CS 188 Spring 2014 Introduction to Artificial Intelligence Final

- You have approximately 2 hours and 50 minutes.
- The exam is closed book, closed notes except your two-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	

First and last name of student to your left	
First and last name of student to your right	

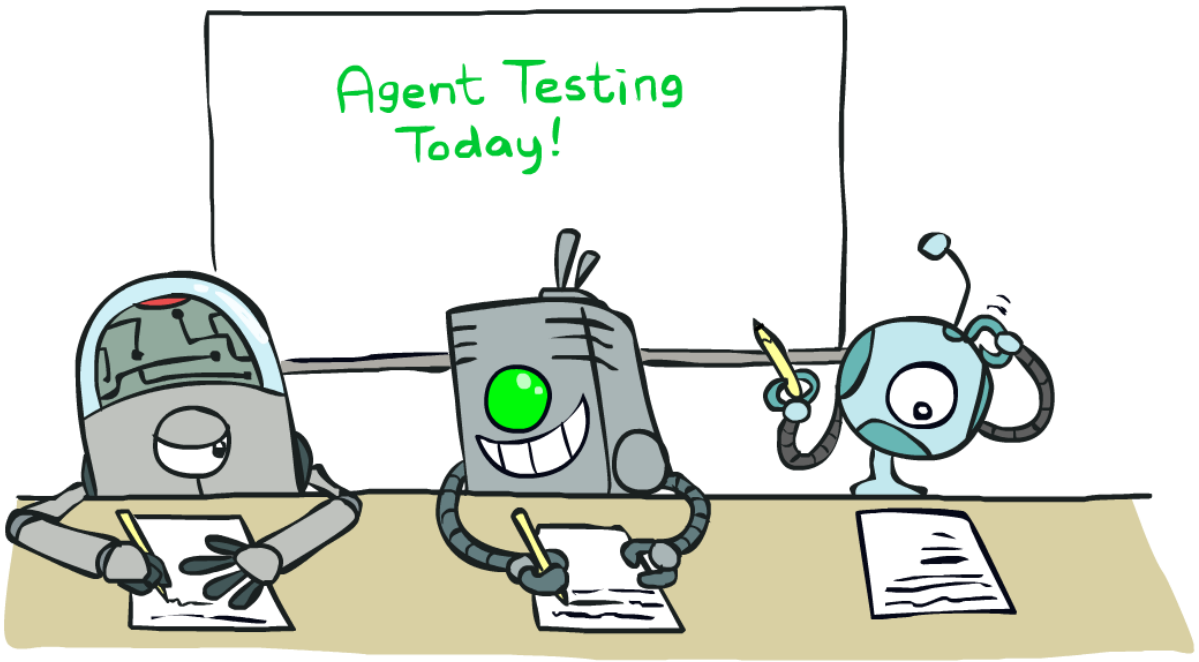
For staff use only:

Q1.	Agent Testing Today!	/1
Q2.	Short questions	/16
Q3.	Finding the Best k Paths	/12
Q4.	Probability and Bayes Nets	/17
Q5.	Kernels and Feature Transforms	/6
Q6.	Stopping Strategy	/9
Q7.	Inference	/13
Q8.	Q-Learning Strikes Back	/8
Q9.	Adversarial VPI	/9
Q10.	Bayes Net CSPs	/9
	Total	/100

THIS PAGE IS INTENTIONALLY LEFT BLANK

Q1. [1 pt] Agent Testing Today!

It's testing time! Not only for you, but for our CS188 robots as well! Circle your favorite robot below.



Any answer was acceptable.

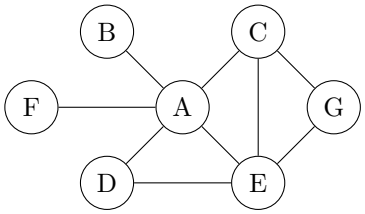
Q2. [16 pts] Short questions

(a) [4 pts] Search. If $f(s)$, $g(s)$ and $h(s)$ are all admissible heuristics then which of the following are also guaranteed to be admissible heuristics:

- ○ $f(s) + g(s) + h(s)$
- ● $f(s)/6 + g(s)/3 + h(s)/2$
- ● $\min(f(s), g(s), h(s))$
- ● $\max(f(s), g(s), h(s))$
- ● $f(s)/3 + g(s)/3 + h(s)/3$
- ○ $f(s) * g(s) * h(s)$
- ● $\min(f(s), g(s) + h(s))$
- ○ $\max(f(s), g(s) + h(s))$

In order to guarantee that a function of admissible heuristics is still admissible, the expression must be less than or equal to the max of the heuristics. Sums and products do not satisfy these, so bubbles 1, 6, and 8 all immediately fail. Bubbles 3, 4, and 7 all work because the max of admissible heuristics is still admissible, as is the min of an admissible heuristic and anything else. Bubble 5 is the average of the heuristics, so it must be less than the max, and is thus admissible. Lastly, bubble 2 is a weighted average, and is thus also less than the max, and is thus admissible.

(b) CSPs. Consider solving the following CSP with backtracking search where we enforce consistency of all arcs before every value assignment. For each of the variable orderings below specify at which variables backtracking might occur. Recall that backtracking occurs at a variable X when after a value from the filtered domain of X has been assigned to the variable X the recursion returns to X without a solution and the next value from the filtered domain of X gets assigned. If enforcing arc consistency results in any empty domains then the ensuing value assignment doesn't happen and the algorithm backtracks.



(i) [1 pt] For ordering A, B, C, D, E, F, G the algorithm might backtrack at variable(s):

● A ● B ● C ○ D ○ E ○ F ○ G

(ii) [1 pt] For ordering G, A, B, C, D, E, F the algorithm might backtrack at variable(s):

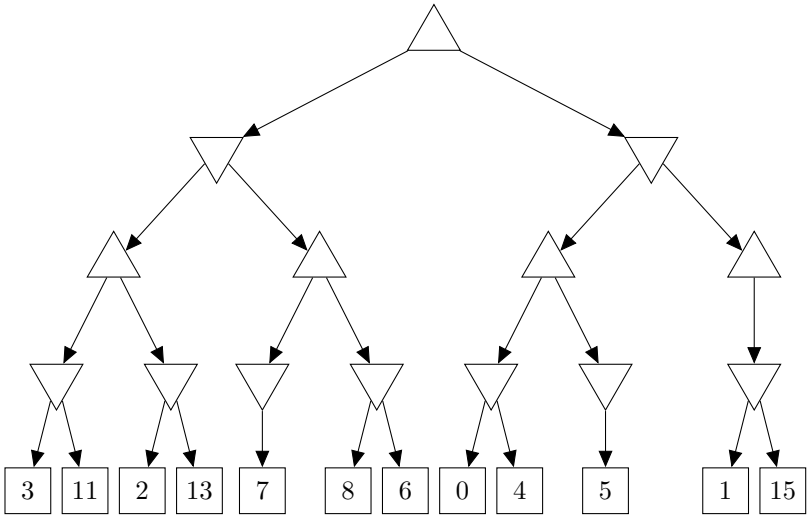
● A ○ B ○ C ○ D ○ E ○ F ● G

(iii) [1 pt] For ordering E, B, F, D, A, C, G the algorithm might backtrack at variable(s):

○ A ○ B ○ C ○ D ● E ○ F ○ G

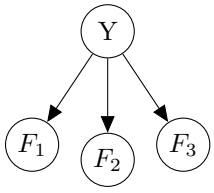
Any node can be backtracked on up until a cutset has been assigned. Note that B's values in the first part has no effect on the rest of the CSP after A has been assigned. However, because of the way that backtracking search is run, B would still be re-assigned before A if there was no consistent solution for a given value of A.

(c) [2 pts] Games. On the minimax game tree below cross out the branches removed by alpha-beta pruning assuming left to right traversal.



13 can be pruned because at that point, $\alpha = 3$, and $2 < 3$. The branch leading to the minimizer with 8 and 6 can be pruned because $\beta = 3$ at that point and $7 > 3$. The 4 can be pruned because $\alpha = 3$ (from the root node), and $0 < 3$. Lastly, 15 can be pruned because $alpha = 3$ at that point, and $1 < 3$.

(d) Naive Bayes. Consider training the Naive Bayes model shown on the left with the training data provided in the table on the right.



F_1	0	0	1	0	1	1	1	1
F_2	0	1	0	1	1	0	1	1
F_3	1	1	1	0	0	1	1	0
Y	0	0	0	1	1	0	0	1

(i) [1 pt] The maximum likelihood estimate of $P(F_1 = 1 \mid Y = 0)$ is $\frac{3}{5}$.
This is found by counting the samples. There are 5 samples where $Y = 0$, and $F_1 = 1$ in 3 of them.

(ii) [1 pt] Assuming Laplace smoothing with $k = 1$, the estimated $P(F_2 = 1 \mid Y = 1)$ is $\frac{4}{5}$.
Laplace smoothing involves counting every occurrence as having happened one more time than it did. There are 3 samples where $Y = 1$, and all of those have $F_2 = 1$. By adding another example where $(Y = 1, F_2 = 0)$ and $(Y = 1, F_2 = 1)$ results in $\frac{4}{5} = 0.8$.

(iii) [1 pt] Assuming Laplace smoothing with $k = 2$, the estimated $P(Y = 1)$ is $\frac{5}{12}$.
The unsmoothed estimate is $\frac{3}{8}$. If each value of Y is counted $k = 2$ extra times, this becomes $\frac{5}{12}$.

(e) Perceptron. We are training a Dual Perceptron for a three-class problem. There are four training examples x_1, x_2, x_3, x_4. The dual weights are currently:

$$\alpha_A =< -1, -1, -1, -1 > \text{ for class } A$$
$$\alpha_B =< -1, +1, +1, -1 > \text{ for class } B$$
$$\alpha_C =< +1, -1, -1, +1 > \text{ for class } C$$

Consider the fourth training example x_4 with correct label A and kernel evaluations:

$$K(x_1, x_4) = 1, \quad K(x_2, x_4) = 2, \quad K(x_3, x_4) = 1, \quad K(x_4, x_4) = 3$$

(i) [1 pt] Which classification label is predicted for the fourth training example x_4 with the current dual weights?

○ A ○ B ● C

The score for a class is equal to the dot product of the weight vector for that class and a vector containing the kernel values for the example. The scores are then: $A = 1*(-1)+2*(-1)+1*(-1)+3*(-1) = -7$, $B = 1*(-1)+2*1+1*1+3*(-1) = -1$, and $C = 1*1+2*(-1)+1*(-1)+3*1 = 1$. C has the highest score, and is the predicted label.

(ii) [3 pts] What are the dual weights after the update that incorporates the fourth training example?

$\alpha_A =$ $< -1, -1, -1, 0 >$

$\alpha_B =$ $< -1, +1, +1, -1 >$

$\alpha_C =$ $< +1, -1, -1, 0 >$

The update for the dual perceptron is simply to add 1 to the dimension corresponding to the example in the correct class's weight vector, and to subtract 1 from the corresponding dimension in the predicted weight vector. In this case, this means that α_B stays the same, the 4^{th} value in α_C is decremented by one, and the 4^{th} value in α_A is incremented.

Q3. [12 pts] Finding the Best k Paths

The optimal search algorithms we covered in CS188 find **one** optimal path (or return failure). We will explore how to find the best k (with $k \geq 1$) paths.

The following assumptions can be made regarding all the questions in this problem :

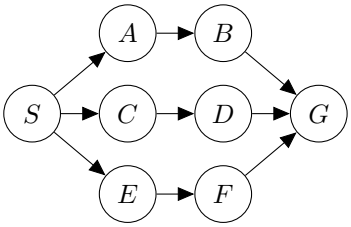
1. There are at least k paths from the start state to a goal state.
2. All edge costs are positive numbers (cost > 0).
3. No ties occur.

Consider a modified implementation of the **Uniform Cost Graph Search (UCS)** algorithm with the following **basic modifications:**

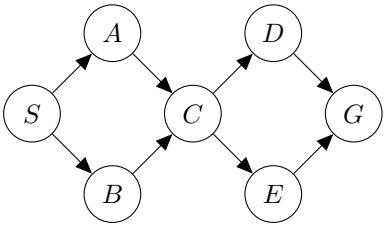
1. Maintain a list of successful paths to a goal state found so far. When a path from the start state to a goal state is found (i.e., whenever a path ending in the goal state is *popped from* the fringe), it is added to this list.
2. Exit the algorithm only if the length of the above list is k (success) or the fringe is empty (failure).

For each of the **additional modifications** on the next page, mark whether or not it would correctly give the top k unique least cost paths from the start state to a goal state. If a modification does not work, select **all** of the below graphs where there exists at least one set of edge weights and value for k (subject to the constraint that there are at least k paths through the graph) that would cause the algorithm to fail. Note that some modifications may even lead to failure for $k = 1$.

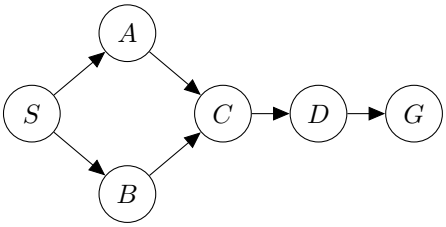
1)



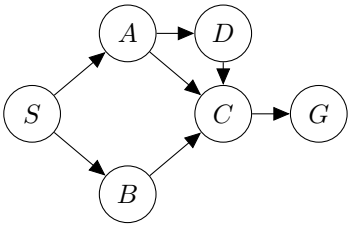
2)



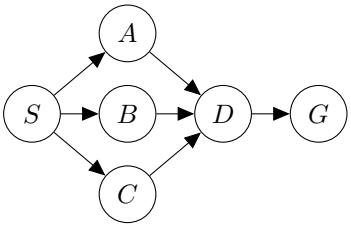
3)



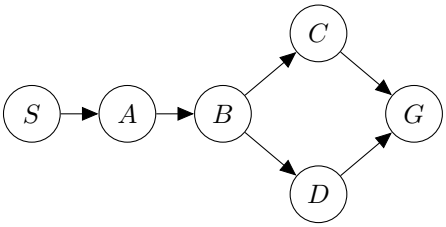
4)



5)



6)



(a) [2 pts] Everytime after a path is found, empty out the closed set.

○ Will work correctly ● Will not work correctly
Graphs for which this modification fails:

○ 1	● 2	● 3
● 4	● 5	○ 6

Whenever two paths intersect prior to the goal state, there is at least one set of weights such that this algorithm will fail for $k > 1$. This occurs in graphs 2, 3, 4, and 5

(b) [2 pts] For each state s, maintain a count `count_expand(s)` of how many times a path ending in state s has been popped from the fringe. Only add a state s to the closed set if `count_expand(s)` $= k$.

● Will work correctly ○ Will not work correctly
Graphs for which this modification fails:

○ 1	○ 2	○ 3
○ 4	○ 5	○ 6

The first k paths ending in a state are guaranteed to be the shortest k paths to that state, because of how uniform cost search works. Further, it is guaranteed that any path ending at a state, s, that is not one of the k shortest paths to s, cannot be part of one of the overall k shortest paths. This means that any path that stops being expanded because a state is on the closed set cannot be one of the overall k shortest paths.

(c) [2 pts] Do not use a closed set.

● Will work correctly ○ Will not work correctly
Graphs for which this modification fails:

○ 1	○ 2	○ 3
○ 4	○ 5	○ 6

This is running tree search, which will consider every possible path through the graph in order of increasing cost. Thus, it will find every path to the goal in order of increasing cost, which will correctly return the k shortest paths given the basic modifications.

(d) [2 pts] Do not use a closed set and, every time after a path is found, change the edge costs along that path by adding C, where C is a number that is at least as large as the sum of the costs of all edges in the graph. Also for each path on the fringe that contains i edges of the path that was just found, add $i \times C$ to the cost associated with this path on the fringe.

○ Will work correctly ● Will not work correctly
Graphs for which this modification fails:

○ 1	● 2	○ 3
● 4	○ 5	○ 6

This modification can fail on graphs in which only a strict subset of the paths share an edge. This applies to graphs 2 and 4. Note that while some of the other graphs share edges, because the edges are common to all paths, changing the value of those edges does not change the order in which the paths are expanded, and thus does not cause the search to fail.

(e) [2 pts] Do not use a closed set and, for each state s, maintain a count `count_fringe(s)` of how many times a node ending in state s has been added to the fringe. Only add a node ending in a state s to the fringe if `count_fringe(s)` $< k$.

○ Will work correctly ● Will not work correctly
Graphs for which this modification fails:

● 1 ● 2 ● 3
● 4 ● 5 ● 6

This modification can fail on any graph in which multiple paths intersect on any node, including the goal, which is the case for all of the graphs provided.

(f) [2 pts] No modification is made except for the Basic Modification described at the beginning of this question.

○ Will work correctly ● Will not work correctly

Graphs for which this modification fails:

○ 1 ● 2 ● 3
● 4 ● 5 ○ 6

This algorithm can fail on any graph that has paths intersecting on any node other then the goal, similar to (a). This occurs in graphs 2, 3, 4, and 5.

Q4. [17 pts] Probability and Bayes Nets

(a) [2 pts] Suppose $A \perp\!\!\!\perp B$. Determine the missing entries (x, y) of the joint distribution $P(A, B)$, where A and B take values in $\{0, 1\}$.

$$P(A = 0, B = 0) = 0.1$$
$$P(A = 0, B = 1) = 0.3$$
$$P(A = 1, B = 0) = x$$
$$P(A = 1, B = 1) = y$$

$x =$ ____.15____, $y =$ ____.45____

Note that $y/x = P(A = 1, B = 1)/P(A = 1, B = 0) = P(A = 0, B = 1)/P(A = 0, B = 0) = P(B = 1)/P(B = 0) = 3$ So $y = 3x$ and $x + y = 0.6$. Solve for x, y.

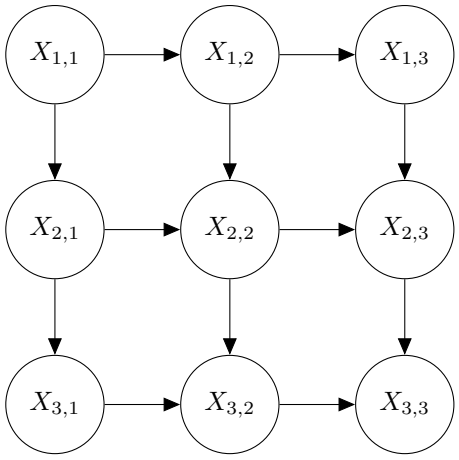
(b) [3 pts] Suppose $B \perp\!\!\!\perp C \mid A$. Determine the missing entries (x, y, z) of the joint distribution $P(A, B, C)$.

$$P(A = 0, B = 0, C = 0) = 0.01$$
$$P(A = 0, B = 0, C = 1) = 0.02$$
$$P(A = 0, B = 0, C = 1) = 0.03$$
$$P(A = 0, B = 1, C = 1) = x$$
$$P(A = 1, B = 0, C = 0) = 0.01$$
$$P(A = 1, B = 0, C = 1) = 0.1$$
$$P(A = 1, B = 1, C = 0) = y$$
$$P(A = 1, B = 1, C = 1) = z$$

$x =$ ____0.06____, $y =$ ____0.07____, $z =$ ____0.7____

First use the same observation about ratios as above to get that $x = 0.03 \cdot \frac{0.02}{0.01} = 0.06$. Then we have that $0.01 + 0.02 + 0.03 + 0.06 + 0.01 + 0.1 + y + z = 1$ so $y + z = 0.77$. The same observation about ratios gives $z/y = 10$. Solving, we get $y = 0.07, z = 0.7$.

(c) [3 pts] For this question consider the Bayes' Net below with 9 variables.



Which random variables are independent of $X_{3,1}$? (Leave blank if the answer is none.)

○ $X_{1,1}$ ○ $X_{1,2}$ ○ $X_{1,3}$ ○ $X_{2,1}$ ○ $X_{2,2}$ ○ $X_{2,3}$ ○ $X_{3,2}$ ○ $X_{3,3}$

There is at least one active path between $X_{3,1}$ and every other node.

Which random variables are independent of $X_{3,1}$ *given* $X_{1,1}$? (Leave blank if the answer is none.)

● $X_{1,2}$ ● $X_{1,3}$ ○ $X_{2,1}$ ○ $X_{2,2}$ ○ $X_{2,3}$ ○ $X_{3,2}$ ○ $X_{3,3}$

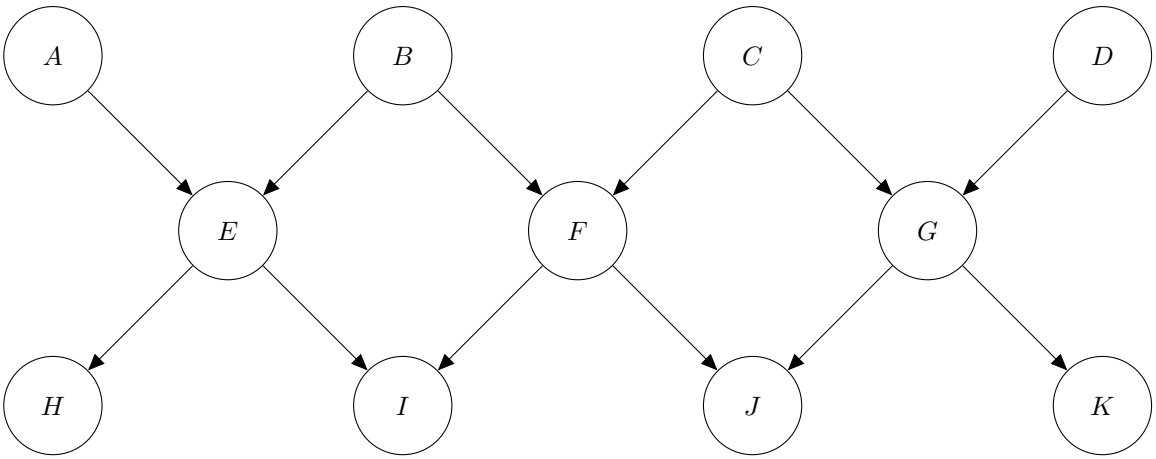
$X_{1,1}$ blocks the only active paths to both $X_{1,2}$ and $X_{1,3}$, so both of those become independent of $X_{3,1}$ given $X_{1,1}$

Which random variables are independent of $X_{3,1}$ *given* $X_{1,1}$ and $X_{3,3}$? (Leave blank if the answer is none.)

○ $X_{1,2}$ ○ $X_{1,3}$ ○ $X_{2,1}$ ○ $X_{2,2}$ ○ $X_{2,3}$ ○ $X_{3,2}$

The path from a node down to $X_{3,3}$ and up to another node is an active path.

For the following questions we will consider the following Bayes' Net:



(d) For each of the following queries, mark which variables' conditional probability tables will affect the answer to the query. For example, by marking F you'd indicate that the values in the conditional probability table $P(F \mid B, C)$ affect the answer to the query.

(i) [1 pt] $P(A \mid +k)$

● A ○ B ○ C ○ D ○ E ○ F ○ G ○ H ○ I ○ J ○ K

(ii) [1 pt] $P(A \mid +d)$

● A ○ B ○ C ○ D ○ E ○ F ○ G ○ H ○ I ○ J ○ K

(iii) [1 pt] $P(A, D)$

● A ○ B ○ C ● D ○ E ○ F ○ G ○ H ○ I ○ J ○ K

(iv) [1 pt] $P(A, D| + i, -j)$

● A ● B ● C ● D ● E ● F ● G ○ H ● I ● J ○ K

(v) [1 pt] $P(A| + j, +k)$

● A ○ B ○ C ○ D ○ E ○ F ○ G ○ H ○ I ○ J ○ K

(vi) [1 pt] $P(A| + i, +k)$

● A ● B ● C ● D ● E ● F ● G ○ H ● I ○ J ● K

We can sum over an unobserved leaf node and remove it from the Bayes net, and its CPT will not affect the query. For parts (i), (ii), (v) below, we can sum over H, I, E in that order to disconnect A from the rest of the graph. For part (iii) note that $A \perp\!\!\!\perp D$ so $P(A, D) = P(A)P(D)$. For part (iv) we can remove H and K this way but none of the other nodes. For part (v) we can also sum out J. All of the other CPTs are necessary. You could construct a numerical example to verify that a given CPT is necessary.

Alternatively, you can consider an augmented Bayes net, which has node A thru K as well as nodes for the CPTs of the original graph, $CPT(A)$ thru $CPT(K)$. In the augmented Bayes net, $CPT(A)$ is a parent of A and so on. Then we can verify that a CPT is necessary by checking conditional independences in the augmented bayes net. For example, we can check that $CPT(I) \perp\!\!\!\perp A|K$.

(e) Consider a run of Gibbs sampling for the query $P(B, C \mid +h, +i, +j)$. The current sample value is $+a, +b, +c, +d, +e, +f, +g, +h, +i, +j, +k$. For each of the following scenarios, write out an expression for the distribution Gibbs sampling would sample from. *Your expression should contain only conditional probabilities available in the network, and your expression should contain a minimal number of such conditional probabilities.*

(i) [1 pt] If A were to be sampled next, the distribution over A to sample from would be:

$P(A| + b, +e) \propto P(+e|A, +b)P(+b)$

Note that only B, E are necessary because all the other variables are independent of A given B, E.

(ii) [1 pt] If F were to be sampled next, the distribution over F to sample from would be:

$P(F|+b,+c,+e,+g,+i,+j) \propto P(F|+b,+c)P(+i|+e,F)P(+j|F,+g)$

(iii) [1 pt] If K were to be sampled next, the distribution over K to sample from would be:

$P(K|+g)$

Q5. [6 pts] Kernels and Feature Transforms

A kernel function $K(x, z)$ is a function that conceptually denotes the similarity between two instances x and z in a transformed space. More specifically, for a feature transform $x \to \phi(x)$, the kernel function is $K(x, z) = \phi(x) \cdot \phi(z)$. The beauty of algorithms using kernel functions is that we never actually need to explicitly specify this feature transform $\phi(x)$ but only the values $K(x, z)$ for pairs (x, z). In this problem, we will explore some kernel functions and their feature transforms. For this problem the input vectors are assumed to be 2 dimensional (i.e. $x = (x_1, x_2)$). Remember that $x \cdot z = x_1 z_1 + x_2 z_2$.

(a) For each of the kernel functions below, mark the corresponding feature transform: (mark a single option only for each question)

(i) [1 pt] $K(x, z) = 1 + x \cdot z$

- ○ $\phi(x) = (x_1, x_2)$
- ● $\phi(x) = (1, x_1, x_2)$
- ○ $\phi(x) = (1, x_1^2, x_2^2)$
- ○ $\phi(x) = (x_1^2, x_2^2)$
- ○ $\phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$
- ○ $\phi(x) = (1, x_1^2, x_2^2, \sqrt{2}x_1x_2)$

(ii) [1 pt] $K(x, z) = (x \cdot z)^2$

- ○ $\phi(x) = (x_1^2, x_2^2)$
- ○ $\phi(x) = (1, x_1^2, x_2^2)$
- ○ $\phi(x) = (1, x_1^2, x_2^2, \sqrt{2}x_1x_2)$
- ● $\phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$
- ○ $\phi(x) = (1, x_1, x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$
- ○ $\phi(x) = (x_1, x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$

(iii) [1 pt] $K(x, z) = (1 + x \cdot z)^2$

- ○ $\phi(x) = (1, x_1^2, x_2^2)$
- ○ $\phi(x) = (1, x_1^2, x_2^2, \sqrt{2}x_1x_2)$
- ○ $\phi(x) = (1, x_1^2, x_2^2, x_1, x_2, \sqrt{2}x_1x_2)$
- ● $\phi(x) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)$
- ○ $\phi(x) = (1, x_1, x_2, \sqrt{2}x_1x_2)$
- ○ $\phi(x) = (1, x_1x_2, x_1^2x_2^2)$

For all the above questions, write out $K(x, z)$ and find a $\phi(x)$ such that $K(x, z) = \phi(x) \cdot \phi(z)$. For example in (iii) $K(x, z) = (1 + x_1z_1 + x_2z_2)^2 = 1 + x_1^2z_1^2 + x_2^2z_2^2 + 2(x_1z_1 + x_2z_2 + x_1x_2z_1z_2) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2) \cdot (1, z_1^2, z_2^2, \sqrt{2}z_1, \sqrt{2}z_2, \sqrt{2}z_1z_2)$

(b) Multiple kernels can be combined to produce new kernel functions. For example $K(x, z) = K_1(x, z) + K_2(x, z)$ is a valid kernel function. For the questions below, kernel K_1 has the associated feature transform ϕ_1 and similarly K_2 has the feature transform ϕ_2. Mark the feature transform associated with K for the expressions given below.

Note: The operator $[*, *]$ denotes concatenation of the two arguments. For example, $[x, z] = (x_1, x_2, z_1, z_2)$.

(i) [1 pt] $K(x, z) = aK_1(x, z)$, for some scalar $a > 0$

- ○ $\phi(x) = \phi_1(x)$
- ○ $\phi(x) = [a, \phi_1(x)]$
- ○ $\phi(x) = a\phi_1(x)$
- ● $\phi(x) = \sqrt{a}\phi_1(x)$
- ○ $\phi(x) = \phi_1(x) + a$
- ○ $\phi(x) = a^2\phi_1(x)$

(ii) [1 pt] $K(x, z) = aK_1(x, z) + bK_2(x, z)$, for scalars $a, b > 0$

- ○ $\phi(x) = a\phi_1(x) + b\phi_2(x)$
- ○ $\phi(x) = \sqrt{a}\phi_1(x) + \sqrt{b}\phi_2(x)$
- ○ $\phi(x) = a^2\phi_1(x) + b^2\phi_2(x)$
- ○ $\phi(x) = [a\phi_1(x), b\phi_2(x)]$
- ● $\phi(x) = [\sqrt{a}\phi_1(x), \sqrt{b}\phi_2(x)]$
- ○ $\phi(x) = [a^2\phi_1(x), b^2\phi_2(x)]$

For (ii) we need a ϕ s.t. $\phi(x) \cdot \phi(z) = a\phi_1(x) \cdot \phi_1(z) + b\phi_2(x) \cdot \phi_2(z) = [\sqrt{a}\phi_1(x), \sqrt{b}\phi_2(x)] \cdot [\sqrt{a}\phi_1(z), \sqrt{b}\phi_2(z)]$. Thus we have $\phi(x) = [\sqrt{a}\phi_1(x), \sqrt{b}\phi_2(x)]$

(c) [1 pt] Suppose you are given the choice between using the normal perceptron algorithm, which directly works with $\phi(x)$, and the dual (kernelized) perceptron algorithm, which does not explictly compute $\phi(x)$ but instead works with the kernel function K. Keeping space and time complexities in consideration, when would you prefer using the kernelized perceptron algorithm over the normal perceptron algorithm.

Note: Here N denotes the total number of training samples and d is the dimensionality of $\phi(x)$.

● $d >> N$ ○ $d << N$ ○ Always ○ Never

For this question, the rationale was when we use a Kernel function, we typically store a Kernel matrix $\mathbb{K}$ with $\mathbb{K}_{ij} = \phi(x_i) \cdot \phi(x_j)$ where x_i and x_j are the i^{th} and j^{th} training instances. This results in an $N \times N$ matrix. If we were to use the transformed d-dimensional feature representation, we would have to store Nd values instead of N^2 values in the Kernel matrix. Thus space-wise, we would prefer kernels when $d >> N$.

Looking at time complexity, (at test time), if we use kernels (e.g. the kernelized perceptron) we need to compute $\sum_{i=1}^{N} \alpha_{i,y} K(x', x_i)$ for a test sample x'. Assuming the kernel function computation takes $\mathcal{O}(1)$ time, we need to do N such computations. In case of using $\phi(x)$, we have the precomputed weight vector as $w = \sum \alpha_{i,y} \phi(x_i)$ which is d-dimensional and the computation of $w.\phi(x')$ takes d $\mathcal{O}(1)$ computations. So again we would prefer kernels if $d >> N$.

Q6. [9 pts] Stopping Strategy

A fair six sided dice is rolled repeatedly and you observe outcomes sequentially. Formally, dice roll outcomes are independently and uniformly sampled from the set $\{1, 2, 3, 4, 5, 6\}$. At every time step before the h^{th} roll you can choose between two actions:

Stop: stop and receive a reward equal to the number shown on the dice or,

Roll: roll again and receive no immediate reward.

If not having stopped before then, at time step h (which would be reached after $h-1$ rolls) you are forced to take the action Stop, you receive the corresponding reward and the game ends.

We will model the game as a finite horizon MDP with six states and two actions. The state at time step k corresponds to the number shown on the dice at the k^{th} roll. Assume that the discount factor, γ, is 1.

Compute the Q function for the two actions: $Q^{h-1}(i, \text{"Roll"}) = \frac{1}{6}\sum_j V^h(j) = 3.5$, and $Q^{h-1}(i, \text{"Stop"}) = i$. Then apply the definition the value function: $V^{h-1}(i) = \max(Q^{h-1}(i, \text{"Roll"}), Q^{h-1}(i, \text{"Stop"}))$.

(a) [2 pts] The value function at time step h, when it is no longer possible to roll the dice again, is $V^h(1) = 1, V^h(2) = 2, \ldots, V^h(6) = 6$. Compute the value function at time step $h-1$:

$V^{h-1}(1) =$ $\max(3.5, 1) = 3.5$

$V^{h-1}(2) =$ $\max(3.5, 2) = 3.5$

$V^{h-1}(3) =$ $\max(3.5, 3) = 3.5$

$V^{h-1}(4) =$ $\max(3.5, 4) = 4$

$V^{h-1}(5) =$ $\max(3.5, 5) = 5$

$V^{h-1}(6) =$ $\max(3.5, 6) = 6$

(b) [2 pts] Express the value function at time step $k-1$, with $2 < k \leq h$ recursively in terms of the value function at roll k, so in terms of $V^k(1), V^k(2), \ldots V^k(6)$:

$V^{k-1}(i) =$ $\max(Q^{k-1}(i, \text{"Roll"}), Q^{k-1}(i, \text{"Stop"})) = \max(1/6\sum_j V^k(j), i)$

The Q function at time step k for action "Roll" does not depend on the state since the number shown by the dice is irrelevant once you decided to roll. We use the shorthand notation $q(k) = Q^k(\text{state}, \text{"Roll"})$ since the only dependence is on k.

(c) [1 pt] Compute $q(h-1)$: $q(h-1) = Q^{h-1}(i, \text{"Roll"}) = \frac{1}{6}\sum_j V^h(j) = 3.5$

(d) [2 pts] Express $q(k-1)$ recursively as a function of $q(k)$, with $2 < k \leq h$.

$q(k-1) =$ $\frac{1}{6}\sum_j V^k(j) = \frac{1}{6}\sum_j \max(Q^k(i, \text{"Roll"}), Q^k(i, \text{"Stop"})) = \frac{1}{6}\sum_j \max(q(k), j)$

(e) [2 pts] What is the optimal policy $\pi^k(s)$ at roll k as a decision rule based on the current state s and $q(k)$?

$\pi^k(s) =$ Roll if $q(k) > s$, stop otherwise

Q7. [13 pts] Inference

(a) Recall that for a standard HMM the Elapse Time update and the Observation update are of the respective forms:

$$P(X_t \mid e_{1:t-1}) = \sum_{x_{t-1}} P(X_t \mid x_{t-1}) P(x_{t-1} \mid e_{1:t-1})$$
$$P(X_t \mid e_{1:t}) \propto P(X_t \mid e_{1:t-1}) P(e_t \mid x_t)$$

We now consider the following two HMM-like models:

Mark the modified Elapse Time update and the modified Observation update that correctly compute the beliefs from the quantities that are available in the Bayes' Net. (Mark one of the first set of six options, and mark one of the second set of six options for (i), and same for (ii).)

(i) [2 pts]

- ● $P(X_t, Z_t \mid e_{1:t-1}) = \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1} \mid e_{1:t-1}) P(X_t \mid x_{t-1}, z_{t-1}) P(Z_t)$
- ○ $P(X_t, Z_t \mid e_{1:t-1}) = \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1} \mid e_{1:t-1}) P(X_t \mid x_{t-1}, z_{t-1})$
- ○ $P(X_t, Z_t \mid e_{1:t-1}) = \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1} \mid e_{1:t-1}) P(X_t, Z_t \mid x_{t-1}, z_{t-1})$
- ○ $P(X_t, Z_t \mid e_{1:t-1}) = \sum_{x_{t-1}} P(x_{t-1}, z_{t-1} \mid e_{1:t-1}) P(X_t \mid x_{t-1}, z_{t-1}) P(Z_t)$
- ○ $P(X_t, Z_t \mid e_{1:t-1}) = \sum_{x_{t-1}} P(x_{t-1}, z_{t-1} \mid e_{1:t-1}) P(X_t \mid x_{t-1}, z_{t-1})$
- ○ $P(X_t, Z_t \mid e_{1:t-1}) = \sum_{x_{t-1}} P(x_{t-1}, z_{t-1} \mid e_{1:t-1}) P(X_t, Z_t \mid x_{t-1}, z_{t-1})$

In the elapse time update, we want to get from $P(X_{t-1}, Z_{t-1} | e_{1:t-1})$ to $P(X_t, Z_t | e_{1:t-1})$.

$$\begin{aligned} P(X_t, Z_t | e_{1:t-1}) &= \sum_{x_{t-1}, z_{t-1}} P(X_t, Z_t, x_{t-1}, z_{t-1} | e_{1:t-1}) \\ &= \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1} | e_{1:t-1}) P(X_t | x_{t-1}, z_{t-1}, e_{1:t-1}) P(Z_t | X_t, x_{t-1}, z_{t-1}, e_{1:t-1}) \\ &= \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1} | e_{1:t-1}) P(X_t | x_{t-1}, z_{t-1}) P(Z_t) \end{aligned}$$

First line: marginalization, second line: chain rule, third line: conditional independence assumptions.

- ● $P(X_t, Z_t \mid e_{1:t}) \propto P(X_t, Z_t \mid e_{1:t-1}) P(e_t \mid X_t, Z_t)$
- ○ $P(X_t, Z_t \mid e_{1:t}) \propto \sum_{X_t} P(X_t, Z_t \mid e_{1:t-1}) P(e_t \mid X_t, Z_t)$
- ○ $P(X_t, Z_t \mid e_{1:t}) \propto \sum_{Z_t} P(X_t, Z_t \mid e_{1:t-1}) P(e_t \mid X_t, Z_t)$
- ○ $P(X_t, Z_t \mid e_{1:t}) \propto P(X_t, Z_t \mid e_{1:t-1}) P(e_t \mid X_t) P(e_t \mid Z_t)$
- ○ $P(X_t, Z_t \mid e_{1:t}) \propto P(X_t, Z_t \mid e_{1:t-1}) P(e_t \mid X_t)$
- ○ $P(X_t, Z_t \mid e_{1:t}) \propto P(X_t, Z_t \mid e_{1:t-1}) \sum_{X_t} P(e_t \mid X_t)$

In the observation update, we want to get from $P(X_t, Z_t | e_{1:t-1})$ to $P(X_t, Z_t | e_{1:t})$.

$$\begin{aligned} P(X_t, Z_t | e_{1:t}) &\propto P(X_t, Z_t, e_t | e_{1:t-1}) \\ &\propto P(X_t, Z_t | e_{1:t-1}) P(e_t | X_t, Z_t, e_{1:t-1}) \\ &\propto P(X_t, Z_t | e_{1:t-1}) P(e_t | X_t, Z_t) \end{aligned}$$

First line: normalization, second line: chain rule, third line: conditional independence assumptions.

(ii) [2 pts]

- ○ $P(X_t, Z_t \mid e_{1:t-1}) = \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1} \mid e_{1:t-1}) P(X_t \mid x_{t-1}, z_{t-1}) P(Z_t \mid e_{t-1})$
- ● $P(X_t, Z_t \mid e_{1:t-1}) = \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1} \mid e_{1:t-1}) P(Z_t \mid e_{t-1}) P(X_t \mid x_{t-1}, Z_t)$
- ○ $P(X_t, Z_t \mid e_{1:t-1}) = \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1} \mid e_{1:t-1}) P(X_t, Z_t \mid x_{t-1}, e_{t-1})$
- ○ $P(X_t, Z_t \mid e_{1:t-1}) = \sum_{x_{t-1}} P(x_{t-1}, z_{t-1} \mid e_{1:t-1}) P(X_t \mid x_{t-1}, z_{t-1}) P(Z_t \mid e_{t-1})$
- ○ $P(X_t, Z_t \mid e_{1:t-1}) = \sum_{x_{t-1}} P(x_{t-1}, z_{t-1} \mid e_{1:t-1}) P(Z_t \mid e_{t-1}) P(X_t \mid x_{t-1}, Z_t)$
- ○ $P(X_t, Z_t \mid e_{1:t-1}) = \sum_{x_{t-1}} P(x_{t-1}, z_{t-1} \mid e_{1:t-1}) P(X_t, Z_t \mid x_{t-1}, e_{t-1})$

In the elapse time update, we want to get from $P(X_{t-1}, Z_{t-1}|e_{1:t-1})$ to $P(X_t, Z_t|e_{1:t-1})$.

$$\begin{aligned} P(X_t, Z_t|e_{1:t-1}) &= \sum_{x_{t-1}, z_{t-1}} P(X_t, Z_t, x_{t-1}, z_{t-1}|e_{1:t-1}) \\ &= \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1}|e_{1:t-1}) P(Z_t|x_{t-1}, z_{t-1}, e_{1:t-1}) P(X_t|Z_t, x_{t-1}, z_{t-1}, e_{1:t-1}) \\ &= \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1}|e_{1:t-1}) P(Z_t|e_{t-1}) P(X_t|x_{t-1}, Z_t) \end{aligned}$$

First line: marginalization, second line: chain rule, third line: conditional independence assumptions.

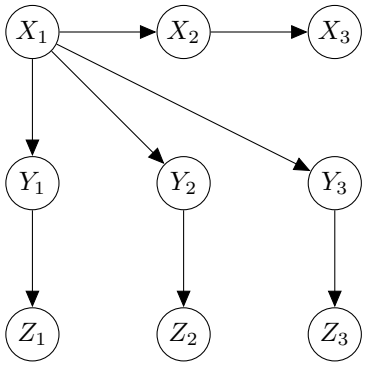
- ● $P(X_t, Z_t \mid e_{1:t}) \propto P(X_t, Z_t \mid e_{1:t-1}) P(e_t \mid X_t, Z_t)$
- ○ $P(X_t, Z_t \mid e_{1:t}) \propto \sum_{X_t} P(X_t, Z_t \mid e_{1:t-1}) P(e_t \mid X_t, Z_t)$
- ○ $P(X_t, Z_t \mid e_{1:t}) \propto \sum_{Z_t} P(X_t, Z_t \mid e_{1:t-1}) P(e_t \mid X_t, Z_t)$
- ○ $P(X_t, Z_t \mid e_{1:t}) \propto P(X_t, Z_t \mid e_{1:t-1}) P(e_t \mid X_t) P(e_t \mid Z_t)$
- ○ $P(X_t, Z_t \mid e_{1:t}) \propto P(X_t, Z_t \mid e_{1:t-1}) P(e_t \mid X_t)$
- ○ $P(X_t, Z_t \mid e_{1:t}) \propto P(X_t, Z_t \mid e_{1:t-1}) \sum_{X_t} P(e_t \mid X_t)$

In the observation update, we want to get from $P(X_t, Z_t|e_{1:t-1})$ to $P(X_t, Z_t|e_{1:t})$.

$$\begin{aligned} P(X_t, Z_t|e_{1:t}) &\propto P(X_t, Z_t, e_t|e_{1:t-1}) \\ &\propto P(X_t, Z_t|e_{1:t-1}) P(e_t|X_t, Z_t, e_{1:t-1}) \\ &\propto P(X_t, Z_t|e_{1:t-1}) P(e_t|X_t, Z_t) \end{aligned}$$

First line: normalization, second line: chain rule, third line: conditional independence assumptions.

(b) In this question we will consider a Bayes' Net with the following structure:



(i) [3 pts] Mark *all* of the following expressions that hold true for distributions represented by the Bayes' Net above.

- ○ $P(X_1, X_2, X_3 \mid +y_1) = P(X_1, X_2, X_3 \mid -y_1)$
- ● $P(Z_1, +x_3) = \sum_{x_1, x_2, y_1} P(x_1)P(x_2 \mid x_1)P(+x_3 \mid x_2)P(y_1 \mid x_1)P(Z_1 \mid y_1)$
- ● $P(Z_1, +x_3) = \sum_{x_1} P(x_1) \sum_{x_2} P(x_2 \mid x_1)P(+x_3 \mid x_2) \sum_{y_1} P(y_1 \mid x_1)P(Z_1 \mid y_1)$
- ● $P(Z_3 \mid +x_1, -y_3) = P(Z_3 \mid -x_1, -y_3)$
- ○ $P(Z_3 \mid +x_1, -y_3) = P(Z_3 \mid +x_1, +y_3)$
- ○ $P(Y_1, Y_2, Y_3) = P(Y_1)P(Y_2)P(Y_3)$

1) True if X_1, X_2, X_3 independent of Y_1. Does not hold in the Bayes Net.
2) This equation sums out all hidden variables from the joint distribution. Note that Y_2, Z_2, Y_3, Z_3 are not present because joining on the factors involving these variables sums to 1. For example, if we join on Z_2, we generate a factor, $f_1(Y_2) = \sum_{z_2} Pr(z_2|Y_2) = 1$, since the conditional distribution must sum to 1. If we join on Y_2 after that, we generate a factor, $f_2(X_1) = \sum_{y_2} f_1(y_2)Pr(y_2|X_1) = \sum_{y_2} Pr(y_2|X_1) = 1$. Same goes for Y_3, Z_3. So this is correct.
3) Same as (2), just reordering the summations. Also correct.
4) True if Z_3 and X_1 are independent given Y_3. This holds from the Bayes Net. Also correct.
5) True if Z_3 and Y_3 are independent given X_1. Not guaranteed true from the Bayes Net.
6) True if Y_1, Y_2, and Y_3 are indepndent. Not guaranteed true from the Bayes Net.
Correct answers: (2), (3), (4).

(ii) [2 pts] For the query $P(Z_1 \mid +x_3, +z_2, +z_3)$:

List a *most efficient* variable elimination ordering: X_2, Y_2, Y_3, X_1, Y_1 (Multiple solutions exist.)

List a *least efficient* variable elimination ordering: X_1, Y_1, Y_2, Y_3, X_2 (Multiple solutions exist.)

Note: efficiency is measured by the size of the single largest factor generated during the variable elimination process.

The most efficient ordering requires first eliminating any permutation of X_2, Y_2, Y_3, since eliminating these terms first joins X with their corresponding leaves, keeping the largest factor of size 1 (in terms of unobserved variables). Then, X_1 must come next, since joining on X_1 creates a factor $f(Y_1, +z_2, +z_3, +x_3)$. Again, largest factor is size 1, Lastly, Y_1 will be eliminated, creating a factor $f(Z_1, +z_2, +z_3, +x_3)$. The largest factor generated contains 1 unobserved variable. The least efficient ordering either requires you to eliminate X_1 first or eliminate Y_1, X_1 first in that order. Doing so will join on 4 unobserved variables.

(iii) [4 pts] Consider sampling through likelihood weighting. For each of the following fill in the weight of the sample and fill in the probability of that sample being the one generated when using likelihood weighting with the provided evidence. (Make sure to use only conditional probabilities available from the Bayes' Net.)

Evidence: $+x_1, +x_2, +x_3$. Sample: $+x_1, +x_2, +x_3, +y_1, +y_2, +y_3, +z_1, +z_2, +z_3$.

Sample weight = $Pr(+x_1)Pr(+x_2| + x_1)Pr(+x_3| + x_2)$

Probability of generating this sample = $(\prod_{i=1}^{3} Pr(+y_i| + x_1))(\prod_{i=1}^{3} Pr(+z_i| + y_i))$

Evidence: $+z_1, +z_2, +z_3$. Sample: $+x_1, +x_2, +x_3, +y_1, +y_2, +y_3, +z_1, +z_2, +z_3$.

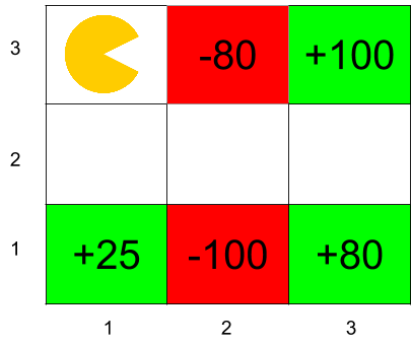
Sample weight = $Pr(+z_1| + y_1)Pr(+z_2| + y_2)Pr(+z_3| + y_3)$

Probability of generating this sample = $Pr(+x_1)Pr(+x_2| + x_1)Pr(+x_3| + x_2) \prod_{i=1}^{3} Pr(+y_i|x_1)$

The sample weight can be found via the likelihood weight calculation: $weight = \prod_i Pr(e_i|parents(e_i))$. The probability of the sample is just the product of the probabilities of sampling the remaining 6 hidden variables. We can take the product of these terms since the sampling process at each hidden variable happens independently.

Q8. [8 pts] Q-Learning Strikes Back

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



(a) [2 pts] What is the value of the optimal value function V^* at the following states:

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

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

(b) [3 pts] 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:

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

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

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

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

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

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

(i) [2 pts] Given that all w_i are initially 0, what are their values after the first episode:

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

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

(ii) [1 pt] 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 $(2+2+4=8)$.

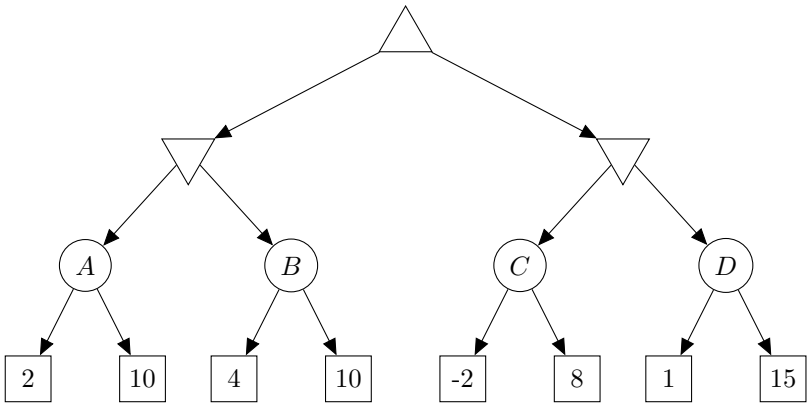
Q9. [9 pts] Adversarial VPI

In this problem you'll be considering VPI of unknown variables in an adversarial game. For this problem, assume that all observations of the random variables encoded as chance nodes are seen by both agents and that all chance nodes have equal probability for all children.

Hint: The properties of VPI presented in class were specifically for VPI when applied to a situation involving a single agent. These properties may or may not hold for situations with multiple agents. For example, the VPI of a node may be negative from the perspective of one of the agents.

When referring to VPI in the questions below, we always refer to the VPI for the maximizer.

(a) In this question we will consider the following game tree:



(i) [1 pt] What is the value of the game, for the maximizer, represented by the search tree above?

Answer: 6

This is regular expectiminimax where chance nodes have value equal to the average of their children, max nodes take the max, and min nodes take the min.

(ii) [1 pt] What is the VPI, for the maximizer, of the outcome of node A being revealed before the game is played?

Answer: -1

The VPI is found as the difference between the weighted sum of each outcome assuming the chance node takes on one of its values and the the current value. Each outcome has equal probability. If $A = 2$, the result is 3, from max going right. If $A = 10$, the result is 7. Thus, VPI(A) $= (0.5 * 3 + 0.5 * 7) - 6 = -1$

(iii) [1 pt] What is the VPI, for the maximizer, of the outcome of node B being revealed before the game is played?

Answer: -1

If $B = 4$, the result is 4, and if $B = 10$, the result is 6. Thus, VPI(A) $= (0.5 * 4 + 0.5 * 6) - 6 = -1$

(iv) [1 pt] What is the VPI, for the maximizer, of the outcome of node C being revealed before the game is played?

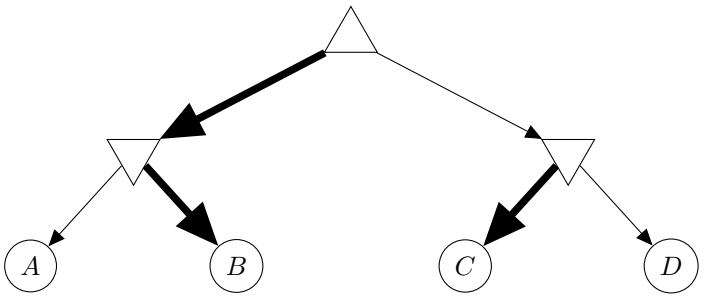
Answer: 1

If $C = -2$, the result is 6, and if $C = 8$, the result is 8. Thus, VPI(A) $= (0.5 * 6 + 0.5 * 8) - 6 = 1$

(v) [1 pt] What is the VPI, for the maximizer, of the outcome of node D being revealed before the game is played?

Answer: ____0____

Because neither player chose the action leading to D, VPI(D)=0. When $D = 1$, max will still choose to go left, and when $D = 15$, min will still choose to go left. Thus, nothing changes and $(0.5 * 6 + 0.5 * 6) - 6 = 0$



(b) The game tree above represents a different game in which the leaf utilities are omitted, but the edges corresponding to the action that would be selected at each node are bolded. Specifically, the maximizer would select left, the left minimizer would select node B, and the right minimizer would select C. For each of the following parts, select the most accurate expression for the VPI of the specified node.

When referring to VPI in the questions below, we always refer to the VPI for the maximizer.

(i) [1 pt] VPI(A):

○ VPI(A) $= 0$	○ VPI(A) > 0	○ VPI(A) < 0
○ VPI(A) $\in \mathbb{R}$	○ VPI(A) ≥ 0	● VPI(A) ≤ 0

One of the values that A can take on will raise its value, while the other will lower it. The option that raises it will not change anything, because the minimizer will not change it's choice. The option that lowers it might cause the minimizer to choose a lower option. This can only reduce the utility of the choices available to the maximizer. The expected outcome after knowing A thus has lower utility than the outcome with A unknown, so $VPI(A) \leq 0$.

(ii) [1 pt] VPI(B):

○ VPI(B) $= 0$	○ VPI(B) > 0	○ VPI(B) < 0
● VPI(B) $\in \mathbb{R}$	○ VPI(B) ≥ 0	○ VPI(B) ≤ 0

Examples: $A = \infty$, B has children $-100, 100$, $C = -50, D = 0$. In this case, $VPI(B) = (0.5 * 100 + 0.5 * -50) - (-50) = 75 \geq 0$.

$A = 0$, B has children $-10, 8$, $C = -\infty, D = -\infty$. In this case, $VPI(B) = (0.5 * -10 + 0.5 * 0) - (-1) = -4 \leq 0$. This shows that VPI(B) is not restricted to be positive or negative.

(iii) [1 pt] VPI(C):

○ VPI(C) $= 0$	○ VPI(C) > 0	○ VPI(C) < 0
○ VPI(C) $\in \mathbb{R}$	● VPI(C) ≥ 0	○ VPI(C) ≤ 0

Because the maximizer originally chose left, the only time the value of the game would change as a result of knowing C would be if the utility was higher. This means that the expected utility of playing after knowing C is greater than or equal to the utility of playing with C unknown, so $VPI(C) \geq 0$.

(iv) [1 pt] VPI(D):

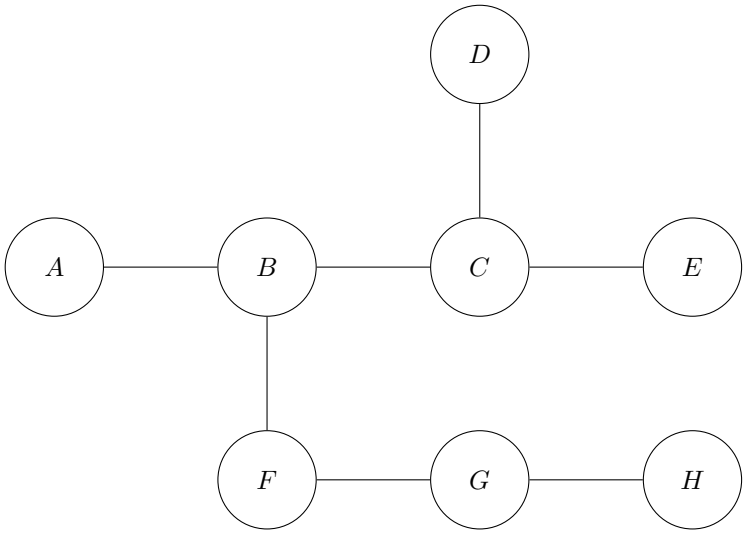
● VPI(D) $= 0$	○ VPI(D) > 0	○ VPI(D) < 0
○ VPI(D) $\in \mathbb{R}$	○ VPI(D) ≥ 0	○ VPI(D) ≤ 0

Because neither player chose to take actions leading to D, it is not possible for the outcome of the game to change as a result of D changing. If the value of D is greater than the expectation, the minimizer will not change its action, and if the value is less than its current value, the maximizer will not change its action.

Q10. [9 pts] Bayes Net CSPs

(a) For the following Bayes' Net structures that are missing a direction on their edges, assign a direction to each edge such that the Bayes' Net structure implies the requested conditional independences and such that the Bayes' Net structure does not imply the conditional independences requested not to be true. Keep in mind that Bayes' Nets cannot have directed cycles.

(i) [2 pts]

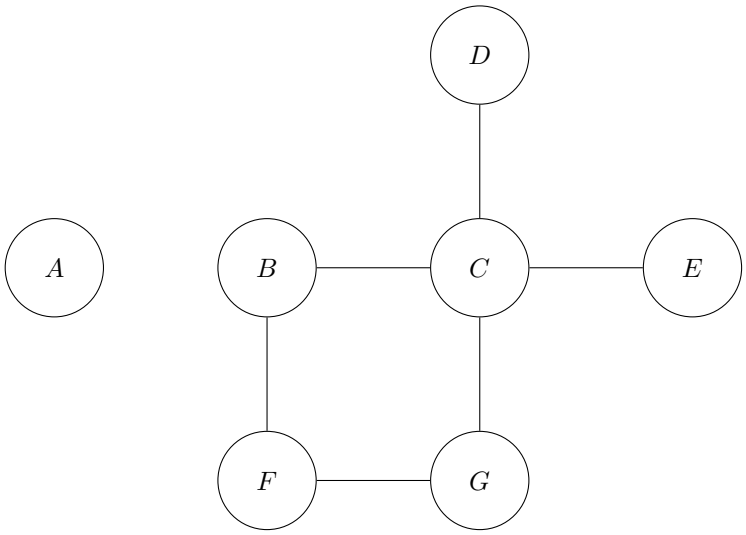


Constraints:

- $D \perp\!\!\!\perp G$
- not $D \perp\!\!\!\perp A$
- $D \perp\!\!\!\perp E$
- $H \perp\!\!\!\perp F$

The following are the directions of the edges:
$B \rightarrow A$
$C \rightarrow B$
$D \rightarrow C$
$E \rightarrow C$
$F \rightarrow B$
$F \rightarrow G$
$H \rightarrow G$

(ii) [2 pts]



Constraints:

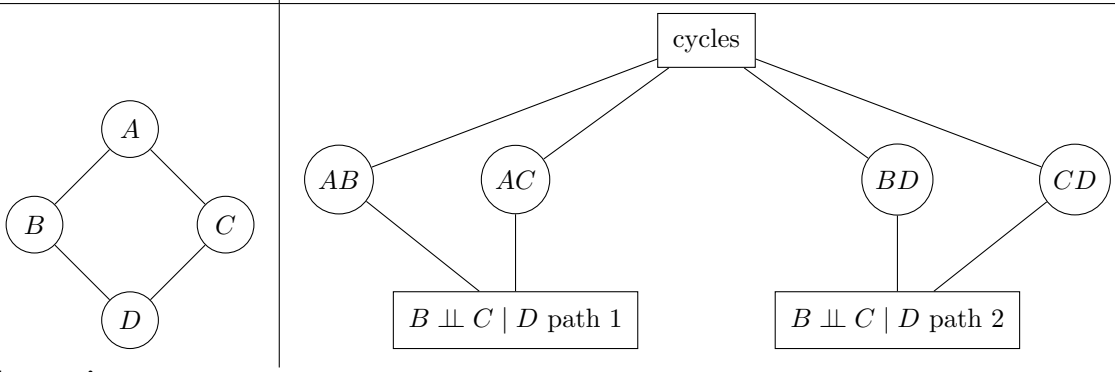
- $D \perp\!\!\!\perp F$
- not $D \perp\!\!\!\perp G$
- $D \perp\!\!\!\perp E$
- Bayes Net has no directed cycles

The following are the directions of the edges:
$C \to B$
$F \to B$
$F \to G$
$C \to G$
$D \to C$
$E \to C$

(b) For each of the following Bayes Nets and sets of constraints draw a constraint graph for the CSP. Remember that the constraint graph for a CSP with non-binary constraints, i.e., constraints that involve more than two variables, is drawn as a rectangle with the constraint connected to a node for each variable that participates in that constraint. A simple example is given below.

Note: As shown in the example below, if a constraint can be broken up into multiple constraints, do so.

Bayes Net | Example Constraint Graph

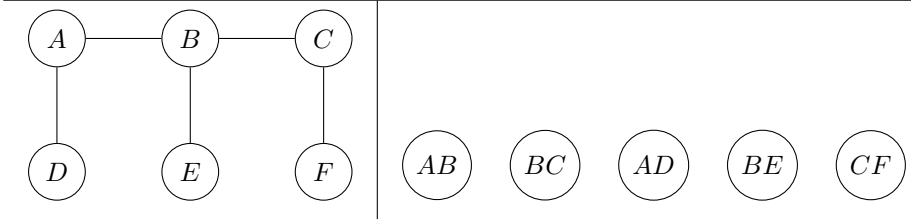


Constraints:

- $B \perp\!\!\!\perp C \mid D$
- No directed cycles

(i) [2 pts]

Bayes Net | Constraint Graph



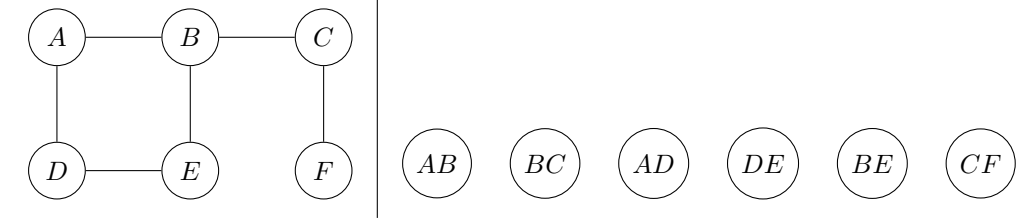
Constraints:

- $A \perp\!\!\!\perp F \mid E$
- not $D \perp\!\!\!\perp C$

Constraint $A \perp\!\!\!\perp F \mid E$: connect AB, BC, BE and CF.
Constraint not $D \perp\!\!\!\perp C$: connect AB, BC and AD.

(ii) [3 pts]

Bayes Net | Constraint Graph



Constraints:

- $A \perp\!\!\!\perp E \mid F$
- $C \perp\!\!\!\perp E$
- No directed cycles

Constraint $A \perp\!\!\!\perp E \mid F$ with path going through path $A - B - E$ with descendant C and F: connect AB, BC, BE, CF.

Constraint $A \perp\!\!\!\perp E \mid F$ with path going through path $A - D - E$: connect AD, DE.

Constraint $C \perp\!\!\!\perp E$ with path going through path $C - B - E$: connect BC, BE.

Constraint $C \perp\!\!\!\perp E$ with path going through path $C - B - A - D - E$: connect AB, BC, AD, DE.

No direct cycles: connect AB, AD, DE and BE.