CS221 Exam

CS221
Fall 2019

Name: _____
by writing my name I agree to abide by the honor code

SUNet ID: _____

Read all of the following information before starting the exam:

- This test has 3 problems printed on 28 pages and is worth 150 points total. It is your responsibility to make sure that you have all of the pages.
- Only the printed (top) side of each page will be scanned so write all your answers on that side of the paper.
- Keep your answers precise and concise. We may award partial credit so show all your work clearly and in order.
- Don't spend too much time on one problem. Read through all the problems carefully and do the easier ones first. Try to understand the problems intuitively; it really helps to draw a picture.
- You cannot use any external aids except one double-sided $8\frac{1}{2}$" x 11" page of notes.
- Good luck!

Problem	Part	Max Score	Score
1	a	15	
	b	10	
	c	10	
	d	15	
2	a	15	
	b	10	
	c	15	
	d	10	
3	a	10	
	b	15	
	c	10	
	d	15	

Total Score: ☐ + ☐ + ☐ = ☐

1. Under Attack (*50 points*)

You work for a cybersecurity company, Sanymtec, to monitor online forums. Recently, you've noticed an increasing number of curious sentences that look like this:

```
"stoafnrd scisetints fnid evenidce taht the erath is falt"
```

These sentences are perfectly readable by humans, but when you feed them into your machine learning models, they are totally confused and make wildly incorrect predictions. Your automatic fake news detection system is under attack!

a. (*15 points*)

As a first step, you would like to classify a sentence x as either adversarial ($y = 1$) or not ($y = -1$). Your boss doesn't want you to use the hinge loss because she's worried that the attacker might be able to more easily reverse engineer the system. So you decide to investigate alternative loss functions.

For each of the five loss functions $\text{Loss}(x, y, \mathbf{w})$ below:

- Determine whether $\text{Loss}(x, y, \mathbf{w})$ is usable for classification if we minimize the training loss with stochastic gradient descent (SGD).
- If the answer is yes, compute its gradient $\nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{w})$.
- If the answer is no, explain in one sentence why it is not usable.

(i) [3 points] $\text{Loss}(x, y, \mathbf{w}) = \max\{1 - \lfloor(\mathbf{w} \cdot \phi(x))y\rfloor, 0\}$, where $\lfloor a \rfloor$ returns a rounded down to the nearest integer.

(ii) [3 points] $\text{Loss}(x, y, \mathbf{w}) = \max\{(\mathbf{w} \cdot \phi(x))y - 1, 0\}$.

(iii) [3 points]

$$\text{Loss}(x, y, \mathbf{w}) = \begin{cases} 1 - 2(\mathbf{w} \cdot \phi(x))y & \text{if } (\mathbf{w} \cdot \phi(x))y \leq 0 \\ (1 - (\mathbf{w} \cdot \phi(x))y)^2 & \text{if } 0 < (\mathbf{w} \cdot \phi(x))y \leq 1 \\ 0 & \text{if } (\mathbf{w} \cdot \phi(x))y > 1 \end{cases}$$

(iv) [3 points] $\text{Loss}(x, y, \mathbf{w}) = \begin{cases} \max(1 - (\mathbf{w} \cdot \phi(x)), 0) + 10 & \text{if } y = +1 \\ \max(1 + (\mathbf{w} \cdot \phi(x)), 0) - 10 & \text{if } y = -1 \end{cases}$.

(v) [3 points] $\text{Loss}(x, y, \mathbf{w}) = \sigma(-(\mathbf{w} \cdot \phi(x))y)$, where $\sigma(z) = (1 + e^{-z})^{-1}$ is the logistic function.

b. (*10 points*)

Your next job is to decide which features to use in order to solve the classification problem. Assume you have a set D of real English words.

(i) [5 points] Suppose you have a sentence x which is a string; e.g., $x =$ `"erath is falt"`. Write the Python code for taking a sentence x and producing a `dict` representing the following two feature templates:

1. x contains word _____
2. number of words in x that are *not* in D

Assume that words in x are separated by spaces; e.g., the words in x above are `"erath"`, `"is"`, `"falt"`.

```
def featureExtractor(x, D):
    phi = defaultdict(float)

    return phi
```

(ii) [2 points] If k is the number of unique words that occur in the training set and $|D|$ is the number of words in the given set of real English words, what is the number of features that the linear classifier will have?

(iii) [3 points] Suppose that an insider leaks Sanymtec's classification strategy to the attackers. The classifier itself was not leaked, just the classification strategy behind it, which reveals that Sanymtec is using a dataset of adversarial sentences to train a classifier with the features defined in part (i).

The attackers then use this information to try modifying any fixed sentence (e.g., `"climate change is a hoax"`) into something readable by humans (e.g., `"clmaite canhge is a haox"`) but classified (incorrectly) as **non-adversarial** by Sanymtec. How can the attackers achieve this?

c. (*10 points*)

Having built a supervised classifier, you find it extremely hard to collect enough examples of adversarial sentences. On the other hand, you have a lot of non-adversarial text lying around.

(i) [3 points] Suppose you have a total of 100,000 training examples that consists of 100 adversarial sentences and 99,900 non-adversarial sentences. You train a classifier and get 99.9% accuracy. Is this a good, meaningful result? Explain why or why not.

(ii) [3 points] You decide to fit a generative model to the non-adversarial text, which is a distribution $p(x)$ that assigns a probability to each sentence x. For simplicity, let's use a unigram model:

$$p(x) = \prod_{i=1}^{n} p_{\text{u}}(w_i),$$

where $w_1, w_2, ..., w_n$ are the words in sentence x, and $p_{\text{u}}(w)$ is a probabilitiy distribution over possible w's.

Suppose you are given a single sentence `"the cat in the hat"` as training data. Compute the maximum likelihood estimate of the unigram model p_{u}:

w	$p_{\text{u}}(w)$

(iv) [4 points] Given an unseen sentence, your goal is to be able to predict whether that sentence is adversarial or not. You have a labeled dataset $\mathcal{D}_{\text{train}} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ and would like to use the unigram model to train your predictor.

How could you use $p(x)$ (from the previous problem) and $\mathcal{D}_{\text{train}}$ to obtain a predictor $f(x)$ that outputs whether a sentence x is adversarial ($y = 1$) or not ($y = -1$)? Be precise in defining $f(x)$. Hint: define a feature vector $\phi(x)$.

$$f(x) =$$

d. (*15 points*)

You notice that the adversarial words are often close to real English words. For example, you might see `"erath"` or `"eatrh"` as misspellings of `"earth"`. Furthermore, the actual number of adversarial words is rather small (it seems like the attacker just wants to reinforce the same messages). This makes you think of another unsupervised approach to try.

Let D be the set of real English words as before and $a_1, \ldots, a_n$ be the list of adversarial words you've found, and let $\text{dist}(a, e)$ be the number of edits to transform some adversarial word a to the English word e (how exactly distance is defined is unimportant).

We wish to choose K English words $e_1, \ldots, e_K \in D$ and assign each adversarial word a_i to one of the chosen English words ($z_i \in \{1, \ldots, K\}$). Each English word $e \in D$ incurs a cost $c(e)$ if we choose it as one of the K words. Our goal is to minimize the total cost of choosing $e_1, \ldots, e_K$ plus the total number of edits from adversarial words $a_1, \ldots, a_n$ to their assigned English words $e_{z_1}, \ldots, e_{z_n}$.

As an example, let $D = \{$`"earth"`, `"flat"`, `"scientists"`$\}$ with $c($`"earth"`$) = 1$, $c($`"flat"`$) = 1, c($`"scientists"`$) = 2$, and $a_1 =$ `"erath"`, $a_2 =$ `"falt"`, $a_3 =$ `"eatrh"`. Then with $K = 2$, one possible assignment (presumably the best one) is $e_1 =$ `"earth"`, $e_2 =$ `"flat"`, $z_1 = 1, z_2 = 2, z_3 = 1$.

(i) [3 points] Define a loss function that captures the optimization problem above:

$$\text{Loss}(e_1, \ldots, e_K, z_1, \ldots, z_n) =$$

(ii) [5 points] Derive an alternating minimization algorithm for optimizing the above objective. We alternate between two steps. In step 1, we optimize $z_1, \ldots, z_n$. Formally write down this update rule as an equation for each z_i where $1 \leq i \leq n$. What is the runtime? You should specify runtime with big-Oh notation in terms of n, K and/or $|D|$.

(iii) [5 points] In step 2, we optimize $e_1, \ldots, e_K$. Formally write down this update rule as an equation for each e_j where $1 \leq j \leq K$. What is the runtime? You should specify runtime with big-Oh notation in terms of n, K and/or $|D|$.

(iv) [2 points] Is the above procedure guaranteed to converge to the minimum cost solution? Explain why or why not. If not, what method what algorithm could you use with such guarantees?

2. Maze (*50 points*)

One day, you wake up to find yourself in the middle of a corn field holding an axe and a map (Figure 1). The corn field consists of an $n \times n$ grid of cells, where some adjacent cells are blocked by walls of corn stalks; specifically, for any two adjacent cells (i, j) and (i', j'), let $W((i, j), (i', j')) = 1$ if there is a wall between the two cells and 0 otherwise. For example, in Figure 1, $W((1, 1), (1, 2)) = 0$ and $W((1, 2), (1, 3)) = 1$.

You can either move to an adjacent cell if there's no intervening wall with cost 1, or you can use the axe to cut down a wall with cost c without changing your position. Your axe can be used to break down at most b_0 walls, and your goal is to get from your starting point (i_0, j_0) to the exit at (n, n) with the minimum cost.

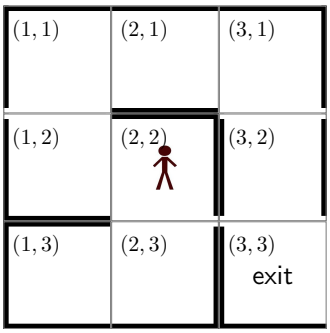

Figure 1: An example of a corn maze. The goal is to go from the initial location $(i_0, j_0) = (2, 2)$ to the exit $(n, n) = (3, 3)$ with the minimum cost.

a. (*15 points*)

(i) [10 points] Fill out the components of the search problem corresponding to the above maze.

- $s_{\text{start}} = ((i_0, j_0), b_0)$.
- $\text{Actions}(((i,j),b)) = \{a \in \{(-1,0),(+1,0),(0,-1),(0,+1)\} :$
  $(i,j)+a$ is in bounds and $(W((i,j),(i,j)+a) = 0$ or $b > 0)\}$.
- $\text{IsEnd}(((i,j),b)) =$

- $\text{Succ}(((i,j),b),a) =$

- $\text{Cost}(((i,j),b),a) =$

(ii) [5 points] When you use an axe to take down a wall, the wall stays down but the set of walls which have been taken down are not tracked in the state. Why does our choice of state still guarantee the minimum cost solution to the problem?

b. (*10 points*)

Solving the search problem above is taking forever and you don't want to be stuck in the corn maze all day long. So you decide to use A*.

(i) [5 points] Define a consistent heuristic function $h(((i, j), b))$ based on finding the minimum cost path using the relaxed state (i, j) where we assume we have an infinite axe budget and therefore do not need to track it. Show why your choice of h is consistent and what you would precompute so that evaluating any $h(((i, j), b))$ takes $O(1)$ time and precomputation takes $O(n^2 \log n)$ time.

(ii) [5 points] Noticing that sometimes h is the true future cost of the original search problem, you wonder when this holds more generally. For what ranges of b_0 and c would this hold? Assume for this part that there is a path that doesn't require breaking down any walls.

$\_\_\_\_\_ \leq b_0$ $\_\_\_\_\_ \leq c$

Your lower bounds need not be tight, but you need to formally justify why they hold.

c. (*15 points*)

Having solved the search problem above, you are eager to set out on your journey through the maze, but you realize that breaking down corn stalks is harder than you thought. Suppose that each attempt to break down a wall has an $\epsilon > 0$ probability of failing. Recall that b_0 is the maximum number of walls you can break down, not the number of attempts, and each attempt to break down a wall has cost c.

(i) [5 points] Suppose that each attempt to break down a wall is independent (e.g., if you fail once, the next attempt at the same wall also has probability ϵ of failing regardless of your previous failures). You are interested in minimizing the expected cost of exiting the maze. While the natural solution is to treat this as an MDP, it turns out you can still cast this problem as a search problem. In particular, define a modified $\text{Cost}(((i, j), b), a)$ function, and write one sentence about why this choice gives you the optimal policy.

(ii) [5 points] Suppose instead that each attempt to break down a wall is perfectly dependent (e.g., if you fail once, you will always fail to break down that wall). Let us model this problem as an MDP. What should the states of the MDP be? What is the number of states in the worst case as a function of b_0 and n (use big-Oh notation)? In this problem suppose $b_0 << n$.

(iii) [5 points] If the probability of successfully breaking down a wall is $(1-\epsilon)/k$, where $k > 0$ is the number of times you've tried to break down a wall. What should the states of the MDP be now?

d. (*10 points*)

Let's actually solve the maze! In this specific 3×3 maze as shown in Figure 1, the initial location is $(2, 2)$ and the exit is $(3, 3)$. For simplicity assume that $b_0 = 1$ and that your axe always succeeds ($\epsilon = 0$).

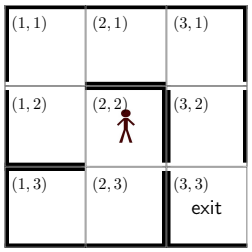

Figure 2: Same corn maze from Figure 1, repeated for convenience.

(i) [5 points] Compute the minimum achievable cost as a function of c.

(ii) [5 points] Let's look at the optimal policy at the initial location. For each value of c, what are corresponding optimal actions? If there is a tie between optimal actions state all of them. Your answer should consist of statements of the form: if $c \in$ _____, then the optimal actions are _____.

3. Faulty Accumulator (*50 points*)

You decide to try your hand at building hardware. Specifically, you will build a simple circuit that takes n numbers and incrementally computes their sum. However, it turns out hardware is hard, and in your first attempt, the accumulator occasionally gets zeroed out randomly.

To capture this precisely, we can define the following generative model whose Bayesian network is shown in Figure 3. Let $Y_0 = 0$ be the initial sum. For each time step $i = 1, \ldots, n$, the circuit:

1. Receives an input number X_i chosen uniformly from $\{1, 2, 3, 4\}$.
2. Decides to remember ($R_i = 1$) with probability $1-\epsilon$ or forget ($R_i = 0$) with probability ϵ.
3. Computes the running sum: $Y_i = R_i Y_{i-1} + X_i$, where Y_{i-1} is added depending on R_i.

As an example:

1. $X_1 = 3$, $R_1 = 1$, $Y_1 = 3$ (remember)
2. $X_2 = 2$, $R_2 = 0$, $Y_2 = 2$ (forget)
3. $X_3 = 4$, $R_3 = 1$, $Y_3 = 6$ (remember)
4. $X_4 = 4$, $R_4 = 1$, $Y_4 = 10$ (remember)

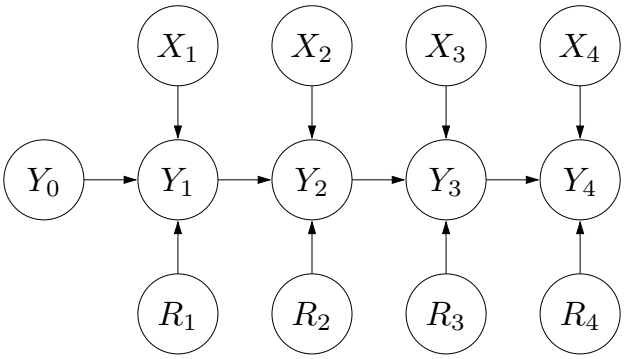

Figure 3: Bayesian network corresponding to the faulty accumulator.

a. (*10 points*)

To speed things up, you want to first prune the domains of variables. Recall that when we enforce arc consistency on a variable A with respect to a factor f, we keep a value v in the domain of A if and only if there exist values for other variables in the scope of f such that f evaluates to a non-zero number.

(i) [5 points] What is the domain of Y_n as a function of n?

(ii) [5 points] Consider the following factor, where we have marginalized out R_2:

$$p(y_2 \mid y_1, x_2) = \epsilon p(y_2 \mid y_1, x_2, r_2 = 0) + (1 - \epsilon) p(y_2 \mid y_1, x_2, r_2 = 1). \tag{1}$$

Suppose $Y_1 \in \{1, 2\}$ and $Y_2 = 3$. What is the domain of X_2 after enforcing arc consistency on X_2?

b. (*15 points*)

Now, disregarding what was done during part a, let us explore how conditioning on evidence changes our beliefs about X_2.

(i) [5 points] Compute:

x_2	$\mathbb{P}(X_2 = x_2)$
1	
2	
3	
4	

(ii) [5 points] Suppose we observe that $Y_2 = 3$. Now what do we believe about X_2?

x_2	$\mathbb{P}(X_2 = x_2 \mid Y_2 = 3)$
1	
2	
3	
4	

(iii) [5 points] Suppose we observe $Y_2 = 3$ and $Y_1 = 2$. Compute

x_2	$\mathbb{P}(X_2 \mid Y_2 = 3, Y_1 = 2)$
1	
2	
3	
4	

c. (*10 points*)

Suppose you wish to compute the posterior distribution over all other variables given $Y_1 = 3, Y_2 = 2, Y_3 = 6, Y_4 = 10$. You're getting tired of doing probabilistic inference by hand, so you decide to implement Gibbs sampling to do it. Suppose you start out with the following configuration:

i	1	2	3	4
X_i	3	2	4	4
Y_i	3	2	6	10
R_i	1	0	1	1

(i) [3 points] Compute the Gibbs sampling update for

$$\mathbb{P}(X_2 \mid \text{everything else}) = \mathbb{P}(X_2 \mid X_1, X_3, X_4, Y_1, \ldots, Y_4, R_1, \ldots, R_4) = \tag{2}$$

(ii) [3 points] Compute the Gibbs sampling update for

$$\mathbb{P}(Y_2 \mid \text{everything else}) = \mathbb{P}(Y_2 \mid X_1, \ldots, X_4, Y_1, Y_3, Y_4, R_1, \ldots, R_4) = \tag{3}$$

(iii) [4 points] What is the problem with running Gibbs sampling on this Bayesian network? What alternative would you suggest?

d. (*15 points*)

You are embarrassed to realize that not only is the circuit faulty, but also you have no clue how faulty it is (what ϵ is). You decide to estimate ϵ from data. Suppose you observe the following variables:

i	1	2	3	4
X_i	3	2	4	4
Y_i	3	2	6	10

In particular, you *do not* observe $R_1, \ldots, R_4$. Your goal is to find the ϵ which maximizes the marginal likelihood of the observed data. Let's use the EM algorithm.

(i) [5 points] Initialize $\epsilon = \epsilon_0$. For the E-step, compute the posterior:

$$\mathbb{P}(R_1, R_2, R_3, R_4 \mid X_1 = 3, X_2 = 2, X_3 = 4, X_4 = 4, Y_1 = 3, Y_2 = 2, Y_3 = 6, Y_4 = 10)$$

(ii) [5 points] For the M-step, use the posterior above to compute the updated value of ϵ (which should be a function of ϵ_0).

(iii) [5 points] Compute what ϵ converges to as you run more iterations of EM. Justify your answer mathematically.

(page left blank for scratch work)

(page left blank for scratch work)