CS 188 Fall 2009 — Introduction to Artificial Intelligence — Final Exam

INSTRUCTIONS

- You have 3 hours.
- The exam is closed book, closed notes except a two-page crib sheet.
- Please use non-programmable calculators only.
- 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.

Last Name	
First Name	
SID	
Login	
GSI	
All the work on this exam is my own. **(please sign)**	

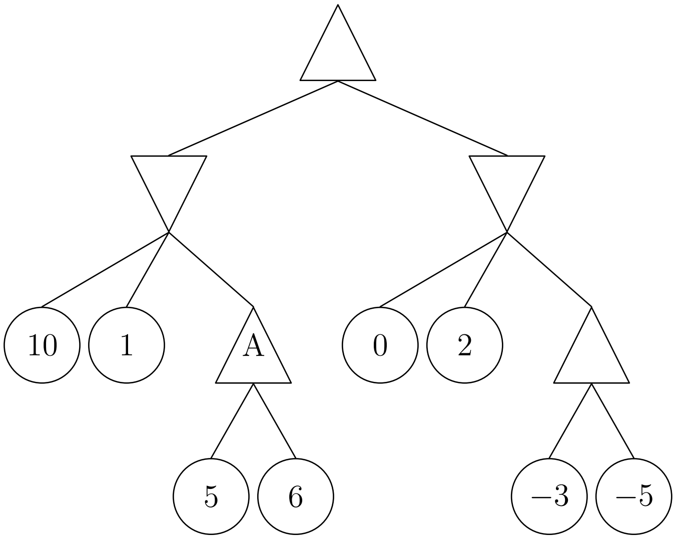
For staff use only

Q. 1	Q. 2	Q. 3	Q. 4	Q. 5	Q. 6	Total
/12	/19	/17	/20	/17	/15	/100

THIS PAGE INTENTIONALLY LEFT BLANK

1. (12 points) Spy Games

Consider the zero-sum minimax game tree shown below. The triangles pointing up, such as the root, correspond to the MAX player, while the triangles pointing down correspond to the MIN player. Leaves represent utilities for the MAX player.



(a) (1 pt) What is the minimax value of the root?

(b) (3 pt) Draw an X through all of the nodes that would be pruned (i.e. not explored at all) by α-β pruning. Assume a left-to-right order of evaluation of children.

(c) (1 pt) What values of α and β will be passed **into** the recursive call to **maxValue** for node A from the call to **minValue** on the parent of A?

(d) (1 pt) What will the final values of α and β be inside the recursive call to **maxValue** for node A just before it returns?

Suppose you are playing a deterministic game against an opponent. You have been covertly surveilling your opponent and have learned that he is a reflex agent.

(e) **(1 pt)** Suppose you also have determined the policy $\pi = \pi_0$ that he is using. What search procedure can you use to play optimally? State *briefly* but precisely how you would apply that procedure here.

(f) **(2 pt)** Your opponent figured out that you know π_0. As a countermeasure, he has switched to randomly picking the policy π from three alternatives, π_1, π_2, and π_3, each with equal probability, at the beginning of each turn. You have been able to determine what the policies are, but naturally do not know which one will be chosen each turn. What search procedure can you use to play optimally? State *briefly* but precisely how you would apply that procedure here.

(g) **(3 pt)** Suppose your opponent switches to randomly picking the policy π at the beginning of the game, rather than at the beginning of each turn (but still chooses randomly among π_1, π_2, and π_3, each with equal probability). The opponent does not switch thereafter. What search procedure can you use to play optimally, and over which state space is this procedure searching? State *briefly* but precisely how you would apply that procedure here.

2. (19 points) Search, MDPs, and CSPs

Consider the following generic search problem formulation with finitely many states:

States: $S = \{s_0, \dots, s_n\}$
Initial state: s_0
Actions: A
Successor function: $\text{Succ}(s, a) = s'$
Cost function: $\text{Cost}(s, a) = c > 0$
Goal test: s_n is the only goal state

(a) (3 pt) Reformulate this problem as an MDP, described in terms of the original search problem elements.

States:

Actions:

Transition function:

Reward function:

Discount:

(b) (3 pt) Imagine that an agent has run value iteration for k rounds, computing V_k for all states, before deciding to switch tactics and do graph search in the original search problem. The agent wants to take advantage of the work it has already done, and so it uses V_k to construct an A* heuristic, h, setting $h(s) = -V_k(s)$. Which of the following are true? Briefly justify your answers.

(i) $\forall s \in S, h(s) \geq 0$

(ii) h is admissible

Now, consider a generic deterministic MDP with all-negative rewards:

States: $S = \{s_0, \ldots, s_n\}$. There is a subset $G \subset S$ of terminal states.
Actions: For $s \in S$, the set of available actions is $A(s)$. For $s \in G$, $A(s) = \emptyset$.
Transition function: For $s \in S, a \in A(s)$, and fixed result state $s'(s,a)$, $T(s,a,s'(s,a)) = 1$.
Otherwise, $T(s,a,s') = 0$.
Reward function: For $s \in S, a \in A(s)$, $R(s,a,s'(s,a)) = r(s,a) < 0$.
Discount: $\gamma < 1$

(c) (4 pt) You are an agent who has just been dropped in some random state $s_i \in S$. You need to find a plan that maximizes the appropriately discounted sum of rewards from now until the end of the episode. Formulate this as a search problem.

States:

Initial state:

Actions:

Successor function:

Cost function:

Goal test:

(d) (3 pt) You're a fairly well prepared agent, who's done some Q-learning before being dropped in the MDP. You want to use this information, so you define a heuristic, $h(s) = -\left(\max_a Q(s,a)\right)$. Which of the following are true? Briefly justify your answers.

(i) $\forall s \in S, h(s) \geq 0$

(ii) h is admissible

Now, imagine trying to solve a generic, not necessarily deterministic, MDP with rewards R and transitions T. Your goal is to find an optimal policy π, which gives the best action from *any* state. You decide to use a CSP to solve this generic MDP, using a formulation in which each state's action under the policy is represented by a variable.

(e) (4 pt) Complete the definition of the CSP below. You may add additional variables beyond those given. Your domains are not limited to binary (or even discrete) values, your constraints are not limited to unary or binary ones, and implicit definitions are allowed. However, make sure that any variables or constraints you add are stated precisely.

Variables: π_s for each $s \in S$

Domains: $\text{Domain}(\pi_s) = A(s)$

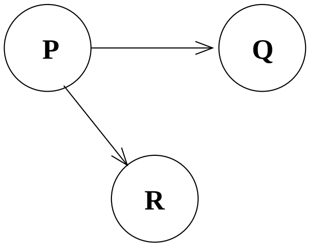
Constraints:

(f) (2 pt) Why would solving this CSP with depth-first search be substantially less efficient than using value iteration or policy iteration?

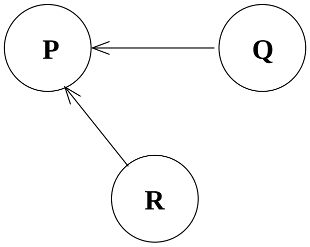
3. (17 points) BNs and MDPs

(a) (3 pt) The figures below show pairs of Bayes Nets. In each, the ORIGINAL network is shown on the left. The REVERSED network, shown on the right, has had all the arrows flipped. Therefore, the REVERSED network may not be able to represent the ORIGINAL distribution. For each pair, add a minimal number of arrows to the REVERSED network such that it is guaranteed to be able to represent the distribution in the ORIGINAL network. If no arrows need to be added, clearly state "none needed."

(i)

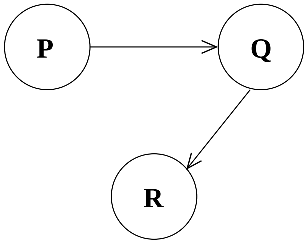


ORIGINAL NET

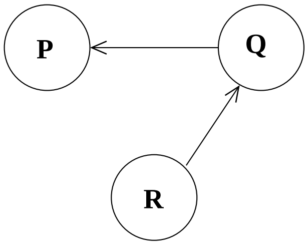


REVERSED: ADD ARROWS

(ii)

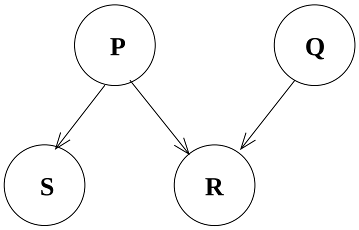


ORIGINAL NET

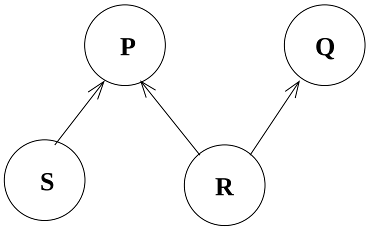


REVERSED: ADD ARROWS

(iii)



ORIGINAL NET



REVERSED: ADD ARROWS

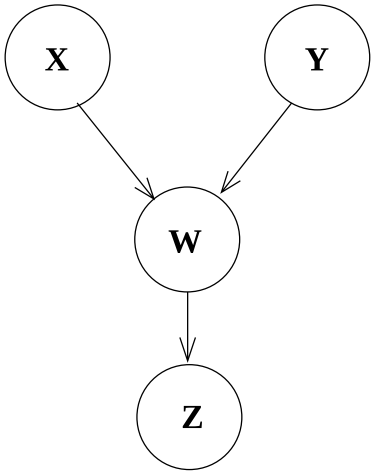
(b) (3 pt) Draw a Bayes net structure over the random variables A, B, C, D which has ALL the conditional independence guarantees as stated below:

(i) $A \perp\!\!\!\perp B$, but not $A \perp\!\!\!\perp B|\{C\}$

(ii) $A \perp\!\!\!\perp D$, but not $A \perp\!\!\!\perp D|\{C\}$

(iii) $C \perp\!\!\!\perp D|\{B\}$ but not $C \perp\!\!\!\perp D$

(c) (3 pt) Consider the Bayes net structure below, which encodes some probability distribution of the form $P(W, X, Y, Z)$. For some specific value of $W = w$, let $Q(X, Y, Z) = P(X, Y, Z|w)$ be the corresponding posterior distribution. On the right, draw the smallest (fewest arcs) Bayes net that is guaranteed to be able to represent the distribution Q.



P(X, Y, Z, W)

Q(X, Y, Z)

(d) (2 pt) Assume you have two BNs, B_{in} and B_{out} which are over the same variables. B_{in} has high indegree and low outdegree, so nodes have many parents. B_{out}, on the other hand, has high outdegree and low indegree, so nodes have many children. For which network will VE be more efficient? Briefly justify.

(e) (2 pt) Consider a generic MDP (non-deterministic, discounted, has cycles, etc.) and an appropriately small learning rate. Under what conditions will a q-learning agent be guaranteed to converge to optimal q-values for the states it visits? Circle all that apply.

(i) It only takes actions which appear optimal under current q-values

(ii) It only takes actions which are actually optimal

(iii) It only takes actions which are actually suboptimal

(iv) It takes completely random actions

(v) It takes apparently optimal actions with probability 0.5 and random ones with probability 0.5

(f) (4 pt) Consider a general, non-deterministic MDP with n states and a actions in each state, in which the length of the longest path is L, the highest-possible total reward along any path is R, the lowest-reward transition has reward $r_{\min}$, and the highest-reward transition has reward $r_{\max}$. Assume the discount is γ.

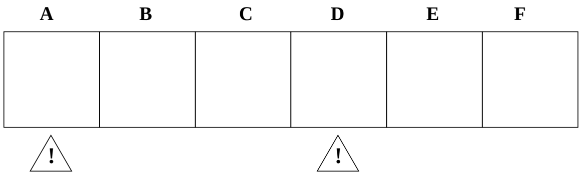
(i) Express the running time of one iteration of value iteration in terms of the (appropriate) above quantities.

(ii) Express the running time of one iteration of policy evaluation in terms of the (appropriate) above quantities.

4. (20 points) HMM: Search and Rescue

You are an interplanetary search and rescue expert who has just received an urgent message: a rover on Mercury has fallen and become trapped in Death Ravine, a deep, narrow gorge on the borders of enemy territory. You zoom over to Mercury to investigate the situation.

Death Ravine is a narrow gorge 6 miles long, as shown below. There are volcanic vents at locations A and D, indicated by the triangular symbols at those locations.



The rover was heavily damaged in the fall, and as a result, most of its sensors are broken. The only ones still functioning are its thermometers, which register only two levels , *hot* and *cold*. The rover sends back evidence $E = hot$ when it is at a volcanic vent (A and D), and $E = cold$ otherwise. There is no chance of a mistaken reading.

The rover fell into the gorge at position A on day 1, so $X_1 = A$. Let the rover's position on day t be $X_t \in \{A, B, C, D, E, F\}$. The rover is still executing its original programming, trying to move 1 mile east (i.e. right, towards F) every day. However, because of the damage, it only moves east with probability 0.5, and it stays in place with probability 0.5. Your job is to figure out where the rover is, so that you can dispatch your rescue-bot.

(a) (3 pt) Three days have passed since the rover fell into the ravine. The observations were ($E_1 = hot$, $E_2 = cold$, $E_3 = cold$). What is $P(X_3|hot_1, cold_2, cold_3)$, the probability distribution over the rover's position on day 3, given the observations?

You decide to attempt to rescue the rover on day 4. However, the transmission of E_4 seems to have been corrupted, and so it is not observed.

(b) (2 pt) What is the rover's position distribution for day 4 given the same evidence, $P(X_4|hot_1, cold_2, cold_3)$?

If you deploy the rescue-bot in the correct location, you will save the rover and be rewarded \$10,000. If not, you will get \$0. You only get one chance to attempt a rescue, at which point you will be paid or fired!

(c) (2 pt) Assuming you attempt a rescue given your current information, what is the MEU (maximum expected utility) location to send the rescue-bot, and what is the corresponding expected utility measured in dollars (your utility)?

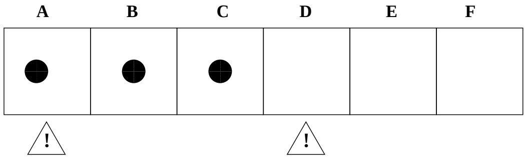
(i) MEU location:

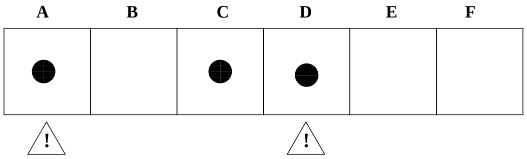
(ii) Expected utility:

(d) (5 pt) The observation wasn't corrupted, it was just accidentally encrypted! You can decrypt the message before attempting the rescue, which will allow you to observe E_4 before deciding where to send the rescue-bot. What is the VPI of E_4?

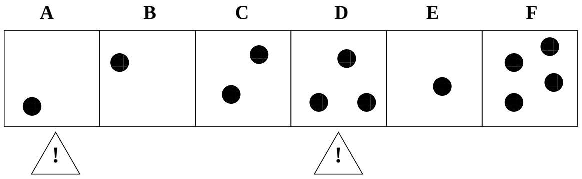
Rescuing robots is hard, so the next time this happens you decide to try approximate inference using particle filtering to track the rover.

(e) (2 pt) If your particles are initially in the top configuration shown below, what is the probability that they will be in the bottom configuration shown below after one day (after time elapses, but before evidence is observed)?

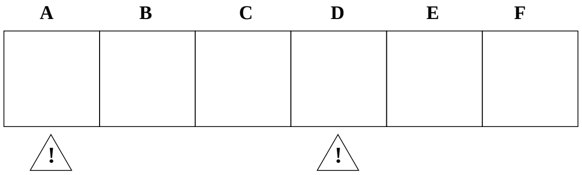




(f) (2 pt) If your particles are initially in the configuration



and the next observation is $E = hot$, what will the maximum likelihood configuration of particles be after this observation has been taken into account? Draw your answer in the diagram below. Assume you keep a fixed number of particles during the observation step.

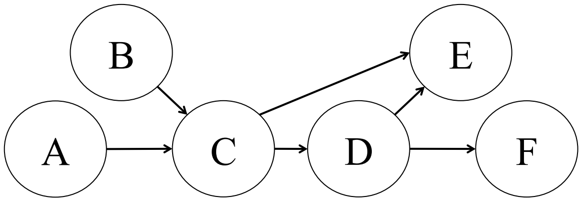


(g) (2 pt) Your co-pilot thinks you should model $P(E|X)$ differently. Even though the sensors are *not* noisy, she thinks you should use $P(hot|$ no volcanic vent$) = \epsilon$, and $P(cold|$ volcanic vent$) = \epsilon$, meaning that a *hot* reading at a non-volcanic location has a small probability, and a *cold* reading at a volcanic location has a small probability. She performs some simulations with particle filtering, and her version does seem to produce more accurate results, despite the false assumption of noise. Explain *briefly* why this could be.

(h) (2 pt) The transition model (east: 0.5 and stay: 0.5) turns out to be an oversimplification. The rover's position X_t is actually determined by both its previous position X_{t-1} and also its current velocity, V_t, which randomly drifts up and down from the previous day's velocity. Draw the dynamic Bayes net that represents this refined model. You do not have to specify the domains of V, X, or E, just the structure of the net.

5. (17 points) Variable Elimination

(a) (2 pt) You are given the following Bayes Net, and you need to compute $P(\mathrm{A}|+e,+f)$. You decide to use variable elimination. What factors will you start out with, taking the evidence into account?



Starting factors:

(b) (2 pt) You start out by eliminating C. What factors will you need to join and what factor will be created after performing the join, but before summing out C?

Factors to join:

Resulting factor:

(c) (2 pt) For any variable, X, let |X| denote the size of X's domain (the number of values it can take). How large is the table for the factor from part (b), again before summing out C ?

Now, instead of a specific Bayes Net, let's consider variable elimination on arbitrary Bayes Nets. At any stage in the variable elimination process, there will be a set of factors $\mathcal{F} = \{F_i \mid i = 1, \ldots, n\}$. Each factor, F_i, can be written as $P(L_i|R_i)$, where L_i and R_i are both sets of variables (for simplicity, assume that there is no evidence). For any variable X, we define $I(\mathrm{X})$ to be the set of indices of factors that include X: $I(\mathrm{X}) = \{i \mid \mathrm{X} \in (L_i \cup R_i)\}$.

When eliminating a specific variable, Y, we start by joining the appropriate factors from $\mathcal{F}$, and creating a single joined factor, called join(Y, $\mathcal{F}$). Note that join(Y, $\mathcal{F}$) is created *before* performing the elimination step, so it still includes Y.

(d) (2 pt) For a given factor, F, we use size(F) to denote the size of the table needed to represent F. Give a precise general expression for size(join(Y, $\mathcal{F}$)), using the notation above.

size(join(Y, $\mathcal{F}$)) =

The variable ordering used can make a large difference in the amount of time and memory needed to run variable elimination. Because of this, it would be useful to figure out the optimal ordering before you actually do any eliminations, making inference as efficient as possible. Consider a generic Bayes Net with variables $\mathrm{X} \in V$, that encodes a set of initial factors $\mathcal{F}_0$.

(e) **(5 pt)** For a query variable Q, consider computing the marginal $P(\mathrm{Q})$. Formulate a search problem that determines the optimal variable elimination ordering, where cost is measured by the sum of the sizes of the tables created during elimination. Note that for this problem we are only concerned with the sum of the sizes of the tables created by the *join* step, not the smaller tables that are subsequently created by the *eliminate* step. A correct answer will explicitly define the initial state in terms of the initial Bayes Net, and will reference the quantity size(F) defined above. You may also find the following notation helpful: for a variable X, and factor F, eliminate(X, F) denotes the new factor created by summing over values of X to eliminate X from F.

States: Pairs $(W, \mathcal{F})$, where W is the set of variables remaining to be eliminated and $\mathcal{F}$ is the set of all remaining factors.

Initial state:

Actions:

Successor function:

Cost function:

Goal test:

(f) **(2 pt)** Let parents(X) and children(X) denote the sets containing all of X's parents/children in the Bayes Net. Which of the following are admissible heuristics for the search problem you defined in part (e)? Circle all that apply.

(i) $|W|$

(ii) $|\mathcal{F}|$

(iii) $\sum_{\mathrm{X}\in W} |\mathrm{X}|$

(iv) $\prod_{\mathrm{X}\in W} |\mathrm{X}|$

(v) $\sum_{\mathrm{X}\in W} \left(|\mathrm{X}| \prod_{\mathrm{Y}\in\mathrm{parents(X)}} |\mathrm{Y}|\right)$

(vi) $\sum_{\mathrm{X}\in W} \left(|\mathrm{X}| \prod_{\mathrm{Y}\in\mathrm{children(X)}} |\mathrm{Y}|\right)$

(g) **(2 pt)** Consider an alternative search problem, where the goal is to find the ordering that minimizes the *maximum* table size instead of the *sum* over all tables created. What should the new state space be?

6. (15 points) Machine Learning

Consider the training data below. X_1 and X_2 are binary-valued features and Y is the label you'd like to classify.

Y	X_1	X_2
1	0	0
0	1	0
1	1	1
1	0	0
0	0	1
0	1	0

(a) (2 pt) Assuming a Naive Bayes model, fill in the quantities learned from the training data in the tables below (no smoothing).

Y	$P(Y)$
0	
1	

X_1	$P(X_1 \mid Y = 0)$	$P(X_1 \mid Y = 1)$
0		
1		

X_2	$P(X_2 \mid Y = 0)$	$P(X_2 \mid Y = 1)$
0		
1		

(b) (2 pt) Fill in the learned quantities below as in (a), but with add-k smoothing, with $k = 1$.

X_1	$P(X_1 \mid Y = 0)$	$P(X_1 \mid Y = 1)$
0		
1		

X_2	$P(X_2 \mid Y = 0)$	$P(X_2 \mid Y = 1)$
0		
1		

(c) (2 pt) Use your model in (b) to calculate $P(Y \mid X_1 = 0, X_2 = 0)$.

(d) (1 pt) What does $P(Y \mid X_1 = 0, X_2 = 0)$ approach as $k \rightarrow \infty$?

Y	X_1	X_2
1	0	0
0	1	0
1	1	1
1	0	0
0	0	1
0	1	0

(e) (4 pt) Circle the feature sets that would enable a perceptron to classify the training data perfectly.

i. $\{X_1\}$
ii. $\{X_2\}$
iii. $\{X_1, X_2\}$
iv. $\{1, X_1, X_2\}$
v. $\{1, \text{abs}(X_1 - X_2)\}$
vi. $\{1, X_1, X_2, X_1 + X_2\}$
vii. $\{1, X_1, X_2, \max(X_1, X_2)\}$
viii. $\{X_1, X_2, X_1 = X_2\}$
ix. $\{1, X_1, (X_1 X_2)\}$

(f) (2 pt) Circle *true* or *false* for each statement about a perceptron classifier in general. Assume weight vectors are initialized to 0s.

(i) (*true* or *false*) Can produce non-integer-valued weight vectors from integer-valued features

(ii) (*true* or *false*) Estimates a probability distribution over the training data

(iii) (*true* or *false*) Assumes features are conditionally independent given the class

(iv) (*true* or *false*) Perfectly classifies any training set eventually

(g) (2 pt) Circle *true* or *false* for each statement about a MIRA classifier in general. Assume weight vectors are initialized to 0s.

(i) (*true* or *false*) Is slower to train than a naive Bayes classifier

(ii) (*true* or *false*) Typically improves on the perceptron by allowing non-linearity in the decision boundary

(iii) (*true* or *false*) Often improves on the perceptron by finding a decision boundary that generalizes better to test data

(iv) (*true* or *false*) Typically tunes the C parameter on the training set