CS221 Spring 2022: Artificial Intelligence: Principles and Techniques

Homework 2: Sentiment Analysis

SUNet ID: jchan7
Name: Jason Alexander Chan
Collaborators: None

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

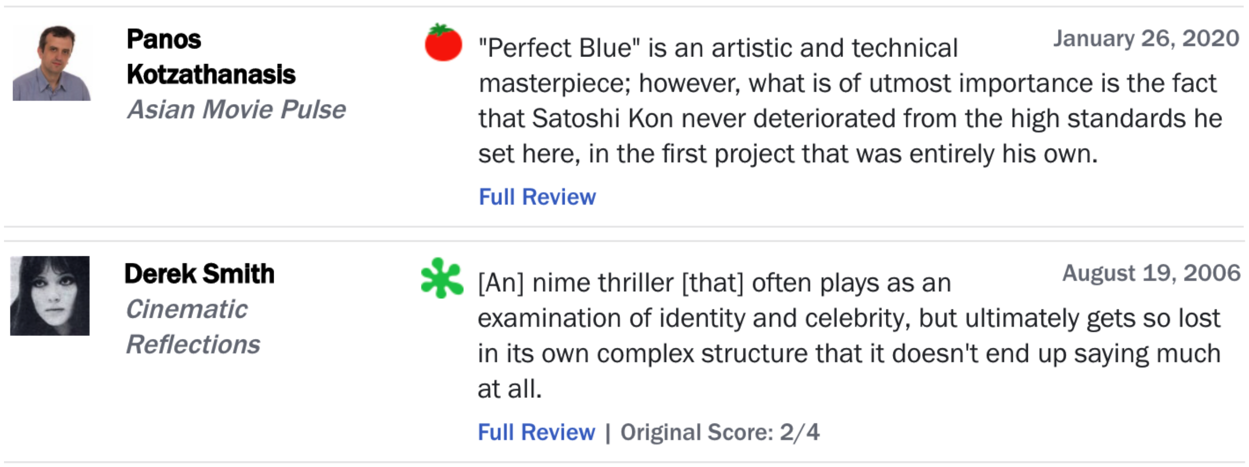
Advice for this homework:

- Words are simply strings separated by whitespace. Note that words which only differ in capitalization are considered separate (e.g. great and Great are considered different words).
- You might find some useful functions in `util.py`. Have a look around in there before you start coding.

Before you get started, please read the Assignments section on the course website thoroughly.

Problem 1: Building intuition

Here are two reviews of *Perfect Blue*, from Rotten Tomatoes:



Panos Kotzathanasis
Asian Movie Pulse

"Perfect Blue" is an artistic and technical masterpiece; however, what is of utmost importance is the fact that Satoshi Kon never deteriorated from the high standards he set here, in the first project that was entirely his own.

January 26, 2020

Full Review

Derek Smith
Cinematic Reflections

[An] nime thriller [that] often plays as an examination of identity and celebrity, but ultimately gets so lost in its own complex structure that it doesn't end up saying much at all.

August 19, 2006

Full Review | Original Score: 2/4

Rotten Tomatoes has classified these reviews as "positive" and "negative," respectively, as indicated by the intact tomato on the top and the splatter on the bottom. In this assignment, you will create a simple text classification system that can perform this task automatically. We'll warm up with the following set of four mini-reviews, each labeled positive $(+1)$ or negative (-1):

1. (-1) pretty bad
2. $(+1)$ good plot
3. (-1) not good
4. $(+1)$ pretty scenery

Each review x is mapped onto a feature vector $\phi(x)$, which maps each word to the number of occurrences of that word in the review. For example, the first review maps to the (sparse) feature vector $\phi(x) = \{\text{pretty} : 1, \text{bad} : 1\}$. Recall the definition of the hinge loss:

$$\text{Loss}_{\text{hinge}}(x, y, \mathbf{w}) = \max\{0, 1 - \mathbf{w} \cdot \phi(x) y\},$$

where x is the review text, y is the correct label, $\mathbf{w}$ is the weight vector.

a. [2 points] Suppose we run stochastic gradient descent once for each of the 4 samples in the order given above, updating the weights according to

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \text{Loss}_{\text{hinge}}(x, y, \mathbf{w}).$$

After the updates, what are the weights of the six words ("pretty", "good", "bad", "plot", "not", "scenery") that appear in the above reviews?

- Use $\eta = 0.1$ as the step size.
- Initialize $\mathbf{w} = [0, 0, 0, 0, 0, 0]$.
- The gradient $\nabla_{\mathbf{w}} \text{Loss}_{\text{hinge}}(x, y, \mathbf{w}) = 0$ when margin is exactly 1.

What we expect: A weight vector that contains a numerical value for each of the tokens in the reviews ("pretty", "good", "bad","plot", "not", "scenery"), **in this order**. For example: $[0.1, 0.2, 0.3, 0.4, 0.5, 0.6]$.

Your Solution: The final weight vector after four iterations on the 4 samples is:

$$\hat{\mathbf{w}} = \left[0, 0, -\frac{1}{10}, \frac{1}{10}, -\frac{1}{10}, \frac{1}{10}\right] \tag{1}$$

This was found by using the below equations to fill out the values in the table to evaluate the updated weights given an initial weight of zeros, a feature vector whose transformation is the count of the words in each data point (pretty, good, bad, plot, not, scenery) and $\eta = 0.1$

$$Loss_{hinge}(x, y, \hat{\mathbf{w}}) = \begin{cases} 0 & otherwise \\ 1 - \hat{\mathbf{w}} \cdot \phi(x)y & \hat{\mathbf{w}} \cdot \phi(x)y < 1 \end{cases} \tag{2}$$

$$\nabla_{\hat{\mathbf{w}}} Loss_{hinge}(x, y, \hat{\mathbf{w}}) = \begin{cases} 0 & otherwise \\ -\phi(x)y & \hat{\mathbf{w}} \cdot \phi(x)y < 1 \end{cases} \tag{3}$$

$$\hat{\mathbf{w}} \leftarrow \hat{\mathbf{w}} + \eta \nabla_{\hat{\mathbf{w}}} Loss_{hinge}(x, y, \hat{\mathbf{w}}) \tag{4}$$

x_i	y_i	$\phi(x_i)$	$-\hat{\mathbf{w}} \cdot \phi(x)y$	$\nabla_{\hat{\mathbf{w}}} Loss_{hinge}(x, y, \hat{\mathbf{w}})$	$\hat{\mathbf{wnew}}$
pretty bad	-1	$[1,0,1,0,0,0]$	0	$[1,0,1,0,0,0]$	$[-\frac{1}{10}, 0, -\frac{1}{10}, 0, 0, 0]$
good plot	1	$[0,1,0,1,0,0]$	0	$[0,-1,0,-1,0,0]$	$[-\frac{1}{10}, \frac{1}{10}, -\frac{1}{10}, \frac{1}{10}, 0, 0]$
not good	-1	$[0,1,0,0,1,0]$	$-\frac{1}{10}$	$[0,1,0,0,1,0]$	$[-\frac{1}{10}, 0, -\frac{1}{10}, \frac{1}{10}, -\frac{1}{10}, 0]$
pretty scenery	1	$[1,0,0,0,0,1]$	$-\frac{1}{10}$	$[-1,0,0,0,0,-1]$	$[0, 0, -\frac{1}{10}, \frac{1}{10}, -\frac{1}{10}, \frac{1}{10}]$

b. [2 points] Given the following dataset of reviews:

1. (-1) bad
2. $(+1)$ good
3. $(+1)$ not bad
4. (-1) not good

Prove that no linear classifier using word features can get zero error on this dataset. Remember that this is a question about classifiers, not optimization algorithms; your proof should be true for any linear classifier, regardless of how the weights are learned. Propose a single additional feature for your dataset that we could augment the feature vector with that would fix this problem.

What we expect:

1. A short written proof (∼3-5 sentences).
2. A viable feature that would allow a linear classifier to have zero error on the dataset (classify all examples correctly).

Your Solution: To get zero error on the data set for Problem 1b where feature vector is the count of (bad, good, not) for each sample, we require $Loss_{hinge}(x, y, \hat{\mathbf{w}}) = 0$

which only occurs if:

$$\hat{\mathbf{w}} \cdot \phi(x)y \geq 1 \tag{5}$$

We can evaluate the conditions for each element of $\hat{\mathbf{w}}$:

$$\hat{\mathbf{w}} \cdot \phi(x)y = (w_1x_1 + w_2x_2 + w_3x_3)y \geq 1 \tag{6}$$

x_i	y_i	$\phi(x_i)$	$\hat{\mathbf{w}} \cdot \phi(x)y \geq 1$
bad	-1	$[1, 0, 0]$	$-w_1 \geq 1$
good	1	$[0, 1, 0]$	$w_2 \geq 1$
not bad	1	$[1, 0, 1]$	$w_1 + w_3 \geq 1$
not good	-1	$[0, 1, 1]$	$-(w_2 + w_3) \geq 1$

Rearrange $w_1 + w_3 \geq 1$ to $w_3 \geq 1 - w_1$ and substitute in $-w_1 \geq 1$:

$$w_3 \geq 2 \tag{7}$$

Examining $-(w_2 + w_3) \geq 1$, we can see a contradiction emerge. Since $w_2 \geq 1$ and $w_3 \geq 2$, their sum is strictly positive. The negative of a strictly positive number $\not\geq 1$. Thus there is no $\hat{\mathbf{w}}$ that can get zero error on the data set for Problem 1b.

A viable feature that would allow a linear classifier to have zero error on the data set is (-1) 'not', 'bad', 'not bad'. This helps us better classify 'not', and 'bad'.

Problem 2: Predicting Movie Ratings

Suppose that we are now interested in predicting a numeric rating for movie reviews. We will use a non-linear predictor that takes a movie review x and returns $\sigma(\mathbf{w} \cdot \phi(x))$, where $\sigma(z) = (1 + e^{-z})^{-1}$ is the logistic function that squashes a real number to the range $(0, 1)$. For this problem, assume that the movie rating y is a real-valued variable in the range $[0, 1]$.
NOTE: Do not use math software such as Wolfram Alpha to solve this problem.

a. [2 points] Suppose that we wish to use **squared loss**. Write out the expression for $\text{Loss}(x, y, \mathbf{w})$ for a single datapoint (x, y).

What we expect: A mathematical expression for the loss. Feel free to use σ in the expression.

Your Solution: Define $\hat{y}$ as the classification prediction.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{8}$$

$$z = \mathbf{w} \cdot \phi(x) \tag{9}$$

$$\hat{y} = \sigma(\mathbf{w} \cdot \phi(x)) \tag{10}$$

Define squared loss as the below with σ in the expression.

$$(\hat{y} - y)^2 = (\sigma(\mathbf{w} \cdot \phi(x)) - y)^2 \tag{11}$$

Expanding the expression we then get

$$(\hat{y} - y)^2 = \left(\frac{1}{1 + e^{-\mathbf{w} \cdot \phi(x)}} - y\right)^2 \tag{12}$$

b. [3 points] Given $\text{Loss}(x, y, \mathbf{w})$ from the previous part, compute the gradient of the loss with respect to $\mathbf{w}$, $\nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{w})$. Write the answer in terms of the predicted value $p = \sigma(\mathbf{w} \cdot \phi(x))$.

What we expect: A mathematical expression for the gradient of the loss.

Your Solution: Let $p = \sigma(\mathbf{w} \cdot \phi(x))$ and substitute into the loss equation.

$$\text{Loss}(x, y, \mathbf{w}) = (\sigma(\mathbf{w} \cdot \phi(x)) - y)^2 \tag{13}$$

$$\text{Loss}(x, y, \mathbf{w}) = (p - y)^2 \tag{14}$$

Using the chain rule, the gradient of the loss is:

$$\nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{w}) = (\nabla_{\mathbf{w}} p) \frac{d\text{Loss}}{dp} \tag{15}$$

First find the gradient of p with respect to **w**

$$p = \sigma(\mathbf{w} \cdot \phi(x)) = \frac{1}{1 + e^{-\mathbf{w} \cdot \phi(x)}} \tag{16}$$

We use the chain rule one more time. Let t $= -\mathbf{w} \cdot \phi(x)$

$$p = \frac{1}{1 + \exp^{-t}} = (1 + e^t)^{-1} \tag{17}$$

$$\nabla_{\mathbf{w}} \text{p} = (\nabla_{\mathbf{w}} t) \frac{d\text{p}}{dt} \tag{18}$$

$$\nabla_{\mathbf{w}} \text{p} = -\phi(x) \cdot -1(1 + e^t)^{-2} \cdot e^t \tag{19}$$

Returning back to the first chain rule to find:

$$\frac{d\text{Loss}}{dp} = 2(p - y) \tag{20}$$

Finally, combining our elements

$$\nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{w}) = \phi(x) \cdot (1 + e^t)^{-2} \cdot e^t \cdot 2(p - y) \tag{21}$$

Factorise in terms of $p = (1 + e^t)^{-1}$.

$$\nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{w}) = \phi(x) \cdot p^2 \cdot e^t \cdot 2(p - y) \tag{22}$$

We can also rearrange p as: $p + p \cdot e^t = 1$ and then into $p \cdot e^t = 1 - p$

$$\nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{w}) = \phi(x) \cdot p \cdot (1 - p) \cdot 2(p - y) \tag{23}$$

c. [3 points] Suppose there is one datapoint (x, y) with some arbitrary $\phi(x)$ and $y = 1$. Specify conditions for $\mathbf{w}$ to make the magnitude of the gradient of the loss with respect to $\mathbf{w}$ arbitrarily small (i.e. minimize the magnitude of the gradient). Can the magnitude of the gradient with respect to $\mathbf{w}$ ever be exactly zero? You are allowed to make the magnitude of $\mathbf{w}$ arbitrarily large but not infinity.

What we expect:

1. 1-2 sentences describing the conditions for $\mathbf{w}$ to minimize the magnitude of the gradient
2. 1-2 sentences explaining whether the gradient can be exactly zero.

HINT: Try to understand intuitively what is going on and what each part of the expression contributes. If you find yourself doing too much algebra, you're probably doing something suboptimal.

MOTIVATION: the reason why we're interested in the magnitude of the gradients is because it governs how far gradient descent will step. For example, if the gradient is close to zero when $\mathbf{w}$ is very far from the optimum, then it could take a long time for gradient descent to reach the optimum (if at all). This is known as the *vanishing gradient problem* when training neural networks.

Your Solution: We defined loss in the question 5b as

$$\nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{w}) = \phi(x) \cdot p \cdot (1 - p) \cdot 2(p - y) \tag{24}$$

For loss to be zero we need to find the conditions for $\mathbf{w}$.

$$0 = \phi(x) \cdot p \cdot (1 - p) \cdot 2(p - y) \tag{25}$$

We have three cases

$$\begin{cases} p = 0 \\ p = 1 \\ p = y \end{cases} \tag{26}$$

Recall

$$p = \sigma(\mathbf{w} \cdot \phi(x)) = \frac{1}{1 + e^{-\mathbf{w} \cdot \phi(x)}} \tag{27}$$

For case one p=0, we want the denominator to be extremely large. Therefore want $e^{-\mathbf{w} \cdot \phi(x)}$ to be large. Thus, we want $\mathbf{w} \to -\infty$

For case two p=1, we want the denominator to be 1. Therefore we want $e^{-\mathbf{w} \cdot \phi(x)}$ to get as close to zero as possible. Thus, we want $\mathbf{w} \to +\infty$.

For case three p=y, the loss will always be zero if the prediction is exactly the same as the actual label. In this case it doesn't matter what $\mathbf{w}$ is set to.

Problem 3: Sentiment Classification

In this problem, we will build a binary linear classifier that reads movie reviews and guesses whether they are "positive" or "negative.".

NOTE: Do not import any outside libraries (e.g. numpy) for any of the coding parts. Only standard python libraries and/or the libraries imported in the starter code are allowed. In this problem, you must implement the functions without using libraries like Scikit-learn.

a. [2 points] Implement the function `extractWordFeatures`, which takes a review (string) as input and returns a feature vector $\phi(x)$ (you should represent the vector $\phi(x)$ as a `dict` in Python).

b. [4 points] Implement the function `learnPredictor` using stochastic gradient descent and minimize the hinge loss. Print the training error and validation error after each epoch to make sure your code is working. You must get less than 4% error rate on the training set and less than 30% error rate on the validation set to get full credit.

c. [2 points] Write the `generateExample` function (nested in the `generateDataset` function) to generate artificial data samples.

   Use this to double check that your `learnPredictor` works! You can do this by using `generateDataset()` to generate training and validation examples. You can then pass in these examples as `trainExamples` and `validationExamples` respectively to `learnPredictor` with the identity function `lambda x:x` as a `featureExtractor`.

d. [2 points] Some languages are written without spaces between words. So is splitting the words really necessary or can we just naively consider strings of characters that stretch across words? Implement the function `extractCharacterFeatures` (by filling in the `extract` function), which maps each string of n characters to the number of times it occurs, ignoring whitespace (spaces and tabs).

e. [3 points] Run your linear predictor with feature extractor `extractCharacterFeatures`. Experiment with different values of n to see which one produces the smallest validation error. You should observe that this error is nearly as small as that produced by word features. Why is this the case? Construct a review (one sentence max) in which character n-grams probably outperform word features, and briefly explain why this is so.

   NOTE: There is a function in `submission.py` that will allow you add a test to `grader.py` to test different values of n. Remember to write your final written solution here.

What we expect:

1. A short paragraph (4-6 sentences). In the paragraph state which value of n produces the smallest validation error, why this is likely the value that produces the smallest error.
2. A one-sentence review and explanation for when character n-grams probably outperform word features.

Your Solution:

- When N=5 for n-grams, the validation error is closest the validation error of word based features, which is 0.273. The likely reason for this is that N=5 matches the average word length in the English language, which is estimated to be five letters per Peter Norvig's English Letter Frequency Counts, accessed via http://norvig.com/mayzner.html.

  When N=7 for n-grams, the lowest validation error of 0.271 is produced. But the validation error for $N = 5, 6, 7$ are all quite similar: 0.274, 0.273 and 0.271 respectively. The validation error begins to increase again when N=8 and N=9.
- For n-grams to outperform word based features I'm hypothesising that the review needs uncommon compound words. One such review could be: *The director's freestyle methodology left us thunderstruck*.

Problem 4: Toxicity Classification and Maximum Group Loss

Recall that models trained (in the standard way) to minimize the average loss can work well on average but poorly on certain groups, and that we can mitigate this issue by minimizing the maximum group loss instead. In this problem, we will compare the average loss and maximum group loss objectives on a toy setting inspired by a problem with real-world toxicity classification models.

Toxicity classifiers are designed to assist in moderating online forums by predicting whether an online comment is toxic or not, so that comments predicted to be toxic can be flagged for humans to review [1]. Unfortunately, such models have been observed to be biased: non-toxic comments mentioning demographic identities often get misclassified as toxic (e.g., "I am a [demographic identity]") [2]. These biases arise because toxic comments often mention and attack demographic identities, and as a result, models learn to *spuriously correlate* toxicity with the mention of these identities.

In this problem, we will study a toy setting that illustrates the spurious correlation problem: The input x is a comment (a string) made on an online forum; the label $y \in \{-1, 1\}$ is the toxicity of the comment ($y = 1$ is toxic, $y = -1$ is non-toxic); $d \in \{0, 1\}$ indicates if the text contains a word that refers to a demographic identity; and $t \in \{0, 1\}$ indicates whether the comment includes certain "toxic" words. The comment x is mapped onto the feature vector $\phi(x) = [1, d, t]$ where 1 is the bias term (the bias term is present to prevent the edge case $\mathbf{w} \cdot \phi(x) = 0$ in the questions that follow). To make this concrete, we provide a few simple examples below, where we underline toxic words and words that refer to a demographic identity:

Comment (x)	Toxicity (y)	Presence of demographic mentions (d)	Presence of toxic words (t)
"Stanford <u>sucks</u>!"	1	0	1
"I'm a <u>woman</u> in computer science!"	-1	1	0
"The hummingbird <u>sucks</u> nectar from the flower"	-1	0	1

Suppose we are given the following training data, where we list the number of times each combination (y, d, t) shows up in the training set.

y	d	t	num. examples
-1	0	0	63
-1	0	1	27
-1	1	0	7
-1	1	1	3
1	0	0	3
1	0	1	7
1	1	0	27
1	1	1	63
		Total num. examples	**200**

From the above table, we can see that 70 out of the 100 of toxic comments include toxic words, and 70 out of the 100 non-toxic comments do not. In addition, the toxicity of the comment t is highly correlated with mentions of demographic identities d (because toxic comments tend to target them) — 90 out of the 100 toxic comments include mentions of demographic identities, and 90 out of the 100 non-toxic comments do not.

We will consider linear classifiers of the form $f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x))$, where $\phi(x)$ is defined above. Normally, we would train classifiers to minimize either the average loss or the maximum group loss, but for simplicity, we will compare two fixed classifiers (which might not minimize either objective):

- Classifier D: $\mathbf{w} = [-0.1, 1, 0]$
- Classifier T: $\mathbf{w} = [-0.1, 0, 1]$

For our loss function, we will be using the zero-one loss, so that the per-group loss is

$$\text{TrainLoss}_g(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}(g)|} \sum_{(x,y)\in\mathcal{D}_{\text{train}}(g)} \mathbf{1}[f_{\mathbf{w}}(x) \neq y].$$

Recall the definition of the maximum group loss:

$$\text{TrainLoss}_{\max}(\mathbf{w}) = \max_g \text{TrainLoss}_g(\mathbf{w}).$$

To capture the spurious correlation problem, let us define groups based on the value of (y, d). There are thus four groups: $(y = 1, d = 1), (y = 1, d = 0), (y = -1, d = 1)$, and $(y = -1, d = 0)$. For example, the group $(y = -1, d = 1)$ refers to non-toxic comments with demographic mentions, for example.

a. [2 points] In words, describe the behavior of Classifier D and Classifier T.

What we expect: For each classifier (D and T), an "if-and-only-if" statement describing the output of the classifier in terms of its features when $y = 1$.

Your Solution: We are given that the feature vector is $\phi(x) = [1, d, t]$, where d is presence of demographic mentions and t is the presence of toxic words, and classifier $D = [-0.1, 0, 0]$ and $T = [-0.1, 0, 1]$

- Classifier D outputs 'toxic' if and only if the input string x contains demographic words.
- Classifier T outputs 'toxic' if and only if the input string x contains toxic words. Both are sub-optimal.

b. [3 points] Compute the following three quantities concerning Classifier D using the dataset above:

1. Classifier D's average loss
2. Classifier D's average loss for each group (fill in the table below)
3. Classifier D's maximum group loss

What we expect: A value for average loss, a complete table (found below) with average loss for each group with the values in the given order, and a value for maximum group loss.

Classifier D	$y = 1$	$y = -1$
$d = 1$	??	??
$d = 0$	??	??

Your Solution: Looking at Classifier D where $\hat{\mathbf{w}} = [-0.1, 0, 0]$

- Classifier D's average loss is = 4/8 = 0.5.

y	d	t	$\hat{\mathbf{w}}\phi(x_i)$	$\hat{\mathbf{w}} \cdot \phi(x)y$	$Loss$
-1	0	0	-0.1	0.1	0
-1	0	1	-0.1	0.1	0
-1	1	0	0.9	-0.9	1
-1	1	1	0.9	-0.9	1
1	0	0	-0.1	-0.1	1
1	0	1	-0.1	-0.1	1
1	1	0	0.9	0.9	0
1	1	1	0.9	0.9	0

- Classifier D's average loss for each group

	y=1	y=-1
d=1	0	1
d=0	1	0

- Classifier D's maximum group loss is = 1.

c. [3 points] Now compute the following three quantities concerning Classifier T using the same dataset:

1. Classifier T's average loss
2. Classifier T's average loss for each group (fill in the table below)
3. Classifier T's maximum group loss

Note the groups are still defined by d, the demographic label.

What we expect: A value for average loss, a complete table with average loss for each group with the values in the given order, and a value for maximum group loss.

Classifier T	$y = 1$	$y = -1$
$d = 1$	??	??
$d = 0$	??	??

Your Solution: For this problem we use the zero-one loss function. Loss = 1 if margin $\hat{\mathbf{w}} \cdot \phi(x)y$ is less than 0, otherwise Loss = 0. We are given that $\phi(x) = [1, d, t]$ Looking at Classifier T where $\hat{\mathbf{w}} = [-0.1, 0, 1]$

- Classifier T's average loss is = 4/8 = 0.5.

y	d	t	$\hat{\mathbf{w}}\phi(x_i)$	$\hat{\mathbf{w}} \cdot \phi(x)y$	$Loss$
-1	0	0	-0.1	0.1	0
-1	0	1	0.9	-0.9	1
-1	1	0	-0.1	0.1	0
-1	1	1	0.9	-0.9	1
1	0	0	-0.1	-0.1	1
1	0	1	0.9	0.9	0
1	1	0	-0.1	-0.1	1
1	1	1	0.9	0.9	0

- Classifier T's average loss for each group

	y=1	y=-1
d=1	0.5	0.5
d=0	0.5	0.5

- Classifier D's maximum group loss is = 0.5.

d. [2 points] Now let's compare the two classifiers. Which classifier has lower average loss? Which classifier has lower maximum group loss?

What we expect: First, indicate which classifier has lower average loss, then indicate which classifier has lower maximum group loss.

Your Solution: Looking at Classifier T:

- Both classifiers have the same average loss.
- Classifier T's has lower maximum group loss.

e. [2 points] As we saw above, different classifiers lead to different numbers of accurate predictions and different people's comments being wrongly rejected. Accurate classification of a non-toxic comment is good for the commenter, but when no classifier has perfect accuracy, how should the correct classifications be distributed across commenters? Here are four well-known principles of fair distribution:

(a) According to **utilitarianism**, we should choose the distribution of accurate classifications that results in the *greatest net benefit* or *greatest average well-being*, where the average is a simple average.[1]

(b) According to **prioritarianism**, we should choose the distribution of accurate classifications that results in the greatest *weighted* average well-being, where the weights prioritize less-well-off groups.[2]

(c) According to John Rawls's **difference principle**, when choosing between distributive systems, we should choose the one that maximizes the well-being of the worst-off people.[3]

(d) In order to **avoid compounding prior injustice**, we should ensure that our classifier does not impose a disadvantage on members of a demographic group that has faced historical discrimination. [4]

[1]There are many varieties of utilitarianism and consequentialism – if you are interested in reading more about these principles, see here and here.

[2]For more on prioritarianism, see here and here.

[3]For more on the difference principle, see here and here.

[4]There are many anti-discrimination principles. This one is drawn from here; for others, see here . Many theories rely on perceived social group membership rather than demographic group membership. This principle also relates to corrective justice, for which see here.

NOTE: The above are only a subset of the ethical frameworks out there. If you feel that another principle is more appropriate, you may invoke it, but you must define it in your answer and link to a source in which it is described. First, choose an objective (average loss or maximum group loss) by appealing to one of the four principles.

Which of Classifier D or Classifier T would you deploy on a real online social media platform in order to flag posts labeled as toxic for review? Refer to one of the above principles when justifying your answer.

What we expect: A 2-4 sentence explanation that justifies your choice of objective by referring to one of the four principles. There are many ways to answer this question well; a good answer explains the connection between a classifier and a principle clearly and concisely.

Your Solution: I choose Classifier D and average Loss. Rationale:

I resonate with John Rawls's difference principle which seeks to maximizes the well-being of the worse-off people. I naturally start with an objective of maximum group loss. If I worked at a social media company, I thus want my content filtering to correctly classify sentences as toxic when demographic identity terms are used because demographic targetting occurs far too regularly. This means I want to have low loss when d=1 and y=1. classifier D has lower loss than Classifier T for this condition.

However, if I actually used maximum group loss as initially intended I would be solving for the group d=0, y=0 since loss = 1. Choosing Maximum Group Loss blindly wouldn't have solved our principle to maximize the well-being of the worse off people. In conclusion, I choose Classifier D and average Loss.

f. [2 points] We've talked about machine learning as the process of turning data into models, but where does the data come from? In the context of collecting data for training a machine learning model for toxicity classification, who determines whether a comment **should** be marked as toxic or not is very important (i.e., whether $y = 1$ or $y = -1$ in the earlier data table). Here are some commonly used choices:

- Recruit people on a crowdsourcing platform (like Amazon Mechanical Turk) to annotate each comment.
- Hire social scientists to establish criteria for toxicity and annotate each comment.
- Ask users of the platform to rate comments.
- Ask users of the platform to deliberate about and decide on community standards and criteria for toxicity, perhaps using a process of participatory design.[5]

[5]For more on participatory design, see here

Which methods(s) would you use to determine the toxicity of comments for use as data to use for training a toxicity classifier, and why? Explain why you chose your method(s) over the others listed.

What we expect: 1-2 sentences explaining what methods(s) you would use and why you chose those method(s), and 1-2 sentences contrasting your chosen method(s) with alternatives.

Your Solution: I'll start by **hiring social scientists to establish criteria for toxicity and annotate each comment.** I want to have conversations with social scientists to understand toxicity, its signals and the severity of harms. It's important to start here because social scientists have spent far more time thinking about the problem than I/the company has. I'd then use this expert knowledge to create an initial framework for the community standards.

After that I'll workshop with user groups, legal, risk, and product teams to develop a final set of community standards. It's important to do this because you want all your stakeholders to buy into and own a shared set of ideals about how our company/platform ought to be run. I extended the stakeholder beyond just users.

Finally, I'll **ask users to tag toxic comments based on these community standards**. When a post is tagged as toxic, users will need to select a reason. These reasons will be traceable to the community standards. Enlisting users to tag toxic comments makes the effort scalable.

I **would not recruit people on a crowdsourcing platform to annotate each comment** because there will be no consistency in the labelled data: one person's idea of toxic is very different to another's. The labelled data would not be useful because of this inconsistency.

Problem 5: K-means Clustering

Suppose we have a feature extractor ϕ that produces 2-dimensional feature vectors, and a toy dataset $\mathcal{D}_{\text{train}} = \{x_1, x_2, x_3, x_4\}$ with

1. $\phi(x_1) = [10, 0]$
2. $\phi(x_2) = [30, 0]$
3. $\phi(x_3) = [10, 20]$
4. $\phi(x_4) = [20, 20]$

a. [2 points] Run 2-means on this dataset until convergence. Please show your work. What are the final cluster assignments z and cluster centers μ? Run this algorithm twice with the following initial centers:

(a) $\mu_1 = [20, 30]$ and $\mu_2 = [20, -10]$
(b) $\mu_1 = [0, 10]$ and $\mu_2 = [30, 20]$

What we expect: Show the cluster centers and assignments for each step.

Your Solution:

Define the below to evaluate the square of the distance between feature vector $\phi(x)$ to the cluster k's centroid μ_k in $\mathbb{R}^2$, where xPos is the horizontal component and yPos is the vertical component in $\mathbb{R}^2$.

$$||\phi(x) - \mu_k||^2 = (xPos_{\phi(x)} - xPos_{\mu_k})^2 + (yPos_{\phi(x)} - yPos_{\mu_k})^2 \quad (28)$$

- Let μ_1 = [20, 30] and μ_2 = [20, -10]
  Iteration 1:

x_i	xPos	yPos	$\|\|\phi(x) - \mu_1\|\|^2$	$\|\|\phi(x) - \mu_2\|\|^2$	Assigned z
x_1	10	0	1000	200	2
x_2	30	0	1000	200	2
x_3	10	20	200	1000	1
x_4	20	20	100	900	1

  Reevaluate μ_1 and μ_2 based on their assigned cluster z.

$$\mu_1 = [\frac{10+20}{2}, \frac{20+20}{2}] = [15, 20] \quad (29)$$

$$\mu_2 = [\frac{10+30}{2}, \frac{0+0}{2}] = [20, 0] \quad (30)$$

Iteration 2:

x_i	xPos	yPos	$\|\|\phi(x)-\mu_1\|\|^2$	$\|\|\phi(x)-\mu_2\|\|^2$	Assigned z
x_1	10	0	425	100	2
x_2	30	0	625	100	2
x_3	10	20	25	500	1
x_4	20	20	25	400	1

Reevaluate μ_1 and μ_2 based on their assigned cluster z.

$$\mu_1 = [\frac{10+20}{2}, \frac{20+20}{2}] = [15, 20] \tag{31}$$

$$\mu_2 = [\frac{10+30}{2}, \frac{0+0}{2}] = [20, 0] \tag{32}$$

- Next, let μ_1 = [0, 10] and μ_2 = [30, 20]

  Iteration 1:

x_i	xPos	yPos	$\|\|\phi(x)-\mu_1\|\|^2$	$\|\|\phi(x)-\mu_2\|\|^2$	Assigned z
x_1	10	0	200	800	1
x_2	30	0	1000	400	2
x_3	10	20	200	400	1
x_4	20	20	500	100	2

Reevaluate μ_1 and μ_2 based on their assigned cluster z.

$$\mu_1 = [\frac{10+10}{2}, \frac{0+20}{2}] = [10, 10] \tag{33}$$

$$\mu_2 = [\frac{30+20}{2}, \frac{0+20}{2}] = [25, 10] \tag{34}$$

Iteration 2:

x_i	xPos	yPos	$\|\|\phi(x)-\mu_1\|\|^2$	$\|\|\phi(x)-\mu_2\|\|^2$	Assigned z
x_1	10	0	100	325	1
x_2	30	0	500	125	2
x_3	10	20	100	325	1
x_4	20	20	200	125	2

Reevaluate μ_1 and μ_2 based on their assigned cluster z.

$$\mu_1 = [\frac{10+10}{2}, \frac{0+20}{2}] = [10, 10] \tag{35}$$

$$\mu_2 = [\frac{30+20}{2}, \frac{0+20}{2}] = [25, 10] \tag{36}$$

b. [5 points] Implement the `kmeans` function. You should initialize your k cluster centers to random elements of `examples`. After a few iterations of k-means, your centers will be very dense vectors. In order for your code to run efficiently and to obtain full credit, you will need to precompute certain dot products. As a reference, our code runs in under a second on cardinal, on all test cases. You might find `generateClusteringExamples` in `util.py` useful for testing your code.

NOTE: Do not use libraries such as Scikit-learn.

c. [2 points] If we scale all dimensions in our initial centroids and data points by some non-zero factor, are we guaranteed to retrieve the same clusters after running k-means (i.e. will the same data points belong to the same cluster before and after scaling)? What if we scale only certain dimensions? If your answer is yes, provide a short explanation; if not, give a counterexample.

What we expect: This response should have two parts. The first should be a yes/no response and explanation or counterexample for the first subquestion (scaling all dimensions). The second should be a yes/no response and explanation or counterexample for the second subquestion (scaling only certain dimensions).

Your Solution: Yes we are guaranteed to retrieve the same clusters after running k-means after scaling all dimensions in initial centroids and data points by some non-zero factor λ.

In the base case, Kmeans performs the below operations for each feature vector $\phi(x_i)$ to assign it to a cluster and then re-evaluate the centroid.

$$Loss_{Kmeans} = \sum_{i=1}^{n} ||\phi(x_i) - \mu_{z_i}||^2 \tag{37}$$

$$z_i \leftarrow \underset{k=1,\dots,K}{\text{argmin}} ||\phi(x_i) - \mu_k||^2 \tag{38}$$

$$\mu_k \leftarrow \frac{1}{|\{i : z_i = k\}|} \sum_{i:z_i=k}^{n} \phi(x_i) \tag{39}$$

If the initial centroids and data points are scaled by some non-zero factor λ, we see that the losses scale by λ^2 and the centroids scale by λ. Even though losses scale by λ^2, it doesn't affect the assignment of the cluster group since the losses consistently scale by λ^2 for all $phi(x_i)$ in the data set.

$$Loss'_{Kmeans} = \sum_{i=1}^{n} ||\lambda\phi(x_i) - \lambda\mu_{z_i}||^2 \tag{40}$$

$$Loss'_{Kmeans} = \lambda^2 \sum_{i=1}^{n} ||\phi(x_i) - \mu_{z_i}||^2 \tag{41}$$

$$z'_i \leftarrow \underset{k=1,...,K}{\operatorname{argmin}} ||\lambda\phi(x_i) - \lambda\mu_k||^2 \tag{42}$$

$$z'_i \leftarrow \lambda^2 \underset{k=1,...,K}{\operatorname{argmin}} ||\phi(x_i) - \mu_k||^2 \tag{43}$$

$$\mu'_k \leftarrow \frac{1}{|\{i : z_i = k\}|} \sum_{i:z_i=k}^{n} \lambda\phi(x_i) \tag{44}$$

$$\mu'_k \leftarrow \frac{\lambda}{|\{i : z_i = k\}|} \sum_{i:z_i=k}^{n} \phi(x_i) \tag{45}$$

Thus $Loss'_{Kmeans} = \lambda^2 Loss_{Kmeans}$, $z'_i = z_i$ and $\mu'_i = \lambda\mu_i$

Submission

Submission is done on Gradescope.

Written: When submitting the written parts, make sure to select **all** the pages that contain part of your answer for that problem, or else you will not get credit. To double check after submission, you can click on each problem link on the right side and it should show the pages that are selected for that problem.

Programming: After you submit, the autograder will take a few minutes to run. Check back after it runs to make sure that your submission succeeded. If your autograder crashes, you will receive a 0 on the programming part of the assignment. Note: the only file to be submitted to Gradescope is `submission.py`.

More details can be found in the Submission section on the course website.