Problem Set 2

Artificial Intelligence
Fall 2021 CS47100-AI

Student name: _____ Student PUID: _____

Note: You are free to use your intuition to find the steps in the proof. But, make sure you do not use your intuition to justify steps in your proofs.[1]

Suppose, there is the following set of four mini-reviews, each labeled positive (+1) or negative (-1):

- (-1) pretty bad
- (+1) good plot
- (-1) not good
- (+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 $\phi(x) = \{pretty : 1, bad : 1\}$. Recall the definition of the hinge loss:

$$Loss_{hinge}(x, y, \mathbf{w}) = \max\{0, 1 - \mathbf{w}.\phi(x)y\}$$

where y is the correct label.
Now, answer the following 2 questions based on this information.

Problem 1. Suppose we run stochastic gradient descent, updating the weights according to,

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_w Loss_{hinge}(x, y, \mathbf{w}),$$

once for each of the four examples in the order given above. After the classifier is trained on the given four data points, what are the weights of the six words ("pretty", "good", "bad", "plot", "not", "scenery") that appear in the above reviews? Please label your weight vector, indicating which word corresponds to each index. Use $\eta = .5$ as the step size and initialize $\mathbf{w} = [0, ..., 0]$. Assume that $\nabla_w Loss_{hinge}(x, y, \mathbf{w}) = 0$ when the margin is exactly 1.

Problem 2. Create a small labeled dataset of four mini-reviews using the words "not", "good", and "bad", where the labels make intuitive sense. Each review should contain one or two words, and no repeated words. No two reviews in the dataset should be the same. The word "not" by itself is not a valid review.

Prove that no linear classifier using word features can get zero error on your 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.

[1] *The contents of this problem set is based on the AI course CS221 taught at Stanford University.*

Propose a single additional feature for your dataset, that we could augment the feature vector with, that would fix this problem. What is the resulting weight vector for your dataset when you introduce this new feature? (Hint: think about the linear effect that each feature has on the classification score.)

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 $\sigma(\mathbf{w}.\phi(x))$, where $\sigma(z) = (1 + e^{-z})^{-1}$ is the logistic function that squashes a real number to the range $(0, 1)$. Suppose that we wish to use the squared loss. For this problem, assume that the movie rating y is a real-valued variable in the range $[0, 1]$.

Now, answer the following 4 questions based on this information.

Problem 3. Write out the expression for $Loss(x, y, \mathbf{w})$ for a single datapoint (x, y). It should be a function of $\phi, x, y, \mathbf{w}$.

Problem 4. Compute the gradient of the loss with respect to $\mathbf{w}$.

Hint: you can write the answer in terms of the predicted value $p = \sigma(\mathbf{w}.\phi(x))$.

Problem 5. Suppose there is one datapoint (x, y) with some arbitrary $\phi(x)$ and $y = 1$. Can you specify conditions for $\mathbf{w}$ to make the magnitude of the gradient of the loss with respect to $\mathbf{w}$ arbitrarily small (minimize the magnitude of the gradient)? If so, how small? 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. You can use your answer from the previous question.

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

Problem 6. Assuming the same datapoint (x, y) as above $(y = 1)$, what is the largest magnitude that the gradient can take $(\max_x \|\nabla_w Loss(x, y, \mathbf{w})\|)$ in terms of $\|\phi(x)\|$. Prove that your chosen value is indeed the maximum.

Problem 7. In this problem, you will build a binary linear classifier that reads movie reviews and guesses whether they are "positive" or "negative."

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) 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) 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) Create an artificial dataset for your `learnPredictor` function by writing the `generateExample` function (nested in the `generateDataset` function). 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) Now we will try a crazier feature extractor. 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) 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. How do you explain this?

Construct a review (one sentence max) in which character n-grams probably outperform word features, and briefly explain why this is so.
Note: This is not a coding problem. This should not be in the python file.

Note: There is code in submission.py that will help you test different values of n. Remember to write your final written solution in the pdf.