CS 188 Introduction to Summer 2015 Artificial Intelligence — Final

- You have approximately 2 hours 50 minutes.
- The exam is closed book, closed calculator, and closed notes except your one-page crib sheet.
- Mark your answers ON THE EXAM ITSELF. If you are not sure of your answer you may wish to provide a *brief* explanation. All short answer sections can be successfully answered in a few sentences AT MOST.

First name	
Last name	
SID	
edX username	
Name of person on your left	
Name of person on your right	

For staff use only:

Q1.	Search and Probability	/10
Q2.	Games	/8
Q3.	Utilities	/10
Q4.	Farmland CSP	/8
Q5.	MDP	/16
Q6.	Bayes Nets	/8
Q7.	Chameleon	/10
Q8.	Perceptron	/10
	Total	/80

THIS PAGE IS INTENTIONALLY LEFT BLANK

Q1. [10 pts] Search and Probability

Each True/False question is worth 1 points. Leaving a question blank is worth 0 points. **Answering incorrectly is worth −1 points.**

(a) Consider a graph search problem where for every action, the cost is at least ϵ, with $\epsilon > 0$. Assume the heuristic is admissible.

(i) [1 pt] [*true* or *false*] Uniform-cost graph search is guaranteed to return an optimal solution.
True. UCS expands paths in order of least total cost so that the optimal solution is found.

(ii) [1 pt] [*true* or *false*] The path returned by uniform-cost graph search may change if we add a positive constant C to every step cost. True. Consider that there are two paths from the start state (S) to the goal (G), $S \to A \to G$ and $S \to G$. $cost(S, A) = 1$, $cost(A, G) = 1$, and $cost(S, G) = 3$. So the optimal path is through A. Now, if we add 2 to each of the costs, the optimal path is directly from S to G. Since uniform cost search finds the optimal path, its path will change.

(iii) [1 pt] [*true* or *false*] A* graph search is guaranteed to return an optimal solution.
False, the heuristic is admissible, but is not guaranteed to be consistent, which is required for optimal graph search.

(iv) [1 pt] [*true* or *false*] A* graph search is guaranteed to expand no more nodes than depth-first graph search.
False. Depth-first graph search could, for example, go directly to a sub-optimal solution.

(v) [1 pt] [*true* or *false*] If $h_1(s)$ and $h_2(s)$ are two admissible A* heuristics, then their average $f(s) = \frac{1}{2}h_1(s) + \frac{1}{2}h_2(s)$ must also be admissible. True. Let $h^*(s)$ be the true distance from s. We know that $h_1(s) \leq h^*(s)$ and $h_2(s) \leq h^*(s)$, thus $h_{avg}(s) = \frac{1}{2}h_1(s) + \frac{1}{2}h_2(s) \leq \frac{1}{2}h^*(s) + \frac{1}{2}h^*(s) = h^*(s)$

(b) [3 pts] A, B, C, and D are random variables with binary domains. How many entries are in the following probability tables and what is the sum of the values in each table? Write a "?" in the box if there is not enough information given.

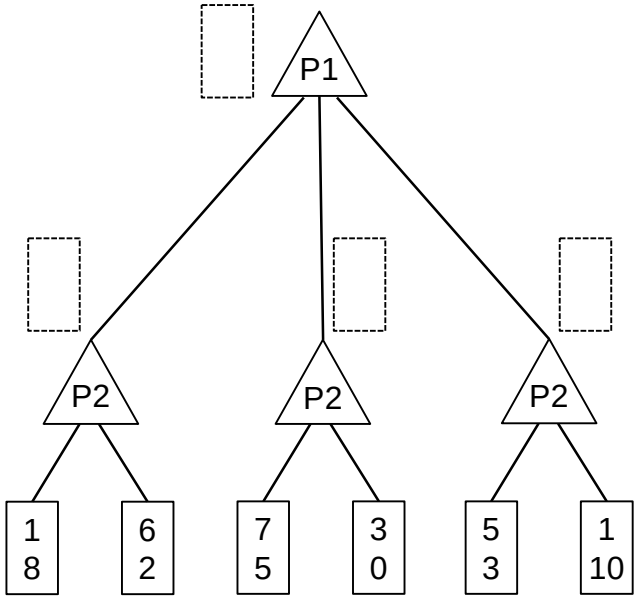
Table	Size	Sum
$P(A\|C)$	4	2
$P(A, D\| + b, +c)$	4	1
$P(B\| + a, C, D)$	8	4

(c) [2 pts] Write all the possible chain rule expansions of the joint probability $P(a, b, c)$. No conditional independence assumptions are made.

$P(a)P(b|a)P(c|a, b)$, $P(a)P(c|a)P(b|a, c)$,
$P(b)P(a|b)P(c|a, b)$, $P(b)P(c|b)P(a|b, c)$,
$P(c)P(b|c)P(a|b, c)$, $P(c)P(a|c)P(b|a, c)$

Q2. [8 pts] Games

For the following game tree, each player maximizes their respective utility. Let x, y respectively denote the top and bottom values in a node. Player 1 uses the utility function $U_1(x, y) = x$.

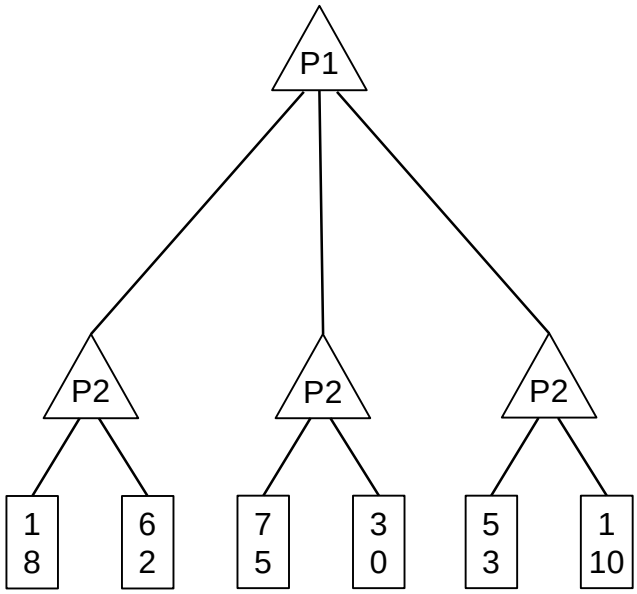


(a) Both players know that Player 2 uses the utility function $U_2(x, y) = x - y$.

(i) [2 pts] Fill in the rectangles in the figure above with pair of values returned by each max node. From top-down, left-right: $(6, 2), (6, 2), (3, 0), (5, 3)$

(ii) [2 pts] You want to save computation time by using pruning in your game tree search. On the game tree above, put an 'X' on branches that do not need to be explored or simply write 'None'. Assume that branches are explored from left to right. None.

Figure repeated for convenience



(b) Now assume Player 2 changes their utility function based on their mood. The probabilities of Player 2's utilities and mood are described in the following table. Let M, U respectively denote the mood and utility function of Player 2.

$P(M = happy)$	$P(M = mad)$
a	b

	$M = happy$	$M = mad$
$P(U_2(x,y) = -x \mid M)$	c	f
$P(U_2(x,y) = x - y \mid M)$	d	g
$P(U_2(x,y) = x^2 + y^2 \mid M)$	e	h

(i) [4 pts] Calculate the maximum expected utility of the game for Player 1 in terms of the values in the game tree and the tables. It may be useful to record and label your intermediate calculations. You may write your answer in terms of a max function.

We first calculate the new probabilities of each utility function as follows.

$P(U_2(x,y) = -x)$	$P(U_2(x,y) = x - y)$	$P(U_2(x,y) = x^2 + y^2)$
$ac + bf$	$ad + bg$	$ae + bh$

$EU(\text{Left Branch}) = (ac + bf)(1) + (ad + bg)(6) + (ae + bh)(6)$
$EU(\text{Middle Branch}) = (ac + bf)(3) + (ad + bg)(3) + (ae + bh)(7)$
$EU(\text{Right Branch}) = (ac + bf)(1) + (ad + bg)(5) + (ae + bh)(1)$

$MEU(\phi) = max((ac+bf)(1)+(ad+bg)(6)+(ae+bh)(6), (ac+bf)(3)+(ad+bg)(3)+(ae+bh)(7), (ac+bf)(1)+(ad+bg)(5)+(ae+bh)(1))$

Q3. [10 pts] Utilities

Davis is on his way to a final exam planning meeting. He is already running late (the meeting is starting now) and he's trying to determine whether he should wait for the bus or just walk.

It takes 20 minutes to get to Cory Hall by walking, and only 5 minutes to get to there by bus. The bus will either come in 10, 20, or 30 minutes, each with probability 1/3.

(a) [3 pts] Davis hates being late; his utility for being late as a function of t, the number of minutes late he is, is

$$U_D(t) = \begin{cases} 0 & : t \leq 0 \\ -2^{t/5} & : t > 0 \end{cases}$$

What is the expected utility of each action? Should he wait for the bus or walk?

$$\begin{aligned} EU(walk) &= -2^{20/5} = -16 \\ EU(bus) &= \frac{1}{3}\big(-2^{(10+5)/5} - 2^{(20+5)/5} - 2^{(30+5)/5}\big) \\ &= \frac{1}{3}\big(-8 - 32 - 128\big) = -168/3 = -56 \end{aligned}$$

Davis should walk.

(b) [3 pts] Pat is running late too. However, Pat reasons that once he's late, it doesn't matter how late he is. Therefore, his utility function is

$$U_P(t) = \begin{cases} 0 & : t \leq 0 \\ -10 & : t > 0 \end{cases}$$

Moreover, Pat prefers riding the bus because it is more comfortable, so riding the bus incurs a utility bonus of 5.

If Pat is deciding whether to take the bus or walk when the meeting is just starting, what are his expected utilities for each action? Should he take the bus or walk?

$$\begin{aligned} EU(walk) &= -10 \\ EU(bus) &= -10 + 5 = -5 \end{aligned}$$

Pat should take the bus.

(c) [2 pts] Give an example of a decreasing utility function in terms of time such that it will favor decisions that always minimize expected time to get to the meeting.

$U(t) = -t$. Any decreasing linear function of t is correct.

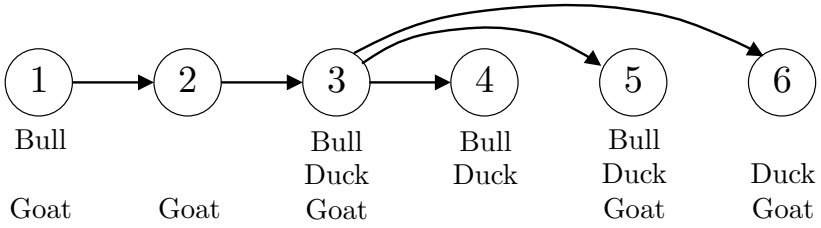
(d) [2 pts] Give an example of a decreasing utility function in terms of time such that it will be risk-seeking; that is, a lottery with expected time of arrival t will be preferred to a guarantee of arrival time t.

$U(t) = -\sqrt{t}$. Any decreasing function with a positive second derivative (concave up) is correct.

Q4. [8 pts] Farmland CSP

The animals in Farmland aren't getting along and the farmers have to assign them to different pens. To avoid fighting, animals of the same type cannot be in connected pens. Fortunately, the Farmland pens are connected in a tree structure.

(a) [2 pts] Consider the following constraint diagram that shows six pens with lines indicating connected pens. The remaining domains for each pen are listed below each node.



After assigning a bull to pen 5, enforce arc consistency on this CSP considering only the *directed* arcs shown in the figure. What are the remaining values for each pen?

Pen	Values
1	Bull
2	Goat
3	Duck, Goat
4	Bull, Duck
5	Bull
6	Duck, Goat

(b) [2 pts] What is the computational complexity of solving general tree structured CSPs with n nodes and d values in the domain? Give an answer of the form $O(\cdot)$.

$O(nd^2)$

(c) This True/False question is worth 1 points. Leaving a question blank is worth 0 points. **Answering incorrectly is worth −1 points.**

(i) [1 pt] [*true* or *false*] If root to leaf arcs are consistent on a general tree structured CSP, assigning values to nodes from root to leaves will not back-track if a solution exists.

True. Because the arcs are consistent, there is a valid value not matter which parent value was assigned.

(d) [3 pts] Given 3 animal types, what is the most number of pens a tree structure could have, such that the computational complexity to solve the tree CSP is no greater than the computational complexity to solve a fully connected CSP with 10 pens?

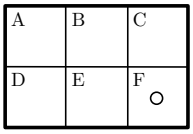
3^8. A fully connected CSP is $O(d^n)$, while a tree structure is $O(nd^2)$. The intent of this question was to show that you could have 3^8 nodes in a tree structure and that would be roughly same amount of computation as a fully connected problem with 10 nodes ($3^{10} = 3^8 3^2$). Unfortunately, this question is poorly worded, because computational complexity doesn't quite work with specific values like this.

Q5. [16 pts] MDP

Pacman is using MDPs to maximize his expected utility. In each environment:

- Pacman has the standard actions {North, East, South, West} unless blocked by an outer wall
- There is a reward of 1 point when eating the dot (for example, in the grid below, $R(C, South, F) = 1$)
- The game ends when the dot is eaten

(a) Consider a the following grid where there is a single food pellet in the bottom right corner (F). The **discount** factor is 0.5. There is no living reward. The states are simply the grid locations.



(i) [2 pts] What is the optimal policy for each state?

State	$\pi(state)$
A	East or South
B	East or South
C	South
D	East
E	East

(ii) [2 pts] What is the optimal value for the state of being in the upper left corner (A)? Reminder: the discount factor is 0.5.

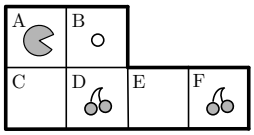
$V^*(A) = 0.25$

k	V(A)	V(B)	V(C)	V(D)	V(E)	V(F)
0	0	0	0	0	0	0
1	0	0	1	0	1	0
2	0	0.5	1	0.5	1	0
3	0.25	0.5	1	0.5	1	0
4	0.25	0.5	1	0.5	1	0

(iii) [2 pts] Using value iteration with the value of all states equal to zero at k=0, for which iteration k will $V_k(A) = V^*(A)$?

$k = 3$ (see above)

(b) Consider a new Pacman level that begins with cherries in locations D and F. Landing on a grid position with cherries is worth 5 points and then the cherries at that position disappear. There is still one dot, worth 1 point. The game still only ends when the dot is eaten.



(i) [2 pts] With no discount ($\gamma = 1$) and a living reward of -1, what is the optimal policy for the states in this level's state space?

State	$\pi(state)$
A	South
C	East
D, F_{Cherry}=true	East
D, F_{Cherry}=false	North
E, F_{Cherry}=true	East
E, F_{Cherry}=false	West
F	West

Larger state spaces with equivalent states and actions are possible too. For example with the state representation of (grid, D-cherry, F-cherry), there could be up to 24 different states, where all four with A are the same, etc.

(ii) [2 pts] With no discount ($\gamma = 1$), what is the range of living reward values such that Pacman eats exactly one cherry when starting at position A?

Valid range for the living reward is (-2.5,-1.25).

Let x equal the living reward.

The reward for eating zero cherries {A,B} is $x + 1$ (one step plus food).

The reward for eating exactly one cherry {A,C,D,B} is $3x + 6$ (three steps plus cherry plus food).

The reward for eating two cherries {A,C,D,E,F,E,D,B} is $7x+11$ (seven steps plus two cherries plus food).

x must be greater than -2.5 to make eating at least one cherry worth it ($3x + 6 > x + 1$).

x must be less than -1.25 to eat less than one cherry ($3x + 6 > 7x + 11$).

(c) Quick reinforcement learning questions [PLEASE WRITE CLEARLY]:

(i) [1 pt] What is the difference between value-iteration and TD-learning?

Value iteration has explicity models for transitions and rewards, while TD-learning relies on active samples.

(ii) [1 pt] What is the difference between TD-learning and Q-learning?

TD-learning stores and updates V(s) while Q-learning stores and updates Q(s,a). Also, Q-learning is able to learn quality policies despite random or suboptimal actions, while TD-learning values are affected by the actions taken.

(iii) [1 pt] What is the purpose of using a learning rate (α) during Q-learning?

The learning rate allows us to average information from previous iterations with the current sample. It allows us to step towards a solution at an incremental rate. This allows us to incorporate random samples while moving away from poor initial estimates.

(iv) [1 pt] In value iteration, we store the value of each state. What do we store during *approximate* Q-learning?

We update and store the weights associated with the features.

(v) [2 pts] Give one advantage and one disadvantage of using approximate Q-learning rather than standard Q-learning.

Pros: Feature representation scales to very large or infinite spaces; learning process generalizes from seen states to unseen states.

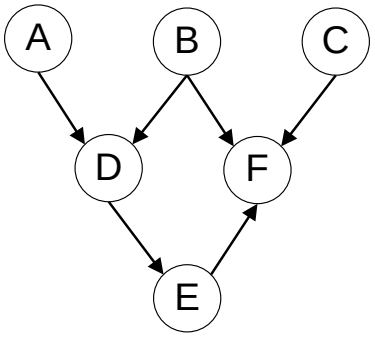
Cons: True Q may not be representable in the chosen form; learning may not converge; need to design feature functions.

Q6. [8 pts] Bayes Nets

(a) For the following graphs, explicitly state the minimum size set of edges that must be removed such that the corresponding independence relations are guaranteed to be true.

Marked the removed edges with an 'X' on the graphs.

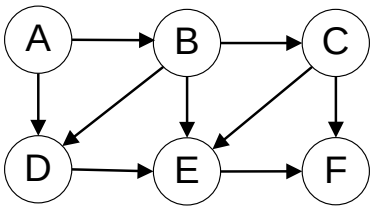
(i) [2 pts]



$A \perp\!\!\!\perp B|F$
$A \perp\!\!\!\perp F|D$
$B \perp\!\!\!\perp C$

AD

(ii) [2 pts]



$A \perp\!\!\!\perp D|B$
$A \perp\!\!\!\perp F|C$
$C \perp\!\!\!\perp D|B$

AD, (*EF* OR *AB*)

(b) You're performing variable elimination over a Bayes Net with variables A, B, C, D, E. So far, you've finished joining over (but not summing out) C, when you realize you've lost the original Bayes Net!

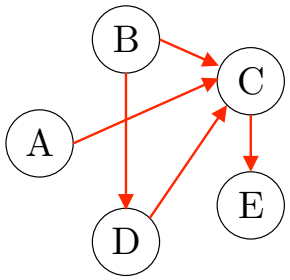
Your current factors are $f(A), f(B), f(B, D), f(A, B, C, D, E)$. Note: these are factors, NOT joint distributions. You don't know which variables are conditioned or unconditioned.

(i) [2 pts] What's the smallest number of edges that could have been in the original Bayes Net? Draw out one such Bayes Net below.

Number of edges = 5

The original Bayes net must have had 5 factors, 1 for each node. f(A) and f(B) must have corresponded to nodes A and B, and indicate that neither A nor B have any parents. f(B, D), then, must correspond to node D, and indicates that D has only B as a parent. Since there is only one factor left, f(A, B, C, D, E), for the nodes C and E, those two nodes must have been joined while you were joining C. This implies two things: 1) E must have had C as a parent, and 2) every other node must have been a parent of either C or E.

The below figure is one possible solution that uses the fewest possible edges to satisfy the above.

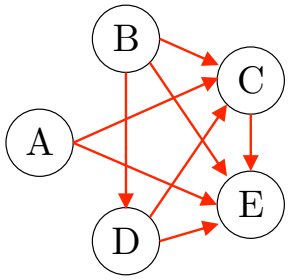


(ii) [2 pts] What's the largest number of edges that could have been in the original Bayes Net? Draw out one such Bayes Net below.

Number of edges = 8

The constraints are the same as outlined in part i). To maximize the number of edges, we make each of A, B, and D a parent of both C and E, as opposed to a parent of one of them.

The below figure is the only possible solution.



Q7. [10 pts] Chameleon

A team of scientists from Berkeley discover a rare species of chameleons. Each one can change its color to be blue or gold, once a day. The probability of colors on a certain day are determined solely by its color on the previous day.

The team spends 5 days observing 10 chameleons changing color from day to day. The recorded counts for the chameleons' color transitions are below.

# of $C_{t+1}\|C_t$	$t=0$	$t=1$	$t=2$	$t=3$
# of $C_{t+1}=gold\|C_t=gold$	0	0	8	2
# of $C_{t+1}=blue\|C_t=gold$	7	0	0	8
# of $C_{t+1}=gold\|C_t=blue$	0	8	2	0
# of $C_{t+1}=blue\|C_t=blue$	3	2	0	0

(a) [3 pts] They suspect that this phenomenon obeys the stationarity assumption – that is, the transition probabilites are actually the same between all the days. Estimate the transition probabilites $P(C_{t+1}|C_t)$ from the above simulation.

	$P(C_{t+1}\|C_t)$
$P(C_{t+1}=gold\|C_t=gold)$	$10/25 = 2/5$
$P(C_{t+1}=blue\|C_t=gold)$	$15/25 = 3/5$
$P(C_{t+1}=gold\|C_t=blue)$	$10/15 = 2/3$
$P(C_{t+1}=blue\|C_t=blue)$	$5/25 = 1/3$

To solve this problem, find the total number of chameleons that were gold ($8+2+7+8=25$) and then split it into those that turned gold ($8+2=10$) and those that turned blue ($7+8=15$). Normalizing yields 10/25 and 15/25 for the first two probabilies. Repeat for the chameleons that were blue.

One common mistake was incorrectly normalizing of the probability table (e.g. dividing by 40 instead of 25). Another was to use only the transitions on t=1 and t=3 to get 0.2, 0.8, 0.8, 0.2, which fails to account for the other observed transitions on t=0 and t=2.

(b) [2 pts] Further scientific tests determine that these chameleons are, in fact, immortal. As a result, they want to determine the distribution of a chameleon's colors over an infinite amount of time.

Given the estimated transition probabilities, what is the steady state distribution for $P(C_\infty)$?

	$P(C_\infty)$
$P(C_\infty=gold)$	10/19
$P(C_\infty=blue)$	9/19

Let $g = P(C_\infty = gold)$ and $b = P(C_\infty = blue)$.

$$g = \frac{2}{5} + \frac{2}{3}b \implies \frac{3}{5}g = \frac{2}{3}b \implies g = \frac{10}{9}b$$

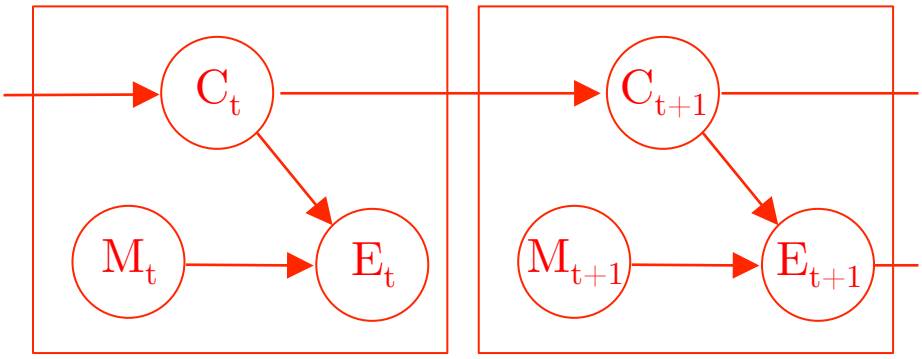
$$g + b = 1$$

Combining the two:

$$\frac{2}{5}b + b = \frac{19}{9}b = 1 \implies b = \frac{9}{19} \implies g = \frac{10}{19}$$

The chameleons, realizing that these tests are being performed, decide to hide. The scientists can no longer observe them directly, but they can observe the bugs that one particular chameleon likes to eat. They know that the chameleon's color influences the probability that it will eat some fraction of a nest. The scientists will observe the size of the nests twice per day: once in the morning, before the chameleon eats, and once in the evening, after the chameleon eats. Every day, the chameleon moves on to a new nest.

(c) [1 pt] Draw a DBN using the variables C_t, C_{t+1}, M_t, M_{t+1}, E_t, and E_{t+1}. C refers to the color of the chameleon, M is the size of a nest in the morning, and E is the size of that nest in the evening.



When the chameleon is blue, it eats half of the bugs in the chosen nest with probability 1/2, one-third of the bugs with probability 1/4, and two-thirds of the bugs with probability 1/4.

When the chameleon is gold, it eats one-third, half, or two-thirds of the bugs, each with probability 1/3.

(d) [4 pts] You would like to use particle filtering to guess the chameleon's color based on the observations of M and E. You observe the following population sizes: $M_1 = 24$, $E_1 = 12$, $M_2 = 36$, and $E_2 = 24$. Fill in the following tables with the weights you would assign to particles in each state at each time step.

State at $t = 1$	Weight
Blue	$\frac{1}{2}$
Gold	$\frac{1}{3}$

State at $t = 2$	Weight
Blue	$\frac{1}{4}$
Gold	$\frac{1}{3}$

The weights in HMM particle filtering are exactly equal to $P(\text{emission} \mid \text{parents of emission})$. In this problem, the change is that the emission is dependent on an addiitonal parameter, the number of morning bugs. For the blue state at t=1, the weight is equal to $P(E_1 = 12 | M_1 = 24, C_1 = Blue) = 1/2$.

One extra step that was commonly made was to normalize the weights afterwards to 1 or some other number; this is extraneous as the resample step of particle filtering only depends on the relative (not absolute) weights of the particles.

Q8. [10 pts] Perceptron

We would like to use a perceptron to train a classifier for datasets with 2 features per point and labels +1 or -1.

Consider the following labeled training data:

Features (x_1, x_2)	Label y^*
(-1,2)	1
(3,-1)	-1
(1,2)	-1
(3,1)	1

(a) [2 pts] Our two perceptron weights have been initialized to $w_1 = 2$ and $w_2 = -2$. After processing the first point with the perceptron algorithm, what will be the updated values for these weights?

For the first point, $y = g(w_1x_1 + w_2x_2) = g(2 \cdot -1 + -2 \cdot 2) = g(-5) = -1$, which is incorrectly classified. To updathe weights, we add the first data point: $w_1 = 2 + (-1) = 1$ and $w_2 = -2 + 2 = 0$.

(b) [2 pts] After how many steps will the perceptron algorithm converge? Write "never" if it will never converge.

Note: one steps means processing one point. Points are processed in order and then repeated, until convergence.

The data is not seperable, so it will never converge.

(c) Instead of the standard perceptron algorithm, we decide to treat the perceptron as a single node neural network and update the weights using gradient descent on the loss function.

The loss function for one data point is $Loss(y, y^*) = (y - y^*)^2$, where y^* is the training label for a given point and y is the output of our single node network for that point.

(i) [3 pts] 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?

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

(ii) [2 pts] For this question, the specific activation function that we will use is:

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

Given the following gradient descent equation to update the weights given a single data point. With initial weights of $w_1 = 2$ and $w_2 = -2$, what are the updated weights after processing the first point?

Gradient descent update equation: $w_i = w_i - \alpha \frac{\partial Loss}{\partial w_1}$

Because the gradient of g is zero, the weights will stay $w_1 = 2$ and $w_2 = -2$.

(iii) [1 pt] 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.

THIS PAGE IS INTENTIONALLY LEFT BLANK