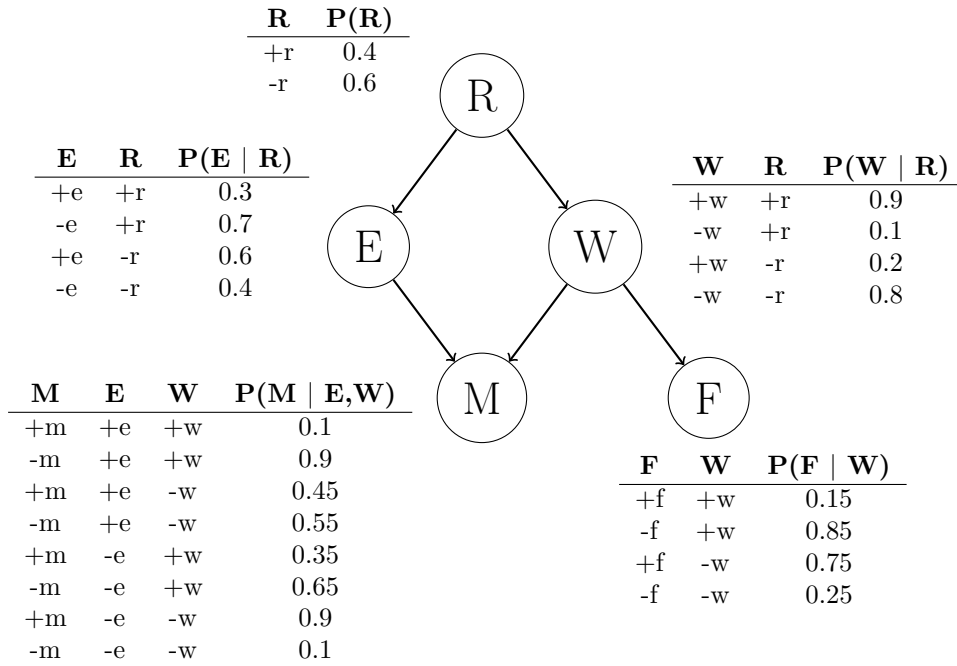
### 3 . Argg! Sampling for the Legendary Treasure

Little did you know that Jasmine and Katie are actually infamous pirates. One day, they go treasure hunting in the Ocean of Bayes, where rumor says a great treasure lies in wait for explorers who dare navigate in the rough waters. After navigating about the ocean, they are within grasp of the treasure. Their current configuration is represented by the boat in the figure below. They can only make one move, and must choose from the actions: (North, South, East, West). Stopping is not allowed. They will land in either a whirlpool (W), an island with a small treasure (S), or an island with the legendary treasure (T). The utilities of the three types of locations are shown below:



| State                  | U(State) |
|------------------------|----------|
| T (Legendary Treasure) | 100      |
| S (Small Treasure)     | 25       |
| W (Whirlpool)          | -50      |

The success of their action depends on the random variable **Movement (M)**, which takes on one of two values: (+m, -m). The Movement random variable has many relationships with other variables: Presence of Enemy Pirates (E), Rain (R), Strong Waves (W), and Presence of Fishermen (F). The Bayes' net graph that represents these relationships is shown below:



In the following questions we will follow a two-step process:

– (1) Jasmine and Katie observed the random variables  $R = -r$  and  $F = +f$ . We then determine the distribution for  $P(M | -r, +f)$  via sampling.

– (2) Based on the estimate for  $P(M | -r, +f)$ , after committing to an action, landing in the intended location of an action successfully occurs with probability  $P(M = +m | -r, +f)$ . The other three possible landing positions occur with probability  $\frac{P(M = -m | -r, +f)}{3}$  each. Use this transition distribution to calculate the optimal action(s) to take and the expected utility of those actions.

- (a) (i) **Rejection Sampling:** You want to estimate  $P(M = +m | -r, +f)$  by rejection sampling. Below is a list of samples that were generated using prior sampling. Cross out those that would be rejected by rejection sampling.

|                                                                       |                                                                       |
|-----------------------------------------------------------------------|-----------------------------------------------------------------------|
| <del>+r</del> <del>+e</del> <del>+w</del> <del>-m</del> <del>-f</del> | -r   -e   +w   -m   +f                                                |
| <del>-r</del> <del>-e</del> <del>+w</del> <del>-m</del> <del>-f</del> | <del>+r</del> <del>-e</del> <del>+w</del> <del>+m</del> <del>-f</del> |
| -r   +e   -w   -m   +f                                                | -r   -e   -w   +m   +f                                                |
| <del>+r</del> <del>-e</del> <del>-w</del> <del>+m</del> <del>-f</del> | <del>+r</del> <del>-e</del> <del>-w</del> <del>+m</del> <del>+f</del> |
| -r   -e   -w   -m   +f                                                | -r   +e   +w   -m   +f                                                |
| -r   +e   -w   -m   +f                                                | -r   +e   -w   -m   +f                                                |

All samples without the conditioning  $-r, +f$  are rejected.

- (ii) What is the approximation for  $P(M = +m | -r, +f)$  using the remaining samples?  
 $\frac{1}{7}$ , the fraction of accepted samples with  $+m$  instantiated.
- (iii) What are the optimal action(s) for Jasmine and Katie based on this estimate of  $P(M = +m | -r, +f)$ ?  
 South, West. As  $p(+m | -r, +f) = \frac{1}{7}$ ,  $p(-m | -r, +f) = \frac{6}{7}$ . Jasmine and Katie will succeed in the selected action  $\frac{1}{7}$  of the time, or take one of the other 3 actions with equal probability of  $\frac{2}{7}$ . In this case,  $p(+m | -r, +f)$  is so low that deciding to head in the direction of the whirlpool actually decreases the chances of landing in it.
- (iv) What is the expected utility for the optimal action(s) based on this estimate of  $P(M = +m | -r, +f)$ ?  
 $\frac{1}{7} * (-50) + \frac{2}{7} * (-50) + \frac{2}{7} * (25) + \frac{2}{7} * (100) = \frac{100}{7}$ , the weighted sum of all four outcomes.

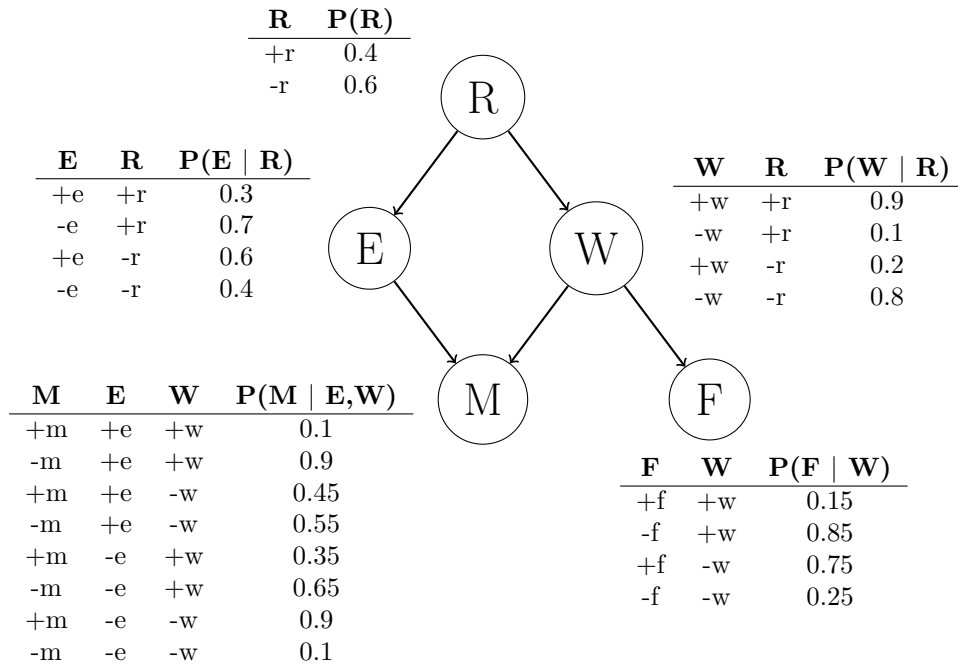
- (b) (i) **Likelihood Weighting:** Suppose instead that you perform likelihood weighting on the following samples to get the estimate for  $P(M = +m | -r, +f)$ . You receive 4 samples consistent with the evidence.

| Sample                 | Weight                                |
|------------------------|---------------------------------------|
| -r   -e   +w   +m   +f | $P(-r)P(+f   +w) = 0.6 * 0.15 = 0.09$ |
| -r   -e   -w   +m   +f | $P(-r)P(+f   -w) = 0.6 * 0.75 = 0.45$ |
| -r   -e   +w   -m   +f | $P(-r)P(+f   +w) = 0.6 * 0.15 = 0.09$ |
| -r   +e   -w   -m   +f | $P(-r)P(+f   -w) = 0.6 * 0.75 = 0.45$ |

- (ii) What is the approximation for  $P(M = +m | -r, +f)$  using the samples above?  
 $\frac{0.09+0.45}{0.09+0.45+0.09+0.45} = \frac{1}{2}$
- (iii) What are the optimal action(s) for Jasmine and Katie based on this estimate of  $P(M = +m | -r, +f)$ ?  
 East
- (iv) What is the expected utility for the optimal action(s) based on this estimate of  $P(M = +m | -r, +f)$ ?  
 $\frac{1}{6} * (-50) + \frac{1}{6} * (-50) + \frac{1}{6} * (25) + \frac{1}{2} * (100) = \frac{75}{2}$



Here is a copy of the Bayes' Net, repeated for your convenience.



- (c) (i) **Gibbs Sampling.** Now, we tackle the same problem, this time using Gibbs sampling. We start out with initializing our evidence:  $R = -r$ ,  $F = +f$ . Furthermore, we start with this random sample:

$-r \ +e \ -w \ +m \ +f$ .

We select variable E to resample. Calculate the numerical value for:

$P(E = +e | R = -r, W = -w, M = +m, F = +f)$ .

$$P(E = +e | R = -r, W = -w, M = +m, F = +f) = \frac{P(+e|-r)P(+m|+e,-w)}{P(+e|-r)P(+m|+e,-w) + P(-e|-r)P(+m|-e,-w)}$$

$$= \frac{0.6 \cdot 0.45}{0.6 \cdot 0.45 + 0.4 \cdot 0.9} = \frac{3}{7}$$

We resample for a long time until we end up with the sample:

$-r \ -e \ +w \ +m \ +f$ .

Jasmine and Katie are happy for fixing this one sample, but they do not have enough time left to compute another sample before making a move. They will let this one sample approximate the distribution:  $P(M = +m | -r, +f)$ .

- (ii) What is the approximation for  $P(M = +m | -r, +f)$ , using this one sample?

1

- (iii) What are the optimal action(s) for Jasmine and Katie based on this estimate of  $P(M = +m | -r, +f)$ ?

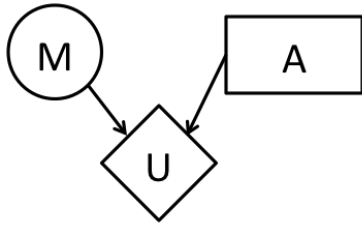
East

- (iv) What is the expected utility for the optimal action(s) based on this estimate of  $P(M = +m | -r, +f)$ ?

100

## 4 . Probability and Decision Networks

The new Josh Bond Movie ( $M$ ), Skyrise, is premiering later this week. Skyrise will either be great ( $+m$ ) or horrendous ( $-m$ ); there are no other possible outcomes for its quality. Since you are going to watch the movie no matter what, your primary choice is between going to the theater (*theater*) or renting (*rent*) the movie later. Your utility of enjoyment is only affected by these two variables as shown below:



| M  | P(M) |
|----|------|
| +m | 0.5  |
| -m | 0.5  |

| M  | A              | U(M,A) |
|----|----------------|--------|
| +m | <i>theater</i> | 100    |
| -m | <i>theater</i> | 10     |
| +m | <i>rent</i>    | 80     |
| -m | <i>rent</i>    | 40     |

### (a) Maximum Expected Utility

Compute the following quantities:

$$EU(\textit{theater}) = P(+m)U(+m, \textit{theater}) + P(-m)U(-m, \textit{theater}) = 0.5 * 100 + 0.5 * 10 = 55$$

$$EU(\textit{rent}) = P(+m)U(+m, \textit{rent}) + P(-m)U(-m, \textit{rent}) = 0.5 * 80 + 0.5 * 40 = 60$$

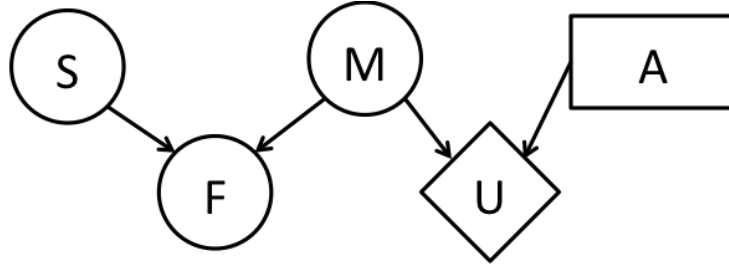
$$MEU(\{\}) = 60$$

Which action achieves  $MEU(\{\}) = \textit{rent}$

(b) **Fish and Chips**

Skyrise is being released two weeks earlier in the U.K. than the U.S., which gives you the perfect opportunity to predict the movie's quality. Unfortunately, you don't have access to many sources of information in the U.K., so a little creativity is in order.

You realize that a reasonable assumption to make is that if the movie ( $M$ ) is great, citizens in the U.K. will celebrate by eating fish and chips ( $F$ ). Unfortunately the consumption of fish and chips is also affected by a possible food shortage ( $S$ ), as denoted in the below diagram.



The consumption of fish and chips ( $F$ ) and the food shortage ( $S$ ) are both binary variables. The relevant conditional probability tables are listed below:

| S  | M  | F  | $P(F S, M)$ |
|----|----|----|-------------|
| +s | +m | +f | 0.6         |
| +s | +m | -f | 0.4         |
| +s | -m | +f | 0.0         |
| +s | -m | -f | 1.0         |

| S  | M  | F  | $P(F S, M)$ |
|----|----|----|-------------|
| -s | +m | +f | 1.0         |
| -s | +m | -f | 0.0         |
| -s | -m | +f | 0.3         |
| -s | -m | -f | 0.7         |

| S  | $P(S)$ |
|----|--------|
| +s | 0.2    |
| -s | 0.8    |

You are interested in the value of revealing the food shortage node ( $S$ ). Answer the following queries:

$$EU(theater| +s) =$$

The shortage variable is independent of the parents of the utility node when no additional evidence is present; thus, the same values hold:

$$EU(theater| +s) = EU(theater) = 55$$

$$EU(rent| +s) = EU(rent) = 60$$

$$MEU(\{+s\}) = 60$$

$$\text{Optimal Action Under } \{+s\} = r \quad (Rent)$$

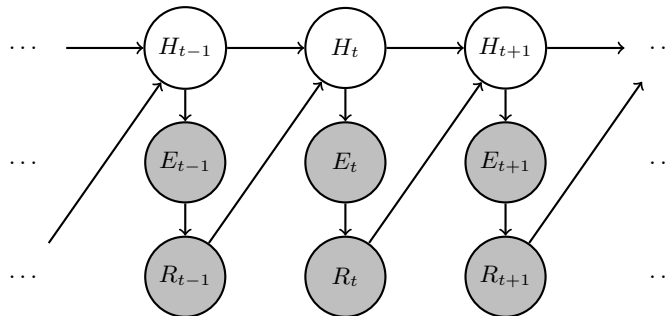
$$MEU(\{-s\}) = 60$$

$$\text{Optimal Action Under } \{-s\} = r \quad (Rent)$$

$VPI(S) = 0$ , since the Value of Perfect Information is the expected difference in MEU given the evidence vs. without the evidence and here the evidence is uninformative.

## 5 . HMM: Human-Robot Interaction

In the near future, autonomous robots would live among us. Therefore, it is important for the robots to know how to properly act in the presence of humans. In this question, we are exploring a simplified model of this interaction. Here, we are assuming that we can observe the robot's actions at time  $t$ ,  $R_t$ , and an evidence observation,  $E_t$ , directly caused by the human action,  $H_t$ . Humans actions and Robots actions from the past time-step affect the Human's and Robot's actions in the next time-step. In this problem, we will remain consistent with the convention that capital letters ( $H_t$ ) refer to random variables and lowercase letters ( $h_t$ ) refer to a particular value the random variable can take. The structure is given below:



You are supplied with the following probability tables:  $P(R_t | E_t)$ ,  $P(H_t | H_{t-1}, R_{t-1})$ ,  $P(H_0)$ ,  $P(E_t | H_t)$ .

Let us derive the forward algorithm for this model. We will split our computation into two components, a **time-elapse update** expression and a **observe update** expression.

- (a) We would like to incorporate the evidence that we observe at time  $t$ . Using the time-lapse update expression we will derive separately, we would like to find the **observe update** expression:

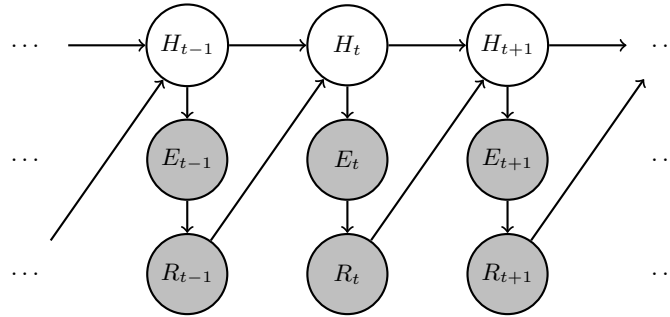
$$O(H_t) = P(H_t | e_{0:t}, r_{0:t})$$

In other words, we would like to compute the distribution of potential human states at time  $t$  given all observations up to and including time  $t$ . In addition to the conditional probability tables associated with the network's nodes, we are given  $T(H_t) = P(H_t | e_{0:t-1}, r_{0:t-1})$ , which we will assume is correctly computed in the time-elapse update that we will derive in the next part. From the options below, select *all* the options that **both** make valid independence assumptions and would evaluate to the observe update expression.

- |                                                                                                                                                                      |                                                                                             |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <input checked="" type="checkbox"/> $\frac{P(H_t   e_{0:t-1}, r_{0:t-1})P(e_t   H_t)P(r_t   e_t)}{\sum_{h_t} P(h_t   e_{0:t-1}, r_{0:t-1})P(e_t   h_t)P(r_t   e_t)}$ | <input type="checkbox"/> $\sum_{r_{t-1}} P(H_t   e_{0:t-1}, r_{0:t-1})P(r_{t-1}   e_{t-1})$ |
| <input checked="" type="checkbox"/> $\frac{P(H_t   e_{0:t-1}, r_{0:t-1})P(e_t   H_t)}{\sum_{h_t} P(h_t   e_{0:t-1}, r_{0:t-1})P(e_t   h_t)}$                         | <input type="checkbox"/> $\sum_{r_t} P(H_t   e_{0:t-1}, r_{0:t-1})P(r_t   r_{t-1}, e_t)$    |
| <input type="checkbox"/> $\frac{\sum_{e_t} P(H_t   e_{0:t-1}, r_{0:t-1})P(e_t   H_t)}{\sum_{h_t} P(h_t   e_{0:t-1}, r_{0:t-1})P(e_t   r_{t-1}, H_{t-1})}$            | <input type="checkbox"/> $\sum_{h_{t+1}} P(H_t   e_{0:t-1}, r_{0:t-1})P(h_{t+1}   r_t)$     |

$$P(H_t | e_{0:t}, r_{0:t}) = \frac{P(H_t, e_{0:t}, e_{0:t})}{\sum_{h_t} P(h_t, e_{0:t}, e_{0:t})} = \frac{P(H_t | e_{0:t-1}, r_{0:t-1})P(e_t | H_t)P(r_t | e_t)}{P(r_t | e_t) \sum_{h_t} P(h_t | e_{0:t-1}, r_{0:t-1})P(e_t | h_t)}$$

The structure below is identical to the one in the beginning of the question and is repeated for your convenience.



- (b) We are interested in predicting what the state of human is at time  $t$  ( $H_t$ ), given all the observations through  $t - 1$ . Therefore, the **time-elapse update** expression has the following form:

$$T(H_t) = P(H_t | e_{0:t-1}, r_{0:t-1})$$

Derive an expression for the given time-elapse update above using the probability tables provided in the question and the observe update expression,  $O(H_{t-1}) = P(H_{t-1} | e_{0:t-1}, r_{0:t-1})$ . Write your final expression in the space provided at below. You may use the function  $O$  in your solution if you prefer.

The derivation of the time-elapse update for this setup is similar to the one we have seen in lecture; however, here, we have additional observations and dependencies.

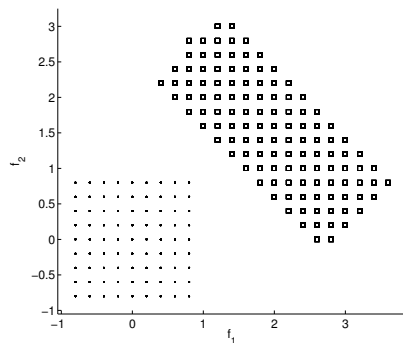
$$\begin{aligned} P(H_t | e_{0:t-1}, r_{0:t-1}) &= \sum_{h_{t-1}} P(H_t, h_{t-1} | e_{0:t-1}, r_{0:t-1}) \\ &= \sum_{h_{t-1}} P(H_t | h_{t-1}, r_{t-1}) P(h_{t-1} | e_{0:t-1}, r_{0:t-1}) \end{aligned}$$

$$P(H_t | e_{0:t-1}, r_{0:t-1}) = \underline{\sum_{h_{t-1}} P(H_t | h_{t-1}, r_{t-1}) P(h_{t-1} | e_{0:t-1}, r_{0:t-1})}$$

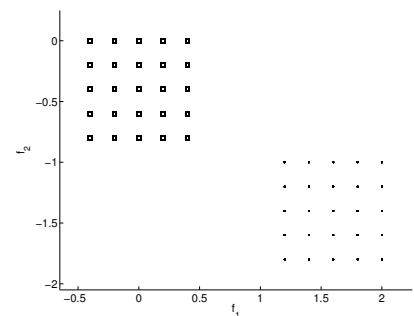
## 6 . Naïve Bayes Modeling Assumptions

You are given points from 2 classes, shown as rectangles and dots. For each of the following sets of points, mark if they satisfy all the Naïve Bayes modelling assumptions, or they do not satisfy all the Naïve Bayes modelling assumptions. Note that in (c), 4 rectangles overlap with 4 dots.

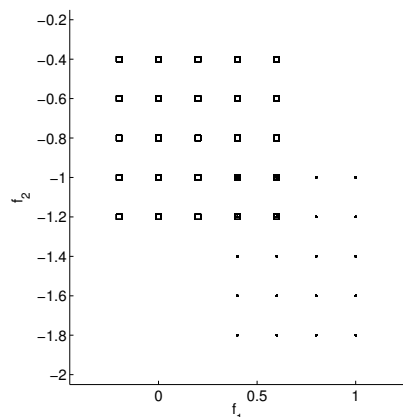
The conditional independence assumptions made by the Naïve Bayes model are that features are conditionally independent when given the class. Features being independent once the class label is known means that for a fixed class the distribution for  $f_1$  cannot depend on  $f_2$ , and the other way around. Concretely, for discrete-valued features as shown below, this means each class needs to have a distribution that corresponds to an axis-aligned rectangle. No other assumption is made by the Naïve Bayes model. Note that linear separability is not an assumption of the Naïve Bayes model—what is true is that for a Naïve Bayes model with all binary variables the decision boundary between the two classes is a hyperplane (i.e., it's a linear classifier). That, however, wasn't relevant to the question as the question examined which probability distribution a Naïve Bayes model can represent, not which decision boundaries.



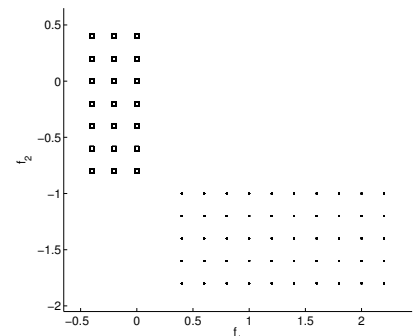
(a) ☐ Satisfies ☒ Does not Satisfy



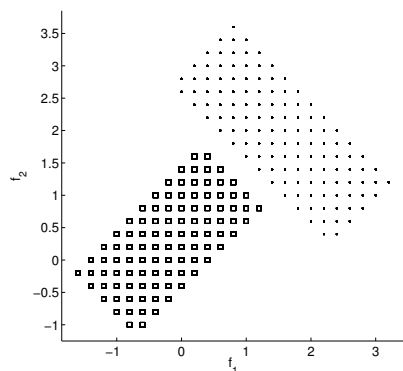
(b) ☒ Satisfies ☐ Does not Satisfy



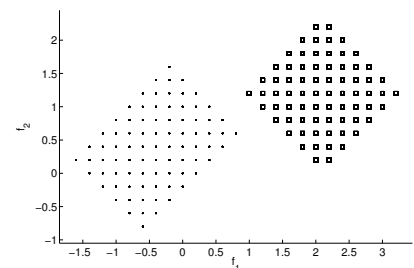
(c) ☒ Satisfies ☐ Does not Satisfy



(d) ☒ Satisfies ☐ Does not Satisfy



(e) ☐ Satisfies ☒ Does not Satisfy



(f) ☐ Satisfies ☒ Does not Satisfy

*A note about feature independence:* The Naïve Bayes model assumes features are conditionally independent given the class. Why does this result in axis-aligned rectangles for discrete feature distributions? Intuitively, this is because fixing one value is uninformative about the other: within a class, the values of one feature are constant across the other. For instance, the dark square class in (b) has  $f_1 \in [-0.5, 0.5]$  and  $f_2 \in [-1, 0]$  and fixing one has no impact on the domain of the other. However, when the features of a class are not axis-aligned then fixing one limits the domain of the other, inducing dependence. In (e), fixing  $f_2 = 1.5$  restricts  $f_1$  to the two points at the top, whereas fixing  $f_2 = 0$  gives a larger domain.

# CS188 Fall 2018 Section 12: Neural Networks and Decision Trees

## 1 Perceptron $\rightarrow$ Neural Nets

Instead of the standard perceptron algorithm, we decide to treat the perceptron as a single node neural network and update the weights using gradient-based optimization.

In lecture, we covered maximizing likelihood using gradient ascent. We can also choose to **minimize** a loss function that calculates the distance between a prediction and the correct label. The loss function for one data point is  $Loss(y, y^*) = \frac{1}{2}(y - y^*)^2$ , where  $y^*$  is the training label for a given point and  $y$  is the output of our single node network for that point.

We will compute a score  $z = w_1x_1 + w_2x_2$ , and then predict the output using an activation function  $g$ :  $y = g(z)$ .

1. Given a general activation function  $g(z)$  and its derivative  $g'(z)$ , what is the derivative of the loss function with respect to  $w_1$  in terms of  $g, g', y^*, x_1, x_2, w_1$ , and  $w_2$ ?

$$\begin{aligned}\frac{\partial Loss}{\partial w_1} &= \frac{\partial}{\partial w_1} \frac{1}{2} (g(w_1x_1 + w_2x_2) - y^*)^2 \\ &= (g(w_1x_1 + w_2x_2) - y^*) * \frac{\partial}{\partial w_1} g(w_1x_1 + w_2x_2) \\ &= (g(w_1x_1 + w_2x_2) - y^*) * g'(w_1x_1 + w_2x_2) * \frac{\partial}{\partial w_1} (w_1x_1 + w_2x_2) \\ &= (g(w_1x_1 + w_2x_2) - y^*) * g'(w_1x_1 + w_2x_2) * x_1\end{aligned}$$

2. We wish to *minimize* the loss, so we will use gradient *descent* (not gradient ascent). What is the update equation for weight  $w_i$  given  $\frac{\partial Loss}{\partial w_i}$  and learning rate  $\alpha$ ?

$$w_i \leftarrow w_i - \alpha \frac{\partial Loss}{\partial w_i}$$

3. For this question, the specific activation function that we will use is

$$g(z) = 1 \text{ if } z \geq 0, \text{ or } -1 \text{ if } z < 0$$

Use gradient descent to update the weights for a single data point. With initial weights of  $w_1 = 2$  and  $w_2 = -2$ , what are the updated weights after processing the data point  $(x_1, x_2) = (-1, 2)$ ,  $y^* = 1$ ?

Because the derivative of  $g$  is always zero,  $g'(z) = 0$  (although it has two pieces, both pieces are constant and so have no slope),  $\frac{\partial Loss}{\partial w_1}$  will be zero, and so the weights will stay  $w_1 = 2$  and  $w_2 = -2$ .

4. What is the most critical problem with this gradient descent training process with that activation function?  
The gradient of that activation function is zero, so the weights will not update.



## 2 Decision Trees

You are a geek who hates sports. Trying to look cool at a party, you join a discussion that you believe to be about football and basketball. You gather information about the two main subjects of discussion, but still cannot figure out what sports they play.

| Sport | Position      | Name           | Height | Weight | Age | College        |
|-------|---------------|----------------|--------|--------|-----|----------------|
| ?     | Guard         | Charlie Ward   | 6'02"  | 185    | 41  | Florida State  |
| ?     | Defensive End | Julius Peppers | 6'07"  | 283    | 32  | North Carolina |

Fortunately, you have brought your CS 188 notes along, and will build some classifiers to determine which sport is being discussed.

You come across a pamphlet from the Atlantic Coast Conference Basketball Hall of Fame, as well as an Oakland Raiders team roster, and create the following table:

| Sport      | Position | Name                 | Height | Weight | Age | College        |
|------------|----------|----------------------|--------|--------|-----|----------------|
| Basketball | Guard    | Michael Jordan       | 6'06"  | 195    | 49  | North Carolina |
| Basketball | Guard    | Vince Carter         | 6'06"  | 215    | 35  | North Carolina |
| Basketball | Guard    | Muggsy Bogues        | 5'03"  | 135    | 47  | Wake Forest    |
| Basketball | Center   | Tim Duncan           | 6'11"  | 260    | 35  | Oklahoma       |
| Football   | Center   | Vince Carter         | 6'02"  | 295    | 29  | Oklahoma       |
| Football   | Kicker   | Tim Duncan           | 6'00"  | 215    | 33  | Oklahoma       |
| Football   | Kicker   | Sebastian Janikowski | 6'02"  | 250    | 33  | Florida State  |
| Football   | Guard    | Langston Walker      | 6'08"  | 345    | 33  | California     |

### 2.1 Entropy

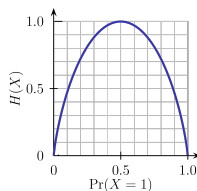
Before we get started, let's review the concept of entropy.

1. Give the definition of entropy for an arbitrary probability distribution  $P(X)$ .

$$H(X) = -\sum_x P(x) \log_2(1/P(x))$$

You can see this as the *expected* information content of the distribution.

2. Draw a graph of entropy  $H(X)$  vs.  $P(X = 1)$  for a binary random variable  $X$ .



3. What is the entropy of the distribution of *Sport* in the training data? What about *Position*?

To calculate the entropy for a random variable, we estimate the probability distribution and use the formula from the part above.

$$P(S = \text{football}) = 1/2, P(S = \text{basketball}) = 1/2 \quad H(S) = \frac{\log_2(2)}{2} + \frac{\log_2(2)}{2} = 1$$

$$P(P = \text{guard}) = 1/2, P(P = \text{kicker}) = 1/4, P(P = \text{center}) = 1/4 \quad H(P) = \frac{\log_2(2)}{2} + \frac{\log_2(4)}{4} + \frac{\log_2(4)}{4} = 3/2$$

## 2.2 Decision Trees

Central to decision trees is the concept of “splitting” on a variable.

1. To review the concept of “information gain”, calculate it for a split on the *Sport* variable.

Since the variable that we want to predict is *Sport*, we want to be calculating the entropy with respect to the variable *Sport*.

- (a)
  - i. Distribution before: 8 examples with  $(1/2, 1/2)$ . (here the first number in the tuple is  $P(\text{basketball})$ , and the second number is  $P(\text{football})$ ).
  - ii. Entropy before:  $\frac{8}{8} \left( \frac{\log(2)}{2} + \frac{\log(2)}{2} \right)$
- (b)
  - i. Distribution after: 4 examples with  $(1, 0)$ , 4 examples with  $(0, 1)$
  - ii. Entropy after:  $\frac{4}{8} \left( \frac{\log(1)}{1} \right) + \frac{4}{8} \left( \frac{\log(1)}{1} \right) = 0$

So, the information gain is  $(1 - 0) = 1$ , which is the greatest possible.

2. Of course, in our situation this would not make sense, as *Sport* is the very variable we lack at test time. Now calculate the information gain for the decision “stumps” (one-split trees) created by first splitting on *Position*, *Name*, and *College*. Do any of these perfectly classify the training data? Does it make sense to use *Name* as a variable? Why or why not?

Note that here we will be splitting on different variables but still need to look at the entropy of the distribution of the variable we need to predict which is *sport*. So, the before case remains same as before.

(a) **Position**

- i. Distribution after: 4 examples with  $(3/4, 1/4)$ , 2 examples with  $(1/2, 1/2)$ , 2 examples with  $(0, 1)$ .
- ii. Entropy after:  $\frac{4}{8} \left( \frac{\log(4/3)}{4/3} + \frac{\log(4)}{4} \right) + \frac{2}{8} \left( \frac{\log(2)}{2} + \frac{\log(2)}{2} \right) + \frac{2}{8} \left( \frac{\log(1)}{1} \right) = 0.66$

(b) **Name**

- i. Distribution after: 1 examples with  $(1, 0)$ , 2 examples with  $(1/2, 1/2)$ , 1 examples with  $(0, 1)$ , 2 examples with  $(1/2, 1/2)$ , 1 example with  $(0,1)$ , 1 example with  $(0,1)$ .
- ii. Entropy after: 0.5

(c) **College**

- i. Distribution after: 2 examples with  $(1, 0)$ , 1 examples with  $(1, 0)$ , 3 examples with  $(1/3, 2/3)$ , 1 examples with  $(0, 1)$ , 1 example with  $(0,1)$ .
- ii. Entropy after: 0.34

Note that none of these variables completely classifies the data.

Regarding using the **Name** as a feature to use in classifying data: since we expect people’s names to be unique, using them as a feature in learning is akin to using the unique ID of each data point. That is to say, it’s quite a bad idea—you will overfit to the training data.

3. Decision trees can represent any function of discrete attribute variables. How can we *best* cast continuous variables (*Height*, *Weight*, and *Age*) into discrete variables?

Use an inequality relation,  $\text{Attribute} > a$ , where  $a$  is a split point chosen to give the highest information gain. E.g., an initial split on  $\text{Age} > 34$  will perfectly classify the training data.

4. Draw a few decision trees that each correctly classify the training data, and show how their predictions vary on the test set. What algorithm are you following? We use the algorithm as given in the slides, and for each split use the variable that gives us the maximum information gain. In this given problem, as we observed above, the variable *Age* correctly classifies all of the training data, so that is the first variable that gets picked up, and the algorithm stops at that.

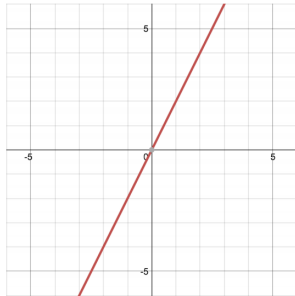
This decision tree would predict test example 1 to be Basketball and test example 2 to be Football.

5. You may have noticed that the testing data has a value for *Position* that is missing in training data. What could we do in this case?

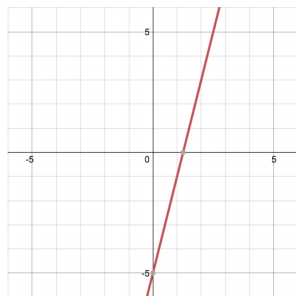
When we come to a split on a variable whose value for the test subject is missing in the tree, we could just choose the most likely branch of the split (the branch that leads to the node with the greatest number of items).

### 3 Neural Network Representations

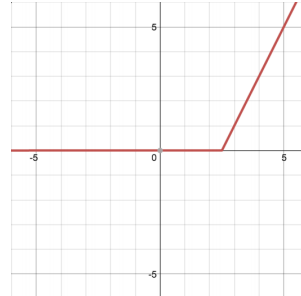
You are given a number of functions (a-h) of a single variable,  $x$ , which are graphed below. The computation graphs on the following pages will start off simple and get more complex, building up to neural networks. For each computation graph, indicate which of the functions below they are able to represent.



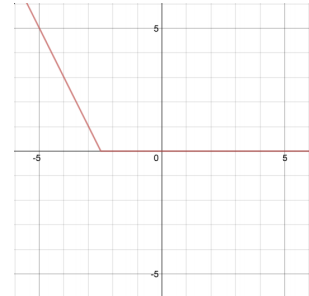
(a)  $2x$



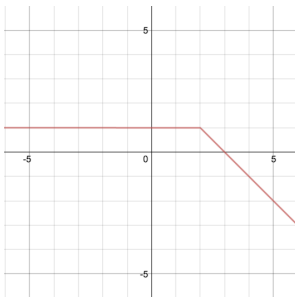
(b)  $4x - 5$



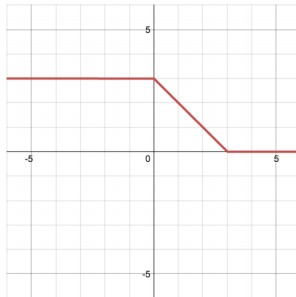
(c)  $\begin{cases} 2x - 5 & x \geq 2.5 \\ 0 & x < 2.5 \end{cases}$



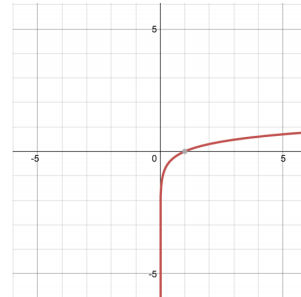
(d)  $\begin{cases} -2x - 5 & x \leq -2.5 \\ 0 & x > -2.5 \end{cases}$



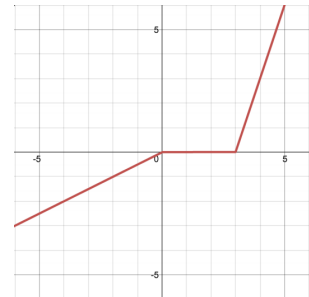
(e)  $\begin{cases} -x + 3 & x \geq 2 \\ 1 & x < 2 \end{cases}$



(f)  $\begin{cases} 3 & x \leq 0 \\ 3 - x & 0 < x \leq 3 \\ 0 & x > 3 \end{cases}$

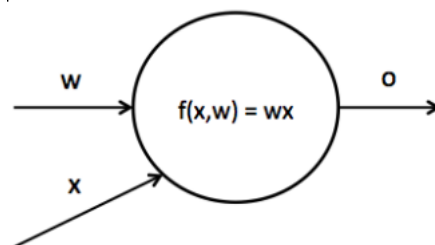


(g)  $\log(x)$



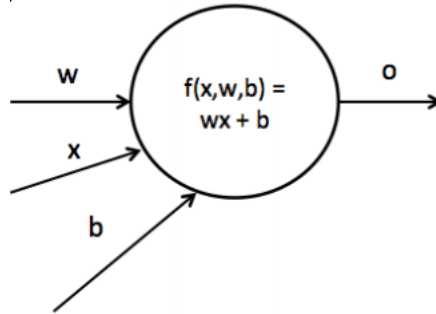
(h)  $\begin{cases} 0.5x & x \leq 0 \\ 0 & 0 < x \leq 3 \\ 3x - 9 & x > 3 \end{cases}$

- Consider the following computation graph, computing a linear transformation with scalar input  $x$ , weight  $w$ , and output  $o$ , such that  $o = wx$ . Which of the functions can be represented by this graph? For the options which can, write out the appropriate value of  $w$ .



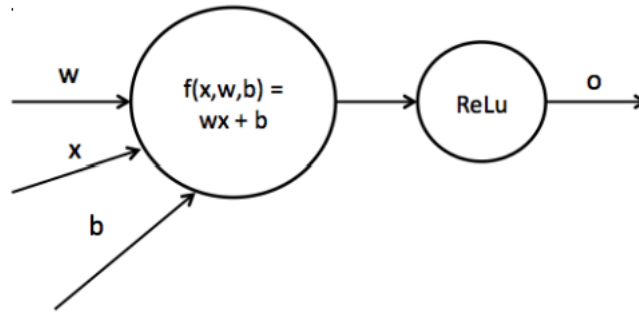
This graph can only represent (a), with  $w = 2$ . Since there is no bias term, the line must pass through the origin.

2. Now we introduce a bias term  $b$  into the graph, such that  $o = wx + b$  (this is known as an *affine* function). Which of the functions can be represented by this network? For the options which can, write out an appropriate value of  $w, b$ .



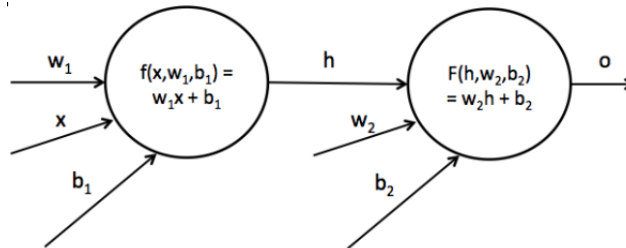
(a) with  $w = 2$  and  $b = 0$ , and (b) with  $w = 4$  and  $b = -5$

3. We can introduce a non-linearity into the network as indicated below. We use the ReLU non-linearity, which has the form  $ReLU(x) = \max(0, x)$ . Now which of the functions can be represented by this neural network with weight  $w$  and bias  $b$ ? For the options which can, write out an appropriate value of  $w, b$ .



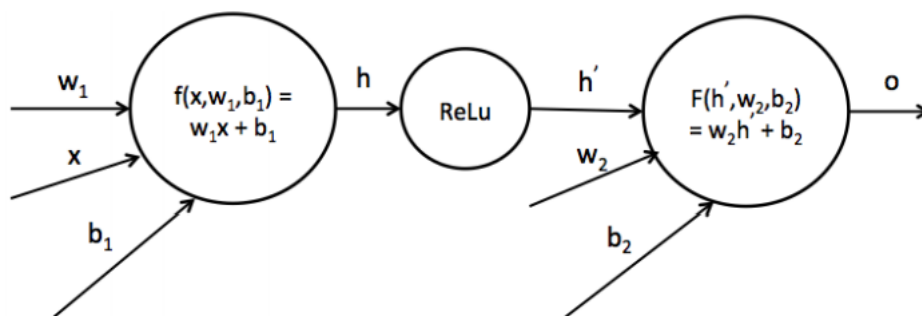
With the output coming directly from the ReLU, this cannot produce any values less than zero. It can produce (c) with  $w = 2$  and  $b = -5$ , and (d) with  $w = -2$  and  $b = -5$

4. Now we consider neural networks with multiple affine transformations, as indicated below. We now have two sets of weights and biases  $w_1, b_1$  and  $w_2, b_2$ . We denote the result of the first transformation  $h$  such that  $h = w_1x + b_1$ , and  $o = w_2h + b_2$ . Which of the functions can be represented by this network? For the options which can, write out appropriate values of  $w_1, w_2, b_1, b_2$ .



Applying multiple affine transformations (with no non-linearity in between) is not any more powerful than a single affine function:  $w_2(w_1x + b_1) + b_2 = w_2w_1x + w_2b_1 + b_2$ , so this is just a affine function with different coefficients. The functions we can represent are the same as in 1, if we choose  $w_1 = w, w_2 = 0, b_1 = 0, b_2 = b$ : (a) with  $w_1 = 2, w_2 = 1, b_1 = 0, b_2 = 0$ , and (b) with  $w_1 = 4, w_2 = 1, b_1 = 0, b_2 = -5$ .

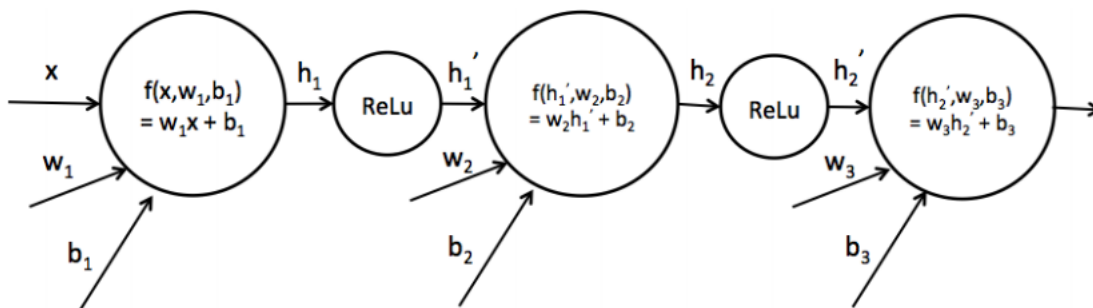
5. Next we add a ReLU non-linearity to the network after the first affine transformation, creating a hidden layer. Which of the functions can be represented by this network? For the options which can, write out appropriate values of  $w_1, w_2, b_1, b_2$ .



(c), (d), and (e). The affine transformation after the ReLU is capable of stretching (or flipping) and shifting the ReLU output in the vertical dimension. The parameters to produce these are:

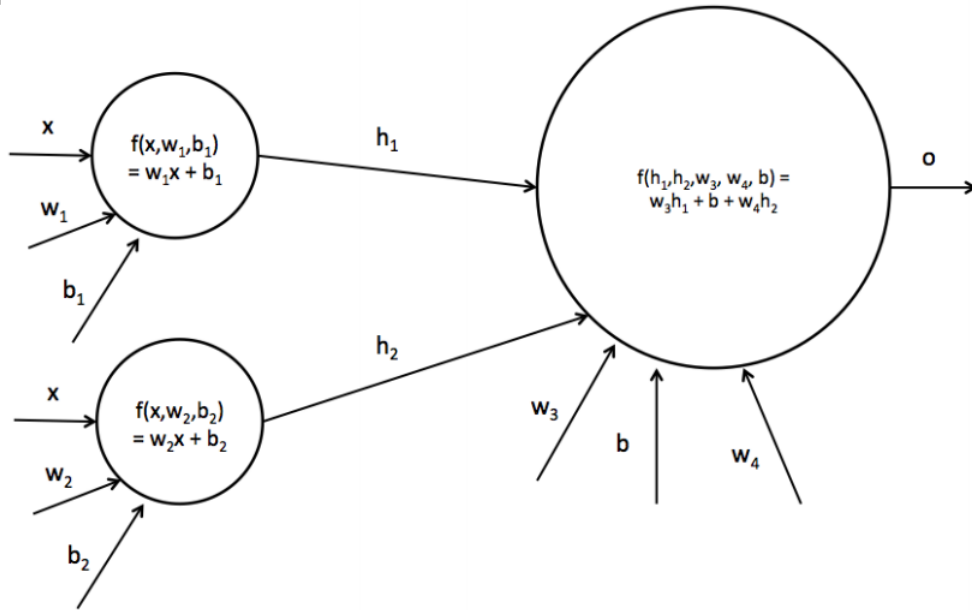
(c) with  $w_1 = 2, b_1 = -5, w_2 = 1, b_2 = 0$ , (d) with  $w_1 = -2, b_1 = -5, w_2 = 1, b_2 = 0$ , and (e) with  $w_1 = 1, b_1 = -2, w_2 = -1, b_2 = 1$

6. Now we add another hidden layer to the network, as indicated below. Which of the functions can be represented by this network?



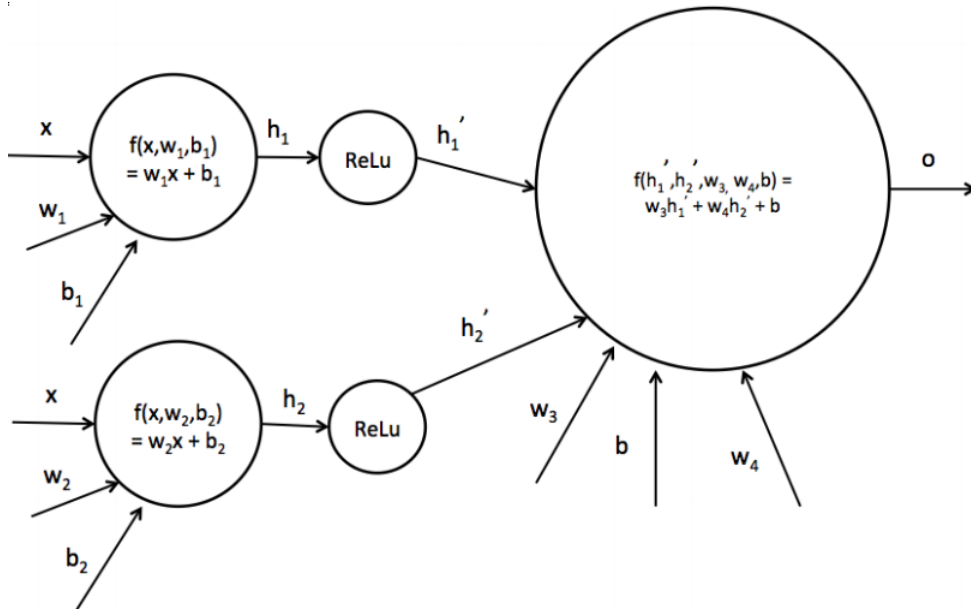
(c), (d), (e), and (f). The network can represent all the same functions as Q5 (because note that we could have  $w_2 = 1$  and  $b_2 = 0$ ). In addition it can represent (f): the first ReLU can produce the first flat segment, the affine transformation can flip and shift the resulting curve, and then the second ReLU can produce the second flat segment (with the final affine layer not doing anything). Note that (h) cannot be produced since its line has only one flat segment (and the affine layers can only scale, shift, and flip the graph in the vertical dimension; they can't rotate the graph).

7. We'd like to consider using a neural net with just one hidden layer, but have it be larger – a hidden layer of size 2. Let's first consider using just two affine functions, with no nonlinearity in between. Which of the functions can be represented by this network?



(a) and (b). With no non-linearity, this reduces to a single affine function (in the same way as Q4)

8. Now we'll add a non-linearity between the two affine layers, to produce the neural network below with a hidden layer of size 2. Which of the functions can be represented by this network?



All functions except for (g). Note that we can recreate any network from (5) by setting  $w_4$  to 0, so this allows us to produce (c), (d) and (e). To produce the rest of the functions, note that  $h'_1$  and  $h'_2$  will be two independent functions with a flat part lying on the x-axis, and a portion with positive slope. The final layer takes a weighted sum of these two functions. To produce (a) and (b), the flat portion of one ReLU should start at the point where the other ends ( $x = 0$  for (a), or  $x = 1$  for (b)). The final layer

then vertically flips the ReLU sloping down and adds it to the one sloping up, producing a single sloped line. To produce (h), the ReLU sloping down should have its flat portion end (at  $x = 0$  before the other's flat portion begins (at  $x = 3$ ). The down-sloping one is again flipped and added to the up-sloping. To produce (f), both ReLUs should have equal slope, which will cancel to produce the first flat portion above the x-axis.



# 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:

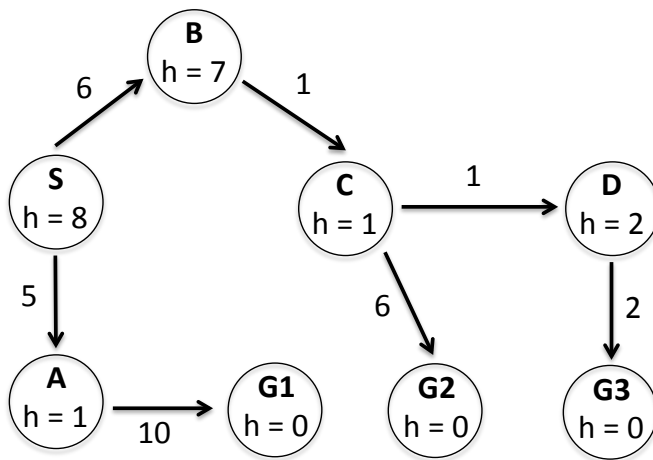
$$f(n) = g(n) + \varepsilon h(n)$$

where  $\varepsilon \geq 1$  is a parameter given to the algorithm. In general, the larger the value of  $\varepsilon$ , 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  $\varepsilon$  as a parameter so it can compute the correct *f*-value of the node.

```
1: function WEIGHTED-A*-TREE-SEARCH(problem, fringe,  $\varepsilon$ )
2:   loop do
3:     if fringe is empty then return failure
4:     node  $\leftarrow$  REMOVE-FRONT(fringe)
5:     if GOAL-TEST(problem, STATE[node]) then return node
6:     for child-node in child-nodes do
7:       fringe  $\leftarrow$  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.

| <i>node</i>                                             | Goal? | <i>fringe</i>                                                                          |
|---------------------------------------------------------|-------|----------------------------------------------------------------------------------------|
| -                                                       | -     | $\{S : g = 0, f = 16\}$                                                                |
| <i>S</i>                                                | No    | $\{S \rightarrow A : g = 5, f = 7; S \rightarrow B : g = 6, f = 20\}$                  |
| <i>S</i> $\rightarrow$ <i>A</i>                         | No    | $\{S \rightarrow A \rightarrow G1 : g = 15, f = 15; S \rightarrow B : g = 6, f = 20\}$ |
| <i>S</i> $\rightarrow$ <i>A</i> $\rightarrow$ <i>G1</i> | 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, remember that  $f(n) = g(n) + \varepsilon h(n)$  comes from the problem statement and the inequality  $g(n) + \varepsilon h(n) \leq \varepsilon (g(n) + h(n))$  is true by algebra.

Since we know that  $g(n)$  is non-negative, it must be true that  $g(n) + \varepsilon h(n) \leq \varepsilon (g(n) + h(n))$ . 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\*

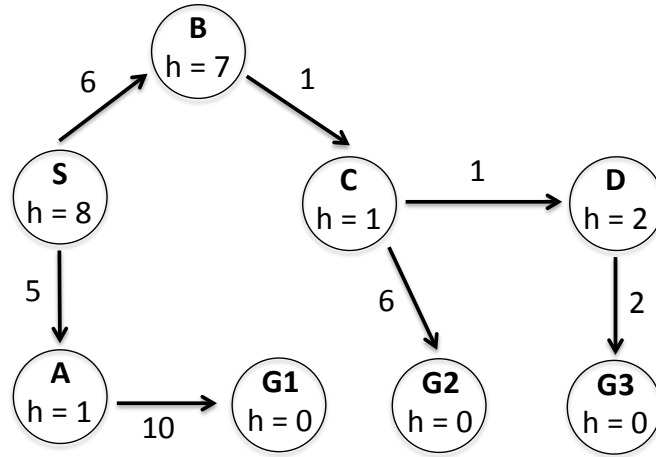
- (ii)  $\varepsilon = 0$ .

Algorithm: UCS

- (iii)  $\varepsilon \rightarrow \infty$  (i.e., as  $\varepsilon$  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    | $\{S \rightarrow A : g = 5, f = 6; S \rightarrow B : g = 6, f = 13\}$                                                                                                                             |
| $S \rightarrow A$                                            | No    | $\{S \rightarrow B : g = 6, f = 13; S \rightarrow A \rightarrow G1 : g = 15, f = 15\}$                                                                                                            |
| $S \rightarrow B$                                            | No    | $\{S \rightarrow B \rightarrow C : g = 7, f = 8; S \rightarrow A \rightarrow G1 : g = 15, f = 15\}$                                                                                               |
| $S \rightarrow B \rightarrow C$                              | No    | $\{S \rightarrow B \rightarrow C \rightarrow D : g = 8, f = 10; S \rightarrow B \rightarrow C \rightarrow G2 : g = 13, f = 13; S \rightarrow A \rightarrow G1 : g = 15, f = 15\}$                 |
| $S \rightarrow B \rightarrow C \rightarrow D$                | No    | $\{S \rightarrow B \rightarrow C \rightarrow D \rightarrow G3 : g = 10, f = 10; S \rightarrow B \rightarrow C \rightarrow G2 : g = 13, f = 13; S \rightarrow A \rightarrow G1 : g = 15, f = 15\}$ |
| $S \rightarrow B \rightarrow C \rightarrow D \rightarrow G3$ | 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  $\varepsilon$
- ☐ Use  $F$  with goal  $G$  removed as new starting fringe, updating the  $f$ -values to account for the new  $\varepsilon$
- ☐ 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  $\varepsilon$
- ☐ 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 to update the  $f$ -values to reflect the new value of  $\varepsilon$ . With these modifications, it is valid to continue searching as the state of the fringe is as if A\* with the new  $\varepsilon$  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).<sup>1</sup> 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 (4, 5) are DAM, and NET.

Example Crossword Grid and Solution

|                |                |                |
|----------------|----------------|----------------|
| <sup>1</sup> D | <sup>2</sup> A | <sup>3</sup> M |
| <sup>4</sup> E | R              | A              |
| <sup>5</sup> 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  $\{d_1, d_2, \dots, d_K\}$ . 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  $k$ th across word. For example, in the crossword above,  $A_1 = \text{DAM}$ ,  $D_1 = \text{DEN}$ ,  $D_2 = \text{ARE}$ , and so on. Let  $D_1[i]$  denote the letter in the  $i$ th 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.

<sup>1</sup><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 | 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.

|                |                |                |                |   |
|----------------|----------------|----------------|----------------|---|
| <sup>1</sup> B | <sup>2</sup> L | <sup>3</sup> O | <sup>4</sup> G | S |
| <sup>5</sup> L | A              | R              | P              |   |
| <sup>6</sup> A | R              | C              | S              |   |
| <sup>7</sup> 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,  $X_1 = D$ ,  $X_2 = A$ , 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  $X_1, X_2, \dots, X_5$  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  $X_1 = G$ , the possible domains for the remaining letters are

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

Upon assigning  $X_1 = B$ , the possible domains remaining are

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

The remaining variables are unaffected since we are using only forward checking. Now, we see that with the assignment  $X_1 = G$ , the minimum size remaining for any domain is 1, while the sum of the sizes remaining domains is 11; for  $X_1 = B$ , 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.

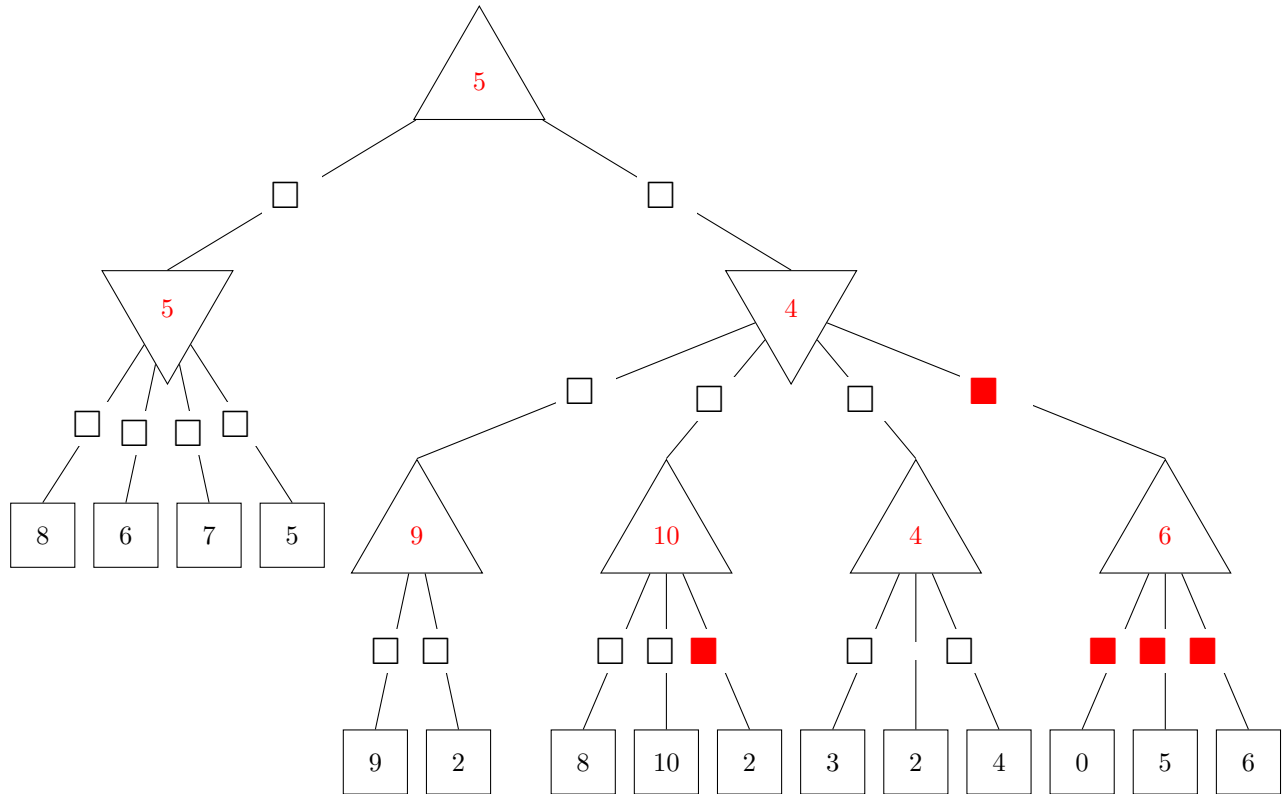
Any choice but  $X_1 = B$  or  $X_1 = G$  will eliminate all values for one of the other variables after forward checking.

### 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

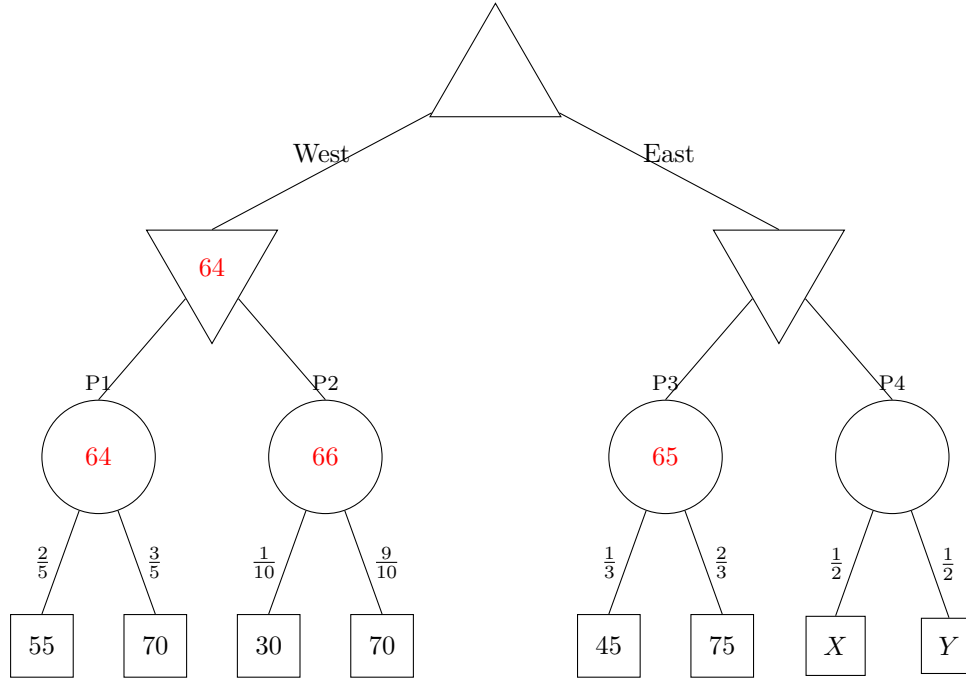
**(b) Food Dimensions**

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:  $X + Y > 128$

To reach P4:  $X + Y = 129$

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  $\frac{X+Y}{2}$ .

$$\Rightarrow \min(65, \frac{X+Y}{2}) > 64$$

$$\Rightarrow \frac{X+Y}{2} > 64$$

$$\text{So, } X + Y > 128 \Rightarrow value(A) > value(B)$$

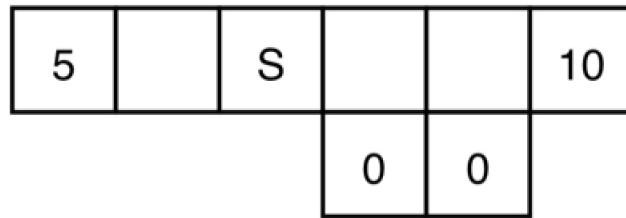
To ensure reaching  $X$  or  $Y$ , apart from the above, we also have  $\frac{X+Y}{2} < 65$

$$\Rightarrow 128 < X + Y < 130$$

$$\text{So, } X, Y \in \mathbb{N} \Rightarrow X + Y = 129$$



## 4 . Discount MDPs

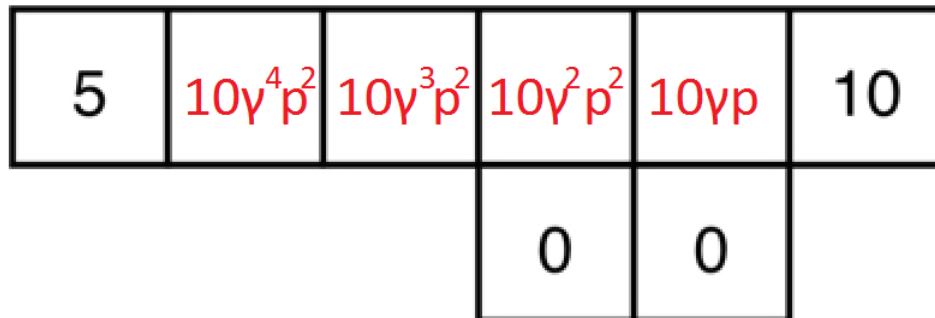


Consider the above gridworld. An agent is currently on grid cell  $S$ , and would like to collect the rewards that lie on both sides of it. If the agent is on a numbered square, its only available action is to Exit, and when it exits it gets reward equal to the number on the square. On any other (non-numbered)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

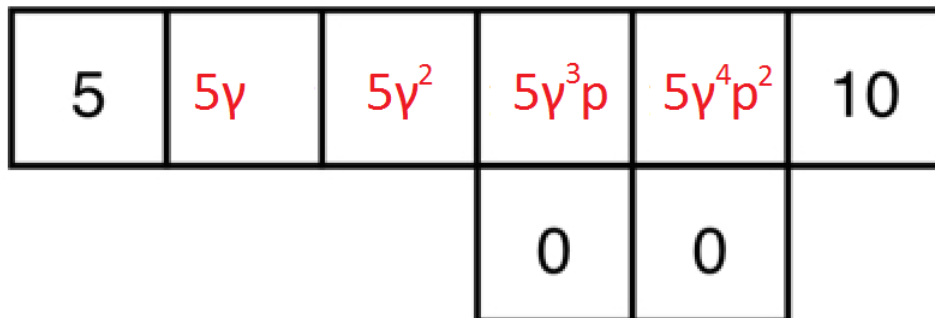
If the agent is in a square with an adjacent square downward, it does not always move successfully: when the agent is in one of these squares and takes a move action, it will only succeed with probability  $p$ . With probability  $1 - p$ , the move action will fail and the agent will instead move downwards.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



- (b) Consider the policy  $\pi_{\text{West}}$ , 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  $\gamma$  and  $p$ .



- (c) For what range of values of  $p$  in terms of  $\gamma$  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  $\gamma$  is  $\pi_{\text{West}}$  the optimal policy?

We need, for each of the four cells, to have the value of that cell under  $\pi_{\text{West}}$  to be at least as large as  $\pi_{\text{East}}$ . 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  $\gamma$  ranges from 0 to 1, the right side of this expression ranges from 2 to  $\infty$ , 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  $\gamma$  is  $\pi_{\text{East}}$  the optimal policy?

We follow the same logic as in the previous part. Specifically, we focus on the leftmost cell, where the condition for  $\pi_{\text{East}}$  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  $\gamma$ .

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:

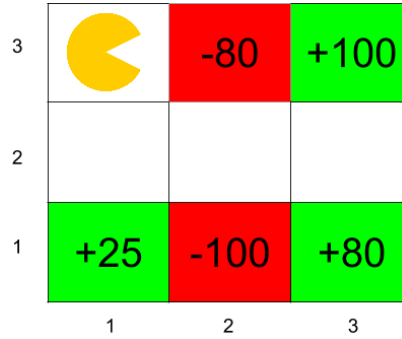
$$\frac{dL_1}{dw_m} = \begin{cases} -f_m(x) & y - \sum_k w_k f_k(x) > 0 \\ f_m(x) & y - \sum_k w_k f_k(x) < 0 \end{cases}$$

- (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 grid). 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:

$$V^*(3, 2) = \underline{100} \quad V^*(2, 2) = \underline{50} \quad V^*(1, 3) = \underline{12.5}$$

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:

$$Q((3,2),N) = \underline{50} \quad Q((1,2),S) = \underline{0} \quad Q((2,2),E) = \underline{12.5}$$

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:

$$w_1 = \underline{\quad -100 \quad}$$

$$w_2 = \underline{\quad -100 \quad}$$

$$w_3 = \underline{\quad -100 \quad}$$

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

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:

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

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

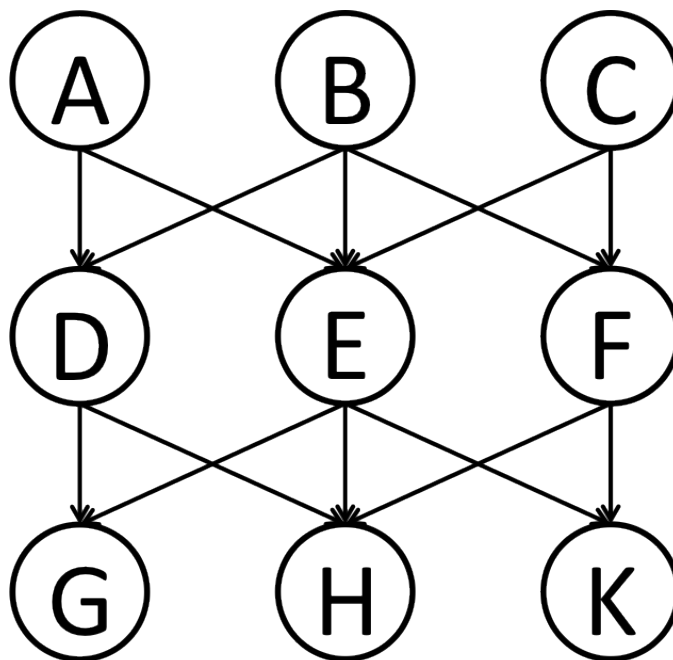
$$\begin{aligned}(1) \left(\frac{1}{2}\right) + \mathbf{P}(B = \text{true} \mid A = \text{false}) \left(\frac{1}{2}\right) &= \frac{3}{4} \\ \implies \mathbf{P}(B = \text{true} \mid A = \text{false}) \left(\frac{1}{2}\right) &= \frac{1}{4} \\ \implies \mathbf{P}(B = \text{true} \mid A = \text{false})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

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:

| $A$  | $B$  | $f_1(A, B)$ | $A$  | $C$  | $D$  | $f_2(A, C, D)$ | $B$  | $D$  | $f_3(B, D)$ |
|------|------|-------------|------|------|------|----------------|------|------|-------------|
| $+a$ | $+b$ | 0.1         | $+a$ | $+c$ | $+d$ | 0.2            | $+b$ | $+d$ | 0.2         |
| $+a$ | $-b$ | 0.5         | $+a$ | $+c$ | $-d$ | 0.1            | $+b$ | $-d$ | 0.2         |
| $-a$ | $+b$ | 0.2         | $+a$ | $-c$ | $+d$ | 0.5            | $-b$ | $+d$ | 0.5         |
| $-a$ | $-b$ | 0.5         | $+a$ | $-c$ | $-d$ | 0.1            | $-b$ | $-d$ | 0.1         |
|      |      |             | $-a$ | $+c$ | $+d$ | 0.5            |      |      |             |
|      |      |             | $-a$ | $+c$ | $-d$ | 0.2            |      |      |             |
|      |      |             | $-a$ | $-c$ | $+d$ | 0.5            |      |      |             |
|      |      |             | $-a$ | $-c$ | $-d$ | 0.2            |      |      |             |

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.

| $A$  | $B$  | $D$  | $f'_4(A, B, D)$    |
|------|------|------|--------------------|
| $+a$ | $+b$ | $+d$ | $0.1 * 0.2 = 0.02$ |
| $+a$ | $+b$ | $-d$ | $0.1 * 0.2 = 0.02$ |
| $+a$ | $-b$ | $+d$ | $0.5 * 0.5 = 0.25$ |
| $+a$ | $-b$ | $-d$ | $0.5 * 0.1 = 0.05$ |
| $-a$ | $+b$ | $+d$ | $0.2 * 0.2 = 0.04$ |
| $-a$ | $+b$ | $-d$ | $0.2 * 0.2 = 0.04$ |
| $-a$ | $-b$ | $+d$ | $0.5 * 0.5 = 0.25$ |
| $-a$ | $-b$ | $-d$ | $0.5 * 0.1 = 0.05$ |

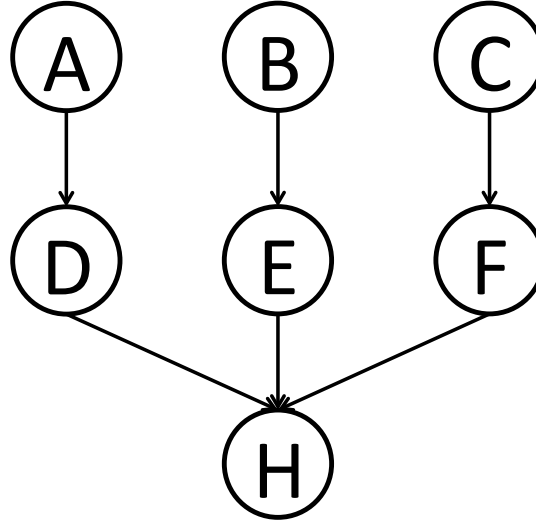
(iii) Perform the summation over  $B$  for the factor you obtained from the join. Your answer should be a table similar to the tables above, it is your job to figure out which variables participate and what the numerical entries are.

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

(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  $P(+a| -b, -c)$ ?

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



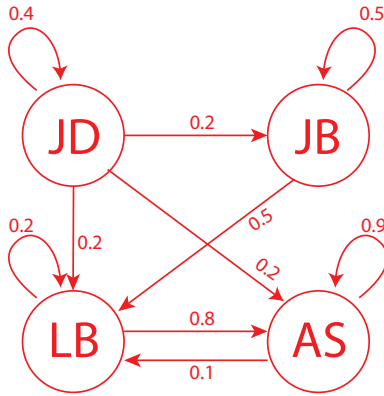
## 8 . HMM: Where is the key?

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  $X_0$ ,  $X_{\text{Mon}}$  and  $X_{\text{Tue}}$  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

$$x = 0.2x + 0.1y$$

$$y = .8x + .9y$$

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:

|    | $P(X_{\text{Mon}})$ | $P(X_{\text{Mon}}   R_{\text{Mon}} = \text{pessim.})$ | $P(X_{\text{Tue}}   R_{\text{Mon}} = \text{pessim.})$ | $P(X_{\text{Tue}}   R_{\text{Mon}} = \text{pessim.}, R_{\text{Tue}} = \text{optim.})$ |
|----|---------------------|-------------------------------------------------------|-------------------------------------------------------|---------------------------------------------------------------------------------------|
| 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,  $R_{\text{Thu}} \perp\!\!\!\perp X_{\text{Fri}} | X_{\text{Thu}}$ , so we compute

|    | $P(X_{\text{Thu}})$ | $P(X_{\text{Fri}})$ | $P(X_{\text{Fri}}   R_{\text{Fri}} = \text{pessim.})$ |
|----|---------------------|---------------------|-------------------------------------------------------|
| 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  $X_{\text{Tue}} \perp\!\!\!\perp X_{\text{Fri}} | X_{\text{Thu}}$

- (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:

- Yes, reports for Saturday and Sunday affect our prediction for the key holder on Friday.
- 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  $i$ th hidden state is the location of the ghost at timestep  $i$  and the  $i$ th 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  $P(X_1, \dots, X_T|e_{1:T})$ ).

- (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  $p < \underline{\hspace{1cm}}$ .
- ☒ If  $p > \underline{0.5}$ .
- ☐ 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  $p < \underline{0.75}$ .
- ☐ If  $p > \underline{\hspace{1cm}}$ .
- ☐ 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?

$$w_A = \underline{(3, -5, 4)} \quad w_B = \underline{(0, -2, 9)} \quad w_C = \underline{(7, 9, -2)}$$

- (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  $\{0^2, 1^2, \dots, 9^2\}$ . 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 individual 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  |

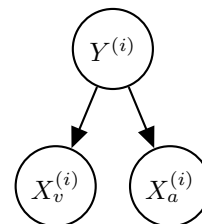
The superscript  $(i)$  denotes that the datum is the  $i$ th one. Now, the individual with  $i = 20$  comes out, and you want to predict the individual's type  $Y^{(20)}$  given that you observed  $X_v^{(20)} = 1$  and  $X_a^{(20)} = 1$ .

- (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).

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

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:

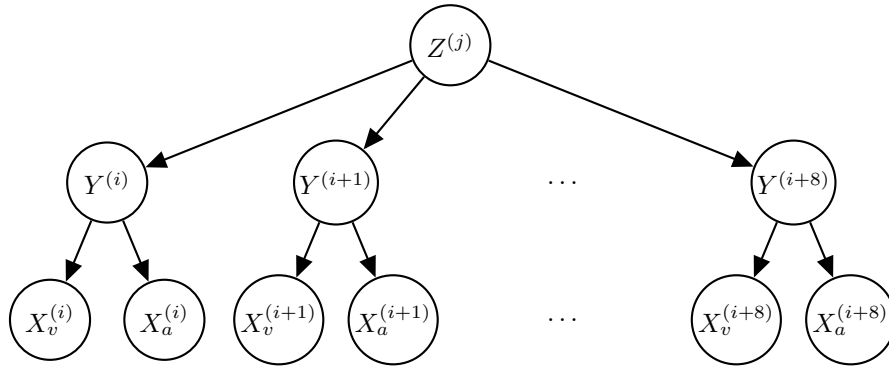
$$P(X_v^{(i)} = x_v | Y^{(i)} = y) = \begin{cases} p_v & \text{if } x_v = y \\ 1 - p_v & \text{if } x_v \neq y \end{cases}$$

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

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

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

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  $r$ ?

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

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).

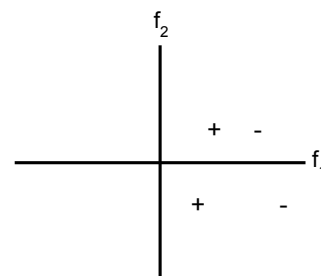


$$P(Y^{(20)} = 1, X_v^{(20)} = 1, X_a^{(20)} = 1, Y^{(19)} = 1, Y^{(18)} = 1) = \frac{p_a p_v [q_0^3(1-r) + q_1^3 r]}{}$$

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

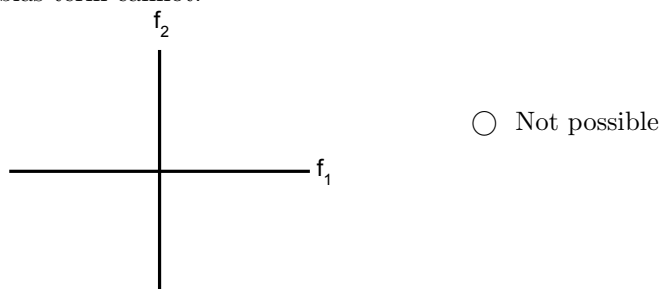
## 12 . Decision Trees and Other Classifiers

- (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.



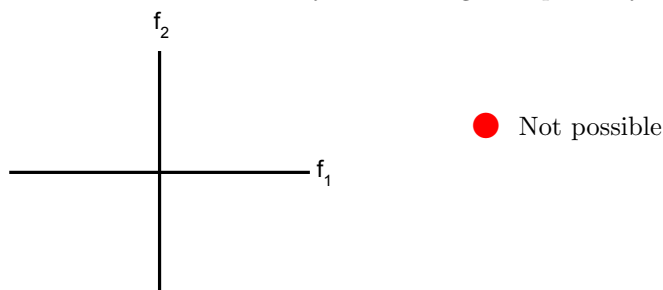
For example, if the conditions were “A depth-1 decision tree can perfectly classify the training data points,” an acceptable answer would be the data points to the right.

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



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, 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:

| $X$ | $P(X)$ | $X$ | $F_1$ | $P(F_1 X)$ | $X$ | $F_2$ | $P(F_2 X)$ |
|-----|--------|-----|-------|------------|-----|-------|------------|
| -   | 0.4    | -   | -1    | 0.4        | -   | -1    | 0.1        |
| +   | 0.6    | -   | 0     | 0.5        | -   | 0     | 0.1        |
|     |        | -   | 1     | 0.1        | -   | 1     | 0.8        |
|     |        | +   | -1    | 0.7        | +   | -1    | 0.6        |
|     |        | +   | 0     | 0.1        | +   | 0     | 0.1        |
|     |        | +   | 1     | 0.2        | +   | 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?

☒  $X = -$  ☐  $X = +$

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

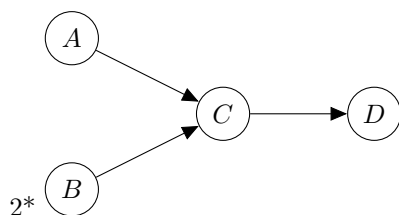
- (iii) If you observe that  $F_1 = 1$  and  $F_2 = 1$ , how will you classify  $X$  using Naive Bayes?

☐  $X = -$  ☒  $X = +$

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

# 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     | Algorithm      | Query     |
|----------------------|-----------|----------------|-----------|
| Likelihood Weighting | $P(C +a)$ | 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.