CS 188 Spring 2006 — Introduction to Artificial Intelligence — Final Exam

You have 180 minutes. The exam is open-book, open-notes, no electronics other than calculators. 100 points total. Don't panic!

Mark your answers ON THE EXAM ITSELF. Write your name, SID, login, and section number at the top of each page.

For true/false questions, CIRCLE *True* OR *False*.

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.

For official use only

Q. 1	Q. 2	Q. 3	Q. 4	Q. 5	Q. 6	Q. 7	Q. 8	Total
/18	/16	/16	/12	/14	/28	/22	/24	/160

1. **(18 points.) True/False**

   Each problem is worth 2 points. Incorrect answers are worth 0 points. Skipped questions are worth 1 point.

   (a) *True/False*: All state space search problems are also MDPs.

   (b) *True/False*: The variable elimination algorithm can require time exponential in the size of a Bayes' net.

   (c) *True/False*: Observing an additional node in a Bayes' net can only increase the number of variable pairs which are conditionally independent.

   (d) *True/False*: There is a distribution over three variables which can be represented by any three node Bayes' net graph topology.

   (e) *True/False*: The largest factor created by the variable elimination algorithm is no bigger than the largest initial factor.

   (f) *True/False*: The starting position for chess has value either 1 (white wins), -1 (black wins), or 0 (draw).

   (g) *True/False*: A rational agent can strictly prefer a lottery between two outcomes over either individual outcome.

   (h) *True/False*: Q-learning is only guaranteed to converge to the optimal q-values if the underlying MDP is deterministic.

   (i) *True/False*: A rational agent will never gamble if the expected monetary value of a game is negative.

2. (16 points.) Search and CSPs

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

States: there are $d+2$ states: $\{s_s, s_g\} \cup \{s_1, \ldots, s_d\}$
Initial state: s_s
Successor function: $Succ(s)$ generates at most b successors
Goal test: s_g is the only goal state
Step cost: each step has a cost of 1

(a) (2 pts) Suppose an optimal solution has cost n. If the goal is reachable, what is the upper bound on n?

(b) (2 pts) Suppose we must solve this search problem using BFS, but with limited memory. Specifically, assume we can only store k states during search. Give a bound on n for which the search will fit in the available memory.

(c) (2 pts) Would any other search procedure allow problems with substantially deeper solutions to be solved? Either argue why not, or give a method along with an improved bound on n.

(d) (5 pts) If we knew the exact value of n, we could formulate a CSP whose complete assignment specifies an optimal solution path $(X_0, X_1 \ldots, X_n)$ for this search problem. State binary and unary constraints which guarantee that a statisfying assignment is a valid solution.

Variables: X_0, X_1, $\ldots$, X_n
Domains: $Dom(X_i) = \{s_s, s_g\} \cup \{s_1, \ldots, s_d\} \quad \forall\, i \in \{0, 1, \ldots, n\}$
Constraints:

(e) (3 pts) How can the successor function be used to efficiently enforce the consistency of an arc $X_i \rightarrow X_{i-1}$? (Note: Enforcing the consistency of this arc prunes values from the domain of X_i, not X_{i-1}.)

(f) (2 pts) After reducing the domains of any variables with unary constraints, suppose we then make all arcs $X_i \rightarrow X_{i-1}$ consistent, processed in order from $i = 1$ to n. Next, we try to assign variables in reverse order, from X_n to X_0, using backtracking DFS. Why is this a particularly good variable ordering?

3. (16 points.) A* and HMMs

Recall that an HMM assigns probabilities to sequences of hidden states $(s_1, \dots s_n)$ (hidden states take values in $s \in S$) along with observations $(o_1, \dots o_n)$ (observations take values $o \in O$). The Viterbi algorithm computes the maximum likelihood sequence $(s_1, \dots s_n)^* = \arg\max_{(s_1, \dots s_n)} P(s_1, \dots s_n, o_1, \dots o_n)$ incrementally, using dynamic programming. In this problem, we will consider the use of heuristic search to solve the same problem.

(a) (1 pt) Write out an expression for $P(s_1, \dots s_n, o_1, \dots o_n)$ in terms of the local transition, emission, and initial probabilities.

(b) (5 pts) Let $(o_1, \dots o_n)$ be a known sequence of observations of length n. Pose the problem of finding the maximum likelihood sequence as a state space search problem in which the states are prefixes of hidden sequences. Provide the initial state, successor function, goal test, and step cost. *Hint:* Recall that $\log ab = \log a + \log b$

States: Sequence prefixes $(s_1, \dots s_k)$

Initial state:

Successor function:

Goal test:

Step cost:

(b) (2 pts) What is the branching factor of the search tree?

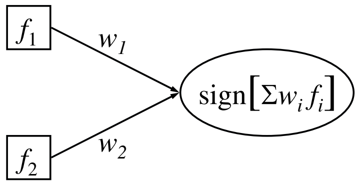
(c) (2 pts) What is the maximum depth of the search tree?

(d) (4 pts) Provide a non-trivial admissible heuristic for this search problem. Explain why your heuristic is admissible. Overly loose bounds will not receive full credit.

(e) (2 pts) Describe a qualitative scenario in which A* search would be more efficient than the Viterbi algorithm from class. Make sure your answer relates specifically to the heuristic function you gave in part (d), i.e. do not simply state that A* will be better if the heuristic is very good.

4. (12 points.) Features and Classification

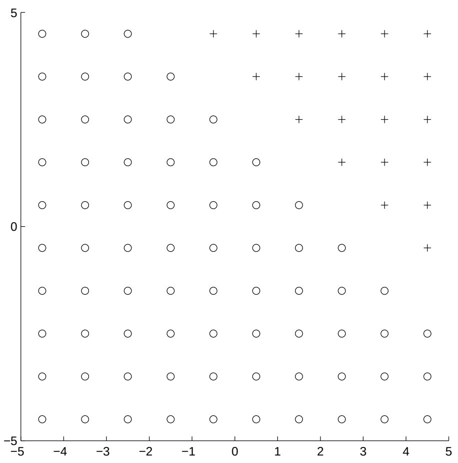
The binary perceptron depicted below has two inputs and two weights (and no fixed bias). It computes a weighted sum of its inputs and produces the symbol + if the sum is positive, and ∘ otherwise.



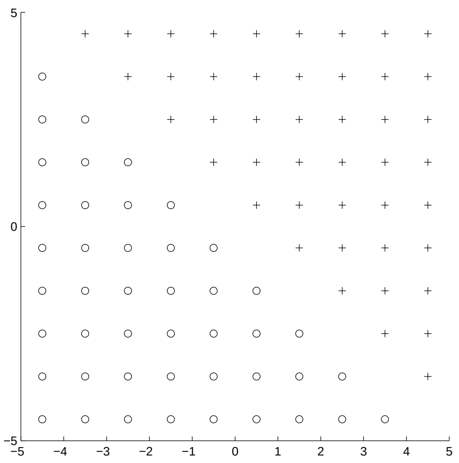
Given labeled training data points (x, y), where the the x-axis is depicted horizontally and the y-axis is vertical, we can compute various features f_1 and f_2 as inputs to the perceptron above. For each feature set (A-H) below, list the training sets (if any) which can be separated (classified perfectly). (12 points total)

(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)
$f_1 = x$	$f_1 = x$	$f_1 = x + y$	$f_1 = x^2$	$f_1 = x^2$	$f_1 = x^2 + y^2$	$f_1 = x + y$	$f_1 = \cos x$
$f_2 = 1$	$f_2 = y$	$f_2 = 1$	$f_2 = 1$	$f_2 = y^2$	$f_2 = 1$	$f_2 = xy$	$f_2 = \sin y$
________	________	________	________	________	________	________	________

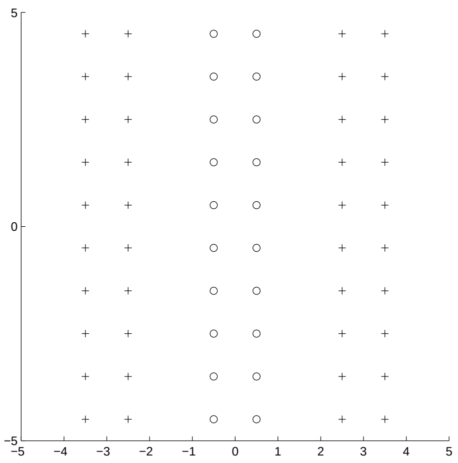
(1)



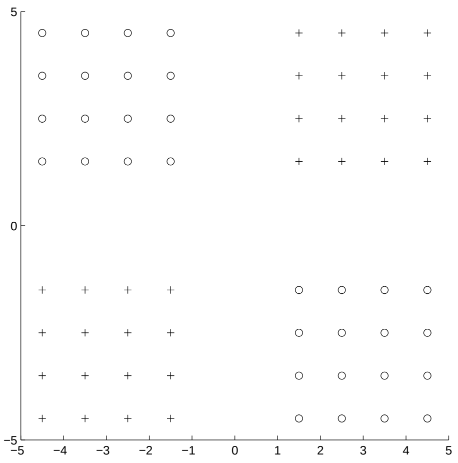
(2)



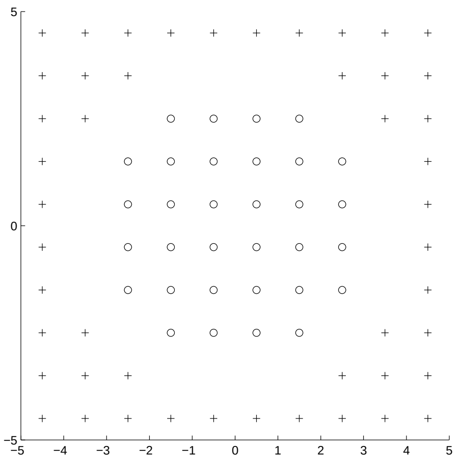
(3)



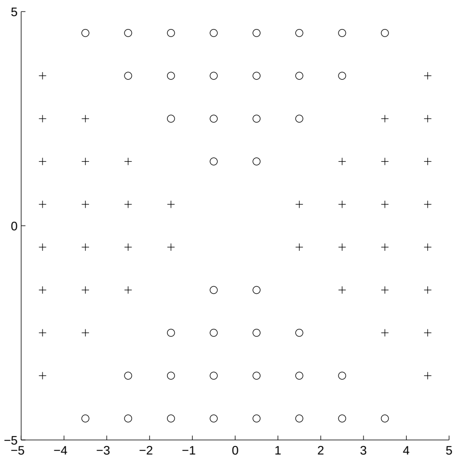
(4)



(5)



(6)



5. (14 points.) Game Trees

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

(a) (3 pts) Draw a (small) game tree in which the root node has a larger value if expectimax search is used than if minimax is used, or argue why it is not possible.

(b) (3 pts) Draw a (small) game tree in which the root node has a larger value if minimax search is used than if expectimax is used, or argue why it is not possible.

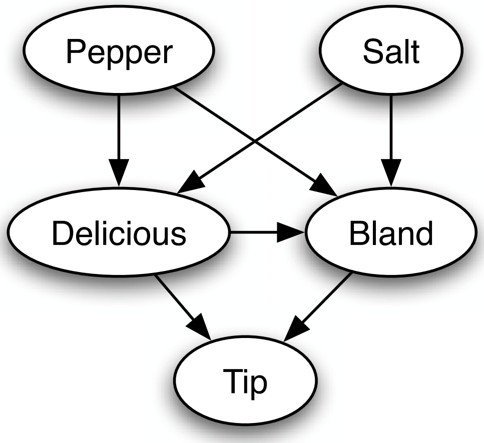
(c) (2 pts) Under what assumptions about player 2 should player 1 use minimax search rather than expectimax search to select a move?

(d) (2 pts) Under what assumptions about player 2 should player 1 use expectimax search rather than minimax search?

(e) (4 pts) Imagine that player 1 wishes to act optimally (rationally), and player 1 knows that player 2 also intends to act optimally. However, player 1 also knows that player 2 (mistakenly) believes that player 1 is moving uniformly at random rather than optimally. Explain how player 1 should use this knowledge to select a move. Your answer should be a precise algorithm involving a game tree search, and should include a sketch of an appropriate game tree with player 1's move at the root. Be clear what type of nodes are at each ply and whose turn each ply represents.

6. (28 points.) Bayes' Nets

An ambitious Bongo Burger employee wants to maximize tips, which he thinks depend on the quality of the chef's cooking. When the chef uses more *Pepper* and more *Salt*, more people say "*Delicious*" and fewer people say "how *Bland*". Better comments come with better tips. He draws the following Bayes' net with 5 Boolean variables to model the scenario.



(a) (3 pts) Which of the following are true of this network? Note that a *marginal distribution* over a subset of variables in a Bayes' net is the result of summing out all other variables from the full joint distribution. (Circle whichever are true, 1 pts each)

(i) It can represent any joint probability distribution over $Salt, Pepper, Delicious, Bland, Tip$.

(ii) For any marginal distribution over $Salt, Delicious, Bland$, the network can represent at least one joint distribution with that marginal distribution.

(iii) For any marginal distribution over $Salt, Tip$, the network can represent at least one joint distribution with that marginal distribution.

(b) (2 pts) How many total probability entries are there in the conditional probability tables for this network. Note: not the number of degrees of freedom, which may be fewer; a coin flip has two probabilities, but only one degree of freedom, and we're asking for the larger number.

(c) (2 pts) List all of the marginal independence assertions made by this graph about pairs of variables.

(d) (3 pts) List all of the conditional independence assertions made by this graph about pairs of variables.

(e) (2 pts) For one of the *conditional* independence assertions above, briefly explain how you might test its correctness from a set of observations.

(f) (6 pts) "No, no, no," the chef replies, "People tip because the food is *Tasty* (influenced by salt and pepper), but comments are more complicated. Customers say *Delicious* or *Bland* in part based on whether they like the food, but mostly based on how *Polite* they are." Draw a new Bayes' net appropriate to this scenario, which includes variables for *Tasty* and *Polite*, but which has as few arcs as possible. You do *not* need to specify the CPTs.

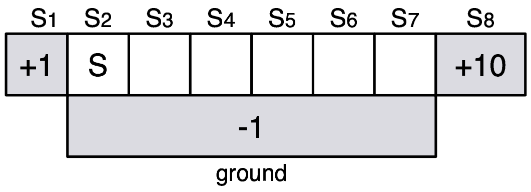
(g) (4 pts) Given $P(Pepper = True) = P(Salt = True) = 0.5$, the CPT below and the Bayes' net you just constructed for part (e), can you give $P(Pepper = True, Salt = False|Tasty = False)$? If so, compute it. If not, explain why not.

$Pepper$	$Salt$	$P(Tasty = True|pepper, salt)$
True	True	0.8
True	False	0.6
False	True	0.6
False	False	0.1

(h) (1 pt) What is the (marginal) probability of the food being $Tasty$?

(i) (3 pts) How might the chef alter her seasoning habits to increase $P(Tasty)$ without using any more salt or pepper overall? What would $P(Tasty)$ be after this change? Would this change alter the Bayes' net you drew for (e)? If so, how? If not, why not?

7. (22 points.) MDPs and Reinforcement Learning



Consider the above MDP, representing a robot on a balance beam. Each grid square is a state and the available actions are *right* and *left*. The agent starts in state s_2, and all states have reward 0 aside from the ends of the grid s_1 and s_8 and the *ground* state, which have the rewards shown. Moving *left* or *right* results in a move left or right (respectively) with probability p. With probability $1 - p$, the robot falls off the beam (transitions to *ground*, and receives a reward of -1. Falling off, or reaching either endpoint, result in the end of the episode (i.e., they are terminal states). Terminal states do have instantaneous rewards, but have zero future rewards.

(a) (3 pts) For what values of p is the optimal action from s_2 to move *right* if the discount γ is 1?

(b) (3 pts) For what values of γ is the optimal action from s_2 to move *right* if $p = 1$?

(c) (5 pts) Given initial value estimates of zero, show the results of one, then two rounds of value iteration. You need only write down the non-zero entries.

	s_{ground}	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
Initial values	0	0	0	0	0	0	0	0	0
One update									
Two updates									

(d) (4 pts) Given initial q-value estimates of zero, show the result of Q-learning with learning rate $\alpha = 0.5$ after two epsiodes: $[s_2, s_3, ground]$ and $[s_2, s_3, s_4, s_5, ground]$ where the agent always moves right. You need only write down the non-zero entries. For the purposes of Q-learning updates, terminal states should be treated as having a single action *die* which leads to future rewards of zero. *Hint:* q-values of terminal states which have been visited should not be zero.

	s_{ground}, *die*	s_1, *die*	s_2, *left*	s_2, *right*	s_3, *left*	s_3, *right*	s_4, *left*	s_4, *right*
Initial q-values	0	0	0	0	0	0	0	0
After first episode								
After second episode								

	s_5, *left*	s_5, *right*	s_6, *left*	s_6, *right*	s_7, *left*	s_7, *right*	s_8, *die*
Initial q-values	0	0	0	0	0	0	0
After first episode							
After second episode							

(e) (3 pts) We can develop learning updates that involve two actions instead of one. Which of the following are true of the utility $U^\pi(s)$ of a state s under policy π, given that $U^\pi(s) = R(s) + \sum_{s'} T(s, \pi(s), s')\gamma U^\pi(s')$?

(i) $U^\pi(s) = R(s) + \sum_{s'} T(s, \pi(s), s') \sum_{s''} T(s', \pi(s), s'')\gamma^2 U^\pi(s'')$

(ii) $U^\pi(s) = R(s) + \sum_{s'} T(s, \pi(s), s') \left[\gamma R(s') + \sum_{s''} T(s', \pi(s), s'')\gamma^2 U^\pi(s'')\right]$

(iii) $U^\pi(s) = R(s) + \sum_{s'} T(s, \pi(s), s') \left[\gamma U^\pi(s') + \sum_{s''} T(s', \pi(s), s'')\gamma^2 U^\pi(s'')\right]$

(iv) $U^\pi(s) = \sum_{s'} T(s, \pi(s), s') \left[\gamma R(s') + U^\pi(s') + \sum_{s''} T(s', \pi(s), s'')\gamma^2 U^\pi(s'')\right]$

(v) $U^\pi(s) = \sum_{s'} T(s, \pi(s), s') \left[R(s) + \frac{1}{2}\gamma U^\pi(s') + \frac{1}{2}(\gamma R(s') + \sum_{s''} T(s', \pi(s), s'')\gamma^2 U^\pi(s''))\right]$

(f) (2 pts) Write a two-step-look-ahead value iteration update that involves $U(s)$ and $U(s'')$, where s'' is the state two time steps later. Why would this update not be used in practice?

(g) (2 pts) Write a two-step-look-ahead TD-learning update that involves $U(s)$ and $U(s'')$ for the observed state-action-state-action-state sequence s, a, s', a', s''.

8. (24 points.) Short answer

Each question should answered by no more than one or two sentences! (3 pts each)

(a) Name three specific techniques for resisting overfitting in classifers.

(b) Write a Bellman equation expressing $Q^\pi(s,a)$ in terms of $U^\pi(s')$ (and other MDP quantities).

(c) Write an equation which expresses that X and Y are conditionally independent given Z.

(d) Give an example of a search algorithm and a search problem where the algorithm is not complete (you may simply describe a qualitative property of the search problem, such as "a single goal state" rather than stating a concrete problem, or you may draw a small search space).

(e) Why might arc consistency require that you process each arc multiple times?

(f) What does the size of a hypothesis class have to do with generalization and overfitting?

(g) In reinforcement learning, why can it be useful to sometimes act in a way which is believed to be suboptimal?

(h) What is the Chinese room argument meant to argue against?

End of Exam