CS 188 Spring 2011 | Introduction to Artificial Intelligence | Final Exam Solutions

Q1. [24 pts] The OMNIBUS

Each question is worth 1 point. Leaving a question blank is worth 0 points. **Answering a multiple choice question with k possible choices incorrectly is worth $-1/(k-1)$ points (so -1 points for true/false questions, $-1/2$ for questions with three options, etc.). This gives you an expected value of 0 for random guessing.**

(a) [1 pt] **CS 188**

Circle the best motto for AI.

1. Maximize your expected utilities.

(b) [5 pts] **Search**

(i) [*true* or *false*] Uniform-cost search will never expand more nodes than A*-search.

(ii) [*true* or *false*] Depth-first search will always expand more nodes than breadth-first search.

(iii) [*true* or *false*] The heuristic $h(n) = 0$ is admissible for every search problem.

(iv) [*true* or *false*] The heuristic $h(n) = 1$ is admissible for every search problem.

(v) [*true* or *false*] The heuristic $h(n) = c(n)$, where $c(n)$ is the true cheapest cost to get from the node n to a goal state, is admissible for every search problem.

(c) [2 pts] **CSPs**

(i) [*true* or *false*] The most-constrained variable heuristic provides a way to select the next variable to assign in a backtracking search for solving a CSP.

(ii) [*true* or *false*] By using the most-constrained variable heuristic and the least-constraining value heuristic we can solve every CSP in time linear in the number of variables.

(d) [3 pts] **Games**

(i) [*true* or *false*] When using alpha-beta pruning, it is possible to get an incorrect value at the root node by choosing a bad ordering when expanding children.

(ii) [*true* or *false*] When using alpha-beta pruning, the computational savings are independent of the order in which children are expanded.

(iii) [*true* or *false*] When using expectimax to compute a policy, re-scaling the values of all the leaf nodes by multiplying them all with 10 can result in a different policy being optimal.

(e) [3 pts] **MDPs** For this question, assume that the MDP has a finite number of states.

(i) [*true* or *false*] For an MDP (S, A, T, γ, R) if we only change the reward function R the optimal policy is guaranteed to remain the same.

(ii) [*true* or *false*] Value iteration is guaranteed to converge if the discount factor (γ) satisfies $0 < \gamma < 1$.

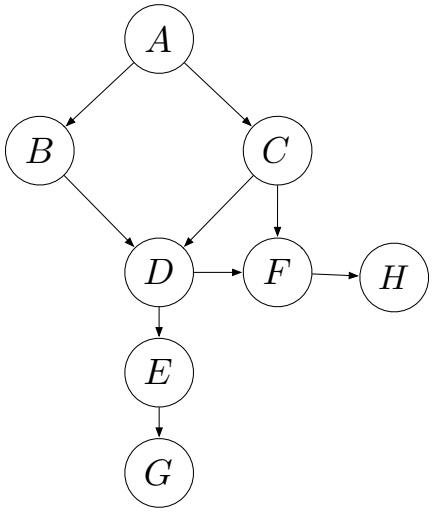
(iii) [*true* or *false*] Policies found by value iteration are superior to policies found by policy iteration.

(f) [2 pts] **Reinforcement Learning**

(i) [*true* or *false*] Q-learning can learn the optimal Q-function Q^* without ever executing the optimal policy.

(ii) [*true* or *false*] If an MDP has a transition model T that assigns non-zero probability for all triples $T(s, a, s')$ then Q-learning will fail.

(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

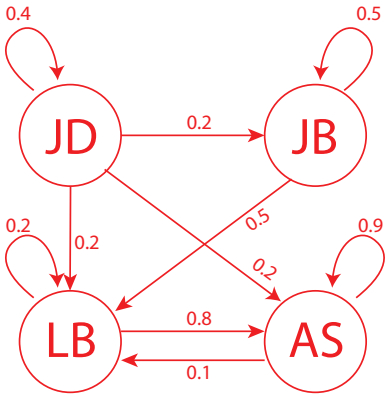
Q2. [21 pts] HMM: Where is the key?

The cs188 staff have a key to the homework bin. It is the master key that unlocks the bins to many classes, so we take special care to protect it.

Every day John Duchi goes to the gym, and on the days he has the key, 60% of the time he forgets it next to the bench press. When that happens one of the other three GSIs, equally likely, always finds it since they work out right after. Jon Barron likes to hang out at Brewed Awakening and 50% of the time he is there with the key, he forgets the key at the coffee shop. Luckily Lubomir always shows up there and finds the key whenever Jon Barron forgets it. Lubomir has a hole in his pocket and ends up losing the key 80% of the time somewhere on Euclid street. However, Arjun takes the same path to Soda and always finds the key. Arjun has a 10% chance to lose the key somewhere in the AI lab next to the Willow Garage robot, but then Lubomir picks it up.

The GSIs lose the key at most once per day, around noon (after losing it they become extra careful for the rest of the day), and they always find it the same day in the early afternoon.

(a) [2 pts] Draw on the left the Markov chain capturing the location of the key and fill in the transition probability table on the right. In this table, the entry of row JD and column JD corresponds to $P(X_{t+1} = \text{JD}|X_t = \text{JD})$, the entry of row JD and column JB corresponds to $P(X_{t+1} = \text{JB}|X_t = \text{JD})$, and so forth.



	JD_{t+1}	JB_{t+1}	LB_{t+1}	AS_{t+1}
JD_t	0.4	0.2	0.2	0.2
JB_t	0	0.5	0.5	0
LB_t	0	0	0.2	0.8
AS_t	0	0	0.1	0.9

Monday early morning Prof. Abbeel handed the key to Jon Barron. (The initial state distribution assigns probability 1 to $X_0 = \text{JB}$ and probability 0 to all other states.)

(b) [4 pts] The homework is due Tuesday at midnight so the GSIs need the key to open the bin. What is the probability for each GSI to have the key at that time? Let X_0, X_{Mon} and X_{Tue} be random variables corresponding to who has the key when Prof. Abbeel hands it out, who has the key on Monday evening, and who has the key on Tuesday evening, respectively. Fill in the probabilities in the table below.

	$P(X_0)$	$P(X_{\text{Mon}})$	$P(X_{\text{Tue}})$
JD	0	0.0	$0*.4+.5*.0+.5*.0+0*.0 = .00$
JB	1	0.5	$0*.2+.5*.5+.5*.0+0*.0 = .25$
LB	0	0.5	$0*.2+.5*.5+.5*.2+0*.1 = .35$
AS	0	0.0	$0*.2+.5*.0+.5*.8+0*.9 = .40$

(c) [3 pts] The GSIs like their jobs so much that they decide to be professional GSIs permanently. They assign an extra credit homework (make computers truly understand natural language) due *at the end of time*. What is the probability that each GSI holds the key at a point infinitely far in the future. Hint:

$$P_\infty(x) = \sum_{x'} P(X_{\text{next day}} = x \mid X_{\text{current day}} = x') P_\infty(x')$$

The goal is to compute the stationary distribution. From the Markov chain it is obvious that $P_\infty(JD) = 0$ and $P_\infty(JB) = 0$. Let $x = P_\infty(LB)$ and $y = P_\infty(AS)$. Then the definition of stationarity implies

$$x = 0.2x + 0.1y$$
$$y = .8x + .9y$$

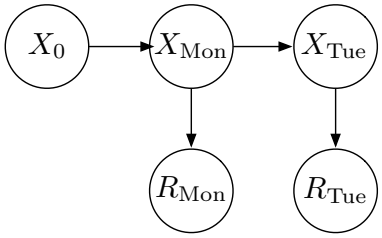
Since we must have $x \geq 0$ and $y \geq 0$, we can choose any $x > 0$ and solve for y. For example, $x = 1$ yields $y = 8$, which normalized results in $P_\infty(LB) = 1/9$ and $P_\infty(AS) = 8/9$.

Every evening the GSI who has the key feels obliged to write a short anonymous report on their opinion about the state of AI. Arjun and John Duchi are optimistic that we are right around the corner of solving AI and have an 80% chance of writing an optimistic report, while Lubomir and Jon Barron have an 80% chance of writing a pessimistic report. The following are the titles of the first few reports:

Monday: Survey: Computers Become Progressively Less Intelligent (pessimistic)
Tuesday: How to Solve Computer Vision in Three Days (optimistic)

(d) [4 pts] In light of that new information, what is the probability distribution for the key on Tuesday midnight given that Jon Barron has it Monday morning? You may leave the result as a ratio or unnormalized.

We are trying to perform inference in an HMM, so we must simply perform the forward algorithm. The HMM described in our problem is



The calculations are as follows:

	$P(X_{\text{Mon}})$	$P(X_{\text{Mon}} \mid R_{Mon} = \text{pessim.})$	$P(X_{\text{Tue}} \mid R_{Mon} = \text{pessim.})$	$P(X_{\text{Tue}} \mid R_{Mon} = \text{pessim.}, R_{Tue} = \text{optim.})$
JD	0.0	$\propto 0.0 * 0.2 \propto 0.0 = 0.0$	0.00	$\propto 0.00 * 0.8 = 0.00/0.44$
JB	0.5	$\propto 0.5 * 0.8 \propto 0.4 = 0.5$	0.25	$\propto 0.25 * 0.2 = 0.05/0.44$
LB	0.5	$\propto 0.5 * 0.8 \propto 0.4 = 0.5$	0.35	$\propto 0.35 * 0.2 = 0.07/0.44$
AS	0.0	$\propto 0.0 * 0.2 \propto 0.0 = 0.0$	0.40	$\propto 0.40 * 0.8 = 0.32/0.44$

On Thursday afternoon Prof. Abbeel noticed a suspiciously familiar key on top of the Willow Garage robot's head. He thought to himself, "This can't possibly be the master key." (He was wrong!) Lubomir managed to snatch the key and distract him before he inquired more about it and is the key holder Thursday at midnight (i.e., $X_{\text{Thu}} = \text{LB}$). In addition, the Friday report is this:

Thursday: ??? (report unknown)
Friday: AI is a scam. I know it, you know it, it is time for the world to know it! (pessimistic)

(e) [3 pts] Given that new information, what is the probability distribution for the holder of the key on Friday at midnight?

In (the extension of) the HMM above, $R_{\text{Thu}} \perp\!\!\!\perp X_{\text{Fri}} \mid X_{\text{Thu}}$, so we compute

	$P(X_{\text{Thu}})$	$P(X_{\text{Fri}})$	$P(X_{\text{Fri}} \mid R_{Fri} = \text{pessim.})$
JD	0	0	$\propto 0.2 * 0.0 = 0.0$
JB	0	0	$\propto 0.8 * 0.0 = 0.0$
LB	1	0.2	$\propto 0.8 * 0.2 = 0.5$
AS	0	0.8	$\propto 0.2 * 0.8 = 0.5$

(f) [3 pts] Prof. Abbeel recalls that he saw Lubomir holding the same key on Tuesday night. Given this new information (in addition to the information in the previous part), what is the probability distribution for the holder of the key on Friday at midnight?

The answer does not change because $X_{\text{Tue}} \perp\!\!\!\perp X_{\text{Fri}} \mid X_{\text{Thu}}$

(g) [2 pts] Suppose in addition that we know that the titles of the reports for the rest of the week are:

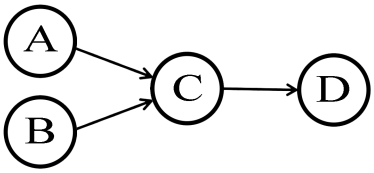
Saturday: Befriend your PC now. Soon your life will depend on its wishes (optimistic)
Sunday: How we got tricked into studying AI and how to change field without raising suspicion (pessimistic)

Will that new information change our answer to (f)? Choose one of these options:

1. Yes, reports for Saturday and Sunday affect our prediction for the key holder on Friday.
2. No, our prediction for Friday depends only on what happened in the past.

Q3. [9 pts] Sampling

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



A	$\mathbb{P}(A)$
$+a$	1/5
$-a$	4/5

B	$\mathbb{P}(B)$
$+b$	1/3
$-b$	2/3

A	B	C	$\mathbb{P}(C\|A,B)$
$+a$	$+b$	$+c$	0
$+a$	$+b$	$-c$	1
$+a$	$-b$	$+c$	0
$+a$	$-b$	$-c$	1
$-a$	$+b$	$+c$	2/5
$-a$	$+b$	$-c$	3/5
$-a$	$-b$	$+c$	1/3
$-a$	$-b$	$-c$	2/3

C	D	$\mathbb{P}(D\|C)$
$+c$	$+d$	1/2
$+c$	$-d$	1/2
$-c$	$+d$	1/4
$-c$	$-d$	3/4

(a) [1 pt] Your task is now to estimate $\mathbb{P}(+c|-a,-b,-d)$ 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 the samples that would be rejected by rejection sampling:

~~$-a \quad -b \quad +c \quad +d$~~
~~$+a \quad -b \quad -c \quad +d$~~
$-a \quad -b \quad +c \quad -d$

~~$+a \quad -b \quad -c \quad -d$~~
~~$-a \quad +b \quad +c \quad +d$~~
$-a \quad -b \quad -c \quad -d$

(b) [2 pts] Using those samples, what value would you estimate for $\mathbb{P}(+c|-a,-b,-d)$ using rejection sampling?

1/2

(c) [3 pts] Using the following samples (which were generated using likelihood weighting), estimate $\mathbb{P}(+c \mid -a,-b,-d)$ using likelihood weighting, or state why it cannot be computed.

$-a \quad -b \quad -c \quad -d$
$-a \quad -b \quad +c \quad -d$
$-a \quad -b \quad +c \quad -d$

We compute the weights of each solution, which are the product of the probabilities of the evidence variables conditioned on their parents.

$$w_1 = P(-a)P(-b)P(-d \mid -c) = 4/5 * 2/3 * 3/4$$
$$w_2 = w_3 = P(-a)P(-b)P(-d \mid +c) = 4/5 * 2/3 * 1/2$$

so normalizing, we have $(w_2 + w_3)/(w_1 + w_2 + w_3) = 4/7$.

(d) [3 pts] Below are three sequences of samples. Circle **any** sequence that could have been generated by Gibbs sampling.

Sequence 1				
1 :	$-a$	$-b$	$-c$	$+d$
2 :	$-a$	$-b$	$-c$	$+d$
3 :	$-a$	$-b$	$+c$	$+d$

Sequence 2				
1 :	$-a$	$-b$	$-c$	$+d$
2 :	$-a$	$-b$	$-c$	$-d$
3 :	$-a$	$-b$	$+c$	$+d$

Sequence 3				
1 :	$-a$	$-b$	$-c$	$+d$
2 :	$-a$	$-b$	$-c$	$-d$
3 :	$-a$	$+b$	$-c$	$-d$

The first and third sequences have at most one variable change per row, and hence could have been generated from Gibss sampling. In sequence 2, the second and third samples have both C and D changing.

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.

The Bellman Optimality Equation can be restated in terms of expected rewards:

$$V^*(s) = \max_a E_{S'}\left[R(s,a,S') + \gamma V^*(S')\right] \tag{1}$$

The worst case utility analogue is:

$$L^*(s) = \max_a \min_{s' \in C(s,a)} \left[R(s,a,s') + \gamma L^*(s')\right] \tag{2}$$

(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.

$$L_{i+1}(s) \leftarrow \max_a \min_{s' \in C(s,a)} \left[R(s,a,s') + \gamma L_i(s')\right] \tag{3}$$

(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.

$$M^*(s,a) = R(s) + \gamma \min_{s' \in C(s,a)} \max_{a'} M^*(s',a') \tag{4}$$

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

The updates (ii) and (iii) will eventually converge, but the third will do so faster, because the environment has deterministic rewards and we only care about the minimax reward. Thus if we ever observe a reward $R(s)$, we know it is correct, and if we see that the state s' can follow state s after taking action a, we must include it in our update. (iii) will be faster, though—it does not use a step size α to slow the rate.

(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)

Minimax (really, maximin)

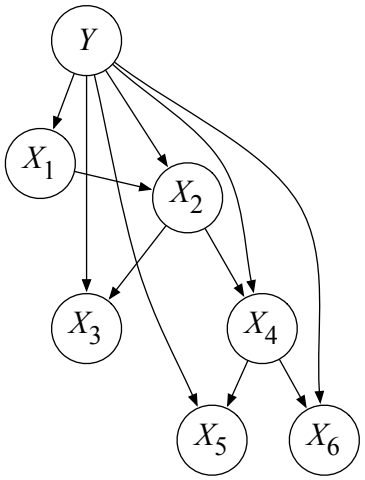
(f) [1 pt] Suppose our agent selected actions to maximize $L_3(s)$ (our estimate of $L^*(s)$ after 3 iterations of our "value-iteration"-like backup in section *b*) and $\gamma = 1$. What non-MDP-related technique from this class would that resemble? (a brief answer will suffice)

three-ply (depth limited) minimax

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.

The solution is simply the maximal y according to the prior probabilites of y: $\text{argmax}_y \ P(y)$.

(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.

We want the most probable y given the variables $X_1, \ldots, X_6$, which is (using the same reasoning as for Naive Bayes)

$$\begin{aligned}
&\operatorname*{argmax}_y \ P(y \mid x_1, \ldots, x_6) \\
&= \operatorname*{argmax}_y \ P(y, x_1, \ldots, x_6) \\
&= \operatorname*{argmax}_y \ P(y)P(x_1 \mid y)P(x_2 \mid x_1, y)P(x_3 \mid x_2, y)P(x_4 \mid x_2, y)P(x_5 \mid x_4, y)P(x_6 \mid x_4, y)
\end{aligned}$$

The rubric for this question is all-or-nothing (correct/incorrect).

(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 (the query variable Y should not be included 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?

We can ignore the variables X_3 and X_6, since when we marginalize them in the elimination order, they will sum to 1. Thus any elimination order including X_3 or X_6 is incorrect. Otherwise, we essentially just walk up the tree:

$$\begin{aligned}
P(Y \mid x_5) \propto P(Y, x_5) &= \sum_{x_1, x_2, x_3, x_4, x_6} P(Y, x_1, \ldots, x_6) \\
&= P(Y) \sum_{x_1, \ldots, x_4, x_6} P(x_1 \mid Y)P(x_2 \mid x_1, Y)P(x_3 \mid x_2, Y)P(x_4 \mid x_2, Y)P(x_5 \mid x_4, Y)P(x_6 \mid x_4, Y) \\
&= P(Y) \sum_{x_1, x_2, x_4} P(x_1 \mid Y)P(x_2 \mid x_1, Y)P(x_4 \mid x_2, Y)P(x_5 \mid x_4, Y) \underbrace{\sum_{x_3} P(x_3 \mid x_1, Y) \sum_{x_6} P(x_6 \mid x_4, Y)}_{=1} \\
&= P(Y) \sum_{x_1} P(x_1 \mid Y) \sum_{x_2} P(x_2 \mid x_1, Y) \sum_{x_4} P(x_4 \mid x_2, Y)P(x_5 \mid x_4, Y)
\end{aligned}$$

So possible orders include $X_4 \prec X_1 \prec X_2$, $X_4 \prec X_2 \prec X_1$, $X_1 \prec X_2 \prec X_4$, and $X_1 \prec X_4 \prec X_2$. Any order must start with one of X_4 or X_1, then so long as X_2 follows one of them it is correct.

Rubric: +1 point for having a correct order (may include X_6 or X_3, +1 point for *not* including X_6 or X_3, and +1 point for getting the correct factor size for the ordering provided by the student. Automatic loss of one point for eliminating X_5.

(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?

Given that X_6 is a child node of X_4, it marginalizes to 1 and has no effect on inference. So any elimination ordering including X_6 is incorrect. Other than that, we can explicitly compute the elimination by marginalizing, and we get

$$P(X_3 \mid x_5) \propto P(X_3, x_5) = \sum_{x_1,x_2,x_4,y} P(y)P(x_1 \mid y)P(x_2 \mid x_1, y)P(X_3 \mid x_2, y)P(x_5 \mid x_4, y)P(x_4 \mid x_2, y)$$
$$= \sum_y P(y) \sum_{x_2} P(X_3 \mid x_2, y) \sum_{x_1} P(x_1 \mid y)P(x_2 \mid x_1, y) \sum_{x_4} P(x_5 \mid x_4, y)P(x_4 \mid x_2, y).$$

The maximum factor size is 3; one such example above is (X_4, X_2, Y). So one possible ordering is $X_4 \prec X_1 \prec X_2 \prec Y \prec X_3$. The possible orders must have one of X_4 and X_1 first, which yields a factor over (X_2, Y) as well as factors (X_2, Y, X_3) and (Y). Then any ordering of X_2, Y will be fine. X_3 must clearly be last. Note that eliminating Y early or X_2 before eliminating one of X_4 and X_1 will yield factors of size 4, so that is incorrect.

Rubric: +1 for a correct order (potentially including X_6), +1 for ignoring X_6, and +1 for calculating the correct factor size of the provided order. Automatic loss of one point for each of eliminating X_3 too early, for eliminating X_5, and failing to eliminate any variable that must be eliminated.

(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.

No, it does not really make sense to perform Gibbs sampling. Inference is always efficient—factors are of size at most 3—because everything is a tree given the node Y. +2 for correct, +1 for indirect reason not to perform Gibbs.

(f) [2 pts] Suppose we are given a dataset of observations of Y and all the variables $X_1, \ldots, X_6$ 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.

There are 3 CPTs involving X_4: $P(X_4 \mid X_2, Y)$, $P(X_5 \mid X_4, Y)$ and $P(X_6 \mid X_4, Y)$. Their estimates are

$$P(X_4 = x_4 \mid X_2 = x_2, Y = y) = \frac{C(X_4 = x_4, X_2 = x_2, Y = y)}{C(X_2 = x_2, Y = y)}$$
$$P(X_5 = x_5 \mid X_4 = x_4, Y = y) = \frac{C(X_5 = x_5, X_4 = x_4, Y = y)}{C(X_4 = x_4, Y = y)}$$
$$P(X_6 = x_6 \mid X_4 = x_4, Y = y) = \frac{C(X_6 = x_6, X_4 = x_4, Y = y)}{C(X_4 = x_4, Y = y)}$$

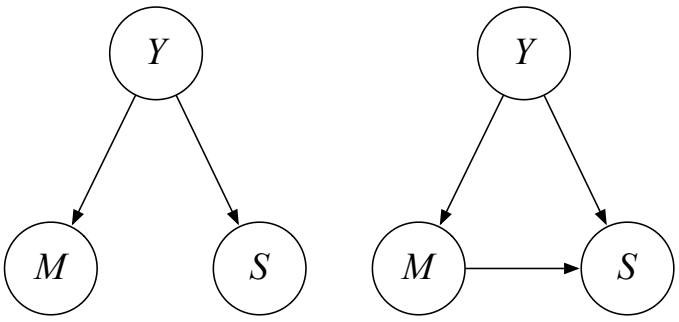
Rubric: +2 for all correct, +1 for getting at least 1, -.5 if denominators left as $\sum_{x_4} C(X_2 = x_2, Y = y, X_4 = x_4)$.

(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).

The three CPTs are the same. Let $|X_i|$ denote the size of the domain of variable X_i. Then

$$P(X_4 = x_4 \mid X_2 = x_2, Y = y) = \frac{C(X_4 = x_4, X_2 = x_2, Y = y) + k}{C(X_2 = x_2, Y = y) + k|X_4|}$$
$$P(X_5 = x_5 \mid X_4 = x_4, Y = y) = \frac{C(X_5 = x_5, X_4 = x_4, Y = y) + k}{C(X_4 = x_4, Y = y) + k|X_5|}$$
$$P(X_6 = x_6 \mid X_4 = x_4, Y = y) = \frac{C(X_6 = x_6, X_4 = x_4, Y = y) + k}{C(X_4 = x_4, Y = y) + k|X_6|}$$

Rubric: +2 for all correct based on the above.



(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	.5
-1	.5

	$\mathbb{P}(M = m \mid Y = y)$	
	$y = 1$	$y = -1$
$m = 1$	.5	.5
$m = -1$	.5	.5

	$\mathbb{P}(S = s \mid Y = y)$	
	$y = 1$	$y = -1$
$s = 1$	.5	.5
$s = -1$	.5	.5

For TANB (right model)

y	$\mathbb{P}(Y = y)$
1	.5
-1	.5

	$\mathbb{P}(M = m \mid Y = y)$	
	$y = 1$	$y = -1$
$m = 1$	.5	.5
$m = -1$	.5	.5

	$\mathbb{P}(S = s \mid Y = y, M = m)$			
	$y = 1$		$y = -1$	
	$m = 1$	$m = -1$	$m = 1$	$m = -1$
$s = 1$	1	0	.5	.5
$s = -1$	0	1	.5	.5

Rubric: +1 for all of left side correct, +1 for all of right side correct. No partial credit.

(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?

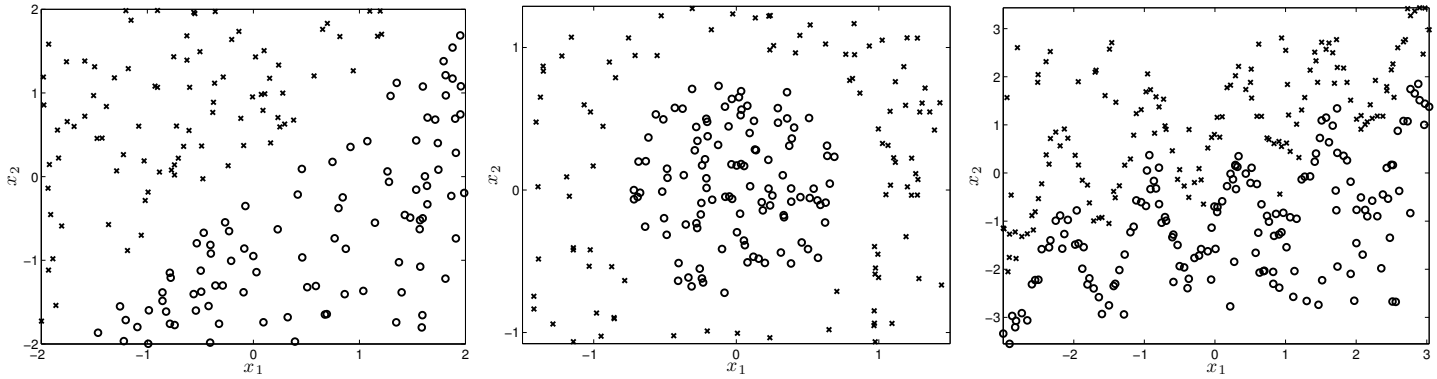
We know for the NB model that $P(Y = y \mid m, s) \propto P(Y = y, m, s) = (1/2)^3$ for any setting of Y, M, and S, so it assigns probability 1/2 to both labels (hence no prediction). The TANB model has

$$P(Y = y \mid M = 1, S = 1) \propto P(M = 1 \mid Y = y)P(S = 1 \mid M = 1, Y = y)P(y) = \begin{cases} (1/2)^2 \cdot 1 & \text{if } y = 1 \\ (1/2)^2 \cdot 1/2 & \text{if } y = -1. \end{cases}$$

Normalizing, we see that the TANB assigns probability 2/3 to $Y = 1$, i.e. Pacman, and 1/3 to $Y = -1$. Since the TANB model encodes all distributions over 3 variables, it assigns the correct posterior probabilities.

Rubric: +1 for correct NB estimate, +1 for correct TANB estimate, only +.5 if ignored normalization or assigned un-normalized values to conditional probabilities $P(Y = 1 \mid M = 1, S = 1) = .25$.

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)

(Rubric: Score is $\max\{6 - \text{Hamming distance to correct selections}, 0\}$.)

Notice that each input x has two initial features: x_1 and x_2.

The **linear** kernel can be represented by the function: $K(x, z) = x * z = x_1 z_1 + x_2 z_2$. Notice that this is a dot product between two inputs x and z, so the corresponding feature mapping is $[x_1, x_2] \to [x_1, x_2]$. When we do classification with this kernel using this feature mapping, we get the decision boundary $w_1 x_1 + w_2 x_2 \geq 0$. This corresponds to a line that passes through the origin, so this kernel can separate Dataset A – just put the corresponding line that separates the dataset.

The **quadratic** kernel can be represented by the function: $K(x, z) = (1 + x * z)^2$. Let's expand this out to $(1 + x_1 z_1 + x_2 z_2)^2 = (1 + 2x_1 z_1 + 2x_2 z_2 + 2x_1 x_2 z_1 z_2 + x_1^2 z_1^2 + x_2^2 z_2^2)$.

And that corresponds to another dot product between these feature vectors:

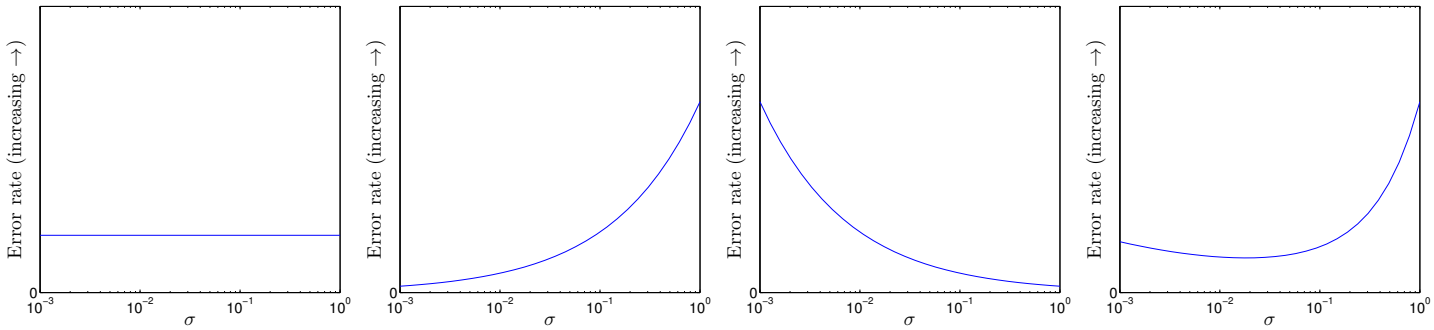
$$[1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2] * [1, \sqrt{2}z_1, \sqrt{2}z_2, \sqrt{2}z_1 z_2, z_1^2, z_2^2]$$

So, the correspond feature space transformation is: $[x_1, x_2] \to [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2]$. That means that the decision boundary rule we get is now: $w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2 \geq 0$ (Dropped the $\sqrt{2}$ term since weights are varied anyways).

Note that we have the power now to express a lot of new boundary lines. For example, if w_1 and w_2 are not zero, and every other weight is 0, then we have a linear decision boundary passing through the origin, so we can represent Dataset A. We also have all the terms for a circle. For example, if w_4, w_5 were 1, w_0 was 1, and the other weights w_1, w_2, w_3 were 0, we get $-1 + x_1^2 + x_2^2 \geq 0 \to x_1^2 + x_2^2 \geq 1$, which defines a decision boundary with a circle of radius 1. We can change the radius by changing w_0, so we can express the circle in Dataset B.

An explanation of the RBF Kernel follows below, but the intuition is that the equation looks like a circle – it's actually a Gaussian with a peak at every training point and fans out radially. The beauty of this kernel is that if you extract out the feature transformation, we jump from $[x_1, x_2]$ to an infinite dimensional feature space. This means that we can have very complex decision boundaries, including the easier ones such as Dataset A and Dataset B (by zeroing out a lot of weights), and something more complicated – like Dataset C.

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.



(a) (b) (c) (d)

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?

(b)—as σ gets very small, the classifier becomes almost like a lookup table. In particular, we see that when $x \neq z$ and $\sigma \approx 0$, then $\exp(-\frac{1}{2\sigma^2}\|x - z\|^2) \approx \exp(-\infty) = 0$, while if $x = z$ then $K(x, z) = 1$.

Again, slower: the Gaussian Kernel (the RBF kernel) is a measurement of similarity. Imagine if $x = z$, then the exponent would be 0, which means that $K(x, z) = 1$. Now, imagine if x and z were very dissimilar. Then $||x - z||^2$ would be a large number, and for a small σ value, the exponent will be a large negative number, so essentially, $K(x, z) = 0$. The shape of this is radial. Graphically, this looks like each training point emitting a radial pulse, where the pulse is strongest near the point and decreases exponentially as it fans out.

For very small σ, the training error would essentially be zero, since nothing is closer to the training point than itself. This means that you will classify your training data perfectly (unless you have two training points at the exact same location). For very large σ, you would be dividing by a large number, so the exponent essentially becomes 0, and you have more or less a uniform kernel function. This would not help in getting your training point correct because you essentially mix in an equal amount of all the other training points during your classification step (remember, you sum over the kernel functions between the current data points and each of the training points).

For the validation data, a small σ would mean that you are overfitting on the training data, and hence your validation data will have a higher error rate. We would expect this to lessen as you generalize (by increasing σ, which means you begin to mix more and more of the other training points). However, if σ is too large, you underfit, which also means that your validation error rate will increase.

(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.

(d)—as σ gets small, the training and test/hold-out error decrease, but eventually for very small σ, the classification becomes like a lookup function. Thus we have an increase in hold-out error rate for small σ

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.

For this solution, the notation $\langle w, x \rangle = w^\top x = w \cdot x$ will be used. We assume that $\texttt{score}\,(k \succ l) = -\texttt{score}\,(l \succ k)$ and that $\texttt{score}\,(k \succ l) \neq 0$ for any pair $(k, l) \in \mathcal{P}$.

The idea here is to use a margin-based perceptron. In this case, given a pair of candidates k, l, we compute $\langle w, f(x^{(k)}) \rangle$ and $\langle w, f(x^{(l)}) \rangle$, and our goal is to have $\langle w, f(x^{(k)}) \rangle \geq \langle w, f(x^{(l)}) \rangle + \texttt{score}\,(k \succ l)$ if $\texttt{score}\,(k \succ l) > 0$ (otherwise, we want the opposite inequality, that is, $\langle w, f(x^{(l)}) \rangle \geq \langle w, f(x^{(k)}) \rangle - \texttt{score}\,(k \succ l) = \langle w, f(x^{(k)}) \rangle + \texttt{score}\,(l \succ k)$).

So our perceptron-like algorithm is as follows: we repeat

1. Choose a random pair $(k, l) \in \mathcal{P}$ (you can also sequentially run through $\mathcal{P}$)
2. If $\texttt{score}\,(k \succ l) > 0$,
   - If $\langle w, f(x^{(k)}) \rangle \geq \langle w, f(x^{(l)}) \rangle + \texttt{score}\,(k \succ l)$ do nothing
   - Otherwise update $w \leftarrow w + f(x^{(k)}) - f(x^{(l)})$
3. Otherwise (if $\texttt{score}\,(k \succ l) < 0$)
   - If $\langle w, f(x^{(l)}) \rangle \geq \langle w, f(x^{(k)}) \rangle - \texttt{score}\,(k \succ l)$ do nothing
   - Otherwise update $w \leftarrow w + f(x^{(l)}) - f(x^{(k)})$

(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?

Lots of answers are possible to this question; essentially anything that is reasonable is acceptable. Some examples: you could use higher-order features, use kernels (the second-order polynomial kernel $K(x^{(k)}, x^{(l)}) = (1 + \langle f(x^{(k)}), f(x^{(l)}) \rangle)^2$ uses all polynomials of the features up to degree 2, which allows taking "ands" of the features), or more complicated perceptron algorithms in which you hold "tournaments," comparing a candidate $x^{(i)}$ against a set of other (less desirable) candidates and performing a perceptron update when $x^{(i)}$ does not outscore all of them.