CS 188 Spring 2010 — Introduction to Artificial Intelligence — Final Exam Solutions

Q1. [14 pts] Search

For the following questions, please choose the best answer (only one answer per question). **Assume a finite search space.**

(a) [2 pts] Depth-first search can be made to return the same solution as breadth-first search using:

(i) Iterative Deepening
(ii) A closed list/list of nodes that have been expanded
(iii) A heuristic function
(iv) This is not possible

(i). Depth-first search with progressive deepening one level at a time will expand nodes in the same order as breadth-first search.

(b) [2 pts] A^* search can be made to perform a breadth-first search by setting (fill in correct values):

1. for all nodes, heuristic =
2. for all nodes, edgecost =

heuristic=0; edgecost=1 (or any constant)

(c) [2 pts] You run A^* search using a heuristic function which you know to be admissible and consistent. Your friend claims he has a search algorithm that is guaranteed to not expand more nodes than your algorithm (and in fact often expands far fewer in practice). He also tells you that his algorithm is guaranteed to find the optimal path.

Could the algorithm your friend claims to have exist? (circle one): yes no
Explain:

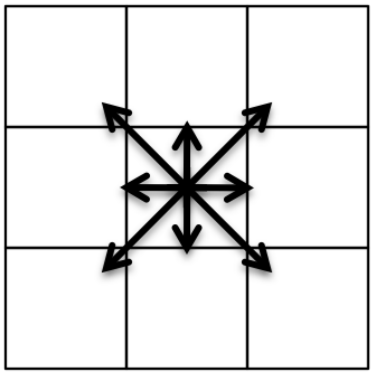
Yes. Your friend could simply be running A^* search but with a different admissible heuristic function that dominates your heuristic function (i.e. it comes closer to approximating the true cost to go)

(d) [2 pts] Depth first search using a closed list/list of nodes that have been expanded is:

(i) Optimal (will find a shortest path to goal)
(ii) Complete (will find a path to goal if at least one exists)
(iii) Both optimal and complete
(iv) Neither optimal nor complete

(ii). DFS is complete when we use an expanded list. However, there is no guarantee of finding an optimal solution.

Consider a grid, a portion of which is shown below:



You would like to search for paths in this grid. Unlike in Pacman, it is possible to move diagonally as well as horizontally and vertically. The distance between neighboring grid squares (horizontally or vertically) is 1, and the distance between diagonally adjacent grid squares is $\sqrt{2}$.

(e) [2 pts] Is the euclidean distance an admissible heuristic? The euclidean distance between two points (x_1, y_1) and (x_2, y_2) is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Yes, euclidean distance is an admissible heuristic. It is an underestimate (or exact value) of the distance it will take to move between any two grid squares.

(f) [2 pts] The Manhattan distance is not an admissible heuristic. Can it be made admissible by adding weights to the x and y terms? The Manhattan distance between two points (x_1, y_1) and (x_2, y_2) is $|x_2 - x_1| + |y_2 - y_1|$. A weighted version with weights α and β would be $\alpha|x_2 - x_1| + \beta|y_2 - y_1|$. Specify the (possibly empty) set of pairs of weights (α, β) such that the weighted Manhattan distance is admissible.

Yes, it is admissible provided that $\alpha < 1$, $\beta < 1$, and $\alpha + \beta < \sqrt{2}$.

(g) [2 pts] Is the L_∞ distance an admissible heuristic? The L_∞ distance between two points (x_1, y_1) and (x_2, y_2) is $max(|x_2 - x_1|, |y_2 - y_1|)$

Yes, L_∞ distance is an admissible heuristic.

Q2. [15 pts] Naive Bayes

Your friend claims that he can write an effective Naive Bayes spam detector with only three features: the hour of the day that the email was received ($H \in \{1, 2, \ldots, 24\}$), whether it contains the word 'viagra' ($W \in \{\text{yes}, \text{no}\}$), and whether the email address of the sender is Known in his address book, Seen before in his inbox, or Unseen before ($E \in \{\text{K}, \text{S}, \text{U}\}$).

(a) [3 pts] Flesh out the following information about this Bayes net:

Graph structure:

Naive Bayes net with three leaves

Parameters:

$\theta_{spam}, \theta_{H,i,c}, i \in \{1, \ldots, 23\}, \theta_{W,c}, \theta_{E,j,c}, j \in \{K, S\}, c \in \{spam, ham\}$

Size of the set of parameters:

$1 + 23 \cdot 2 + 2 + 2 \cdot 2$. Overcomplete representation should also accepted.

Suppose now that you labeled three of the emails in your mailbox to test this idea:

spam or ham?	H	W	E
spam	3	yes	S
ham	14	no	K
ham	15	no	K

(b) [2 pts] Use the three instances to estimate the maximum likelihood parameters.

$\theta_{spam} = 1/3, \theta_{H,3,spam} = 1, \theta_{H,14,ham} = 1/2, \theta_{H,15,ham} = 1/2, \theta_{W,spam} = 1.0, \theta_{E,S,spam} = 1, \theta_{E,K,ham} = 1$

(c) [2 pts] Using the maximum likelihood parameters, find the predicted class of a new datapoint with $H = 3$, $W = \text{no}$, $E = U$. Both assign a likelihood of zero, so no prediction is specified by the maximum likelihood parameters.

(d) [4 pts] Now use the three to estimating the parameter using Laplace smoothing and $k = 2$. Do not forget to smooth both the class prior parameters and the feature values parameters.

$\theta_{spam} = 3/7, \theta_{H,3,spam} = 3/49, \theta_{H,other,spam} = 2/49, \theta_{H,14,ham} = 3/50, \theta_{H,15,ham} = 3/50, \theta_{H,other,ham} = 2/50, \theta_{W,spam} = 3/5, \theta_{W,ham} = 2/6, \theta_{E,S,spam} = 3/7, \theta_{E,K,ham} = 4/8.$

(e) [1 pt] Using the parameters obtained with Laplace smoothing, find the predicted class of a new datapoint with $H = 3$, $W =$ no, $E = U$.

If spam:

(f) [3 pts] You observe that you tend to receive spam emails in batches. In particular, if you receive one spam message, the next message is more likely to be a spam message as well. Explain a new graphical model which most naturally captures this phenomena.

Graph structure:

A HMM, except that there are three observations at each node.

Parameters:

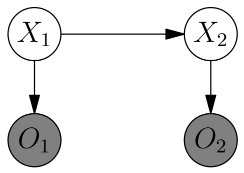
Add 2 parameters: transition to spam from spam and from ham.

Size of the set of parameters:

Add 2 to the expression in the first question.

Q3. [15 pts] Hidden Markov Models

Consider the following Hidden Markov Model.



X_1	$\Pr(X_1)$
0	0.3
1	0.7

X_t	X_{t+1}	$\Pr(X_{t+1}\|X_t)$
0	0	0.4
0	1	0.6
1	0	0.8
1	1	0.2

X_t	O_t	$\Pr(O_t\|X_t)$
0	A	0.9
0	B	0.1
1	A	0.5
1	B	0.5

Suppose that $O_1 = A$ and $O_2 = B$ is observed.

(a) [3 pts] Use the Forward algorithm to compute the probability distribution $\Pr(X_2, O_1 = A, O_2 = B)$. Show your work. You do not need to evaluate arithmetic expressions involving only numbers.

X_1	$\Pr(X_1, O_1 = A)$
0	$0.3 \cdot 0.9$
1	$0.7 \cdot 0.5$

X_2	$\Pr(X_2, O_1 = A, O_2 = B)$
0	$0.1 \cdot [0.4 \cdot (0.3 \cdot 0.9) + 0.8 \cdot (0.7 \cdot 0.5)] = 0.0388$
1	$0.5 \cdot [0.6 \cdot (0.3 \cdot 0.9) + 0.2 \cdot (0.7 \cdot 0.5)] = 0.1160$

(b) [3 pts] Use the Viterbi algorithm to compute the maximum probability sequence X_1, X_2. Show your work.

X_1	$\Pr(X_1, O_1 = A)$
0	$0.3 \cdot 0.9$
1	$0.7 \cdot 0.5$

X_2	$\max_{x_1} \Pr(X_1 = x_1, X_2, O_1 = A, O_2 = B)$	arg max
0	$0.1 \cdot \max(0.4 \cdot (0.3 \cdot 0.9), 0.8 \cdot (0.7 \cdot 0.5)) = 0.1 \cdot \max(0.108, 0.28) = 0.028$	$X_1 = 1$
1	$0.5 \cdot \max(0.6 \cdot (0.3 \cdot 0.9), 0.2 \cdot (0.7 \cdot 0.5)) = 0.5 \cdot \max(0.162, 0.07) = 0.081$	$X_1 = 0$

Thus, in the maximum probability sequence, $X_2 = 1$ and $X_1 = 0$.

For the next two questions, use the specified sequence of random numbers $\{a_i\}$ generated independently and uniformly at random from $[0, 1)$ to perform sampling. Specifically, to obtain a sample from a distribution over a variable $Y \in \{0, 1\}$ using the random number a_i, pick $Y = 0$ if $a_i < \Pr(Y = 0)$, and pick $Y = 1$ if $a_i \geq \Pr(Y = 0)$. Similarly, to obtain a sample from a distribution over a variable $Z \in \{A, B\}$ using the random number a_i, pick $Z = A$ if $a_i < \Pr(Z = A)$, and pick $Z = B$ if $a_i \geq \Pr(Z = A)$. Use the random numbers $\{a_i\}$ in order starting from a_1, using a new random number each time a sample needs to be obtained.

(c) [3 pts] Use likelihood-weighted sampling to obtain 2 samples from the distribution $\Pr(X_1, X_2 | O_1 = A, O_2 = B)$, and then use these samples to estimate $\mathrm{E}[\sqrt{X_1 + 3X_2} | O_1 = A, O_2 = B]$.

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
0.134	0.847	0.764	0.255	0.495	0.449	0.652	0.789	0.094	0.028

Sample 1: Sampling $X_1 = 0$ since $a_1 = 0.134 < 0.3$. Sampling $X_2 = 1$ since $a_2 = 0.847 \geq 0.4$. Weight is $0.9 \cdot 0.5 = 0.45$.

Sample 2: Sampling $X_1 = 1$ since $a_3 = 0.764 \geq 0.3$. Sampling $X_2 = 0$ since $a_4 = 0.255 < 0.8$. Weight is $0.5 \cdot 0.1 = 0.05$.

$\mathrm{E}[\sqrt{X_1 + X_2}] = 0.45/0.5 \cdot 1.0 + 0.05/0.5 \cdot 1.0 = 1.0$.

(d) [2 pts] [*true* or *false*] In general, particle filtering using a single particle is equivalent to rejection sampling in the case that there is no evidence. Explain your answer.

Both will produce samples from the prior.

(e) [2 pts] [*true* or *false*] Performing particle filtering twice, each time with 50 particles, is equivalent to performing particle filtering once with 100 particles. Explain your answer.

False. Due to re-weighting, it is not equivalent. (Consider the case of 1 particle vs. 2 particles.)

(f) [2 pts] [*true* or *false*] Variable elimination is generally more accurate than the Forward algorithm. Explain your answer.

They both perform exact inference.

Q4. [13 pts] Machine Learning

(a) [4 pts] Indicate which, if any, of the statements are correct, and explain your answer. Naive Bayes trained using maximum-likelihood parameter estimation

(i) is guaranteed not to perform worse if more features are added.

(ii) generally performs better on the training set if add-k smoothing is used.

(i) False. If additional dependent features are added, accuracy may be worse.

(ii) False. Add-k smoothing typically hurts accuracy on the training set, if anything.

(b) [2 pts] For data points that are *not* normalized (normalized means that for all data vectors x, $||x|| = 1$), we can implement nearest neighbor classification using the distance metric $||x - y||$ defined between two data vectors x and y to determine the nearest neighbor of each test point. Given a kernel function $K(x, y)$, describe how to implement a kernelized version of this algorithm. *Hint*: For a pair of vectors a and b, $||a-b||^2 = (a-b) \cdot (a-b)$.

Given a kernel function $K(x, y)$, we can define the corresponding distance metric $d(x, y) = \langle x - y, x - y\rangle_K = K(x, x) - 2K(x, y) + K(y, y)$. To classify a test point y, pick the label of the training point x that minimizes $d(x, y)$.

(c) [2 pts] Indicate which, if any, of the statements are correct, and explain your answer. Assuming a linearly separable dataset, Support Vector Machines typically

(i) achieve higher training set accuracy than Perceptrons.

(ii) achieve higher test set accuracy than Perceptrons because SVMs can be used with kernels.

(i) False. Both will achieve perfect accuracy.

(ii) False. Support Vector Machines typically achieve higher accuracy than Perceptrons due to maximizing the margin, but both Perceptrons and Support Vector Machines can be kernelized.

(d) [3 pts] Consider the problem of detecting human faces in images of larger scenes that may contain one or more faces, or no faces at all. The specific goal is to identify the *location and size* of each face in an image, rather than merely determining *whether* a face is present in the image. Suppose you are given as training data a set of images in which all of the faces have been labeled (specified as a list of pixel positions for each face). Describe one approach to solving this problem using *binary classification* machine learning. Specify what will serve as training examples for the binary classification algorithm, and how the binary classifier can be used to detect faces in new images. You don't need to specify the specific features or classification algorithm to use.

A binary classifier can be trained to classify fixed-size rectangular regions of an images as being of a face or not. A particular size of rectangle is chosen. Positive training examples can be obtained by picking, for each identified face in the training images, a minimum-size bounding box of the correct aspect ratio that contains all of the pixels in the face. Negative training examples can be obtained by randomly picking bounding boxes of the correct aspect ratio from the training images that do not contain any face pixels. These examples are then rescaled to be of the correct size.

To process a new image, every possible bounding box on the image of the correct aspect ratio at one of several scales is classified by rescaling as necessary and then invoking the classifier.

(e) [2 pts] [[✓ *true*] or *false*] In the case of a binary class and all binary features, Naive Bayes is a linear classifier. *Briefly* justify your answer.

$\arg\max_c \Pr(C = c | X_{1:n} = x_{1:n}) = \arg\max_c \log \Pr(C = c, X_{1:n} = x_{1:n})$, and $\log \Pr(C = c, X_{1:n} = x_{1:n})$ is linear in $x_{1:n}$.

Q5. [10 pts] Markov Decision Processes

(a) In the standard MDP formulation, we define the utility of a state-action sequence to be the sum of discounted rewards obtained from that sequence, i.e.

$$U_+(s_0, a_0, s_1, \ldots, a_n, s_n) = \sum_{i=0}^{n-1} \gamma^i R(s_i, a_i, s_{i+1}),$$

and the goal is to obtain a policy that maximizes the expected utility of the resultant state sequence. Under this utility function, the optimal policy depends only on the current state and not on any previous states or actions.

Suppose that we redefine the utility of a state-action sequence to be the *maximum* reward obtained from that sequence, i.e.

$$U_{\max}(s_0, a_0, s_1, \ldots, a_n, s_n) = \max_{i=0}^{n-1} R(s_i, a_i, s_{i+1}).$$

We still wish to obtain a policy that maximizes the expected utility of the resultant state sequence.

(i) [3 pts] Show by way of a counter-example that with this modified utility function, the optimal policy does not depend only on the current state.

Consider an MDP with three states, s_1, s_2, s_3, and two actions A, and B. $T(s_1, A, s_1) = 1$ and $R(s_1, A, s_1) = 1$; $T(s_1, B, s_2) = 0.5$ and $R(s_1, B, s_2) = 2$; $T(s_1, B, s_3) = 0.5$ and $R(s_1, B, s_3) = 0$; $T(s_2, \cdot, s_2) = 1$, $R(s_2, \cdot, \cdot) = 0$; $T(s_3, \cdot, s_3) = 1$, $R(s_3, \cdot, \cdot) = 0$. Then the optimal policy in s_1 is to choose action A once, and then choose action B.

(ii) [7 pts] Suppose you are given an arbitrary MDP $M = (S, A, T, R)$ with the *modified utility function* $U_{\max}$, where S denotes the state set, A denotes the action set, $T(s, a, s')$ denotes the transition probability function, and $R(s, a, s')$ denotes the reward function. Define a corresponding MDP $M' = (S', A', T', R', \gamma')$ with the property that the optimal policy for M' under the standard utility function U_+ specifies the optimal policy for M under the modified utility function $U_{\max}$.

$S' = S \times \mathbf{R}$

$A' = A$

Transition function: $T'((s, v), a, (s', v')) = \mathbf{1}[v' = \max(v, R(s, a, s'))] \cdot T(s, a, s')$.

Reward function: $R'((s, v), a, (s', v')) = v' - v$

$\gamma' = 1$

Q6. [33 pts] Short answer

Each true/false question is worth 1 point. Leaving a question blank is worth 0 points. **Answering incorrectly is worth -1 point.**

(a) For a search problem, the path returned by uniform cost search may change if we

(i) [*true* or [✓ *false*]] rescale all step costs by a scalar α: $0 < \alpha < 1$.

(ii) [*true* or [✓ *false*]] rescale all step costs by a scalar α: $1 < \alpha < 2$.

(iii) [[✓ *true*] or *false*] add a positive constant C to every step cost.

(b) Assume we are running A^* graph search with a consistent heuristic h. Let p be the node in the fringe about to be expanded in the search. When expanding p, we find that exactly one of its children is the goal state G and the cost of the found path through p to G is K. Let pathcost(p) denote the cost of the path to p that led to p being inserted in the queue. Then we have that

(i) [*true* or [✓ *false*]] the found path through p to the goal is a shortest path.

(ii) [[✓ *true*] or *false*] the found path through p to the goal is guaranteed to be at most $K - \text{pathcost}(p)$ longer than the shortest path.

(iii) [[✓ *true*] or *false*] the found path through p to the goal is guaranteed to be at most $K - \text{pathcost}(p) - h(p)$ longer than the shortest path.

(iv) [*true* or [✓ *false*]] the found path through p to the goal is guaranteed to be the shortest path going through p to the goal state.

(c) Consider a zero-sum game adversarial game. The minimizer is played by a computer program that is fast enough to perform min-max search all the way to the end of the game and it plays according to the thus-found moves. It is the minimizer's turn to play, and the minimizer's computer program returns a win of some positive value for the minimizer. Then we have that

(i) [*true* or [✓ *false*]] the minimizer is guaranteed to win the game only if the maximizer also plays the min-max strategy.

(ii) [*true* or [✓ *false*]] the minimizer is guaranteed to win the game only if the maximizer plays a deterministic strategy.

(iii) [*true* or [✓ *false*]] if the minimizer were to use alpha-beta instead of min-max search then the game could still end up in a tie.

(iv) [[✓ *true*] or *false*] if the maximizer is known to make moves uniformly at random every other turn, then the minimizer is not necessarily maximizing pay-off.

(d) Jeremy, Jie, Woody and Alex all get to act in an MDP $(S, A, T, \gamma, R, s_0)$. Jeremy runs value iteration until he finds V^* which satisfies $\forall s \in S : V^*(s) = \max_{a \in A} \sum_{s'} T(s,a,s')(R(s,a,s') + \gamma V^*(s'))$ and acts according to $\pi_{\text{Jeremy}} = \arg\max_{a \in A} \sum_{s'} T(s,a,s')(R(s,a,s') + \gamma V^*(s'))$. Jie acts according to an arbitrary policy π_{Jie}. Woody takes Jie's policy π_{Jie} and runs one round of policy iteration to find his policy π_{Woody}. Alex takes Jeremy's policy and runs one round of policy iteration to find his policy π_{Alex}. Then we have that

(i) [*true* or [✓ *false*]] there are MDP's in which Jie would outperform Woody.

(ii) [*true* or [✓ *false*]] there are MDP's in which Woody would outperform Alex.

(iii) [*true* or [✓ *false*]] there are MDP's in which Alex would outperform Jeremy.

(iv) [*true* or [✓ *false*]] there are MDP's in which Jeremy would outperform Alex.

(v) [*true* or [✓ *false*]] there are MDP's in which Jie would outperform Jeremy.

(e) Bob notices value iteration converges more quickly with smaller γ and rather than using the true discount factor γ, he decides to use a discount factor of $\alpha\gamma$ with $0 < \alpha < 1$when running value iteration. Then we have that

(i) [*true* or [✓ *false*]] while Bob will not find the optimal value function, he could simply rescale the values he finds by $\frac{1-\gamma}{1-\alpha}$ to find the optimal value function.

(ii) [[✓ *true*] or *false*] if the MDP has zero rewards everywhere, except for a single transition at the goal with a positive reward, then Bob will still find the optimal policy.

(iii) [*true* or [✓ *false*]] if the MDP's transition model is deterministic, then Bob will still find the optimal policy.

(iv) [[✓ *true*] or *false*] Bob's policy will tend to more heavily favor short-term rewards over long-term rewards compared to the optimal policy.

(f) In the Bayes Net to the right, which of the following conditional independence assertions are true?

$$\begin{tikzcd}
 & & A \arrow[dl] \arrow[dr] & \\
 & B \arrow[dl] \arrow[dr] & & C \arrow[dl] \\
F & & D \arrow[d] & \\
 & & E &
\end{tikzcd}$$

(i) [*true* or [✓ *false*]] $A \perp\!\!\!\perp E$
(ii) [[✓ *true*] or *false*] $B \perp\!\!\!\perp C|A$
(iii) [[✓ *true*] or *false*] $F \perp\!\!\!\perp C|A$
(iv) [*true* or [✓ *false*]] $B \perp\!\!\!\perp C|A, E$

(g) In variable elimination,

(i) [*true* or [✓ *false*]] when changing a Bayes net by removing a parent from a variable, the maximum factor size (where size is the number of non-fixed variables involved in the factor) generated during the optimally ordered variable elimination is reduced by at most 1.

(ii) [*true* or [✓ *false*]] the ordering of variables in variable elimination affects the maximum factor size generated by at most a factor of two.

(iii) [*true* or [✓ *false*]] the size of factors generated during variable elimination is upper-bounded by twice the size of the largest conditional probability table in the original Bayes net.

(iv) [*true* or [✓ *false*]] the size of factors generated during variable elimination is the same if we exactly reverse the elimination ordering.

(h) Consider two particle filtering implementations:

Implementation 1:
Initialize particles by sampling from initial state distribution and assigning uniform weights.

1. Propagate particles, retaining weights
2. Resample according to weights
3. Weight according to observations

Implementation 2:
Initialize particles by sampling from initial state distribution.

1. Propagate unweighted particles
2. Weight according to observations
3. Resample according to weights

(i) [[✓ *true*] or *false*] Implementation 2 will typically provide a better approximation of the estimated distribution than implementation 1.

(ii) [[✓ *true*] or *false*] If the transition model is deterministic then both implementations provide equally good estimates of the distribution.

(iii) [*true* or [✓ *false*]] If the observation model is deterministic then both implementations provide equally good estimates of the distribution.

(i) Particle filtering:

(i) [[✓ *true*] or *false*] With a deterministic transition model and a stochastic observation model, as time goes to infinity, when running a particle filter we will end up with all identical particles.

(ii) [[✓ *true*] or *false*] With a deterministic observation model, all particles might end up having zero weight.