cs221 Assignment #2: sentiment

Problem 1. Building intuition

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

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

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? Use $\eta = .5$ as the step size and initialize $\mathbf{w} = [0, ..., 0]$. Assume that $\nabla_{\mathbf{w}} \text{Loss}_{\text{hinge}}(x, y, \mathbf{w}) = 0$ when the margin is exactly 1.

sol)

$$\nabla_{\mathbf{w}} \text{Loss}_{\text{hinge}}(x, y, \mathbf{w}) = \nabla_{\mathbf{w}} \max\{0, 1 - \mathbf{w} \cdot \phi(x)y\} = \begin{cases} -\phi(x)y, & \text{if } \mathbf{w} \cdot \phi(x)y < 1 \\ 0, & \text{elsewhere} \end{cases}$$

Let's denote $\mathbf{w} = [\text{pretty}, \text{good}, \text{bad}, \text{plot}, \text{not}, \text{scenery}]$

for the first example, initialized $\mathbf{w} = [0, ..., 0]$, $\phi(x) = \{\text{pretty} : 1, \text{bad} : 1\}$, $y = -1$.
Since $\mathbf{w} \cdot \phi(x)y = 0$, $\nabla_{\mathbf{w}} = -\phi(x)y = [1, 0, 1, 0, 0, 0]$ and $\mathbf{w} - \eta\nabla_{\mathbf{w}} = [-0.5, 0, -0.5, 0, 0, 0]$.

for the second example, $\mathbf{w} = [-0.5, 0, -0.5, 0, 0, 0]$, $\phi(x) = \{\text{good} : 1, \text{plot} : 1\}$, $y = +1$.
Since $\mathbf{w} \cdot \phi(x)y = 0$, $\nabla_{\mathbf{w}} = -\phi(x)y = [0, -1, 0, -1, 0, 0]$ and $\mathbf{w} - \eta\nabla_{\mathbf{w}} = [-0.5, +0.5, -0.5, +0.5, 0, 0]$.

for the third example, $\mathbf{w} = [-0.5, +0.5, -0.5, +0.5, 0, 0]$, $\phi(x) = \{\text{not} : 1, \text{good} : 1\}$, $y = -1$.
Since $\mathbf{w} \cdot \phi(x)y = [-0.5, +0.5, -0.5, +0.5, 0, 0] \cdot [0, 1, 0, 0, 1, 0] \times -1 = 0.5$,
$\nabla_{\mathbf{w}} = -\phi(x)y = [0, 1, 0, 0, 1, 0]$ and $\mathbf{w} - \eta\nabla_{\mathbf{w}} = [-0.5, 0, -0.5, +0.5, -0.5, 0]$.

for the second example, $\mathbf{w} = [-0.5, 0, -0.5, +0.5, -0.5, 0]$, $\phi(x) = \{\text{pretty} : 1, \text{scenery} : 1\}$, $y = +1$.
Since $\mathbf{w} \cdot \phi(x)y = [-0.5, 0, -0.5, +0.5, -0.5, 0] \cdot [1, 0, 0, 0, 0, 1] \times -1 = -0.5$,
$\nabla_{\mathbf{w}} = -\phi(x)y = [-1, 0, 0, 0, 0, -1]$ and $\mathbf{w} - \eta\nabla_{\mathbf{w}} = [0, 0, -0.5, +0.5, -0.5, +0.5]$.

$$\mathbf{w} = [0, 0, -0.5, +0.5, -0.5, +0.5]$$

b. 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. 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.

After providing such a dataset, propose a single additional feature that we could augment the feature vector with that would fix this problem. (Hint: think about the linear effect that each feature has on the classification score.)

sol)

Using the words with given rule, we can form four types of mini-reviews.

- good: +1
- bad: -1
- not good: -1
- not bad: +1

if 'good' is positive, 'bad' is negative, and 'not good' is negative, if such linear classifier exists, 'not' should be toward negative axis. however, in that sense, 'not bad' should be negative since both 'not' and 'bad' are toward negative axis, which contradicts the real meaning of 'not bad'.

To fix the problem, a two-word feature such as 'not bad' is needed.

Problem 2: Predicting Movie Ratings

a. Write out the expression for $\text{Loss}(x, y, \mathbf{w})$ for a single datapoint (x, y).

sol)

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

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

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

sol)

Due to the property of sigmoid function, $\frac{\partial}{\partial x}\sigma(x) = \sigma(x)(1 - \sigma(x))$.

$$\begin{aligned}
\therefore \nabla_{\mathbf{w}}\text{Loss} &= \frac{\partial}{\partial \mathbf{w}}(\sigma(\mathbf{w} \cdot \phi(x)) - y)^2 \\
&= 2(\sigma(\mathbf{w} \cdot \phi(x)) - y)\frac{\partial}{\partial \mathbf{w}}(\sigma(\mathbf{w} \cdot \phi(x)) - y) \\
&= 2(p - y)\phi(x)(\sigma(\mathbf{w} \cdot \phi(x))(1 - \sigma(\mathbf{w} \cdot \phi(x)) = 2(p - y)\phi(x)p(1 - p)
\end{aligned}$$

c. 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.

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.

sol)

$\nabla_{\mathbf{w}}\text{Loss} = 2(p - y)\phi(x)p(1 - p) = 0$ for any $\phi(x)$, $y = 1 \Leftrightarrow (p - 1)p(1 - p) = 0$

Therefore, if and only if $p = 1$ or $p = 0$, the magnitude of the gradient gets zero.

However, to make $p = \sigma(\mathbf{w} \cdot \phi(x)) = 1$, $\mathbf{w} \cdot \phi(x) \to \infty \Leftrightarrow \mathbf{w}$ is very large.

On the other hand, to make $p = \sigma(\mathbf{w} \cdot \phi(x)) = 0$, $\mathbf{w} \cdot \phi(x) \to -\infty \Leftrightarrow \mathbf{w}$ is very small.

d. Assuming the same datapoint (x, y) as above, what is the largest magnitude that the gradient can take? Leave your answer in terms of $\|\phi(x)\|$. Prove that your chosen value is indeed the largest magnitude that the gradient can take.

sol)

$\nabla_{\mathbf{w}}\text{Loss} = 2(p - y)\phi(x)p(1 - p) = -2\phi(x) \times p(p - 1)^2$ for some $\phi(x)$, $y = 1$

To find the local extremum,

$$\frac{\partial}{\partial p}p(p - 1)^2 = \frac{\partial}{\partial p}p^3 - 2p^2 + p = 3p^2 - 4p + 1 = (3p - 1)(p - 1) = 0$$

When $3p - 1 = 0 \Leftrightarrow p = 1/3$, $p(p - 1)^2 = 4/27$ and when $p - 1 = 0 \Leftrightarrow p = 1$, $p(p - 1)^2 = 0$

Hence, the largest magnitude $= 2\|\phi(x)\| \times 4/27 = \|\phi(x)\| \times 8/27$

e. The problem with the loss function we have defined so far is that is it is non-convex, which means that gradient descent is not guaranteed to find the global minimum. In general these types of problems can be difficult to solve, so let us try to reformulate the problem as plain old linear regression. Suppose you have a dataset $\mathbf{D}$ consisting of (x, y) pairs, and that there exists a weight vector $\mathbf{w}$ that yields zero loss on this dataset. The dataset $\mathbf{D}$ uses the non-linear predictor (i.e.,

$\sigma(\mathbf{w} \cdot \phi(x)))$ as described above. Show that there is an easy transformation to a modified dataset $\mathbf{D}'$ of (x, y') pairs such that performing least squares regression (using a linear predictor and the squared loss) on $\mathbf{D}'$ converges to a vector $\mathbf{w}^*$ that yields zero loss on $\mathbf{D}'$. Concretely, write an expression for y' in terms of y and justify this choice. This expression should not be a function of $\mathbf{w}$.

Hint: write down the loss functions for both datasets and set them equal to 0.

sol)

While we have $\mathbf{w}$ and y such that $(\sigma(\mathbf{w} \cdot \phi(x)) - y)^2 = 0$, we have to find $\mathbf{w}^*$ and y' satisfying $(\mathbf{w}^* \cdot \phi(x) - y')^2 = 0$

$$
\begin{aligned}
(\sigma(\mathbf{w} \cdot \phi(x)) - y)^2 = 0 \Leftrightarrow \sigma(\mathbf{w} \cdot \phi(x)) - y = 0 \Leftrightarrow \frac{1}{1 + e^{-\mathbf{w} \cdot \phi(x)}} - y = 0 \\
\Leftrightarrow 1 + e^{-\mathbf{w} \cdot \phi(x)} = \frac{1}{y} \Leftrightarrow e^{-\mathbf{w} \cdot \phi(x)} = \frac{1}{y} - 1 \\
\Leftrightarrow -\mathbf{w} \cdot \phi(x) = \ln(\frac{1}{y} - 1) \\
\Leftrightarrow (\mathbf{w} \cdot \phi(x) + \ln(\frac{1}{y} - 1))^2 = 0
\end{aligned}
$$

So when $\mathbf{w}^* = \mathbf{w}$ and $y' = -\ln(\frac{1}{y} - 1)$, a modified dataset $\mathbf{D}'$ of (x, y') pairs satisfies such relationship.

Problem 3: Sentiment Classification

d. When you run the grader.py on test case `3b-2`, it should output a `weights` file and a `error-analysis` file. Look through some example incorrect predictions and for five of them, give a one-sentence explanation of why the classification was incorrect. What information would the classifier need to get these correct? In some sense, there's not one correct answer, so don't overthink this problem. The main point is to convey intuition about the problem.

sol)

- 'it's painful to watch witherspoon's talents wasting away inside unnecessary films like legally blonde and sweet home abomination , i mean , alabama . '

  Truth: -1, Prediction: 1 [WRONG]

  - In the weight vector, movie names which should be neutral in sentiment classification like 'legally blonde' or 'sweet home' makes positive bias. Moreover, while 'painfully' is negative, 'painful' is regarded as positive word. So the model need to understand prefix/suffix.

- . . . standard guns versus martial arts cliche with little new added .

  Truth: -1, Prediction: 1 [WRONG]

  - In the context, phrase 'with little new added' is negative meaning while summing up the weights gets positive bias. It might arise from using lots of optimistic word like 'with', 'new', 'added'. therefore, the model needs to read the phrase, not each word.

- o ótimo esforço do diretor acaba sendo frustrado pelo roteiro , que , depois de levar um bom tempo para colocar a trama em andamento , perde-se de vez a partir do instante em que os estranhos acontecimentos são explicados .

  Truth: -1, Prediction: 1 [WRONG]

  - While 'de' and 'em' got huge positive bias, about half of words got zero weights since it is written in spanish. Therefore it is hard to predict the sentiment.

- this may be the dumbest , sketchiest movie on record about an aspiring writer's coming-of-age .

  Truth: -1, Prediction: 1 [WRONG]

  - In this review, the most important keywords to read the sentiment may be dumbest, sketchiest. However, those two words were uncommon due to its superlative form so that those word were zero weight. Hence, the model need to understand the superlative form of adjective or adverb.

- a lovely and beautifully photographed romance .
  Truth: 1, Prediction: -1 [WRONG]
  - The review dataset may have a positive bias toward genre movie or artistic film, since 'lovely' or 'romance' have negative weight. therefore, the model should be trained in neutral, unbiased dataset.

f. Run your linear predictor with feature extractor `extractCharacterFeatures`. Experiment with different values of n to see which one produces the smallest test 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: You should replace the featureExtractor in test3b2() in grader.py, i.e., let featureExtractor = submission.extractCharacterFeatures(___) and report your results. Don't forget to recover test3b2() after finishing this question.

sol)

When $n = 5$, the character level feature extractor produces the lowest test error and the error rate 26.8% is similar to that of word level extractor. With average length of word, $n = 5$, it can capture the meaning rare or uncommon word in character level. For instance, the character level extractor outperformed to catch the sentiment of the review, 'even the hastily and amateurishly drawn animation cannot engage .'. In the word level, 'amateurishly' is uncommon but the character level feature extracts the meaning of 'amateur' in the review.

Problem 4: K-means clustering

a. 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:

sol)

1. $\mu_1 = [2, 3]$ and $\mu_2 = [2, -1]$
   At the first step,

   $$z_1 = \text{argmin}\{(1-2)^2 + (0-3)^2, (1-2)^2 + (0-(-1))^2\} = 2$$
   $$z_2 = \text{argmin}\{(1-2)^2 + (2-3)^2, (1-2)^2 + (2-(-1))^2\} = 1$$
   $$z_3 = \text{argmin}\{(3-2)^2 + (0-3)^2, (3-2)^2 + (0-(-1))^2\} = 2$$
   $$z_4 = \text{argmin}\{(2-2)^2 + (2-3)^2, (2-2)^2 + (2-(-1))^2\} = 1$$

   Since $z_1 = z_3 = 2, \mu_2 = \frac{1}{2}\{\phi(x_1) + \phi(x_3)\} = [2, 0]$ and $z_2 = z_4 = 1, \mu_1 = \frac{1}{2}\{\phi(x_2) + \phi(x_4)\} = [1.5, 2]$.
   At the second step,

   $$z_1 = \text{argmin}\{(1-1.5)^2 + (0-2)^2, (1-2)^2 + (0-0)^2\} = 2$$
   $$z_2 = \text{argmin}\{(1-1.5)^2 + (2-2)^2, (1-2)^2 + (2-0)^2\} = 1$$
   $$z_3 = \text{argmin}\{(3-1.5)^2 + (0-2)^2, (3-2)^2 + (0-0)^2\} = 2$$
   $$z_4 = \text{argmin}\{(2-1.5)^2 + (2-2)^2, (2-2)^2 + (2-0)^2\} = 1$$

   Since no z_i has changed, $z_1 = z_3 = 2, \mu_2 = [2, 0]$ and $z_2 = z_4 = 1, \mu_1 = [1.5, 2]$.
2. $\mu_1 = [0, 1]$ and $\mu_2 = [3, 2]$
   At the first step,

   $$z_1 = \text{argmin}\{(1-0)^2 + (0-1)^2, (1-3)^2 + (0-2)^2\} = 1$$
   $$z_2 = \text{argmin}\{(1-0)^2 + (2-1)^2, (1-3)^2 + (2-2)^2\} = 1$$
   $$z_3 = \text{argmin}\{(3-0)^2 + (0-1)^2, (3-3)^2 + (0-2)^2\} = 2$$
   $$z_4 = \text{argmin}\{(2-0)^2 + (2-1)^2, (2-3)^2 + (2-2)^2\} = 2$$

Since $z_1 = z_2 = 1, \mu_1 = \frac{1}{2}\{\phi(x_1) + \phi(x_2)\} = [1, 1]$ and $z_3 = z_4 = 2, \mu_2 = \frac{1}{2}\{\phi(x_2) + \phi(x_4)\} = [2.5, 1]$.

At the second step,

$$z_1 = \text{argmin}\{(1-1)^2 + (0-1)^2, (1-2.5)^2 + (0-1)^2\} = 1$$
$$z_2 = \text{argmin}\{(1-1)^2 + (2-1)^2, (1-2.5)^2 + (2-1)^2\} = 1$$
$$z_3 = \text{argmin}\{(3-1)^2 + (0-1)^2, (3-2.5)^2 + (0-1)^2\} = 2$$
$$z_4 = \text{argmin}\{(2-1)^2 + (2-1)^2, (2-2.5)^2 + (2-1)^2\} = 2$$

Since no z_i has changed, $z_1 = z_1 = 1, \mu_1 = [1, 1]$ and $z_3 = z_4 = 1, \mu_1 = [2.5, 1]$.

c. Sometimes, we have prior knowledge about which points should belong in the same cluster. Suppose we are given a set S of example pairs (i, j) which must be assigned to the same cluster. For example, suppose we have 5 examples; then $S = \{(1, 5), (2, 3), (3, 4)\}$ says that examples 2, 3, and 4 must be in the same cluster and that examples 1 and 5 must be in the same cluster. Provide the modified k-means algorithm that performs alternating minimization on the reconstruction loss:

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

where μ_{z_i} is the assigned centroid for the feature vector $\phi(x_i)$.

Recall that alternating minimization is when we are optimizing two variables jointly by alternating which variable we keep constant.

sol)

follow the link: `https://www.cs.cmu.edu/~./dgovinda/pdf/icml-2001.pdf`

1. Let $C_1, \cdots C_k$ be the initial cluster centroids.
2. For each point d_i in the dataset D, assign it to the closest cluster C_j while must-link constraint preserves.
3. For each cluster C_i, update its center by averaging all of the points d_j that hav been assigned to it.
4. Iterate between (2) and (3) until convergence.
5. Return $\{C_1, \cdots C_k\}$.

d. What is the advantage of running K-means multiple times on the same dataset with the same K, but different random initializations?

sol)

By different random initializations, K-means algorithm can reach different centroids and different reconstruction loss since K-means algorithm can have lots of local optimal points. Therefore, we can compare the loss from different outcomes and find more optimal centroids.

e. If we scale all dimensions in our initial centroids and data points by some 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 it is no, provide a counterexample.

sol)

If we scale all dimensions by a constant c, the reconstruction loss of scaled K-means

$$\sum_{i=1}^{n} \|c\mu_{z_i} - c\phi(x_i)\|^2 = c^2 \sum_{i=1}^{n} \|\mu_{z_i} - \phi(x_i)\|^2$$

for the constant c. Since we multiply the L_2 norm by a positive c^2, the norm order preserves and we can get the same cluster as before.

However, if we scale only certain dimension, the clustering results can not preserve.
counterexample)
Assume that we have $\phi(x) = (0, 0, 0)$, $\mu_1 = (1, 2, 2)$, $\mu_2 = (2, 2, 0)$.
when not scaling, $z_x = \text{argmin}\{(0-1)^2 + (0-2)^2 + (0-2)^2, (0-2)^2 + (0-2)^2 + (0-0)^2\} = 2$.
but when scaling the third dimension by $c = 1/2$,

$$z_x = \text{argmin}\{(0-1)^2 + (0-2)^2 + (\frac{1}{2} \cdot 0 - \frac{1}{2} \cdot 2)^2, (0-2)^2 + (0-2)^2 + (\frac{1}{2} \cdot 0 - \frac{1}{2} \cdot 0)^2\} = 1.$$