

CS221 Fall 2018 Homework 2

SUNet ID: 05794739

Name: Luis Perez

Collaborators:

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

Problem 1

(a) We first need to know the $\nabla_{\mathbf{w}} \text{Loss}_{\text{hinge}}(x, y, \mathbf{w})$. We have:

$$\begin{aligned}\nabla_{\mathbf{w}} \text{Loss}_{\text{hinge}}(x, y, \mathbf{w}) &= 0 & (1 - \mathbf{w} \cdot \phi(x)y \leq 0) \\ \nabla_{\mathbf{w}} \text{Loss}_{\text{hinge}}(x, y, \mathbf{w}) &= -\phi(x)y\end{aligned}$$

We see then that \mathbf{w} changes only when $\mathbf{w} \cdot \phi(x)y < 1$ such that the count for the word in that review either goes up by η (for a positive review) or down by η (for a negative review) – this is due to using subtracting the gradient, which has a scaling factor of $-\eta y$ on the feature vector.

We start with $w = [0, 0, 0, 0, 0, 0]$ where the features are {pretty, bad, good, plot, not, scenery}. On the first update, we note that $\mathbf{w} \cdot \phi(x)y = 0 < 1$, so we now have $w = [-0.5, -0.5, 0, 0, 0, 0]$.

For the second review, we now have $\mathbf{w} \cdot \phi(x)y = 0 < 1$, so we update to have $w = [-0.5, -0.5, 0.5, 0.5, 0, 0]$.

On the third review, we have $\mathbf{w} \cdot \phi(x)y = -0.5 < 1$, so we update to have $w = [-0.5, -0.5, 0, 0.5, -0.5, 0]$.

On the fourth review, we have $\mathbf{w} \cdot \phi(x)y = -0.5 < 1$, so we update to have $w = [0.0, -0.5, 0.0, 0.5, -0.5, 0.5]$.

(b) The data set we can use is as follows:

- (-1) not, $\phi(x_1) = [1, 0, 0]^T$
- (+1) good, $\phi(x_2) = [0, 1, 0]^T$
- (-1) bad, $\phi(x_3) = [0, 0, 1]^T$
- (+1) not bad, $\phi(x_4) = [1, 0, 1]^T$

Then consider any linear classifier with weight vector $\mathbf{w} = [w_1, w_2, w_3]$ where we have $\hat{y}(x) = \mathbf{w} \cdot \phi(x)$ (we predict +1 if non-negative and -1 if negative). Suppose a linear classifier exists which can correctly classify the data above. Then we must have the

below be true, assuming correct classification:

$$\begin{aligned}\mathbf{w}\phi(x_1) &= w_1 < 0 \\ \mathbf{w}\phi(x_2) &= w_2 \geq 0 \\ \mathbf{w}\phi(x_3) &= w_3 < 0 \\ \mathbf{w}\phi(x_4) &= w_1 + w_3 \geq 0\end{aligned}$$

This is a contradiction, as we can't have $w_1 < 0, w_3 < 0$ and $w_1 + w_3 \geq 0$ (two negative values can't add to a non-negative value). Therefore, the above dataset is not linearly-seperable and a linear classifier cannot possible achieve zero loss.

To fix the problem, we could add an additional feature which is 0 for reviews with one word and 1 for reviews with two words.

- (-1) not, $\phi(x_1) = [1, 0, 0, 0]^T$
- (+1) good, $\phi(x_2) = [0, 1, 0, 0]^T$
- (-1) bad, $\phi(x_3) = [0, 0, 1, 0]^T$
- (+1) not bad, $\phi(x_4) = [1, 0, 1, 1]^T$

Then note that the weight vector $w = [-1, 1, -1, 3]$ on a linear classifier $w\phi(x)$ will now correctly classify the items in the data set.

$$\begin{aligned}\mathbf{w}\phi(x_1) &= -1 \\ \mathbf{w}\phi(x_2) &= +1 \\ \mathbf{w}\phi(x_3) &= -1 \\ \mathbf{w}\phi(x_4) &= +1\end{aligned}$$

Problem 2

(a) As described, we have the following loss:

$$\begin{aligned}\text{Loss}(x, y, \mathbf{w}) &= (\sigma(\mathbf{w} \cdot \phi(x)) - y)^2 \\ &= \left(y - \frac{1}{1 + e^{-\mathbf{w}\phi(x)}} \right)^2\end{aligned}$$

(b) We can take the gradient directly, letting $p = \sigma(\mathbf{w} \cdot \phi(x))$

$$\begin{aligned}\nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{w}) &= -2(y - p) \nabla_{\mathbf{w}} \sigma(\mathbf{w} \cdot \phi(x)) && \text{(chain rule)} \\ &= -2(y - p)p(1 - p)\phi(x) && \text{(derivative of sigmoid as detailed here)}\end{aligned}$$

- (c) Suppose we have some arbitrary $\phi(x)$ and $y = 1$. Then we can simplify the gradient expression slightly.

$$\begin{aligned}\nabla_{\mathbf{w}} \text{Loss}(x, 1, \mathbf{w}) &= -2(1-p)^2 p \phi(x) \\ &= -2[p - 2p^2 + p^3] \phi(x)\end{aligned}$$

We can make the above arbitrarily small by taking $\|\mathbf{w}\| \rightarrow \infty$ with the additionally restriction that $\mathbf{w} \cdot \phi(x) \neq 0$. To see why, let us see how p is affected as $\|\mathbf{w}\|$ changes.

$$\begin{aligned}\lim_{\|\mathbf{w}\| \rightarrow \infty} p &= \lim_{\|\mathbf{w}\| \rightarrow \infty} \frac{1}{1 - e^{-\mathbf{w} \cdot \phi(x)}} \\ &= \lim_{\|\mathbf{w}\| \rightarrow \infty} \frac{1}{1 - e^{-\|\mathbf{w}\| \mathbf{u} \cdot \phi(x)}} \quad (\mathbf{u} = \frac{\mathbf{w}}{\|\mathbf{w}\|})\end{aligned}$$

At this point, we have two options. The first, is $\mathbf{u} \cdot \phi(x) > 0$ or $\mathbf{u} \cdot \phi(x) < 0$ (the $= 0$ case is not possible by our constraints). In the first case, we'll have:

$$\lim_{\|\mathbf{w}\| \rightarrow \infty} p = 1$$

while in the second case, we have:

$$\lim_{\|\mathbf{w}\| \rightarrow \infty} p = 0$$

In either scenario, we have:

$$\lim_{\|\mathbf{w}\| \rightarrow \infty} \nabla_{\mathbf{w}} \text{Loss}(x, y, -c \frac{\phi(x)}{\|\phi(x)\|_2^2}) = 0$$

From the above, we see that we can make the gradient be as small as we'd like. The intuition is that we can make the gradient arbitrarily small as long as we can make $\|\mathbf{w}\|$ arbitrarily large.

However, we note that the magnitude of the gradient will never be exactly zero.

- (d) In terms of making the gradient be large, this is achieved when $p - 2p^2 + p^3$ is maximized in the interval $[0, 1]$. We note that the derivative is $1 - 4p + 3p^2 = (3p - 1)(p - 1)$ which has roots at $p = 1$ and $p = \frac{1}{3}$. From the results above, we know that $p = 1$ is a local minimum. We note that the function is convex on $[0, 1]$, and as such, $p = \frac{1}{3}$ is a local maximum.

Therefore, maximum magnitude that the gradient can take occurs at $p = \frac{1}{3}$ and is given by:

$$2 \left(\frac{4}{27} \right) \|\phi(x)\|_2 = \frac{8}{27} \|\phi(x)\|_2$$

(e) In order to generate our new dataset, we simply transform $y \rightarrow y'$ as follows:

$$y' = \ln \left(\frac{y}{1-y} \right)$$

We claim that there $\exists \mathbf{w}^*$ such that \mathbf{D}' has zero loss when the prediction is given by $\hat{y}' = \mathbf{w}^* \cdot \phi(x)$ (a linear predictor). To see why, first let us solve for y given our above definition of y' :

$$\begin{aligned} y' &= \ln \left(\frac{y}{1-y} \right) \\ \implies e^{y'} &= \frac{y}{1-y} \\ \implies e^{y'} &= \frac{1}{\frac{1}{y} - 1} \\ \implies \frac{1}{y} &= e^{-y'} + 1 \\ \implies y &= \frac{1}{1 + e^{-y'}} \end{aligned}$$

With the above in hand, let us now see why we can achieve zero loss on our new dataset with standard linear regression. We note that:

$$\begin{aligned} \left(y - \frac{1}{1 + e^{-\mathbf{w} \cdot \phi(x)}} \right)^2 &= 0 \quad (\text{given in the problem that such a } \mathbf{w} \text{ exists}) \\ \implies \frac{1}{1 + e^{-y'}} - \frac{1}{1 + e^{-\mathbf{w} \cdot \phi(x)}} &= 0 \\ & \quad (\text{solving for } y \text{ given our definition of } y' \text{ and substituting}) \\ \implies y' &= \mathbf{w} \cdot \phi(x) \\ \implies (y' - \mathbf{w} \cdot \phi(x))^2 &= 0 \end{aligned}$$

We therefore have $\mathbf{w}^* = \mathbf{w}$ which converges to zero loss on \mathbf{D}' .

Problem 3

- (a) In “submissions.py”.
- (b) In “submissions.py”.
- (c) In “submissions.py”.
- (d) We look through some incorrect predictions for our system.

- (a) “this is a superior horror flick .” was misclassified as negative (-1) mostly because of the large negative weights associated with “horror” and “flick” despite the fact that these are mostly noun descriptors for the movie genre, rather than the opinion of the reviewer.
 - (b) “you’re better off staying home and watching the x-files .” was misclassified as positive (+1), mostly because it appears to be near the middle – almost all of the words have low weights ($-0.15 < x < 0.23$) and this barely passes as positive.
 - (c) “wickedly funny , visually engrossing , never boring , this movie challenges us to think about the ways we consume pop culture .” was misclassified as negative (-1) almost entirely due to two words (“never” and “boring”) which have some of the largest negative weights, despite semantically negating (and becoming positive) in this case.
 - (d) “accuracy and realism are terrific , but if your film becomes boring , and your dialogue isn’t smart , then you need to use more poetic license .” was misclassified as positive (+1) due to its complex structure, conditionals, and sheer number of positive to neutral words which become negative only due to the negations and conditionals.
 - (e) “a solid film . . . but more conscientious than it is truly stirring .” was misclassified as positive (+1) mostly due to its use of strongly positive words, such as “solid” and “truly” which were semantically used as a contrast rather than direct descriptors of the film.
- (e) In “submissions.py”.
- (f) We can see our exploration for different values of n in Table 1. The lowest test-error given our tested values is found for $n = 7$, which appears to be a minimum with smaller values increases the test error (insufficient model capacity) and larger values also increasing the test error (overfitting on the training set, or also insufficient training data due to longer n-grams extracting fewer features).

We can explain the $n \in \{4, 5, 6, 7\}$ nearing and surpassing our performance with the word extractor by nothing a few factors. With a such n-grams, most “words” will end-up being partially extracted in their entirety, thereby explaining why a match in performance with our first model is non unexpected. The improvement in performance likely stems from the fact that the n-gram can actually take into account the relationships between words. For example, “not bad” will have two word features (“not” and “bad”) which will likely both be negative (despite its positive sentiment). However, a 4-gram can capture the near-word feature (“notb”, “tbad”) as well as the transitional feature sof “otba”, thereby giving the model the ability to differentiate between just “not” (negative) and “not bad” (positive).

We can imagine a review such as the following:

“not bad horror flick, never slow, truly marvelous.”

is far more likely to be correctly classified by an n-gram model than a bag-of-words model due to the constant double negations and inter-word relations (“horror flick”) which can only be captured by considering multiple words at once. We can verify this, as our trained bag-of-words models predicts a (-1), mostly due to all the negative words, while our 5-gram model classifies it correctly.

	train error	dev error
1	0.479	0.492
2	0.317	0.410
3	0.002	0.323
4	0.0	0.281
5	0.0	0.275
6	0.0	0.275
7	0.0	0.273
8	0.001	0.291
9	0.001	0.308
10	0.001	0.333
15	0.002	0.45
19	0.006	0.49

Table 1: Test and Train Error with n-gram feature extractions

Problem 3

(a) We run 2-means on the given dataset.

(a) We begin the algorithm with the centroids $\mu_1 = [2, 3]$ and $\mu_2 = [2, -1]$.

(1) We first compute the cluster assignments. We have:

$$d(\mu_1, \phi(x_1)) = 10$$

$$d(\mu_2, \phi(x_1)) = 2$$

$$d(\mu_1, \phi(x_2)) = 2$$

$$d(\mu_2, \phi(x_2)) = 10$$

$$d(\mu_1, \phi(x_3)) = 10$$

$$d(\mu_2, \phi(x_3)) = 2$$

$$d(\mu_1, \phi(x_4)) = 2$$

$$d(\mu_2, \phi(x_4)) = 9$$

which implies our assignments are:

$$z_1 = 2$$

$$z_2 = 1$$

$$z_3 = 2$$

$$z_4 = 1$$

(2) Given the above cluster assignments, we can compute the new centroids as:

$$\mu_1 = [1.5, 2]$$

$$\mu_2 = [1.5, 0]$$

(3) We now recompute the assignments, and we have:

$$d(\mu_1, \phi(x_1)) = 4.25$$

$$d(\mu_2, \phi(x_1)) = 2.25$$

$$d(\mu_1, \phi(x_2)) = 0.25$$

$$d(\mu_2, \phi(x_2)) = 4.25$$

$$d(\mu_1, \phi(x_3)) = 8.25$$

$$d(\mu_2, \phi(x_3)) = 4.25$$

$$d(\mu_1, \phi(x_4)) = 0.25$$

$$d(\mu_2, \phi(x_4)) = 4.25$$

which implies:

$$z_1 = 2$$

$$z_2 = 1$$

$$z_3 = 2$$

$$z_4 = 1$$

The assignments are unchanged. As such, we can stop the algorithm with the above assignments and centroids.

(b) We now do another iteration with the initialize centroids as $\mu_1 = [0, 1]$ and $\mu_2 = [3, 2]$.

i. We first compute the assignments:

$$\begin{aligned}
d(\mu_1, \phi(x_1)) &= 2 \\
d(\mu_2, \phi(x_1)) &= 8 \\
d(\mu_1, \phi(x_2)) &= 2 \\
d(\mu_2, \phi(x_2)) &= 4 \\
d(\mu_1, \phi(x_3)) &= 10 \\
d(\mu_2, \phi(x_3)) &= 4 \\
d(\mu_1, \phi(x_4)) &= 5 \\
d(\mu_2, \phi(x_4)) &= 1
\end{aligned}$$

which implies our assignments are:

$$\begin{aligned}
z_1 &= 1 \\
z_2 &= 1 \\
z_3 &= 2 \\
z_4 &= 2
\end{aligned}$$

ii. Given the above cluster assignments, we can compute the new centroids as:

$$\begin{aligned}
\mu_1 &= [1, 1] \\
\mu_2 &= [2.5, 1]
\end{aligned}$$

iii. We now recompute the assignments, and we have:

$$\begin{aligned}
d(\mu_1, \phi(x_1)) &= 1 \\
d(\mu_2, \phi(x_1)) &= 3.25 \\
d(\mu_1, \phi(x_2)) &= 1 \\
d(\mu_2, \phi(x_2)) &= 3.25 \\
d(\mu_1, \phi(x_3)) &= 5 \\
d(\mu_2, \phi(x_3)) &= 1.25 \\
d(\mu_1, \phi(x_4)) &= 2 \\
d(\mu_2, \phi(x_4)) &= 1.25
\end{aligned}$$

which implies:

$$\begin{aligned}
z_1 &= 1 \\
z_2 &= 1 \\
z_3 &= 2 \\
z_4 &= 2
\end{aligned}$$

The assignments are unchanged. As such, we can stop the algorithm with the above assignments and centroids.