CS 188 Fall 2014 — Introduction to Artificial Intelligence — Final

INSTRUCTIONS

- You have 2 hours 50 minutes.
- The exam is closed book, closed notes except a one-page crib sheet.
- Please use non-programmable calculators only. Laptops, phones, etc., may not be used. If you have one in your bag please switch it off.
- 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.
- Questions are not sequenced in order of difficulty. Make sure to look ahead if stuck on a particular question.

Last Name	
First Name	
SID	
Email	
First and last name of person on left	
First and last name of person on right	
All the work on this exam is my own. **(please sign)**	

For staff use only

Q. 1	Q. 2	Q. 3	Q. 4	Q. 5	Q. 6	Q. 7	Q. 8	Q. 9	Q. 10	Total
/10	/10	/ 11	/ 6	/12	/12	/11	/11	/7	/10	/100

1. (10 points) Agents, Search, CSPs

Please CIRCLE either *True* OR *False* for the following questions.

(a) (1 pt) *True*/ *False*: A hill-climbing algorithm that never visits states with lower value (or higher cost) is guaranteed to find the optimal solution if given enough time to find a solution.

Such an algorithm will reach a local optimum and stop (or wander on a plateau).

(b) (1 pt) *True*/ *False*: Suppose the temperature schedule for simulated annealing is set to be constant up to time N and zero thereafter. For any finite problem, we can set N large enough so that the algorithm is returns an optimal solution with probability 1.

If the temperature is fixed the algorithm always has a certain amount of randomness, so when it stops there is no guarantee it will be in an optimal state.

(c) (1 pt) *True*/ *False*: For any local-search problem, hill-climbing will return a global optimum if the algorithm is run starting at any state that is a neighbor of a neighbor of the state corresponding to the global optimum.

The intervening neighbor could be a local minimum and the current state is on another slope leading to a local maximum.

(d) (1 pt) *True* / *False*: Given a fully observable task environment where there is more than one action at every particular state, there exists a simple reflex agent that can take a different action upon revisiting the same state.

The question was not worded precisely enough, since it's not clear if randomized actions are allowed. Technically they are, so the answer is True, but we allowed both.

(e) (1 pt) *True*/ *False*: Running forward checking after the assignment of a variable in backtracking search will ensure that every variable is arc consistent with every other variable.

Forward checking only checks variables that are connected to the variable being assigned.

(f) (1 pt) *True* / *False*: Suppose we use the min-conflicts algorithm to try to solve a CSP. It is possible the algorithm will not terminate even if the CSP has a solution.

Since min-conflicts takes an integer number of steps as input, it always terminates; but it is true that it can get stuck in local minima where it loops through the same set of states, so we allowed both answers.

(g) (1 pt) *True*/ *False*: In a CSP constraint graph, a link (edge) between any two variables implies that those two variables may not take on the same values.

A link represents any constraint, including, e.g., equality!

(h) **(1 pt)** *True*/*False:* A model-based reflex agent maintains some internal state that it updates as the agent collects more information through percepts.

The internal state represents in some form what the agent knows about the external state of the world given the percepts to date, which is a good idea for partially observable environments.

(i) **(1 pt)** *True*/*False*: For any particular CSP constraint graph, there exists exactly one minimal cutset.

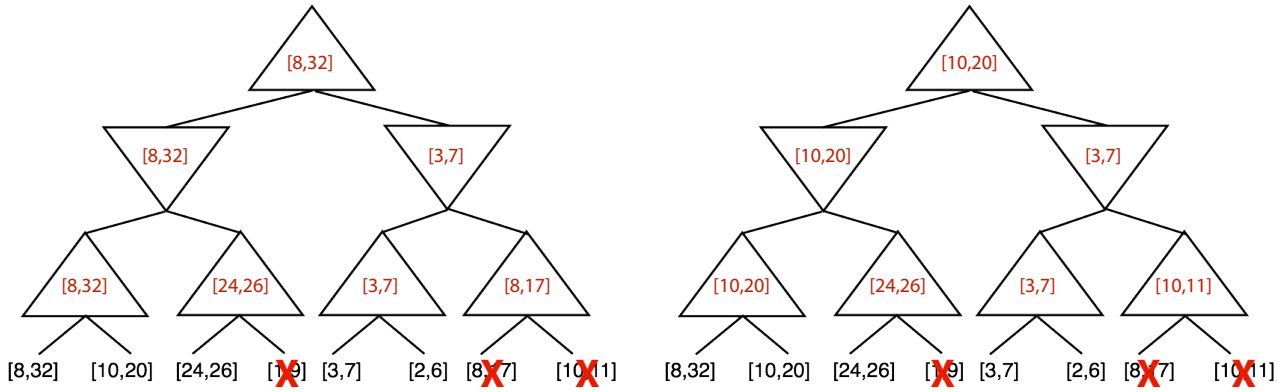
There may be several minimal cutsets of equal size, particularly if the graph has symmetries.

(j) **(1 pt)** *True*/*False:* Suppose agent A1 is rational and agent A2 is irrational. There exists a task environment where A2's actual score will be greater than A1's actual score.

Rational decisions are defined by expected outcomes, not actual outcomes. A1 might just be unlucky.

2. (10 points) Game tree search on intervals

Consider a modification of a game tree where, rather than numbers, leaf nodes are now intervals, given by [lower, upper], and where the true value is somewhere inside the interval. Here is an example (with a spare copy for later use):



For parts (a) and (b), suppose the min and max players are both trying to optimize $avg(lower, upper)$; that is, max is trying to maximize the average value of the interval chosen and min is trying to minimize it.

(a) (2 pt) In each node of the tree above, fill in the interval defining what the agent knows about the value of that node.

(b) (2 pt) In the tree above, cross out the leaves that would be pruned by alpha-beta pruning based on average values.

For parts (c) and (d), assume that $upper(interval)$ is the upper bound of current interval we are looking at, $lower(interval)$ is the lower bound, α is the best value for max seen so far and β the best value for min.

(c) (3 pt) Suppose that min and max are both pessimistic: both try to optimize their worst possible outcomes. Is pruning possible in this scenario? If so, write the pruning condition which, if true, would lead to maximal pruning while evaluating a max node. If not, explain why not.

At a max node, choose the child with the highest lower bound; at a min node, choose the node with the lowest upper bound. The right-hand tree above shows that pruning is possible. Once the third grandchild of the root is known to be [3,7], its parent, a min node, will choose a child with an upper bound of 7 or less. The lower bound of this node has to be 7 or less, by definition, so at the root max will always choose the left branch; thus the fourth grandchild and its two leaves can be pruned.

(d) (3 pt) Suppose that min and max are both optimistic: both try to optimize their best possible outcomes. Is pruning possible in this scenario? If so, write the pruning condition which, if true, would lead to maximal pruning while evaluating a max node. If not, explain why not.

In this case pruning is not possible. Because the players are optimistic, both will think that a leaf node with interval $[-\infty, \infty]$ is ideal, so no leaf can be pruned in case it might be such a node.

3. (11 points) Propositional Logic

(a) (3 pt) *True*/*False*: $A \wedge B \implies C$ entails $(A \implies C) \vee (B \implies C)$

A truth table shows that they are actually equivalent. Alternatively, one can argue that the LHS is false in a single case: TTF; and for the RHS to be false, both implications have to be false, again just the TTF case. Finally, one could convert to clause form, getting $\neg A \vee \neg B \vee C$ in both cases.

(b) (2 pt) *True*/*False*: Every nonempty propositional clause, by itself, is satisfiable.

Obviously every literal is satisfiable; and disjoining means unioning the satisfying models. Many people thought that clauses were conjunctions of literals; a conjunction such as $A \wedge \neg A$ is *two* clauses!

(c) (2 pt) *True*/*False*: Suppose a propositional clause contains three literals, each mentioning a different variable; then the clause is satisfied in exactly 7 of the 8 possible models for the three variables.

The clause can be false only if all three literals are false, which fixes a single model of the 8

(d) (4 pt) Now explain why the following set of clauses is unsatisfiable *without* using truth tables:

$A \vee B \vee C$	$\neg A \vee B \vee C$
$A \vee B \vee \neg C$	$\neg A \vee B \vee \neg C$
$A \vee \neg B \vee C$	$\neg A \vee \neg B \vee C$
$A \vee \neg B \vee \neg C$	$\neg A \vee \neg B \vee \neg C$

Since each clause is false in exactly one model, and all clauses are different and therefore false in different models, all 8 models are ruled out hence the conjunction is unsatisfiable. There are other elegant proofs: for example, suppose there is a satisfying model $[A = a, B = b, C = c]$; that falsifies the clause $\neg a \vee \neg b \vee \neg c$, which must be a member of the set, hence no such model can exist. Some more elaborate proofs based on splitting on variable values amount to truth tables in disguise. One approach definitely won't work: claiming that, say, $A \vee B \vee C$ and $\neg A \vee \neg B \vee \neg C$ are contradictory; in fact, there are 6 models where both are true!

4. (6 points) Bayes Nets

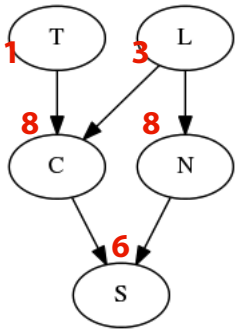
You're interested in predicting whether boba stores in Berkeley will be successful or go out of business. You decide to solve this problem using probablistic inference over a model with the following variables:

- S, whether or not the store will be successful. May take two values: $true$ or $false$.
- T, whether or not the store makes boba with real tea leaves. May take two values: $true$ or $false$.
- N, the number of other boba stores nearby. May take three values: 0, 1, or > 1.
- L, location of the store relative to campus. May take four values: $north$, $south$, $east$, or $west$.
- C, the cost of the boba. May take two values: $cheap$ or $expensive$.

(a) (1 pt) Your first idea for a probability model is a joint probability table over all of the variables. How many free parameters would this joint probability table have (after taking into account sum-to-1 constraints)?

$2 * 2 * 3 * 4 * 2 - 1 = 95$

(b) (2 pt) You decide this is too many parameters. To fix this, you decide to model the problem with the following Bayes net instead:



For this network, write next to each node the total number of free parameters in its conditional probability table, taking into account sum-to-1 constraints.

(c) (3 pt) A new boba store is opening up! You don't know how expensive it will be, but you have heard that it's going to use real tea, it will be North of campus, and there will be one other boba store nearby. According to your model, what is the probability it will be successful? (Just write out the expression *in terms of conditional probabilities from the model*; no need to simplify, though you may include a normalizing constant.)

The question asks for $P(S \mid T = true, L = north, N = 1)$. The most straightforward way to calculate this is to compute the full joint distribution with the given variables and then sum out the unknown variable, C. Recall that to calculate the full joint distribution in a Bayes net, we only need the conditional probabilities in our conditional probability table. The full calculation is: $P(S \mid T = true, L = north, N = 1)$
$= \alpha P(S, T = true, L = north, N = 1)$
$= \alpha \sum_p P(C = c, S, T = true, L = north, N = 1)$
$= \alpha \sum_p P(T = true) P(L = north) P(C = c \mid T = true, L = north) P(N = 1 \mid L = north) P(S \mid N = 1, C = c)$

5. (12 points) Bayes Nets

A *k-zigzag* network has k Boolean root variables and $k+1$ Boolean leaf variables, where root i is connected to leaves i and $i+1$. Here is an example for $k=3$, where each D_i represents a Boolean disease variable and each S_j is a Boolean symptom variable:

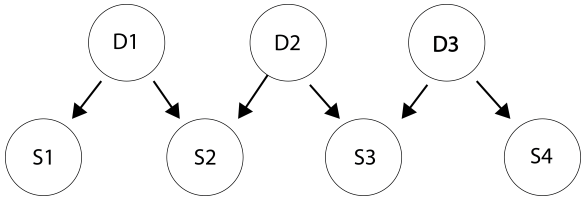


Figure 1: A 3-zigzag Bayes net.

(a) (1 pt) Does having symptom S_4 affect the probability of disease D_1? Why or why not?

No. D_1 is independent of its nondescendants (including S_4) given its parents (empty set), i.e., D_1 is absolutely independent of S_4.

(b) (3 pt) *Using only conditional probabilities from the model*, express the probability of having symptom S_1 but not symptom S_2, given disease D_1.

$P(s_1 \wedge \neg s_2 \mid D_1) = P(s_1 \mid D_1)P(\neg s_2 \mid D_1) = P(s_1 \mid D_1) \sum_{d_2} P(d_2)P(\neg s_2 \mid D_1, d_2)$

(c) (1 pt) *True*/*False*: Exact inference in a k-zigzag net can be done in time $O(k)$.

The network is a polytree.

(d) (1 pt) Suppose the values of all the symptom variables have been observed, and you would like to do Gibbs sampling on the disease variables (i.e., sample each variable given its Markov blanket). What is the largest number of non-evidence variables that have to be considered when sampling any particular disease variable? Explain your answer.

The Markov blanket of a disease includes its parents, children, children's other parents; for a disease variable, the non-evidence variables in the Markov blanket are just the two neighboring disease variables.

(e) **(2 pt)** Suppose $k = 50$. You would like to run Gibbs sampling for 1 million samples. Is it a good idea to precompute all the sampling distributions, so that when generating each individual sample no arithmetic operations are needed? Explain.

Yes. Precomputation requires space for 200 numbers and saves tens of millions of computations.

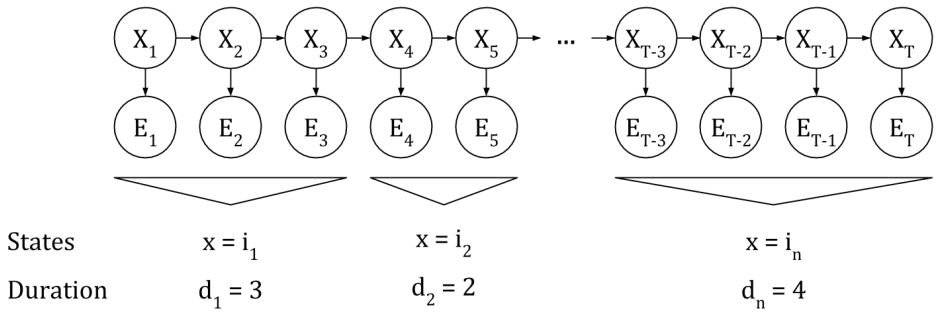
(f) **(2 pt)** A k-zigzag++ network is a k-zigzag network with two extra variables: one is a root connected to all the leaves and one is a leaf to which all the roots are connected. You would like to run Gibbs sampling for 1 million samples in a 50-zigzag++ network. Is it a good idea to precompute all the sampling distributions? Explain.

No. The Markov blanket of every disease now includes all other diseases, and a table of 2^{50} numbers is impossible to compute or store.

(g) **(2 pt)** Let us assume that in every case the values of all symptom variables are observed, which means that we can consider training a neural net to predict diseases from symptoms rather than using a Bayes net. Mary from MIT claims that an equivalent neural net can be obtained by reversing the arrows in the figure. She argues that because D_1 doesn't affect S_3 and S_4, they are not needed for predicting D_1. Is Mary right? Explain.

No. While D_1 is absolutely independent of S_3, it is not independence given S_1 and S_2. The reason is that the value of S_3 affects the probability of D_2; D_2 in turn can "explain away" S_2, thereby changing the probability of D_1. The same argument applies, by extension, to all symptoms.

6. (12 points) Hidden Semi-Markov Models



A Hidden Semi-Markov Model is an extention of HMMs where the system is allowed reside in a given state for some *duration* before a transition occurs to the next state (which may be the same state as before). In a regular HMM the durations are all 1. The HSMM formulation is as follows:

- Chain of state variables $X_1, \ldots, X_T$; X_t is the state of the system at time t; actual states are labeled by integers.
- Chain of evidence variables $E_1, \ldots, E_T$; E_t is the observation at time t; actual observations are labeled by integers.
- State sequence durations $D_1, \ldots, D_M$, which take on positive integer values; D_m is the number of time steps the system resides in a state i_m between transition $m-1$ and transition m.

It is helpful also to define the variables S_m and F_m, the start and finish times for period m, where $S_1 = 1$, $F_1 = D_1$, $S_2 = D_1 + 1$, $F_2 = D_1 + D_2$, etc. Then we can write $X_{S_m:F_m} = i_m$ as shorthand for $X_{S_m} = i_m, X_{S_m+1} = i_m, \ldots, X_{F_m} = i_m$.

Suppose that a transition occurs at time t which is the end of residence period m with duration D_m. The new state depends on the old state and its duration, while the new duration depends only on the new state:

$$P(X_{t+1}, D_{m+1} \mid X_t, D_m) = P(X_{t+1}|X_t, d_m)P(D_{m+1}|X_{t+1}) \,. \tag{1}$$

During a residence period, of course, the state and duration remain constant. In all cases the evidence at t depends only on the state at t. Thus, the entire probability model can be built from the following probability tables:

(A) Initial distribution of X: $P(X_1)$;

(B) Sensor model: $P(E|X)$;

(C) Transition model: $P(X_{S_{m+1}} = i|X_{F_m} = j, D_m = d)$;

(D) Duration model: $P(D|X)$.

(a) (2 pt) Which of the following expressions are correct for $P_1 = P(D_1, X_1, \ldots, X_{D_1}, E_1, \ldots, E_{D_1})$, the probability that the first period has duration D_1 and the state sequence is $X_1, \ldots, X_{D_1}$ and the observation sequence is $E_1, \ldots, E_{D_1}$, assuming $X_1 = X_2 = \ldots = X_{D_1}$?

⊗ $P(X_1)P(D_1 \mid X_1)\prod_{t=1}^{D_1} P(E_t \mid X_t)$

○ $P(X_1)\prod_{t=1}^{D_1} P(E_t \mid X_t)P(D_t \mid X_t)$

○ $P(X_1)\prod_{t=1}^{D_1} P(E_t \mid X_t)$

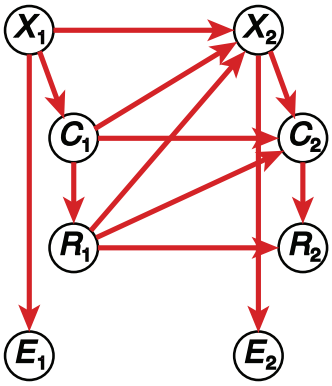
⊗ $P(X_1, E_1)P(D_1 \mid X_1)\prod_{t=2}^{D_1} P(E_t \mid X_t)$

(b) (2 pt) The following expression is supposed to be the probability that the first M periods have durations $D_1, \ldots, D_M$ and the state sequence is $X_1, \ldots, X_{F_M}$ and the observation sequence is $E_1, \ldots, E_{F_M}$, but one term is missing. What is that term and where does it go?

$$P_1 \prod_{m=2}^{M} P(X_{S_m} \mid X_{F_{m-1}}, D_{m-1}) \prod_{t=S_m}^{F_m} P(E_t \mid X_t)$$

A term is needed for D_m, i.e., $P(D_m \mid X_{S_m})$. It goes in the scope of the product over m.

(c) **(3 pt)** Suppose we want to describe an HSMM as an ordinary dynamic Bayes net that yields an exactly equivalent distribution over state and observation sequences. In addition to X_t and E_t, we will handle durations using C_t, the duration of the current period, and R_t, the amount of time left in the current period. In the figure below, add an appropriate (and minimal) set of links to define the model.



(d) **(2 pt)** Describe precisely the conditional distribution for C_2 in your model.

If $R_1 = 1$ then C_2 depends on X_2 according to the model $P(D \mid X)$; else $C_2 = C_1$.
Some students interpreted C_t as "the time spent so far in this period," which is not quite what the question states but still results in a feasible (Markovian) model so we allowed it.

(e) **(3 pt)** The forward equation for an ordinary HMM is given by

$$P(X_{t+1} \mid e_{1:t+1}) = \alpha\, P(e_{t+1} \mid X_{t+1}) \sum_{x_t} P(X_{t+1} \mid x_t) P(x_t \mid e_{1:t}) \,.$$

Write the forward equation for this dynamic Bayes net; i.e., write an equation for $P(X_{t+1}, C_{t+1}, R_{t+1} \mid e_{1:t+1})$ in terms of $P(X_t, C_t, R_t \mid e_{1:t})$ and quantities given in the model.

The key is to realize that the state of the process is now constituted by X, C, R instead of just X but otherwise the equation has the same structure. And any distribution for X becomes a joint distribution for X, C, R, expressed as a product of the appropriate conditionals:

$$\alpha\, P(e_{t+1} \mid X_{t+1}) \sum_{x_t, c_t, r_t} P(X_{t+1} \mid x_t, c_t, r_t) P(C_{t+1} \mid X_{t+1}, c_t, r_t) P(R_{t+1} \mid C_{t+1}, r_t) P(x_t, c_t, r_t \mid e_{1:t})$$

7. (11 points) MDPs

(a) (3 pt) The MDP of life has two actions $Party$ and $Study$ in the start state. After that, there is only one choice per state. If the agent parties, it receives a reward of +10 followed by an infinite sequence of rewards of –1. If the agent studies, it receives a reward of –10 followed by an infinite sequence of rewards of +1. For what value of the discount factor γ is the agent indifferent between partying and studying? (Recall that $1+x+x^2+\cdots=1/(1-x)$.)

At indifference, $10-\gamma/(1-\gamma)=-10+\gamma/(1-\gamma)$, so $\gamma=10/11$.

The GSIs of 188 have started a game company. Their first game, GhostBlushers, starts with two ghosts, Blinky and Pinky. On each turn, the player clicks on a ghost. Clicking on any Blinky (action a_B) produces a new Blinky with probability $(2b+p)/2(b+p)$ and a new Pinky otherwise, whereas clicking on any Pinky (action a_P) produces a new Pinky with probability $(b+2p)/2(b+p)$ and a new Blinky otherwise, where b and p are the numbers of Blinkies and Pinkies in the current state. There is a reward of +1 for producing an offspring of the same kind. The game ends after T steps.

(b) (4 pt) Define the game as an MDP, with a minimal state space. (No need to describe both actions, just a_B.)

States: pairs (b,p). (Time step is not needed as it's encoded in $b+p$.) Actions: a_B, a_P.
Transition model: $T((b,p),a_B,(b+1,p))=(2b+p)/2(b+p)$; $T((b,p),a_B,(b,p+1))=p/2(b+p)$; 0 otherwise.
Reward function: $T((b,p),a_B,(b+1,p))=1$; 0 otherwise.

(c) (1 pt) The total number of states in the MDP is ◯ $O(T)$ ⊗ $O(T^2)$ ◯ $O(T^3)$ ◯ None of these

It's just an arithmetic series up to the last time step which has $O(T)$ states.

(d) (2 pt) Explain why value iteration applied to this problem converges exactly after $O(T)$ iterations.

The key property of this MDP is that the state space is acyclic. After iteration t, states t steps from the end have exact values and no changes take place after that. After T iterations the root is correct. This is in direct contrast to the typical process for a model with cycles in the state space, where convergence is geometric and continues for ever. Notice, on the other hand, that each iteration of VI updates ALL the states; it does not "start from the end and work back."

(e) (1 pt) Assuming that Bellman backups at each state consider only successor states that have nonzero probability, the total runtime of value iteration on this problem is
◯ $O(T^2)$ ⊗ $O(T^3)$ ◯ $O(T^4)$ ◯ $O(2^T)$ ◯ None of these

For each state the backup work is constant (only two successors); T iterations times $O(T^2)$ states is $O(T^3)$.

8. (11 points) Approximate Q-Learning

The UC system is experimenting with new transportation options that will help students move between all 10 campuses.

Each possible **state** for a student is a tuple of $location$ (one of 10 campuses) and $mood$ ($happy$ or $upset$).

Available **actions**: Bus, $Taxi$, $Motorcycle$. Actions have the following properties:

Action	Makes Stops	Max. Passengers	Has Wi-Fi
Bus	Yes	50	Yes
Taxi	No	4	No
Motorcycle	No	1	No

We will use a linear, feature-based approximation of the Q-values:

$$\text{Linear value function: } Q_{\mathbf{w}}(s,a) = \sum_{i=0}^{3} f_i(s,a) w_i$$

Features	Initial Weights
$f_0(state,\ action) = 1$ (this is a bias feature that is always 1)	$w_0 = 1$
$f_1(state,\ action) = \begin{cases} 1 & \text{if } action \text{ makes stops} \\ 0 & \text{otherwise} \end{cases}$	$w_1 = 2.5$
$f_2(state,\ action) = \begin{cases} 1 & \text{if } action \text{ has max. passengers } > 1 \\ 0 & \text{otherwise} \end{cases}$	$w_2 = 0.5$
$f_3(state,\ action) = \begin{cases} 1 & \text{if } action \text{ has Wi-Fi} \\ 0 & \text{otherwise} \end{cases}$	$w_3 = 1$

Remember that the approximate Q-values are a function of the weights, so make sure to use the chain rule as follows whenever updating the weights:

$$w_i \leftarrow w_i + \alpha \left[r + \gamma \max_{a'} Q_{\mathbf{w}}(s',a') - Q_{\mathbf{w}}(s,a) \right] \frac{\partial}{\partial w_i} Q_{\mathbf{w}}(s,a)$$

(a) (3 pt) Calculate the following initial Q values given the initial weights above:

$Q(\ (UCLA, upset),\ Bus\ )$	$1+2.5+0.5+1 = 5$
$Q(\ (UCLA, upset),\ Taxi\ )$	$1+0+0.5+0 = 1.5$
$Q(\ (UCLA, upset),\ Motorcycle\ )$	$1+0+0+0 = 1$

(b) (2 pt) The initial Q-values for the state $(UCLA,\ upset)$ happen to be equal to the corresponding initial Q-values for the state $(Berkeley,\ happy)$. As you update the weights, will these values always be the equal to each other? In other words, will $Q_{\mathbf{w}}(\ (UCLA, upset),\ a\ ) = Q_{\mathbf{w}}(\ (Berkeley, happy),\ a\ )$ given any action a and vector of weights $\mathbf{w}$?

⊗ Yes ◯ No

Why / Why not? (in just one sentence)

They are always equal because our features are not dependent on the state.

(c) **(2 pt)** Exploration / exploitation: Given the Q-values for the state $(UCLA,\ upset)$ you calculated on the previous page, what is the probability that each action could be chosen when using ϵ-greedy exploration (assuming any random movements are chosen uniformly from all actions)?

Action	Probability (in terms of ϵ)
Bus	$(1-\epsilon)+\frac{\epsilon}{3}=1-\frac{2\epsilon}{3}$
Taxi	$\frac{\epsilon}{3}$
Motorcycle	$\frac{\epsilon}{3}$

(d) **(2 pt)** Given a sample with start state = $(Berkeley,\ happy)$, action= *Taxi*, successor state = $(UCLA,\ upset)$, and reward = -7: Using a learning rate of $\alpha = 0.5$ and discount of $\gamma = 0.5$, update each of the weights.

Difference $= -7 + 0.5 * max(5, 1.5, 1) - 1.5 = -6$ $\frac{\partial}{\partial w_i} Q_{\mathbf{w}}(s,a) = f_i$ $w_0 = 1 + 0.5 * (-6) * 1 = -2$
$w_1 = 2.5 + 0.5 * (-6) * 0 = 2.5$
$w_2 = 0.5 + 0.5 * (-6) * 1 = -2.5$
$w_3 = 1 + 0.5 * (-6) * 0 = 1$

(e) **(2 pt)** 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.

9. (7 points) Decision Trees

After taking CS188, Alice wants to construct a decision tree classifier to predict flight delays. She has collected the data for a few months. A summary of the data is provided in table below. Read it **carefully**.

Feature	Feature value = Yes		Feature value = No	
	# Delayed	# not Delayed	# Delayed	# not Delayed
Rain	30	10	10	30
Wind	25	15	15	25
Summer	5	35	35	5
Winter	20	10	20	20
Day	20	20	20	20
Night	15	10	25	30

(a) **(1 pt)** Alice thinks she was tired when filling in the last column and may have made a mistake. Please correct it in the table for her.

The 4th number is wrong, as the row total should be 80 no matter what feature is used to classify the data. We should change 20 to 30.

(b) **(2 pt)** Which of " Rain" and "Day" has the higher information gain for predicting delay? Explain without numerical calculations.

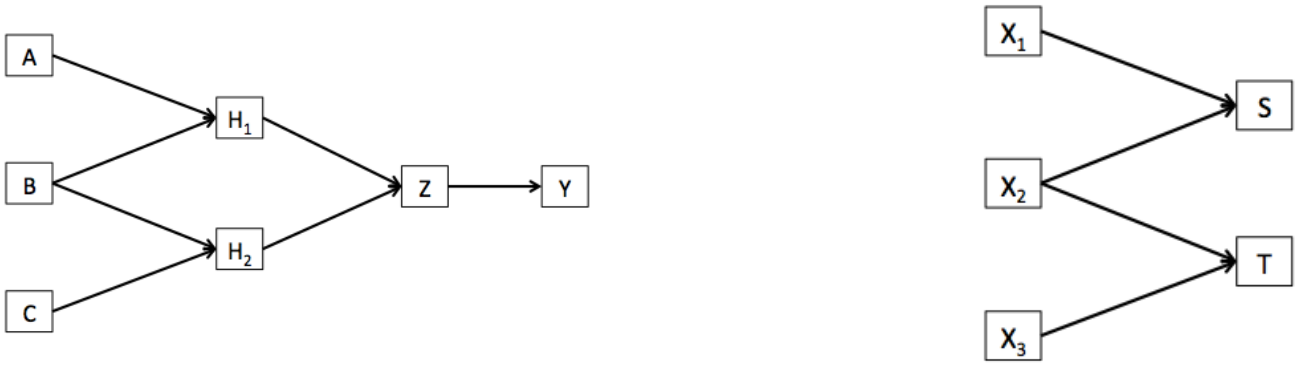
Day is useless: the subsets have the same delay proportions as the original set; Rain is better as the proportions are different from the prior.

(c) **(2 pt)** Bob the Stanford student says **among all six features**, "Rain" should be at the root of the decision tree. Do you agree with him? If not, which feature should be at the root? Explain why **qualitatively**.

No. "Rain" is not the best feature. Note that the distribution of "Delay" conditioned on "Summer" is more skewed than conditioned on all other features, which implies the maximum information gain can be achieved when "Summer" is chosen as the root.

(d) **(2 pt)** Based **only** on the table, can you determine which feature should be on the second level (the level just beneath the root) of the decision tree? If yes, which one is it? Support your answer **qualitatively**.

No, it is impossible to tell. To determine the second level feature we need to know the breakdown of delays for the other features given the value of "Summer". Since we only have summaries for delay given one feature in this table, there is not enough information to determine the feature that would lead to the highest information gain AFTER we used "Summer".



(a) (b)

Figure 2: (a) A neural network with two hidden layers; (b) a multiclass perceptron.

10. (10 points) Neural Networks

(a) (2 pt) Consider the (non-fully connected) neural network in Figure 2(a), where each node is assumed to have a bias input of constant value 1 that is not shown. Assume the inputs to A, B, C and the output Y are boolean (1 or 0). Could this network correctly classify each of the following logical operations? Briefly justify.

$(A \wedge B \wedge C) \vee (\neg A \wedge B \wedge C)$

Yes, this is equivalent to $B \wedge C$ which can be classified using H_2.

A XOR B

No, because (1) H_1 can only separate the inputs into two sets of size two, each with one positive and one negative example, or into a set of size three and a singleton. (2) H_2 can only test on B, which is not enough to separate the sets from (1) correctly.

Now assume we have the (non-fully connected) *multi-class* perceptron shown in Figure 2(b); here we assume no bias inputs. Such a perceptron returns as a prediction the label of the output node with the highest output value.

(b) (4 pt) Suppose we start with the weights [1, 1] for output S and the weights [0, 1] for output T and a learning rate of 1. Given the following example run one iteration of weight update.

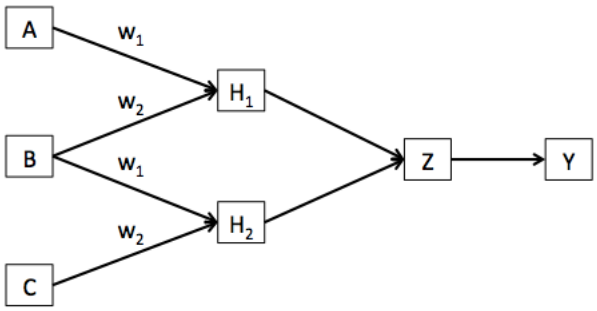
$X_1 = -1, X_2 = 0.5, X_3 = 1, label = S$

Here we do normal perceptron weight updates, the only difference is that S only has weights for X_1 and X_2 while T only has weights for X_2 and X_3.
1) Feed Forward
$output_S = 1 * -1 + 1 * 0.5 = -0.5$
$output_T = 0 * 0.5 + 1 * 1 = 1$
So we predict T which is incorrect so we update the weights.
$W_S = [1, 1] + [-1, 0.5] = [0, 1.5]$
$W_T = [0, 1] - [0.5, 1] = [-0.5, 0]$

(c) **(2 pt)** Independent of your answer to the previous problem, assume you arrived at the conclusion that the weights for S are [0, 2] and the weights for T are [-1, 0]. What is the condition on the inputs such that this network chooses S rather than T? Express the condition as simply as possible.

The condition for choosing S over T is $(0 \cdot X_1) + (2 \cdot X_2) > (-1 \cdot X_2) + (0 \cdot X_3)$ which simplifies to $X_2 > 0$.

Many neural networks used in practice have weights that are shared across multiple connections. In the example below, there are two weights in the first layer, w_1 and w_2, each of which appears twice.



(d) **(1 pt)** What effect would you expect weight sharing to have on the training error?
◯ It should go down ⊗ It should go up ◯ Cannot tell given this information

This reduces the ability of the model to fit the data well.

(e) **(1 pt)** What effect would you expect weight sharing to have on the test error?
◯ It should go down ◯ It should go up ⊗ Cannot tell given this information

It could go either way, depending on the tradeoff between reduced expressive power and whether the true function agrees or nearly agrees with the constrained function space.