CS221 Practice Final

Autumn 2012

1 Other Finals

The following pages are excerpts from similar classes' finals. The content is similar to what we've been covering this quarter, so that it should be useful for practicing. Note that the topics and terminology differ slightly, so feel free to ignore the questions that we did not cover.

Certain topics are less emphasized in the past exams, but will be **more emphasized** in the final for the class. These include:

- Weighted CSPs and Markov Nets (the practice exams place more of an emphasis on Bayes Nets).
- Loss-based learning (the practice exams place an emphasis on Naive Bayes instead).
- Unsupervised learning (e.g., EM)
- Logic (covered in much greater depth in our class)

In contrast, the practice exams cover state space models fairly deeply. State space models will be **less emphasized** in the final for the class.

The first portion of the practice exam comes with solutions; the rest are provided as example problems, but without solutions. In terms of other miscellaneous notes:

- *Perceptron* refers to a classifier using the perceptron loss (see slide 34 in the lecture on loss minimization).
- The *forward* (and *backward*) algorithm for HMMs is just an instance of variable elimination, as you did in the first part of your Pacman projects, before implementing particle filtering. Relatedly, *Viterbi* is an algorithm to decode the MAP estimate in an HMM.

1. (17 points.) Search: A* Variants

Queuing variants: Consider the following variants of the A* *tree search* algorithm. In all cases, g is the cumulative path cost of a node n, h is a lower bound on the shortest path to a goal state, and n' is the parent of n. Assume all costs are non-negative.

(i) Standard A*

(ii) A*, but we apply the goal test before enqueuing nodes rather than after dequeuing

(iii) A*, but prioritize n by $g(n)$ only (ignoring $h(n)$)

(iv) A*, but prioritize n by $h(n)$ only (ignoring $g(n)$)

(v) A*, but prioritize n by $g(n) + h(n')$

(vi) A*, but prioritize n by $g(n') + h(n)$

(a) (3 points) Which of the above variants are complete, assuming all heuristics are admissible?

(b) (3 points) Which of the above variants are optimal, again assuming all heuristics are admissible?

Upper Bounds: A* exploits lower bounds h on the true completion cost h^*. Suppose now that we also have an *upper bound* $k(n)$ on the best completion cost (i.e. $\forall n, k(n) \geq h^*(n)$). We will now consider A* variants which still use $g + h$ as the queue priority, but save some work by using k as well. Consider the point at which you are inserting a node n into the queue (fringe).

(c) (3 points) Assume you are required to preserve optimality. In response to n's insertion, can you ever delete any nodes m currently on the queue? If yes, state a general condition under which nodes m can be discarded, if not, state why not. Your answer should involve various path quantities (g, h, k) for both the newly inserted node n and other nodes m on the queue.

In a *satisficing* search, you are only required to find *some* solution of cost less than some threshold t (if one exists). You need not be optimal.

(d) (3 points) In the satisficing case, in response to n's insertion, can you ever delete any nodes m currently on the queue? If yes, state a general condition, if not, state why not. Your answer should involve various path quantities (g, h, k) for both the newly inserted node n and other nodes m on the queue.

ϵ-Admissible Heuristics: Suppose that we have a heuristic function which is not admissible, but ϵ-*admissible*, meaning for some known $\epsilon > 0$,

$$h(n) \leq h^*(n) + \epsilon \qquad \text{for all nodes } n$$

where $h^*(n)$ is the optimal completion cost. In other words, h is never more than ϵ from being optimal.

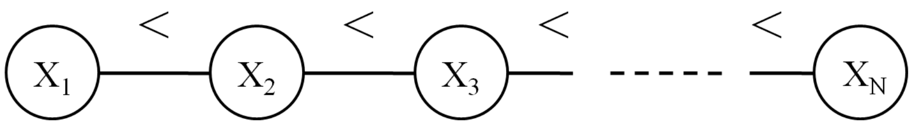
(e) (1 point) Is using A* with an ϵ-admissible heuristic complete? Briefly justify.

(f) (2 points) Assuming we utilize an ϵ admissible heuristic in standard A* search, how much worse than the optimal solution can we get? I.e., if c^* is the optimal cost for a search problem, what is the worst cost solution an ϵ admissible heuristic would yield? Justify your answer.

(g) (2 points) Suggest a modification to the A* algorithm which will be guaranteed to yield an optimal solution using an ϵ-admissible heuristic with fixed, known ϵ. Justify your answer.

2. (13 points.) CSPs: A Greater (or Lesser) Chain

Consider the general less-than chain CSP below. Each of the N variables X_i has the domain $\{1 \ldots M\}$. The constraints between adjacent variables X_i and X_{i+1} require that $X_i < X_{i+1}$.



For now, assume $N = M = 5$.

(a) (1 point) How many solutions does the CSP have?

(b) (1 point) What will the domain of X_1 be after enforcing the consistency of *only* the arc $X_1 \rightarrow X_2$?

(c) (2 points) What will the domain of X_1 be after enforcing the consistency of *only* the arcs $X_2 \rightarrow X_3$ then $X_1 \rightarrow X_2$?

(d) (2 points) What will the domain of X_1 be after fully enforcing arc consistency?

Now consider the general case for arbitrary N and M.

(e) (3 points) What is the minimum number of arcs (big-O is ok) which must be processed by AC-3 (the algorithm which enforces arc consistency) on this graph before arc consistency is established?

(f) (4 points) Imagine you wish to construct a similar family of CSPs which forces one of the two following types of solutions: either all values must be ascending *or* all values must be descending, from left to right. For example, if $M = N = 3$, there would be exactly two solutions: $\{1, 2, 3\}$ and $\{3, 2, 1\}$. Explain how to formulate this variant. Your answer should include a constraint graph and precise statements of variables and constraints.

3. (18 points.) RL and MDPs: Two-Armed Bandit

Imagine you have two slot machine levers. You are playing a game where at each time step, you must pull exactly one lever. Lever A always pays a reward of 6. Lever B pays a reward of either 10 or 0. If B is a lucky lever (L=ℓ), it pays 10 with probability 4/5. If it is an unlucky one (L=$\neg\ell$), it pays 10 with probability 1/5. B is equally likely to lucky or unlucky a priori. Assume $\gamma = 1$ (which is ok for finite games, not a trick).

(a) (2 points) If you can only pull a lever once, what is the MEU?

(b) (1 point) Which action(s) (A or B or both) give that MEU?

If you play this game multiple times, it becomes more difficult to figure out what actions to take. The game is formally a POMDP, but we can turn it into an MDP in which the states encode our past outcomes from lever B. In particular, a state will be of the form (m, n), where m is the number of times we pulled B and got 10 and n is the number of times we pulled B and got 0. We begin in state $(0, 0)$. If we then pull lever B and get the outcome 10 we will go to state $(1, 0)$, while getting the 0 outcome puts us in state $(0, 1)$. Your actions are {A, B}, and the rewards are as described above.

(c) (3 points) If you will play exactly two rounds, draw the computation tree which represents the possible outcomes of the MDP. Clearly indicate which nodes are of which type (min, max, expectation, etc).

Note that if you pull lever B, the resulting payoff should change your beliefs about what kind of lever B is, and therefore what future payoffs from B might be. For example, if you get the 10 reward, your belief that B is lucky should increase.

(d) (2 points) If you are in state $(0, 1)$ and select action B, list the states you might land in and the probability you will land in them.

(e) (2 points) If you are in state $(1, 0)$ and select action B, list the states you might land in and the probability you will land in them.

(f) (3 points) On the computation tree in (c), clearly mark the probabilities on each branch of any chance nodes.

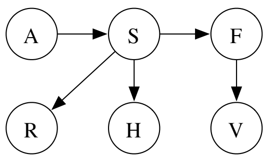
(g) (3 points) Again in this two-round setting, what is the MEU from the state state, and which first action(s) (A or B or both) give it?

(h) (2 points) If the number of plays N is large enough, the optimal first action will eventually be to pull lever B. Explain why this makes sense using concepts from reinforcement learning.

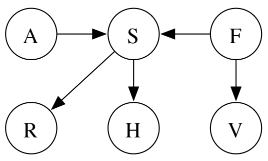
4. (15 points.) Bayes Nets: Snuffles

Assume there are two types of conditions: (S)inus congestion and (F)lu. Sinus congestion is is caused by (A)llergy or the flu.

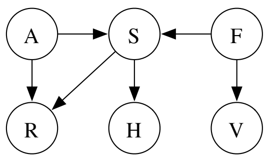
There are three observed symptoms for these conditions: (H)eadache, (R)unny nose, and fe(V)er. Runny nose and headaches are directly caused by sinus congestion (only), while fever comes from having the flu (only). For example, allergies only cause runny noses indirectly. Assume each variable is boolean.



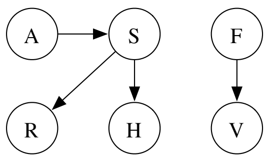
(i)



(ii)



(iii)



(iv)

(a) (2 points) Consider the four Bayes Nets shown. Circle the one which models the domain (as described above) best.

(b) (3 points) For each network, if it models the domain exactly as above, write *correct*. If it has too many conditional independence properties, write *extra independence* and state one that it has but should not have. If it has too few conditional independence properties, write *missing independence* and state one that it should have but does not have.

(i)

(ii)

(iii)

(iv)

(c) (3 points) Assume we wanted to remove the Sinus congestion (S) node. Draw the minimal Bayes Net over the remaining variables which can encode the original model's marginal distribution over the remaining variables.

(d) (2 points) In the original network you chose, which query is more efficient to compute using variable elimination: $P(F|r, v, h, a, s)$ or $P(F)$? Briefly justify.

Assume the following samples were drawn from prior sampling:

$$a, s, r, \neg h, \neg f, \neg v$$
$$a, s, \neg r, h, f, \neg v$$
$$a, \neg s, \neg r, \neg h, \neg f, \neg v$$
$$a, \neg s, \neg r, h, f, \neg v$$
$$a, s, \neg r, h, \neg f, \neg v$$

(e) (1 point) Give the sample estimate of $P(f)$ or state why it cannot be computed.

(f) (1 point) Give the sample estimate of $P(f|h)$ or state why it cannot be computed.

(g) (1 point) Give the sample estimate of $P(f|v)$ or state why it cannot be computed.

(h) (2 points) For rejection sampling in general (not necessarily on these samples), which query will require more samples to compute to a certain degree of accuracy, $P(f|h)$ or $P(f|h, a)$? Justify your answer in general terms.

5. (19 points.) HMMs: Tracking a Jabberwock

You have been put in charge of a Jabberwock for your friend Lewis. The Jabberwock is kept in a large tugley wood which is conveniently divided into an $N \times N$ grid. It wanders freely around the N^2 possible cells. At each time step $t = 1, 2, 3, \ldots$, the Jabberwock is in some cell $X_t \in \{1, \ldots, N\}^2$, and it moves to cell X_{t+1} randomly as follows: with probability $1 - \epsilon$, it chooses one of the (up to 4) valid neighboring cells uniformly at random; with probability ϵ, it uses its magical powers to teleport to a random cell uniformly at random among the N^2 possibilities (it might teleport to the same cell). Suppose $\epsilon = \frac{1}{2}, N = 10$ and that the Jabberwock always starts in $X_1 = (1, 1)$.

(a) (2 points) Compute the probability that the Jabberwock will be in $X_2 = (2, 1)$ at time step 2. What about $P(X_2 = (4, 4))$?

$P(X_2 = (2, 1)) =$

$P(X_2 = (4, 4)) =$

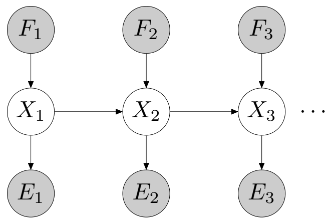
At each time step t, you don't see X_t but see E_t, which is the row that the Jabberwock is in; that is, if $X_t = (r, c)$, then $E_t = r$. You still know that $X_1 = (1, 1)$.

(b) (4 points) Suppose we see that $E_1 = 1, E_2 = 2, E_3 = 10$. Fill in the following table with the distribution over X_t after each time step, taking into consideration the evidence. Your answer should be concise. *Hint: you should not need to do any heavy calculations.*

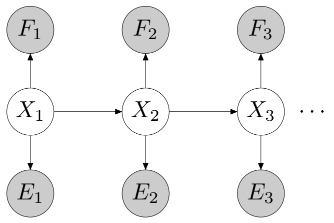
t	$P(X_t, e_{1:t-1})$	$P(X_t, e_{1:t})$
1		
2		
3		

You are a bit unsatisfied that you can't pinpoint the Jabberwock exactly. But then you remembered Lewis told you that the Jabberwock teleports only because it is frumious on that time step, and it becomes frumious independently of anything else. Let us introduce a variable $F_t \in \{0, 1\}$ to denote whether it will teleport at time t. We want to to add these frumious variables to the HMM.

Consider the two candidates:



(A)



(B)

(A)	(B)
$X_1 \perp X_3 \mid X_2$	$X_1 \perp X_3 \mid X_2$
$X_1 \perp E_2 \mid X_2$	$X_1 \perp E_2 \mid X_2$
$X_1 \perp F_2 \mid X_2$	$X_1 \perp F_2 \mid X_2$
$X_1 \perp E_4 \mid X_2$	$X_1 \perp E_4 \mid X_2$
$X_1 \perp F_4 \mid X_2$	$X_1 \perp F_4 \mid X_2$
$E_3 \perp F_3 \mid X_3$	$E_3 \perp F_3 \mid X_3$
$E_1 \perp F_2 \mid X_2$	$E_1 \perp F_2 \mid X_2$
$E_1 \perp F_2 \mid E_2$	$E_1 \perp F_2 \mid E_2$

(c) (3 points) For each model, circle the conditional independence assumptions above which are true in that model.

(d) (2 points) Which Bayes net is more appropriate for the problem domain here, (A) or (B)? Justify your answer.

For the following questions, your answers should be fully general for models of the structure shown above, not specific to the teleporting Jabberwock. For full credit, you should also simplify as much as possible (including pulling constants outside of sums, etc.).

(e) (2 points) For (A), express $P(X_{t+1}, e_{1:t+1}, f_{1:t+1})$ in terms of $P(X_t, e_{1:t}, f_{1:t})$ and the CPTs used to define the network. Assume the E and F nodes are all observed.

(f) (2 points) For (B), express $P(X_{t+1}, e_{1:t+1}, f_{1:t+1})$ in terms of $P(X_t, e_{1:t}, f_{1:t})$ and the CPTs used to define the network. Assume the E and F nodes are all observed.

Suppose that we don't actually observe the F_ts.

(g) (2 points) For (A), express $P(X_{t+1}, e_{1:t+1})$ in terms of $P(X_t, e_{1:t})$ and the CPTs used to define the network.

(h) (2 points) For (B), express $P(X_{t+1}, e_{1:t+1})$ in terms of $P(X_t, e_{1:t})$ and the CPTs used to define the network.

6. (18 points.) Classification and VPI: Cat Cravings

Consider the following Naive-Bayes model for diagnosing whether your cat is (H)ungry. Signs of hunger include that the cat is (T)hin, (M)eowing, or (W)eak.

H	$P(H)$
h	0.5
¬h	0.5

H	T	$P(T\|H)$
h	t	0.6
h	¬t	0.4
¬h	t	0.4
¬h	¬t	0.6

H	M	$P(M\|H)$
h	m	0.6
h	¬m	0.4
¬h	m	0.4
¬h	¬m	0.6

H	W	$P(W\|H)$
h	w	0.5
h	¬w	0.5
¬h	w	0.0
¬h	¬w	1.0

(a) (3 points) If your cat is thin and meowing, but not weak, what is the probability that he is hungry?

(b) (2 points) Which of the following smoothing options *might* have been applied to produce the CPTs above from training data? Circle the best answer:

(i) Laplace smoothing only might have been applied

(ii) Linear interpolation only might have been applied

(iii) Neither could have been applied

(iv) Either might have been applied

(c) (2 points) Assume that no smoothing has been applied (so these are the maximum likelihood estimates). Compute the linear interpolation smoothed estimate of $P_{\text{LIN}}(w|h)$ using $\alpha = 0.5$.

(d) (2 points) In a single word, state why smoothing is necessary.

Imagine you cannot tell whether your cat is weak or not.

(e) (2 points) Is it correct to simply skip over any unobserved evidence variables when classifying in a Naive Bayes model? That is, will you get the same answer as if you had marginalized out the missing nodes? Briefly justify why or why not.

Now return to the original probabilities, reprinted here:

H	$P(H)$
h	0.5
¬h	0.5

H	T	$P(T\|H)$
h	t	0.6
h	¬t	0.4
¬h	t	0.4
¬h	¬t	0.6

H	M	$P(M\|H)$
h	m	0.6
h	¬m	0.4
¬h	m	0.4
¬h	¬m	0.6

H	W	$P(W\|H)$
h	w	0.5
h	¬w	0.5
¬h	w	0.0
¬h	¬w	1.0

You can decide whether or not to give your cat a mega-feast (F) to counteract his (possible) hunger. Your resulting utilities are below:

H	F	$U(H, F)$
h	f	0
h	¬f	-100
¬h	f	0
¬h	¬f	10

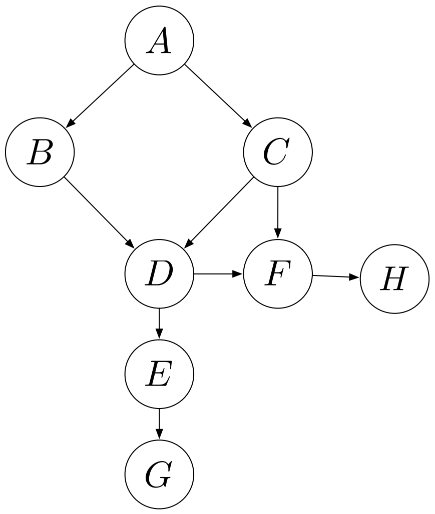
(f) (2 points) Draw the decision diagram corresponding to this decision problem.

If you do not know W, but wish to determine whether your cat is weak, you can apply the weak-o-meter test, which reveals the value of W.

(g) (3 points) In terms of *high-level quantities* (MEUs, EUs, conditional probabilities, or similar) and variables, give an expression for the maximum utility you should be willing pay to apply the weak-o-meter, assuming the cat is again thin and meowing?

(h) (2 point) What is the maximum utility you should be willing to pay, as a specific real number?

(g) [8 pts] **Bayes' Nets** For each of the conditional independence assertions given below, circle whether they are guaranteed to be true, guaranteed to be false, or cannot be determined for the given Bayes' net.



$B \perp\!\!\!\perp C$	Guaranteed true	Guaranteed false	Cannot be determined
$B \perp\!\!\!\perp C \mid G$	Guaranteed true	Guaranteed false	Cannot be determined
$B \perp\!\!\!\perp C \mid H$	Guaranteed true	Guaranteed false	Cannot be determined
$A \perp\!\!\!\perp D \mid G$	Guaranteed true	Guaranteed false	Cannot be determined
$A \perp\!\!\!\perp D \mid H$	Guaranteed true	Guaranteed false	Cannot be determined
$B \perp\!\!\!\perp C \mid A, F$	Guaranteed true	Guaranteed false	Cannot be determined
$F \perp\!\!\!\perp B \mid D, A$	Guaranteed true	Guaranteed false	Cannot be determined
$F \perp\!\!\!\perp B \mid D, C$	Guaranteed true	Guaranteed false	Cannot be determined

Q4. [12 pts] Worst-Case Markov Decision Processes

Most techniques for Markov Decision Processes focus on calculating $V^*(s)$, the maximum expected utility of state s (the expected discounted sum of rewards accumulated when starting from state s and acting optimally). This maximum expected utility $V^*(s)$ satisfies the following recursive expression, known as the Bellman Optimality Equation:

$$V^*(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s')\right].$$

In this question, instead of measuring the quality of a policy by its expected utility, we will consider the worst-case utility as our measure of quality. Concretely, $L^\pi(s)$ is the minimum utility it is possible to attain over all (potentially infinite) state-action sequences that can result from executing the policy π starting from state s. $L^*(s) = \max_\pi L^\pi(s)$ is the optimal worst-case utility. In words, $L^*(s)$ is the *greatest lower bound* on the utility of state s: the discounted sum of rewards that an agent acting optimally is guaranteed to achieve when starting in state s.

Let $C(s,a)$ be the set of all states that the agent has a non-zero probability of transferring to from state s using action a. Formally, $C(s,a) = \{s' \mid T(s,a,s') > 0\}$. This notation may be useful to you.

(a) [3 pts] Express $L^*(s)$ in a recursive form similar to the Bellman Optimality Equation.

(b) [2 pts] Recall that the Bellman update for value iteration is:

$$V_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s')\right]$$

Formally define a similar update for calculating $L_{i+1}(s)$ using L_i.

(c) [3 pts] From this point on, you can assume that $R(s,a,s') = R(s)$ (rewards are a function of the current state) and that $R(s) \geq 0$ for all s. With these assumptions, the Bellman Optimality Equation for Q-functions is

$$Q^*(s,a) = R(s) + \sum_{s'} T(s,a,s') \left[\gamma \max_{a'} Q^*(s',a')\right]$$

Let $M(s,a)$ be the *greatest lower bound* on the utility of state s when taking action a (M is to L as Q is to V). (In words, if an agent plays optimally after taking action a from state s, this is the utility the agent is guaranteed to achieve.) Formally define $M^*(s,a)$, in a recursive form similar to how Q^* is defined.

(d) [2 pts] Recall that the Q-learning update for maximizing expected utility is:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(R(s) + \gamma \max_{a'} Q(s',a')\right),$$

where α is the learning rate, $(s, a, s', R(s))$ is the sample that was just experienced ("we were in state s, we took action a, we ended up in state s', and we received a reward $R(s)$). Circle the update equation below that results in $M(s,a) = M^*(s,a)$ when run sufficiently long under a policy that visits all state-action pairs infinitely often. If more than one of the update equations below achieves this, select the one that would converge more quickly. Note that in this problem, we do not know T or C when starting to learn.

(i) $C(s,a) \leftarrow \{s'\} \cup C(s,a)$ (i.e. add s' to $C(s,a)$)

$$M(s,a) \leftarrow (1-\alpha)M(s,a) + \alpha\left(R(s) + \gamma \sum_{s' \in C(s,a)} \max_{a'} M(s',a')\right)$$

(ii) $C(s,a) \leftarrow \{s'\} \cup C(s,a)$ (i.e. add s' to $C(s,a)$)

$$M(s,a) \leftarrow (1-\alpha)M(s,a) + \alpha\left(R(s) + \gamma \min_{s' \in C(s,a)} \max_{a'} M(s',a')\right)$$

(iii) $C(s,a) \leftarrow \{s'\} \cup C(s,a)$ (i.e. add s' to $C(s,a)$)

$$M(s,a) \leftarrow R(s) + \gamma \min_{s' \in C(s,a)} \max_{a'} M(s',a')$$

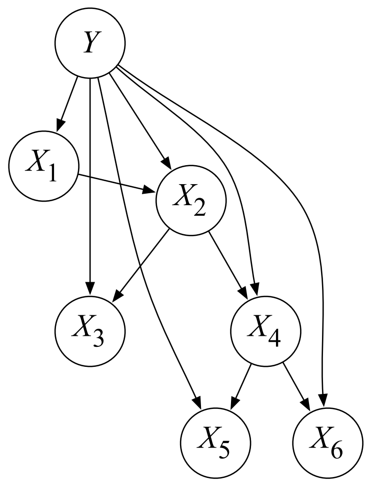
(iv) $$M(s,a) \leftarrow (1-\alpha)M(s,a) + \alpha \min\left\{M(s,a), R(s) + \gamma \max_{a'} M(s',a')\right\}.$$

(e) [1 pt] Suppose our agent selected actions to maximize $L^*(s)$, and $\gamma = 1$. What non-MDP-related technique from this class would that resemble? (a one word answer will suffice)

(f) [1 pt] Suppose our agent selected actions to maximize $L_3(s)$ (our estimate of $L^*(s)$ after 3 iterations of our "value-iteration"-like backup in section b) and $\gamma = 1$. What non-MDP-related technique from this class would that resemble? (a brief answer will suffice)

Q5. [19 pts] Tree-Augmented Naive Bayes

In section, we twice have tried to help Pacbaby distinguish his father, Pacman, from ghosts. Now Pacbaby has been transported back in time to the 1970s! Pacbaby has noticed that in the 1970s, nearly everyone who wears sunglasses also has a moustache, whether the person in question is Pacman, a ghost, or even a young Ms. Pacman. So Pacbaby decides that it's time for an upgrade from his Naive Bayes brain: he's getting a tree-augmented Naive Bayes brain so that the features he observes don't have to be independent.



In this question, we'll explore learning and inference in an abstraction of Pacbaby's new brain. A tree-augmented Naive Bayes model (TANB) is identical to a Naive Bayes model, except the features are no longer assumed conditionally independent given the class Y. Specifically, if $(X_1, X_2, \ldots, X_n)$ are the variables representing the features that Pacbaby can observe, a TANB allows $X_1, \ldots, X_n$ to be in a tree-structured Bayes net in addition to having Y as a parent. The example we explore is to the right.

(a) [1 pt] Suppose we observe no variables as evidence in the TANB above. What is the classification rule for the TANB? Write the formula in terms of the CPTs (Conditional Probability Tables) and prior probabilities in the TANB.

(b) [2 pts] Assume we observe all the variables $X_1 = x_1, X_2 = x_2, \ldots, X_6 = x_6$ in the TANB above. What is the classification rule for the TANB? Write the formula in terms of the CPTs and prior probabilites in the TANB.

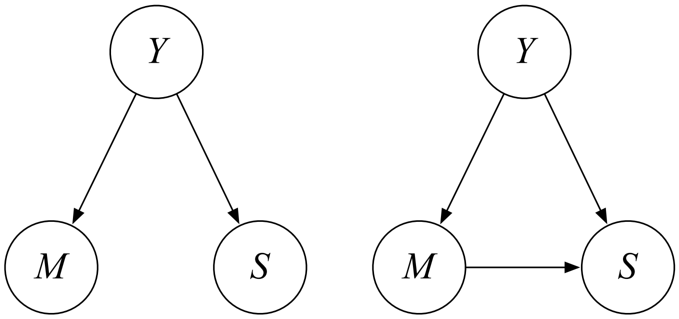
(c) [3 pts] Specify an elimination order that is efficient for the query $\mathbb{P}(Y \mid X_5 = x_5)$ in the TANB above (including Y in your ordering). How many variables are in the biggest factor (there may be more than one; if so, list only one of the largest) induced by variable elimination with your ordering? Which variables are they?

(d) [3 pts] Specify an elimination order that is efficient for the query $P(X_3 \mid X_5 = x_5)$ in the TANB above (including X_3 in your ordering). How many variables are in the biggest factor (there may be more than one; if so, list only one of the largest) induced by variable elimination with your ordering? Which variables are they?

(e) [2 pts] Does it make sense to run Gibbs sampling to do inference in a TANB? In two or fewer sentences, justify your answer.

(f) [2 pts] Suppose we are given a dataset of observations of Y and all the variables $X_1, \dots, X_6$ in the TANB above. Let C denote the total count of observations, $C(Y = y)$ denotes the number of observations of the event $Y = y$, $C(Y = y, X_i = x_i)$ denotes the count of the times the event $Y = y, X_i = x_i$ occurred, and so on. Using the C notation, write the maximum likelihood estimates for all CPTs involving the variable X_4.

(g) [2 pts] In the notation of the question above, write the Laplace smoothed estimates for all CPTs involving the variable X_4 (for amount of smoothing k).



(NB) (TANB)

(h) [2 pts] Consider the two graphs on the nodes Y (Pacbaby sees Pacman or not), M (Pacbaby sees a moustache), and S (Pacbaby sees sunglasses) above. Pacbaby observes $Y = 1$ and $Y = -1$ (Pacman or not Pacman) 50% of the time. Given $Y = 1$ (Pacman), Pacbaby observes $M = +m$ (moustache) 50% of the time and $S = +s$ (sunglasses on) 50% of the time. When Pacbaby observes $Y = -1$, the frequency of observations are identical (i.e. 50% $M = \pm m$ and 50% $S = \pm s$). In addition, Pacbaby notices that when $Y = +1$, anyone with a moustache also wears sunglasses, and anyone without a moustache does not wear sunglasses. If $Y = -1$, the presence or absence of a moustache has no influence on sunglasses. Based on this information, fill in the CPTs below (you can assume that Pacbaby has the true probabilities of the world).

For NB (left model)

y	$\mathbb{P}(Y = y)$
1	
-1	

	$\mathbb{P}(M = m \mid Y = y)$	
	$y = 1$	$y = -1$
$m = 1$		
$m = -1$		

	$\mathbb{P}(S = s \mid Y = y)$	
	$y = 1$	$y = -1$
$s = 1$		
$s = -1$		

For TANB (right model)

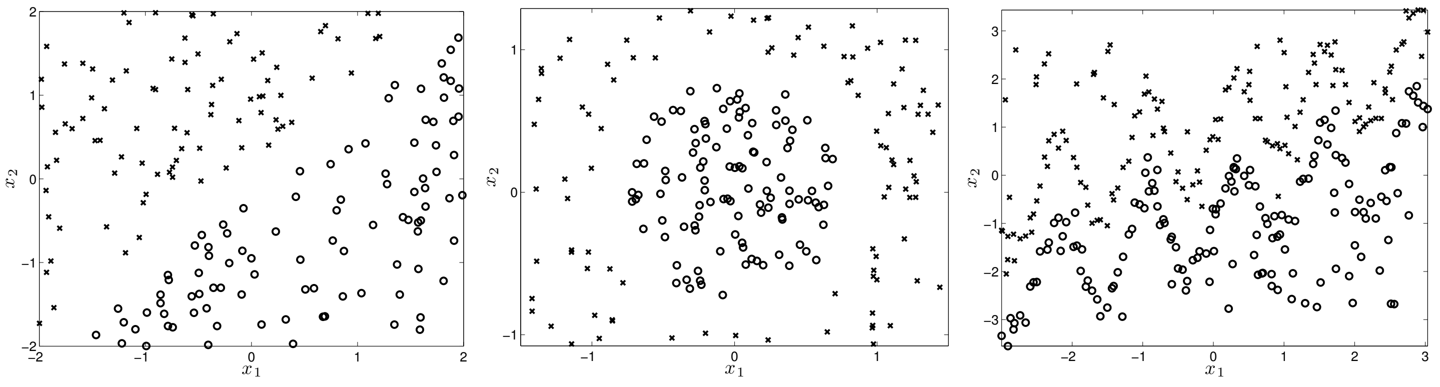
y	$\mathbb{P}(Y = y)$
1	
-1	

	$\mathbb{P}(M = m \mid Y = y)$	
	$y = 1$	$y = -1$
$m = 1$		
$m = -1$		

	$\mathbb{P}(S = s \mid Y = y, M = m)$			
	$y = 1$		$y = -1$	
	$m = 1$	$m = -1$	$m = 1$	$m = -1$
$s = 1$				
$s = -1$				

(i) [2 pts] Pacbaby sees a character with a moustache and wearing a pair of sunglasses. What prediction does the Naive Bayes model NB make? What probability does the NB model assign its prediction? What prediction does the TANB model make? What probability does the TANB-brained Pacbaby assign this prediction? Which (if any) of the predictions assigns the correct posterior probabilities?

Q6. [10 pts] Finding Working Kernels



(A) **(B)** **(C)**

The above pictures represent three distinct two-dimensional datasets with positive examples labeled as o's and negative examples labeled as x's. Consider the following three kernel functions (where $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$):

(i) Linear kernel: $K(x, z) = x^\top z = x \cdot z = x_1 z_1 + x_2 z_2$

(ii) Polynomial kernel of degree 2: $K(x, z) = (1 + x^\top z)^2 = (1 + x \cdot z)^2$

(iii) RBF (Gaussian) kernel: $K(x, z) = \exp\left(-\frac{1}{2\sigma^2}\|x - z\|^2\right) = \exp\left(-\frac{1}{2\sigma^2}(x - z)^\top (x - z)\right)$

(a) [6 pts] For each dataset (A, B, C) circle all kernels that make the dataset separable (assume $\sigma = .01$ for the RBF kernel):

Dataset (A): (i) (ii) (iii)
Dataset (B): (i) (ii) (iii)
Dataset (C): (i) (ii) (iii)

For parts (b) and (c), assume you train the perceptron using RBF (Gaussian) kernels: $K(x, z) = \exp\left(-\frac{1}{2\sigma^2}\|x - z\|^2\right)$. You run the perceptron algorithm on dataset (C) until you either encounter no more errors on the training data or you have encountered an error 1 million times and performed the associated update each time, whichever comes first.

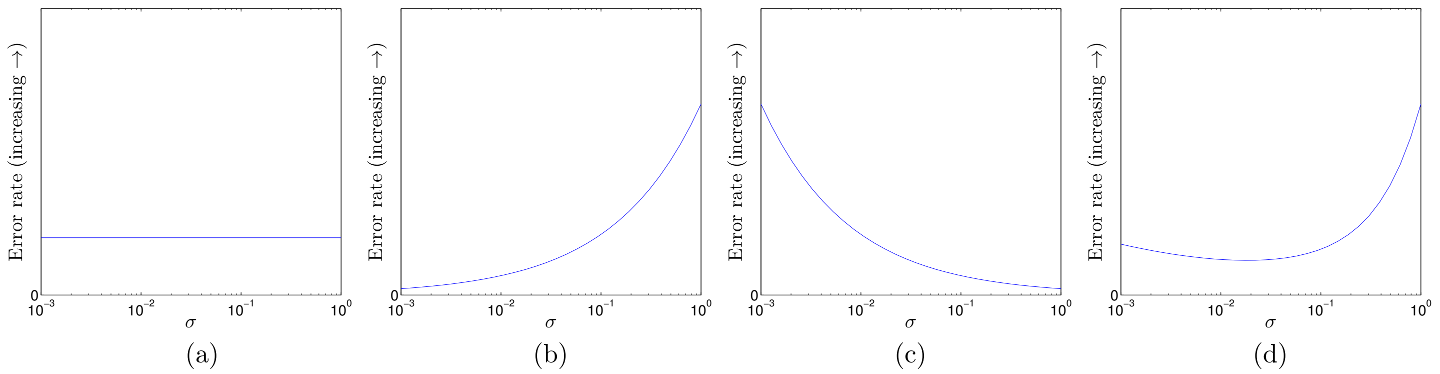


Figure 1: Possible plots of error rate (vertical axis) versus σ (horizontal axis)

(b) [2 pts] Which of the plots (a), (b), (c), or (d) in Fig. 1 is most likely to reflect the training set error rate of the learned classifier as a function of σ?

(c) [2 pts] Which of the plots (a), (b), (c), or (d) in Fig. 1 is most likely to reflect the hold-out error rate as a function of σ? Recall that "hold-out error-rate" is the error rate obtained by evaluating the classifier that was learned on training data on held-out (unused) data.

Q7. [10 pts] Learning a Ranking for Twoogle Hiring

You were just hired by Twoogle. Twoogle is expanding rapidly, and you decide to use your machine learning skills to assist them in their attempts to hire the best. To do so, you have the following available to you for each candidate i in the pool of candidates $\mathcal{I}$: (i) Their GPA, (ii) Whether they took CS164 with Hilfinger and achieved an A, (iii) Whether they took CS188 and achieved an A, (iv) Whether they have a job offer from GBook, (v) Whether they have a job offer from FacedIn, (vi) The number of misspelled words on their resume. You decide to represent each candidate $i \in \mathcal{I}$ by a corresponding 6-dimensional feature vector $f(x^{(i)})$. You believe that if you just knew the right weight vector $w \in \mathbb{R}^6$ you could reliably predict the quality of a candidate i by computing $w \cdot f(x^{(i)})$. To determine w your boss lets you sample pairs of candidates from the pool. For a pair of candidates (k, l) you can have them face off in a "twoogle-fight." The result is $\texttt{score}\,(k \succ l)$, which tells you that candidate k is at least $\texttt{score}\,(k \succ l)$ better than candidate l. Note that the score will be negative when l is a better candidate than k. Assume you collected scores for a set of pairs of candidates $\mathcal{P}$.

(a) [8 pts] Describe how you could use a perceptron-like algorithm to learn the weight vector w. Make sure to describe (i) Pseudo-code for the entire algorithm, (ii) In detail how the weight updates would be done.

(b) [2 pts] You notice that your perceptron-like algorithm is unable to reach zero errors on your training data. You ask your boss if you could get access to more information about the candidates, but you are not getting it. Is there anything else you could do to potentially improve performance on your training data?

2. (24 points.) Search and Bayes' Nets

Consider the problem of finding the *most likely explanation* in a general Bayes' net. The input is a network G in which some variables $X_{e_1} \ldots X_{e_k}$ are observed, and the output is an assignment to all the variables $X_1 \ldots X_n$, consistent with the observations, which has maximum probability. You will formulate this problem as a state space search problem. Assume that the network is constructed such that for any variable X_i, its parents Parents(X_i) are variables X_j for $j < i$.

States: each partial assignment to a prefix of the variables, of the form $\{X_1 = x_1, X_2 = x_2, \ldots X_k = x_k\}$
Initial state: the empty assignment $\{\}$
Successor function: ??
Goal test: the assignment is complete (i.e. assigns all variables)
Step cost: ??

(a) **(3 pts)** Give an expression for the size of the state space if each variable X_i has D_i elements in its domain.

(b) **(3 pts)** What is the successor function for this search problem?

(c) **(4 pts)** What is the cost function for this search problem? *Hint: Recall that* $\log ab = \log a + \log b$ *and that search minimizes total cost.*

(d) **(4 pts)** Give two reasons why BFS would be a poor choice for solving this problem.

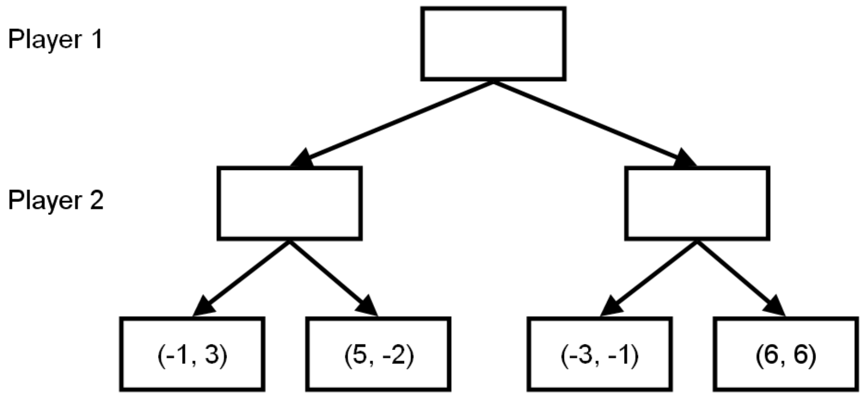
(e) **(6 pts)** Give a non-trivial admissible heuristic for this problem. You heuristic should be efficient to compute. Justify the admissibility of your heuristic briefly.

(f) **(4 pts)** Briefly describe how we might use local search to solve this problem.

3. (16 points.) Game Trees

In a two-player *non-zero-sum* game, players 1 and 2 alternate moves, just as in a minimax game. However, terminal states are not labeled with a single value $V(s)$, but rather with a pair of values $V(s) = (V_1(s), V_2(s))$ representing the utility of that terminal outcome to players 1 and 2, respectively. Each player tries to maximize their own utility, under the assumption that the other player is playing optimally (again, similar to minimax).

(a) **(4 pts)** Label each node in the following search tree with its value pair. 1-nodes represent player 1's move, while 2-nodes represent player 2's move.



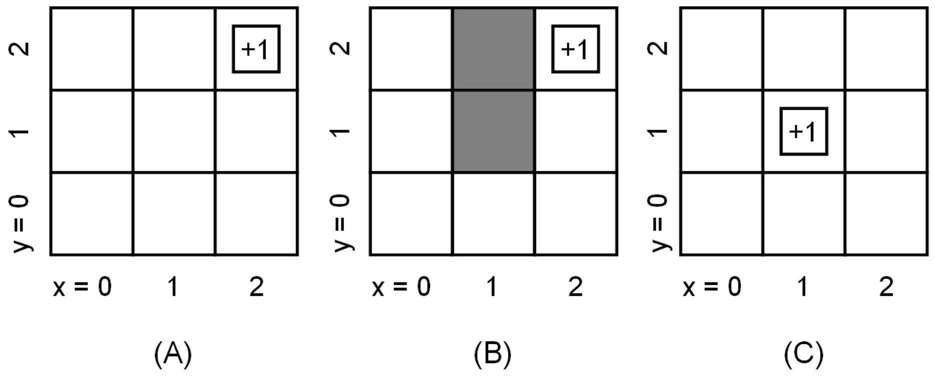
(b) **(4 pts)** Describe formally how to compute the value pair $V(s) = (V_1(s), V_2(s))$ of a node at state s under the control of player 1 (the analog of a max node).

(c) **(4 pts)** Is it possible to prune the search in a manner similar to α-β pruning? Either describe a pruning algorithm, or describe why such pruning is not possible.

(d) **(4 pts)** Would the knowledge that the game is nearly zero-sum, such as knowing that $|V_1(s) + V_2(s)| \leq k$ for all terminal states s allow you to improve your pruning algorithm or enable pruning (depending on your answer to (c))? Describe why or why not. Do not write more than a few sentences at most!

4. (16 points.) Reinforcement Learning

For the following gridworld problems, the agent can take the actions *N, S, E, W*, which move the agent one square in the respective directions. There is no noise, so these actions always take the agent in the direction attempted, unless that direction would lead off the grid or into a blocked (grey) square, in which case the action does nothing. The boxed +1 squares also permit the action X which causes the agent to exits the grid and enter the terminal state. The reward for all transitions are zero, except the exit transition, which has reward +1. Assume a discount of 0.5.



(a) **(4 pts)** Fill in the optimal values for grid (A) (*hint: this should require very little calculation*).

(b) **(3 pts)** Specify the optimal policy for grid (B) by placing an arrow in each empty square.

Imagine we have a set of real-valued features $f_i(s)$ for each non-terminal state $s = (x, y)$, and we wish to approximate the optimal utility values $V^*(s)$ by $V(s) = \sum_i w_i \cdot f_i(s)$ (linear feature-based approximation).

(c) **(3 pts)** If our features are $f_1(x, y) = x$ and $f_2(x, y) = y$, give values of w_1 and w_2 for which a one-step look-ahead policy extracted from V will be optimal in grid (A).

(d) **(2 pts)** Can we represent the actual optimal values V^* for grid (A) using these two features? Why or why not?

(e) **(4 pts)** For each of the feature sets listed below, state which (if any) of the grid MDPs above can be 'solved', in the sense that we can express some (possibly non-optimal) values which produce optimal one-step look-ahead policies.

i. $f_1(x, y) = x$ and $f_2(x, y) = y$.

ii. For each (i, j), a feature $f_{i,j}(x, y) = 1$ if $(x, y) = (i, j)$, 0 otherwise.

iii. $f_1(x, y) = (x - 1)^2$, $f_2(x, y) = (y - 1)^2$, and $f_3(x, y) = 1$.

5. (30 points.) Bayes' Nets

In the game of Minesweeper, there are bombs placed on a grid; you do not know where or how many. Assume that each square (i, j) independently has a bomb ($B_{i,j} = true$) with probability b. What you can observe for a given square is a reading $N_{i,j}$ of the number of bombs in adjacent squares (i.e. the eight closest squares *not including the square itself*). The variables $N_{i,j}$ can therefore take the values 0 through 8, plus a special value *bomb* if the square itself has a bomb (at which point the adjacent bomb count has no effect on the reading). If a square has less than 8 neighbors, such as on the boundaries, its N has an appropriately limited domain. In classic Minesweeper, you lose if you try to reveal a square with a bomb; you will ignore that complication in this problem.

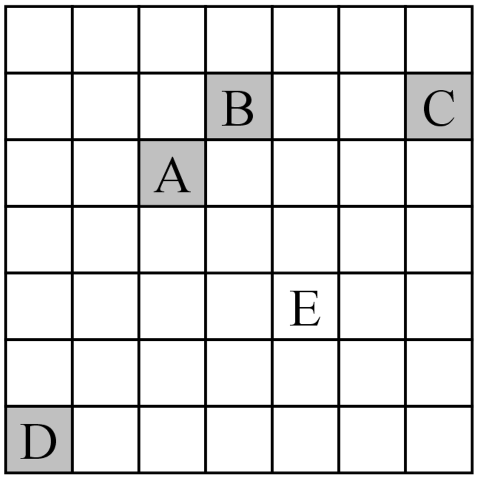
(a) **(3 pts)** Draw a Bayes' net for a one-dimensional 4x1 Minesweeper grid, showing all eight variables ($B_1 \ldots B_4$ and $N_1 \ldots N_4$). Show the minimal set of arcs needed to correctly model the domain above.

(b) **(3 pts)** Fully specify the CPTs for B_1 and N_1, assuming that there is no noise in the readings (i.e. that the number of adjacent bombs (or *bomb*) is reported exactly, deterministically). Your answers may use the bomb rate b if needed.

(c) **(3 pts)** What are the posterior probabilities of bombs in each of the four squares, given no information?

(d) **(4 pts)** If we observe $N_2 = 1$, what are the posterior probabilities of bombs in each square?

(e) **(4 pts)** On the following two-dimensional grid, assume we know the value of N_A, N_B, N_C, and N_D, and we are about to observe N_E. Shade in the squares whose posterior bomb probabilities can change as a result of this new observation.



(f) **(3 pts)** On a 2x1 grid, imagine that you must take an action by declaring which squares have bombs and which do not, so there are four possible actions on the 2x1 grid (again, note that there is no fixed number of bombs, unlike in your project or in classic Minesweeper). The utility of correctly declaring a bomb is $+1$, the utility of correctly declaring a clear square is $+1$, the utility of overlooking a bomb is -10 and the utility of declaring a bomb where there is none is -1. If the initial probability of a bomb is 0.5, what is the MEU action, and what is its EU?

(g) **(6 pts)** On the same 2x1 grid, what is the value of information about N_1?

(h) **(4 pts)** How would you modify the network in (a) if you knew that there were exactly two bombs? Draw a new network below and briefly describe/justify any new nodes you introduce.

7. (22 points.) HMMs

You sometimes get colds, which make you sneeze. You also get allergies, which make you sneeze. Sometimes you are well, which doesn't make you sneeze (much). You decide to model the process using the following HMM, with hidden states $X \in \{well, allergy, cold\}$ and observations $E \in \{sneeze, quiet\}$:

$P(X_1)$

well	1
allergy	0
cold	0

$P(X_t \mid X_{t-1} = well)$

well	0.7
allergy	0.2
cold	0.1

$P(X_t \mid X_{t-1} = allergy)$

well	0.6
allergy	0.3
cold	0.1

$P(X_t \mid X_{t-1} = cold)$

well	0.2
allergy	0.2
cold	0.6

Transitions

$P(E_t \mid X_t = well)$

quiet	1.0
sneeze	0.0

$P(E_t \mid X_t = allergy)$

quiet	0.0
sneeze	1.0

$P(E_t \mid X_t = cold)$

quiet	0.0
sneeze	1.0

Emissions

Note that colds are "stickier" in that you tend to have them for multiple days, while allergies come and go on a quicker time scale. However, allergies are more frequent. Assume that on the first day, you are well.

(a) **(2 pts)** Imagine you observe the sequence *quiet, sneeze, sneeze.* What is the probability that you were well all three days and observed these effects?

(b) **(4 pts)** What is the posterior distribution over your state on day 2 (X_2) if $E_1 = quiet$, $E_2 = sneeze$?

(c) **(4 pts)** What is the posterior distribution over your state on day 3 (X_3) if $E_1 = quiet$, $E_2 = sneeze$, $E_3 = sneeze$?

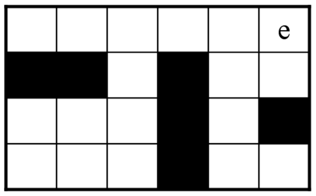
(d) **(4 pts)** What is the Viterbi (most likely) sequence for the observation sequence *quiet, sneeze, sneeze, sneeze, quiet, quiet, sneeze, quiet, quiet*? *Hint: you should not have to do extensive calculations.*

Imagine you are monitoring your state using the particle filtering algorithm, and on a given day you have 5 particles on *well*, 2 on *cold*, and 3 on *allergy* before making an observation on that day.

(e) **(4 pts)** If you observe *sneeze*, what weight will each of your particles have?

(f) **(4 pts)** After resampling, what is the expected number of particles you will have on *cold*?

1. (17 points.) Search and Utilities: Conformant Problems



Consider an agent in a maze-like grid, as shown to the right. Initially, the agent might be in any location x (including the exit), *but it does not know where it is.* The agent can move in any direction (*N, S, E, W*). Moving into a wall is legal, but does not change the agent's position. For now, assume that all actions are deterministic. The agent is trying to reach a designated exit location e where it can be rescued. However, while the agent knows the layout of the maze, it has no sensors and cannot tell where it is, or even what walls are nearby.

The agent must devise a plan which, on completion, guarantees that the agent will be in the exit location, *regardless of the (unknown) starting location.* For example, here, the agent might execute [*W,N,N,E,E,N,N,E,E,E*], after which it will be at e regardless of start position. You may find it useful to refer to $pre(x, a)$, the either empty or singleton set of squares which lead to x on a successful action a, and/or $post(x, a)$, the square resulting from x on a successful action a.

(a) (4 points) Formally state this problem as a single agent state-space search problem. You should formulate your problem so that your state space is finite (e.g. do not use an encoding where each partial plan is a state).

States:

Size of State Space:

Start state:

Successor function:

Goal test:

(b) (4 points) Give a non-trivial admissible heuristic for this problem.

Imagine the agent's movement actions may fail, causing it to stay in place with probability f. In this case, the agent can never be sure where it is, no matter what actions it takes. However, after any sequence of actions $a = a_1 \dots a_k$, we can calculate a belief state $P(x|\mathbf{a})$ over the locations x in the grid.

(c) (3 points) Give the expression from an incremental algorithm for calculating $P(x|a_1 \dots a_k)$ in terms of $P(x|a_1 \dots a_{k-1})$. Be precise (e.g., refer to x, f, and so on).

$P(x|a_1 \dots a_k) =$

Imagine the agent has a new action $a = Z$ which signals for pick-up at the exit. The agent can only use this action once, at which point the game ends. The utility for using Z if the agent is actually at the exit location is +100, but -1000 elsewhere.

(d) (3 points) If the agent has already executed movement actions $a_1 \dots a_k$, give an expression for the utility of then executing Z in terms of the quantity computed in (c).

$U(A_{k+1} = Z|a_1 \dots a_k) =$

Imagine that the agent receives a reward of -1 for each movement action taken, and wishes to find a plan which maximizes its expected utility. Assume there is no discounting. Note that despite the underlying uncertainty, this problem can be viewed as a deterministic state space search over the space of plans. Unlike your answer in (a), this formulation does not guarantee a finite search space.

(e) (3 points) Complete the statement of this version of the problem as a single agent state-space search problem. Remember that state space search *minimizes* cost and costs should be non-negative!

States: partial plans, which are strings of the form $\{N, S, E, W\}^*$ possibly followed by Z

Size of State Space: infinite

Start state: the empty plan

Successor function: append N, S, E, W, or Z if current plan does not end in Z, no successors otherwise

Goal test:

Step cost:

2. (11 points.) CSPs: Layout

You are asked to determine the layout of a new, small college. The campus will have three structures: an administration building (A), a bus stop (B), a classroom (C), and a dormitory (D). Each building must be placed somewhere on the grid below. The following constraints must be satisfied:

(i) The bust stop (B) must be adjacent to the road.

(ii) The administration building (A) and the classroom (C) must both be adjacent to the bus stop (B).

(iii) The classroom (C) must be adjacent to the dormitory (D).

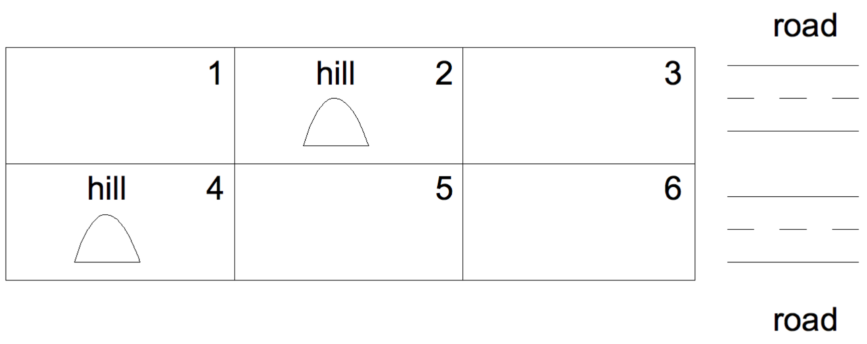
(iv) The administration building (A) must *not* be adjacent to the dormitory (D).

(v) The administration building (A) must not be on a hill.

(vi) The dormitory (D) must be on a hill or near the road.

(vii) All buildings must be in different grid squares.

Here, "adjacent" means that the buildings must share a grid edge, not just a corner.



(a) (3 points) Express the non-unary above constraints as implicit *binary* constraints over the variables A,B,C,D. Precise but evocative notation such as *different(X, Y)* is acceptable.

(b) (3 points) Cross out eliminated values to show the domains of all variables after unary constraints and arc consistency have been applied (but no variables have been assigned).

$$\begin{array}{cc} A & [\ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ ] \\ B & [\ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ ] \\ C & [\ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ ] \\ D & [\ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ ] \end{array}$$

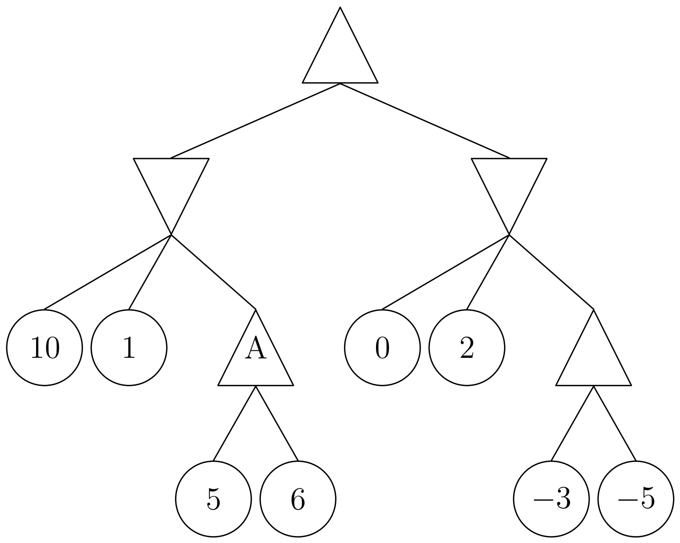
(c) (3 points) Cross out eliminated values to show the domains of the variables after $B = 3$ has been assigned and arc consistency has been rerun.

$$\begin{array}{cc} A & [\ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ ] \\ B & [\ \ \ \ \ \ \ \ 3\ \ \ \ \ \ \ \ \ \ \ ] \\ C & [\ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ ] \\ D & [\ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ ] \end{array}$$

(d) (2 points) Give a solution for this CSP or state that none exist.

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: $S = \{s_0, \ldots, s_n\}$
Initial state: s_0
Actions: A
Successor function: $\mathrm{Succ}(s, a) = s'$
Cost function: $\mathrm{Cost}(s, a) = c > 0$
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, $h(s) = -(\max_a Q(s,a))$. 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 $s \in S$

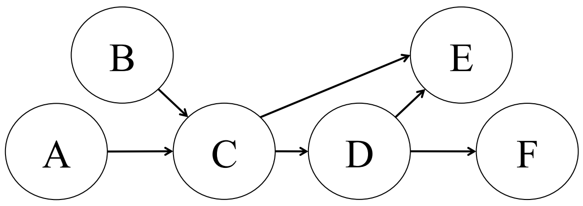
Domains: $\text{Domain}(\pi_s) = A(s)$

Constraints:

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

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 $X \in V$, that encodes a set of initial factors $\mathcal{F}_0$.

(e) (5 pt) For a query variable Q, consider computing the marginal $P(Q)$. 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) $\sum_{X \in W} |X|$
(iv) $\prod_{X \in W} |X|$
(v) $\sum_{X \in W} \left(|X| \prod_{Y \in \text{parents}(X)} |Y| \right)$
(vi) $\sum_{X \in W} \left(|X| \prod_{Y \in \text{children}(X)} |Y| \right)$

(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 $k \to \infty$?

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. $\{1, X_1, (X_1X_2)\}$

(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

1. (21 points) Everything

(a) (1 pt) CS 188

Circle the best motto for AI.

i. Maximize your expected utilities.

(b) (2 pt) Search

Circle all of the following statements that are true, if any. **Ignore ties** in all cases.

i. Breadth-first search is a special case of depth-first search. (There is a way to get depth-first search to generate the same search order as breadth-first search).

ii. Depth-first search is a special case of uniform-cost search. (There is a way to get uniform-cost search to generate the same search order as depth-first search).

iii. Uniform-cost search is a special case of A* search. (There is a way to get A* search to generate the same search order as uniform-cost search).

iv. A* search can perform breadth-first search under some class of admissible heuristics and cost functions.

v. A* search can perform depth-first search under some class of admissible heuristics and cost functions.

(c) (2 pt) CSP

For each of the heuristics, circle the **single choice** that best describes what they're doing.

A. Minimum Remaining Values (MRV)

i. Focuses on the hard parts of the graph in order to fail quickly.

ii. Focuses on the easy parts of the graph in order to postpone failure.

iii. Maximizes the chance of the solution succeeding without backtracking.

iv. Minimizes the chance of the solution succeeding without backtracking.

B. Least Constraining Value

i. Focuses on the hard parts of the graph in order to fail quickly.

ii. Focuses on the easy parts of the graph in order to postpone failure.

iii. Maximizes the chance of the solution succeeding without backtracking.

iv. Minimizes the chance of the solution succeeding without backtracking.

(d) (3 pt) Games

Say we have player MAX and player MIN playing a game with a finite number of possible moves. MAX calculates the minimax value of the root to be M. You may assume that each player has at least 2 possible actions at every turn. Also, you may assume for all parts that a different sequence of moves will always lead to a different score (**no two sequences yield the same score**). Circle all of the following statements that are true, if any.

i. Assume MIN is playing suboptimally, and MAX **does not** know this. The outcome of the game can be better than M (i.e. higher for MAX).

ii. Assume MAX **knows** player MIN is playing randomly. There exists a policy for MAX such that MAX can guarantee a better outcome than M.

iii. Assume MAX **knows** MIN is playing suboptimally at all times and **knows** the policy π_{MIN} that MIN is using (MAX knows exactly how MIN will play). There exists a policy for MAX such that MAX can guarantee a better outcome than M.

iv. Assume MAX **knows** MIN is playing suboptimally at all times but **does not know** the policy π_{MIN} that MIN is using (MAX knows MIN will choose a suboptimal action at each turn, but does not know which suboptimal action). There exists a policy for MAX such that MAX can guarantee a better outcome than M.

(e) (2 pt) MDPs

Circle all of the following statements that are true, if any.

i. If one is using value iteration and the policy (the greedy policy with respect to the values) has converged, the values must have converged as well.

ii. If one is using value iteration and the values have converged, the policy must have converged as well.

iii. Expectimax will generally run in the same amount of time as value iteration on a given MDP.

iv. Policy iteration will converge to an optimal policy.

(f) (3 pt) Reinforcement Learning

We are given an MDP (S, A, T, γ, R), and a policy π (not necessarily the optimal policy). For each of the following Bellman-like update equations, **circle the single correct choice** to match the equations with the quantity being computed (V^π, Q^π, V^*, Q^*, π^*, or none of these).

i. $g_1(s) = \max_{a \in A} \sum_{s' \in S} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a' \in A} Q^*(s',a')\right]$

(a) V^π (b) Q^π (c) V^* (d)Q^* (e)π^* (f) none of these.

ii. $g_2(s) = \arg\max_{a \in A} Q^*(s,a)$

(a) V^π (b) Q^π (c) V^* (d)Q^* (e)π^* (f) none of these.

iii. $g_3(s,a) = \sum_{s' \in S} T(s,a,s') \left[R(s,a,s') + \gamma g_3(s', \pi(s'))\right]$

(a) V^π (b) Q^π (c) V^* (d)Q^* (e)π^* (f) none of these.

iv. $g_4(s) = \sum_{s' \in S} T(s,\pi(s),s') \left[R(s,\pi(s),s') + \gamma g_4(s')\right]$

(a) V^π (b) Q^π (c) V^* (d)Q^* (e)π^* (f) none of these.

v. $g_5(s) = \max_{a \in A} \sum_{s' \in S} T(s,a,s') \left[R(s,a,s') + \gamma \max_{s'' \in S} g_5(s'')\right]$

(a) V^π (b) Q^π (c) V^* (d)Q^* (e)π^* (f) none of these.

(g) (2 pt) Probability

Circle all of the following equalities that are **always** true, if any.

i. $\mathbf{P}(A, B) = \mathbf{P}(A)\mathbf{P}(B)$

ii. $\mathbf{P}(A|B) = \mathbf{P}(A)\mathbf{P}(B)$

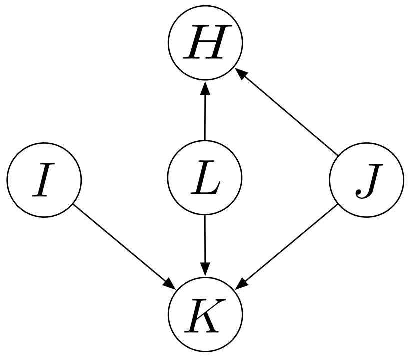
iii. $\mathbf{P}(A, B) = \mathbf{P}(A)\mathbf{P}(B) - \mathbf{P}(A|B)$

iv. $\mathbf{P}(A, B, C) = \mathbf{P}(A|B, C)\mathbf{P}(B|C)\mathbf{P}(C)$

v. $\mathbf{P}(A, B) = \sum_{c \in C} \mathbf{P}(A|B, C = c)\mathbf{P}(B|C = c)\mathbf{P}(C = c)$

(h) (2 pt) Bayes' Nets

For the Bayes' Net shown below, you start with the factors $\mathbf{P}(I), \mathbf{P}(J), \mathbf{P}(L), \mathbf{P}(K|I, L, J)$, and $\mathbf{P}(H|L, J)$.



What are the factors after joining on and eliminating J?

(i) (2 pt) Particle Filtering

Circle all of the following statements that are true, if any.

i. It is possible to use particle filtering when the state space is continuous.

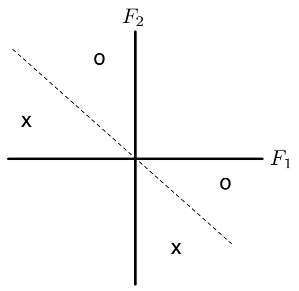
ii. It is possible to use particle filtering when the state space is discrete.

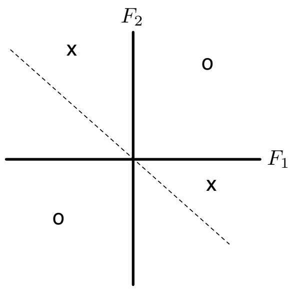
iii. As the number of particles goes to infinity, particle filtering will represent the same probability distribution that you'd get by using exact inference.

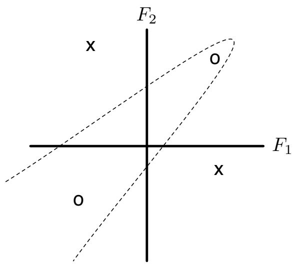
iv. Particle filtering can represent a flat distribution (i.e. uniform) with fewer particles than it would need for a more concentrated distribution (i.e. Gaussian).

(j) (1 pt) Perceptron

Suppose you have a classification problem with classes $Y = \text{X}, \text{O}$ and features F_1, F_2. You decide to use the perceptron algorithm to classify the data. Suppose you run the algorithm for each of the data sets shown below, stopping either after convergence or 1000 iterations (you may assume that if the algorithm will converge, it will within 1000 iterations). Circle all of the examples, if any, where the decision boundary could possibly be created by the perceptron algorithm.







(k) (1 pt) Inverse Reinforcement Learning

What quantity of an MDP is inverse reinforcement learning trying to estimate?

$Q(s, a)$ $R(s, a, s')$ $V(s)$ $T(s, a, s')$

3. (9 points) VPI: Crack the Code

You are defusing a bomb constructed by the evil Dr. Xor. You know the shutdown sequence is three bits B_1, B_2, B_3, $B_i \in \{0, 1\}$ and you know that an odd number of the B_i are 1. Otherwise, all sequences are equally likely. You must pick a sequence, at which point the bomb either deactivates, or not. You get a utility of 100 if you guess the correct sequences, and 0 otherwise. (Hint: you should not need to do much calculation.)

(a) (1 pt) Draw a minimal (fewest arcs) Bayes' Net that can represent the joint distribution over these variables. You only need to include B_1, B_2, and B_3 (not the utility).

(b) (1 pt) What is the MEU given no evidence?

(c) (1 pt) What is the VPI of B_1 given no information?

(d) (1 pt) What is the VPI of B_2 given B_1?

(e) (1 pt) What is the VPI of B_3 given B_1 and B_2?

At the last second, you discover that the bomb was actually set by Xor's uncreative henchman, Repeato. Repeato always uses all 1's or all 0's in his code (and is not restricted to an odd number of 1's).

(f) (1 pt) Draw a minimal (fewest arcs) Bayes' Net that can represent the joint distribution over these variables. You only need to include B_1, B_2, and B_3 (not the utility).

(g) (1 pt) What is the VPI of B_1 given no information?

(h) (1 pt) What is the VPI of B_2 given B_1?

(i) (1 pt) What is the VPI of B_3 given B_1 and B_2?

4. (11 points) MDPs: Micro-Blackjack

In micro-blackjack, you repeatedly draw a card (with replacement) that is equally likely to be a 2, 3, or 4. You can either Draw or Stop if the total score of the cards you have drawn is less than 6. Otherwise, you must Stop. When you Stop, your utility is equal to your total score (up to 5), or zero if you get a total of 6 or higher. When you Draw, you receive no utility. There is no discount ($\gamma = 1$).

(a) (2 pt) What is the state space for this MDP?

(b) (2 pt) What is the reward function for this MDP?

(c) (2 pt) Give the optimal policy for this MDP.

(d) (2 pt) What is the smallest number of rounds (k) of value iteration for which this MDP will have its exact values (if value iteration will never converge exactly, state so).

(e) (3 pt) Imagine that you run Q-learning instead of calculating values offline. You play many games, and frequently choose all actions from states that you visit. However, due to bizarre luck, each card is a 2. What will the final q-values approach in the limit if they are initialized to zero and you use a learning rate of 1/2?

5. (7 points) Search: Expanded Nodes

Consider tree search (i.e. no closed set) on an arbitrary search problem with max branching factor b. Each search node n has a backward (cumulative) cost of $g(n)$, an admissible heuristic of $h(n)$, and a depth of $d(n)$. Let c be a minimum-cost goal node, and let s be a shallowest goal node.

For each of the following, you will give an expression that characterizes the set of nodes that are expanded before the search terminates. For instance, if we asked for the set of nodes with positive heuristic value, you could say $h(n) \geq 0$. Don't worry about ties (so you won't need to worry about $>$ versus $\geq$). If there are no nodes for which the expression is true, you must write "none."

(a) **(1 pt)** Give an expression (i.e. an inequality in terms of the above quantities) for which nodes n will be expanded in a breadth-first search.

(b) **(1 pt)** Give an expression for which nodes n will be expanded in a uniform cost search.

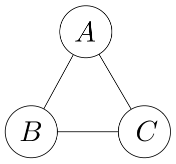
(c) **(1 pt)** Give an expression for which nodes n will be expanded in an A* search with heuristic $h(n)$.

(d) **(2 pt)** Let h_1 and h_2 be two admissible heuristics such that $\forall n, h_1(n) \geq h_2(n)$. Give an expression for the nodes which will be expanded in an A* search using h_1 but not when using h_2.

(e) **(2 pt)** Give an expression for the nodes which will be expanded in an A* search using h_2 but not when using h_1.

6. (6 points) CSPs: Arc Consistency

Consider the following CSP graph. Each variable is binary valued (0 or 1). For each of the following sets of constraints, circle all true statements, if any.



(a) (2 pt) $A = B$, $B = C$, $C = A$

(i) The CSP has no solutions, and enforcing arc consistency will expose it.

(ii) The CSP has no solutions, but enforcing arc consistency will not expose it.

(iii) The CSP has exactly one solution, and arc consistency will narrow domains to this solution.

(iv) The CSP has exactly one solution, but arc consistency will not narrow domains to this solution.

(v) The CSP has multiple solutions, and arc consistency will rule out all but one.

(vi) The CSP has multiple solutions, and arc consistency will leave domains so that all are possible.

(b) (2 pt) $A \neq B$, $B \neq C$, $C \neq A$

(i) The CSP has no solutions, and enforcing arc consistency will expose it.

(ii) The CSP has no solutions, but enforcing arc consistency will not expose it.

(iii) The CSP has exactly one solution, and arc consistency will narrow domains to this solution.

(iv) The CSP has exactly one solution, but arc consistency will not narrow domains to this solution.

(v) The CSP has multiple solutions, and arc consistency will rule out all but one.

(vi) The CSP has multiple solutions, and arc consistency will leave domains so that all are possible.

(c) (2 pt) $A < B$, $B < C$, $C < A$

(i) The CSP has no solutions, and enforcing arc consistency will expose it.

(ii) The CSP has no solutions, but enforcing arc consistency will not expose it.

(iii) The CSP has exactly one solution, and arc consistency will narrow domains to this solution.

(iv) The CSP has exactly one solution, but arc consistency will not narrow domains to this solution.

(v) The CSP has multiple solutions, and arc consistency will rule out all but one.

(vi) The CSP has multiple solutions, and arc consistency will leave domains so that all are possible.

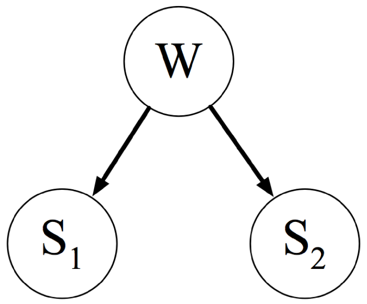
9. (12 points) Naive Bayes and Perceptron

Stoplights S_1 and S_2 can each be in one of two states: green (g) or red (r). Additionally, the machinery behind both stoplights (W) can be in one of two states: working (w) or broken (b). We collect data by observing the stoplights and the state of their machinery on seven different days. Here's a naive Bayes graphical model for the stoplights:

Data:

Day	S_1	S_2	W
1	g	r	w
2	g	r	w
3	g	r	w
4	r	g	w
5	r	g	w
6	r	g	w
7	r	r	b

Model:



(a) (1 pt) Fill in tables for $P(W)$, $P(S_1|W)$, $P(S_2|W)$ with probabilities that give the naive Bayes joint distribution that assigns highest probability to the data we observed.

W	$\mathbf{P}(W)$
w	
b	

S_1	W	$\mathbf{P}(S_1|W)$
g	w	
r	w	
g	b	
r	b	

S_2	W	$\mathbf{P}(S_2|W)$
g	w	
r	w	
g	b	
r	b	

(b) (2 pt) What's the posterior probability $P(W = b|S_1 = r, S_2 = r)$?

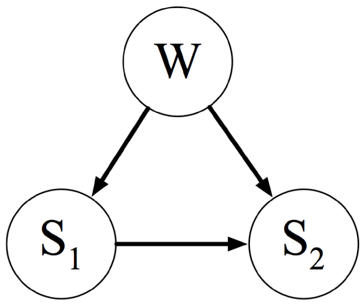
(c) (2 pt) Estimate each of $P(W)$, $P(S_1|W)$, and $P(S_2|W)$ using add-k smoothing with $k = 1$ (also smooth $P(W)$). Fill in their values in the tables below:

W	$\mathbf{P}(W)$
w	
b	

S_1	W	$\mathbf{P}(S_1|W)$
g	w	
r	w	
g	b	
r	b	

S_2	W	$\mathbf{P}(S_2|W)$
g	w	
r	w	
g	b	
r	b	

What if instead of naive Bayes we use the following graphical model and fill in probability tables with estimates that assign highest probability to the data we observed:



(d) (2 pt) What's the posterior probability $P(W = b|S_1 = r, S_2 = r)$? (Hint: you should not have to do a lot of work.)

(e) (1 pt) What is it about the problem that makes the second graphical model more apt?

(f) (2 pt) Let's see what perceptron does with the data we observed. Use only the two features f_{S_1} and f_{S_2} where $f_{S_1} = +1$ if $S_1 = g$ and $f_{S_1} = -1$ if $S_1 = r$, and similarly for f_{S_2}. Treat W as the label.

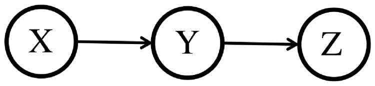
Initialize all weights to 0 and perform one pass of perceptron training on the data, doing updates in the order that the data points were observed. Break ties by choosing $W = w$. What are the final weights?

Day	S_1	S_2	W
1	g	r	w
2	g	r	w
3	g	r	w
4	r	g	w
5	r	g	w
6	r	g	w
7	r	r	b

(g) (2 pt) Will perceptron converge if you run it long enough? Justify your answer.

10. (7 points) Sampling

Assume the following Bayes net, and the corresponding distributions over the variables in the Bayes net:



X	$\mathbf{P}(X)$
$+x$	$2/5$
$-x$	$3/5$

Y	X	$\mathbf{P}(Y\|X)$
$+y$	$+x$	$2/3$
$-y$	$+x$	$1/3$
$+y$	$-x$	$3/4$
$-y$	$-x$	$1/4$

Z	Y	$\mathbf{P}(Z\|Y)$
$+z$	$+y$	$1/3$
$-z$	$+y$	$2/3$
$+z$	$-y$	$1/5$
$-z$	$-y$	$4/5$

(a) (1 pt) Your task is now to estimate $\mathbf{P}(+y|+x, +z)$ using rejection sampling. Below are some samples that have been produced by prior sampling (that is, the rejection stage in rejection sampling hasn't happened yet). Cross out whichever of the following samples that would be rejected by rejection sampling:

$$\begin{array}{lll} +x, & +y, & +z \\ -x, & +y, & +z \\ -x, & -y, & +z \\ +x, & -y, & -z \\ +x, & -y, & +z \end{array}$$

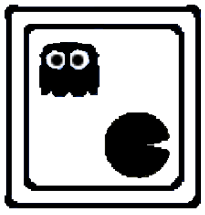
(b) (2 pt) Using rejection sampling, give an estimate of $\mathbf{P}(+y| + x, +z)$ from these samples, or state why it cannot be computed.

(c) (2 pt) Using the following samples (which were generated using likelihood weighting), estimate $\mathbf{P}(+y| + x, +z)$ using likelihood weighting, or state why it cannot be computed.

$$\begin{array}{lll} +x, & +y, & +z \\ +x, & -y, & +z \\ +x, & +y, & +z \end{array}$$

(d) (2 pt) Which query is better suited for likelihood weighting, $\mathbf{P}(Z|X)$ or $\mathbf{P}(X|Z)$? Justify your answer.

11. (7 points) Pursuit Evasion



Pacman is trapped in the following 2 by 2 maze with a hungry ghost (the horror)! When it is his turn to move, Pacman must move one step horizontally or vertically to a neighboring square. When it is the ghost's turn, he must also move one step horizontally or vertically. The ghost and Pacman alternate moves. After every move (by either the ghost or Pacman) if Pacman and the ghost occupy the same square, Pacman is eaten and receives utility -100. Otherwise, he receives a utility of 1. The ghost attempts to minimize the utility that Pacman receives. **Assume the ghost makes the first move**.

For example, with a discount factor of $\gamma = 1.0$, if the ghost moves down, then Pacman moves left, Pacman earns a reward of 1 after the ghost's move and -100 after his move for a total utility of -99.

Note that this game is not guaranteed to terminate.

(a) (1 pt) Assume a discount factor $\gamma = 0.5$, where the discount factor is applied once every time either Pacman or the ghost moves. What is the minimax value of the truncated game after 2 ghost moves and 2 Pacman moves? (Hint: you should not need to build the minimax tree)

(b) (1 pt) Assume a discount factor $\gamma = 0.5$. What is the minimax value of the complete (infinite) game? (Hint: you should not need to build the minimax tree)

(c) (2 pt) Why is value iteration superior to minimax for solving this game?

(d) (3 pt) This game is similar to an MDP because rewards are earned at every timestep. However, it is also an adversarial game involving decisions by two agents.

Let s be the state (e.g. the position of Pacman and the ghost), and let $A_P(s)$ be the space of actions available to Pacman in state s (and similarly let $A_G(s)$ be the space of actions available to the ghost). Let $N(s, a) = s'$ denote the successor function (given a starting state s, this function returns the state s' which results after taking action a). Finally, let $R(s)$ denote the utility received after moving to state s.

Write down an expression for $P^*(s)$, the value of the game to Pacman as a function of the current state s (analogous to the Bellman equations). Use a discount factor of $\gamma = 1.0$. Hint: your answer should include $P^*(s)$ on the right hand side.

$P^*(s) =$

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?

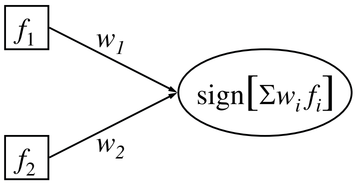
(c) (2 pts) What is the maximum depth of the search tree?

(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

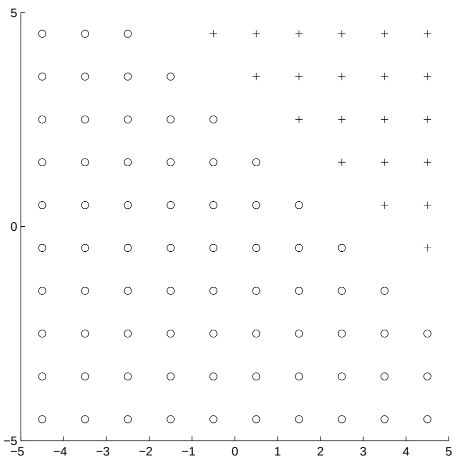
The binary perceptron depicted below has two inputs and two weights (and no fixed bias). It computes a weighted sum of its inputs and produces the symbol + if the sum is positive, and ∘ otherwise.



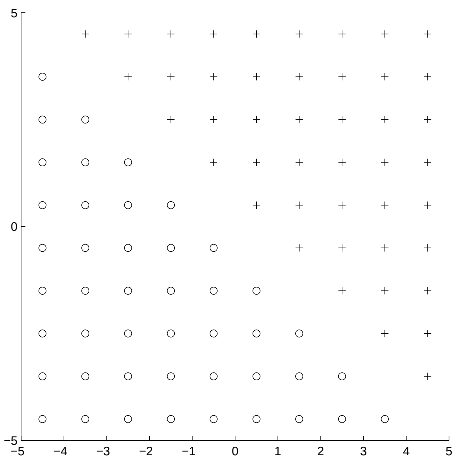
Given labeled training data points (x, y), where the the x-axis is depicted horizontally and the y-axis is vertical, we can compute various features f_1 and f_2 as inputs to the perceptron above. For each feature set (A-H) below, list the training sets (if any) which can be separated (classified perfectly).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

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

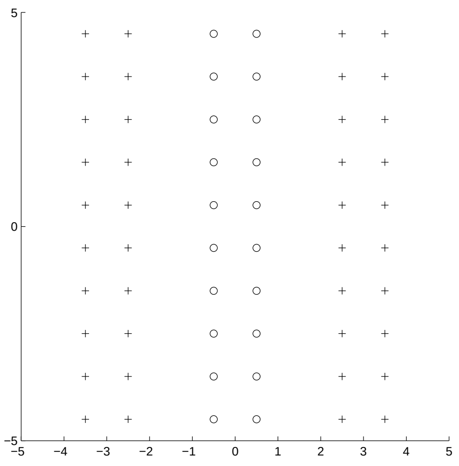
(1)



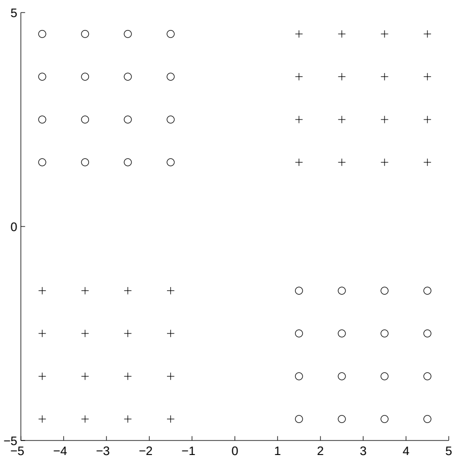
(2)



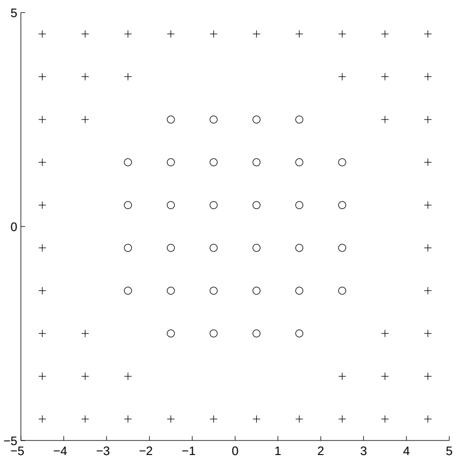
(3)



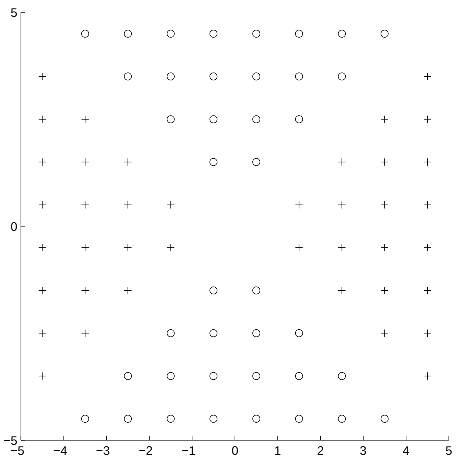
(4)



(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.

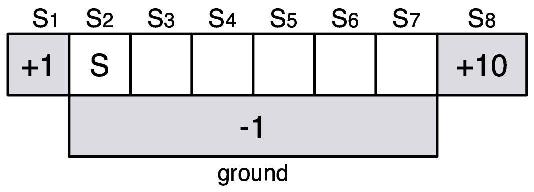
(b) (3 pts) Draw a (small) game tree in which the root node has a larger value if minimax search is used than if expectimax is used, or argue why it is not possible.

(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.

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