CS221 Fall 2018 Homework [Sentiment]

SUNet ID: prabhjot
Name: Prabhjot Singh Rai

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) Mapping reviews into feature vectors as follows,

$$\phi_{x1} = \{pretty : 1, bad : 1\}, y_1 = -1$$
$$\phi_{x2} = \{good : 1, plot : 1\}, y_2 = +1$$
$$\phi_{x3} = \{not : 1, good : 1\}, y_3 = -1$$
$$\phi_{x4} = \{pretty : 1, scenery : 1\}, y_4 = +1$$

Recalling from the graph, gradient of hinge loss, for margin less than one, will be $-\phi_{(x)}y$ and 0 for margin greater than one.

$$\nabla_w Loss_{hinge}(x, y, w) = \begin{cases} -\phi_{(x)}y & \text{when}(w.\phi)y < 1 \\ 0 & \text{when}(w.\phi)y > 1 \end{cases}$$

Stochastic gradient descent is defined as

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

Initialising $\mathbf{w} = [0, \ldots 0]$, or $\mathbf{w} = \{pretty : 0, bad : 0 \ldots scenery : 0\}$, and iterating over each feature vector to update w
First iteration, $w.\phi_{x1}y = 0, \nabla Loss = -\phi_{(x)}y = \{pretty : 1, bad : 1\}$

$$w = w - \eta(\{pretty : 1, bad : 1\})$$
$$w = \{pretty : 0, bad : 0 \ldots scenery : 0\} - \{pretty : 0.5, bad : 0.5\}$$
$$w = \{pretty : -0.5, bad : -0.5\}$$

Second iteration, $w.\phi_{x2}y = 0, \nabla Loss = -\phi_{(x)}y = \{good : -1, plot : -1\}$

$$\begin{aligned} w &= w - \eta\{good : -1, plot : -1\} \\ &= \{pretty : -0.5, bad : -0.5\} - 0.5\{good : -1, plot : -1\} \\ &= \{pretty : -0.5, bad : -0.5, good : 0.5, plot : 0.5\} \end{aligned}$$

Third iteration, $w.\phi_{x3}y = -0.5, \nabla Loss = -\phi_{(x)}y = \{not : 1, good : 1\}$

$$\begin{aligned} w &= w - 0.5\{not : 1, good : 1\} \\ &= \{not : -0.5, bad : -0.5, plot : 0.5, pretty : -0.5\} \end{aligned}$$

Fourth iteration, $w.\phi_{x4}y = -0.5, \nabla Loss = -\phi_{(x)}y = \{pretty : -1, scenery : -1\}$

$$\begin{aligned} w &= w - \{pretty : -0.5, scenery : -0.5\} \\ &= \{scenery : 0.5, plot : 0.5, bad : -0.5, not : -0.5\} \end{aligned}$$

Therefore, weights of the six words are $\{pretty : 0, good : 0, bad : -0.5, plot : 0.5, not : -0.5, scenery : 0.5\}$

(b) Labelled dataset:
1. "not good" (-1)
2. "good" (+1)
3. "bad" (-1)
4. "not bad" (+1)

Let $\{not : x, good : y, bad : z\}$ be the weights assigned to each feature(our feature extractor considering only single words as per the question), where x, y and z can be both positive and negative. In order to get a total of zero error on all data points, we need to get zero error on each data point. For "good" review, score should be positive, therefore, $y > 0$. For bad, score should be negative, therefore, $z < 0$. For "not good", $x + y$ should be < 0, therefore, since $y > 0$, $x < 0$ and $x < -y$. For "not bad", $x + z$ should be > 0, but through former calculations, $x + z$ will be < 0. Therefore, no linear classifier using word features can get zero error on this dataset.

In order to get a zero error, we can append our feature vector with a bi-gram of words, for example "not bad". This additional feature would add weight t to "not bad" in the last datum. In such a scenario, weights of each feature would be "good" > 0, "bad" < 0, "not" < 0 and "not bad" > 0 ("not bad"'s weight surpassing weights of "not" and "bad" combined). This way we can achieve zero error.

Problem 2

(a)

$$\begin{aligned} f_w(x) &= \sigma(w.\phi_x) \\ &= (1 + e^{-w.\phi_x})^{-1} \\ Loss_{squared}(x, y, w) &= (f_w(x) - y)^2 \\ &= ((1 + e^{-w.\phi_x})^{-1} - y)^2 \end{aligned}$$

(b) Let $p = \sigma(w.\phi_x) = (1 + e^{-w.\phi_{(x)}})^{-1}$

$$\begin{aligned}
\nabla_w Loss &= \frac{d(p-y)^2}{dw} \\
&= \frac{d(p-y)^2}{dp}\frac{dp}{dw} \\
&= 2(p-y)\frac{dp}{dw} && \textbf{...(1)} \\
\frac{dp}{dw} &= \frac{d(1+e^{-w.\phi_{(x)}})^{-1}}{dw} \\
&= (-1)(1+e^{-w.\phi_{(x)}})^{-2}e^{-w.\phi(x)}(-\phi_{(x)}) \\
&= (p)^2\frac{(1-p)}{p}\phi_{(x)} && (\text{since } p = (1+e^{-w.\phi_x})^{-1}, \text{therefore } e^{-w.\phi_{(x)}} = \frac{1-p}{p}) \\
&= p(1-p)\phi_{(x)}
\end{aligned}$$

Therefore, substituting the value of $\frac{dp}{dw}$ in (1), we get gradient of the loss is

$$\nabla_w Loss = 2(p-y)p(1-p)\phi_{(x)}$$

(c) Substituting $y = 1$ and arbitary $\phi_{(x)}$ in the above equation,

$$\begin{aligned}
\nabla_w Loss &= 2(p-1)p(1-p)\phi_{(x)} \\
&= -2(p-1)^2 p\phi_{(x)} && \textbf{(...2)}
\end{aligned}$$

In order to make the magnitude of the gradient of the loss arbitrarily small, we need to make the above equation close to zero. That can happen when p approaches 1 and p approaches zero.
When p approaches 1:

$$\begin{aligned}
p &= 1 \\
(1+e^{-w.\phi_{(x)}})^{-1} &= 1 \\
e^{-w.\phi_{(x)}} &= 0
\end{aligned}$$

w approaches ∞.
When p approaches 0:

$$\begin{aligned}
p &= 0 \\
(1+e^{-w.\phi_{(x)}})^{-1} &= 0 \\
e^{-w.\phi_{(x)}} &\text{ approaches } \infty
\end{aligned}$$

w approaches $-\infty$. Therefore, for w approaching very large magnitude(∞ and $-\infty$), magnitude of the gradient of the loss is arbitrarily small(approaching zero).
No, the magnitude of the gradient can never be zero.

(d) For largest magnitude of the gradient, in the equation **2** in **2c** above, we need to maximize $(p-1)^2 p$. Differentiating and equating to zero.

$$\begin{aligned} 2p(p-1) + (p-1)^2 &= 0 \\ 2p^2 - 2p + p^2 + 1 - 2p &= 0 \\ 3p^2 - 4p + 1 &= 0 \qquad \textbf{(...3)} \\ 3p^2 - 3p - p + 1 &= 0 \\ (3p-1)(p-1) &= 0 \\ p &= 1, \frac{1}{3} \end{aligned}$$

To check if it's minima or maxima, we differentiate equation 3 and check signs

$$6p - 4 \text{ is negative only when } p = \frac{1}{3}$$

Therefore, maximum ma of function $p(p-1)^2$ when p ranges from 0 to 1 is

$$p = \frac{1}{3}$$

Substituting value of p in equation **2** in **2c** above

$$\begin{aligned} \nabla_w Loss &= ||2(\frac{2}{3})^2 \frac{1}{3} \phi_{(x)}|| \\ &= ||(\frac{2}{3})^3 \phi_{(x)}|| \\ &= ||\frac{8}{27}\phi(x)|| \end{aligned}$$

(e) As we know that Loss is difference between the predicted value and actual value:

$$\begin{aligned} Loss_{w,x,y} &= PredictedOutput - Actual \\ Loss_{w,x,y} &= \sigma(w\phi_{(x)}) - y \end{aligned}$$

Simplifying the loss equation for a given point,

$$
\begin{aligned}
Loss_{w,x,y} &= (\sigma(w\phi_{(x)}) - y)^2 && \textbf{...(1)} \\
0 &= \sigma(w\phi_{(x)}) - y \\
y &= \frac{1}{(1 + e^{-w\phi_{(x)}})} \\
e^{-w\phi_{(x)}} &= \frac{1-y}{y} \\
-w\phi_{(x)} &= \ln \frac{1-y}{y} \\
w\phi_{(x)} &= \ln \frac{y}{1-y} \\
w\phi_{(x)} - \ln \frac{y}{1-y} &= 0 && \textbf{(...(2))}
\end{aligned}
$$

The above equation is similar to linear predictor's loss, $w^*\phi_{(x)} - y'$. Since it's given that the loss is zero for the given equation 1, squared loss on every data point has to be zero. From the modified equation 2, least squares regression also yields zero loss on D', which has data points x, y'. In this dataset, we compute y' by

$$
\begin{aligned}
y' &= \ln \frac{y}{1-y} \\
&= \ln y - \ln(1-y)
\end{aligned}
$$

The solution of the squared loss will yield vector w*, and since the total squared loss is zero, D' should converge to a vector w^*.

Justification of $y' = \ln y - \ln(y-1)$
As we can deduce from equation 1 that y can take a value between zero and 1 (since it's equal to a sigmoid), ln y - ln (y - 1) in the range of 0 to 1 can take up any value between $-\infty$ to $+\infty$. Since $\phi_(x)$ can take up any value, this range for y' can easily compensate for any value of the dot product with w to get a zero loss.

Problem 3

(d)

(1) home alone goes hollywood , a funny premise until the kids start pulling off stunts not even steven spielberg would know how to do . besides , real movie producers aren't this nice .
Truth: -1, Prediction: 1 [WRONG]

This is predicted wrong because "funny" has a count of 1 with a heavy positive weight of 0.4, since with most of the movies, funny sounds positive but in the context, the review starts off as positive but the tone changes as the review proceeds to negative, which isn't given accurate weight to by the model.

(2) '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 our weights, sweet has high positive weight of 0.55, whose score overshoots scores of other negative words. but is being used to describe the "home" and not the review in general.

(3) patchy combination of soap opera , low-tech magic realism and , at times , ploddingly sociological commentary
Truth: -1, Prediction: 1 [WRONG]

We see many negative words here, but their weights are zero, for example patchy, low-tech, ploddingly etc. That means our model hasn't seen such words in training, therefore based on other words classifies this as positive.

(4) the best thing i can say about this film is that i can't wait to see what the director does next
Truth: 1, Prediction: -1 [WRONG]

The words such as "does" and "film" have very high positive weights(0.6 and 0.36 to be precise), but in general have equal probability of coming in a positive or negative review. But in our training data, these words occur more frequently in positive than negative reviews, that's why they have high positive values. Ignoring such stop words (words which do not tell about actual sentiment) can make the model better.

(5) even during the climactic hourlong cricket match , boredom never takes hold
Truth: 1, Prediction: -1 [WRONG]

Words like "never", "boredom" etc increase negative sentiment a lot, but in context, the movie review says that negative doesn't happen. Since we aren't considering context through n-grams or some other technique, this is classified only on single word basis hence wrong.

(f) When n = 1, $Testerror = 0.478615644344$
When n = 2, $TestError = 0.419527293191$
When n = 3, $TestError = 0.318232976927$

When n = 4, $TestError = 0.283342712437$
When n = 5, $TestError = 0.270680922904$
When n = 6, $TestError = 0.271525042206$
When n = 7, $TestError = 0.270962296005$
When n = 8, $TestError = 0.292909397862$
When n = 9, $TestError = 0.308384918402$

When n=5, the test error is the minimum. Test error reflects how well our feature vector has defined features which eventually are used in classification. If we look at the weights data from the weights file when our feature extractor is word, maximum number of words which "matter"(having high positive or negative weights which will help increase the score) are for word lengths 5. Here are the word lengths against their counts for weights less than -0.5 and greater than $+0.5$:

$\{1 : 2.0, 2 : 1.0, 3 : 4.0, 4 : 29.0, 5 : 42.0, 6 : 27.0, 7 : 27.0, 8 : 18.0, 9 : 13.0, 10 : 11.0, 11 : 8.0, 12 : 6.0, 13$

Therefore, having feature vectors with length 5 help to classify the sentiments better.

A review such as "not good" will be better classified by n-gram model than word one. The reason is, while learning, it will take context of words into account as well and will encounter more examples having "notgo", "otgoo", "tgood" etc. as negatively classified than a word one. "not" and "good" can be present in both positive and negative review. For example, in the 3-b example, for word classifier, "not" has -0.14 and "good" has 0.29, therefore gets classified as positive, whereas in character 5-gram,{ "notgo": -0.05, "otgoo": -0.02, "tgood": -0.07 } gets classified as negative.

Problem 4

(a)

$$\phi_{x1} = [1, 0]$$
$$\phi_{x2} = [1, 2]$$
$$\phi_{x3} = [3, 0]$$
$$\phi_{x4} = [2, 2]$$

(1) **Iteration 1:** $\mu_1 = [2, 3]$ and $\mu_2 = [2, -1]$

$$
\begin{aligned}
|\phi_{x1} - \mu_1|^2 &= 1^2 + 3^2 \\
&= 1 + 9 \\
&= 10 \\
|\phi_{x1} - \mu_2|^2 &= 1^2 + 1^2 = 2
\end{aligned}
$$

Therefore, ϕ_{x1} gets assigned to μ_2.

$$
\begin{aligned}
|\phi_{x2} - \mu_1|^2 &= 1^2 + 1^2 = 2 \\
|\phi_{x2} - \mu_2|^2 &= 1^2 + 3^2 = 10
\end{aligned}
$$

Therefore, ϕ_{x2} gets assigned to μ_1.

$$
\begin{aligned}
|\phi_{x3} - \mu_1|^2 &= 1^2 + 3^2 = 10 \\
|\phi_{x3} - \mu_2|^2 &= 1^2 + 1^2 = 2
\end{aligned}
$$

Therefore, ϕ_{x3} gets assigned to μ_2.

$$
\begin{aligned}
|\phi_{x4} - \mu_1|^2 &= 1^2 = 1 \\
|\phi_{x4} - \mu_2|^2 &= 3^2 = 9
\end{aligned}
$$

Therefore, ϕ_{x4} gets assigned to μ_1.
ϕ_{x1} and ϕ_{x3} belong to μ_2, ϕ_{x2} and ϕ_{x4} belong to μ_1
New Means:

$$
\begin{aligned}
\mu_1 &= \frac{\phi_{x2} + \phi_{x4}}{2} = [1.5, 2] \\
\mu_2 &= \frac{\phi_{x1} + \phi_{x3}}{2} = [2, 0]
\end{aligned}
$$

Iteration 2: $\mu_1 = [1.5, 2]$ and $\mu_2 = [2, 0]$

$$|\phi_{x1} - \mu_1|^2 = 0.5^2 + 2^2 = 4.25$$
$$|\phi_{x1} - \mu_2|^2 = 1^2 = 1$$

Therefore, ϕ_{x1} gets assigned to μ_2.

$$|\phi_{x2} - \mu_1|^2 = 0.5^2 = 0.25$$
$$|\phi_{x2} - \mu_2|^2 = 1^2 + 2^2 = 5$$

Therefore, ϕ_{x2} gets assigned to μ_1.

$$|\phi_{x3} - \mu_1|^2 = 1.5^2 = 2.25$$
$$|\phi_{x3} - \mu_2|^2 = 1^2 = 1$$

Therefore, ϕ_{x3} gets assigned to μ_2.

$$|\phi_{x4} - \mu_1|^2 = 0.5^2 = 0.25$$
$$|\phi_{x4} - \mu_2|^2 = 2^2 = 4$$

Therefore, ϕ_{x4} gets assigned to μ_1. ϕ_{x1} and ϕ_{x3} belong to μ_2, ϕ_{x2} and ϕ_{x4} belong to μ_1, which is same as previous iteration. Hence $[z_1, z_2, z_3, z_4] = [2, 1, 2, 1]$

(2) **Iteration 1:** $\mu_1 = [0, 1]$ and $\mu_2 = [3, 2]$

$$|\phi_{x1} - \mu_1|^2 = 1^2 + 1^2 = 2$$
$$|\phi_{x1} - \mu_2|^2 = 2^2 + 2^2 = 8$$

Therefore, ϕ_{x1} gets assigned to μ_1.

$$|\phi_{x2} - \mu_1|^2 = 1^2 + 1^2 = 2$$
$$|\phi_{x2} - \mu_2|^2 = 2^2 = 4$$

Therefore, ϕ_{x2} gets assigned to μ_1.

$$|\phi_{x3} - \mu_1|^2 = 3^2 + 1^2 = 10$$
$$|\phi_{x3} - \mu_2|^2 = 2^2 = 4$$

Therefore, ϕ_{x3} gets assigned to μ_2.

$$|\phi_{x4} - \mu_1|^2 = 2^2 + 1^2 = 5$$
$$|\phi_{x4} - \mu_2|^2 = 1^2 = 1$$

Therefore, ϕ_{x4} gets assigned to μ_2.
ϕ_{x1} and ϕ_{x2} are assigned to μ_1 and ϕ_{x3} and ϕ_{x4} are assigned to μ_2.
New Means:

$$\mu_1 = \frac{\phi_{x1} + \phi_{x2}}{2} = [1, 1]$$
$$\mu_2 = \frac{\phi_{x3} + \phi_{x4}}{2} = [2.5, 1]$$

Iteration 2: $\mu_1 = [1, 1]$ and $\mu_2 = [2.5, 1]$

$$|\phi_{x1} - \mu_1|^2 = 1^2 = 1$$
$$|\phi_{x1} - \mu_2|^2 = 1.5^2 + 1^2 = 3.25$$

Therefore, ϕ_{x1} gets assigned to μ_1.

$$|\phi_{x2} - \mu_1|^2 = 1^2 = 1$$
$$|\phi_{x2} - \mu_2|^2 = 1.5^2 + 2^2 = 3.25$$

Therefore, ϕ_{x2} gets assigned to μ_1.

$$|\phi_{x3} - \mu_1|^2 = 2^2 = 4$$
$$|\phi_{x3} - \mu_2|^2 = 0.5^2 + 1^2 = 1.25$$

Therefore, ϕ_{x3} gets assigned to μ_2.

$$|\phi_{x4} - \mu_1|^2 = 1^2 + 1^2 = 2$$
$$|\phi_{x4} - \mu_2|^2 = 0.5^2 + 1^2 = 1.25$$

Therefore, ϕ_{x4} gets assigned to μ_2.
ϕ_{x1} and ϕ_{x2} are assigned to μ_1 and ϕ_{x3} and ϕ_{x4} are assigned to μ_2, which is same as previous iteration. Hence $[z_1, z_2, z_3, z_4] = [1, 1, 2, 2]$

(b) Let the points be $x_1, x_2...x_n$. As given, S is the set of tuples. Let's define them as $S\epsilon\{(x_{t_1}, x_{t_2}), (x_{t_3}, x_{t_4})...(x_{t_{2d-1}}, x_{t_{2d}})\}$, where d is the unique number of relations described in S, $x_{t_1}, x_{t_2}, ...x_{t_{2d}}\epsilon\{x_1, x_2...x_n\}$. Let d' be the unique number of points in S.

As the first step, we can group the terms which should belong to same cluster together. Let K' be the number of such clusters. Let's call this set S' and,

$$S' = \{(x_{11}, x_{12}, x_{13}...x_{1l_1}), (x_{21}, x_{22}, x_{23}...x_{2l_2}), ...(x_{k'1}, x_{k'2}, x_{k'3}...x_{k'l_{K'}})\}$$

where $l_1, l_2, ...l_{K'}$ are the length of the clusters 1 to K'. Also, as per the question, each element in $(x_{11}, x_{12}, x_{13}...x_{1l_1})$ to $(x_{k'1}, x_{k'2}, x_{k'3}...x_{k'l_{K'}})$ is unique within their own group and also belongs to only one cluster at a time.
We can consider the centroid for each group of vectors to proceed for the clustering with unpaired/other paired group of vectors. Let $\{\mu_{l1}^{k'}, \mu_{l2}^{k'}, ...\mu_{l_{K'}}^{k'}\}$ be the centroid of each cluster.
Thus, in our modified algorithm, the feature vectors will be consisting of two sets, one being $\phi_{x_i'}$ for every $x_{i'}\epsilon\{x_1, x_2, ...x_n\}$ when $x_{i'}$ not in S and other being centroids of points of each pre defined clusters, $\{\mu_1^{k'}, \mu_2^{k'}, ...\mu_{K'}^{k'}\}$. Also, stating again, length of each cluster is assumed to be $l_1, l_2, ...l_{K'}$.

For dividing the set of points to K clusters

Step 1
Total length of points will be $n - d' + K'$ (Total points minus number of unique points in S, added to number of pre defined clusters).
Setting assignments Z given μ, where we initialise μ to a random value.
For each point $i' = 1, ...n - d'$
Assign i' to cluster with closest centroid:

$$Z_{i'} \leftarrow argmin_{k=1,...K}||\phi_{x_i'} - \mu_k||$$

For remaining points which are centroids of points which should belong to same cluster, $i' = 1, ...K'$
Assign i' to cluster with closest centroid:

$$Z_{n-d'+i'} \leftarrow argmin_{k=1,...K}||\mu_{i'}^{k'} - \mu_k||$$

Step 2
For each cluster $k = 1, ..., K$:
Set μ_k to average of points assigned to cluster k:

$$\mu_k \leftarrow \frac{1}{|\{i' : z = k\}| + \sum_{i':z=k} li'} \sum_{i':z_i'=k} \phi(x_i') + \sum_{i':z=k} \mu_{i'}^{k'} li'$$

which is essentially the centroid of cluster formed by "feature vector for every $x_{i'}$ when $x_{i'}$ not in S and centroid of pre defined clusters multiplied by their lengths" divided by "sum of lengths of pre defined clusters and number of points considered in $x_{i'}$".

(c) The advantage of running K-means multiple times on the same dataset with the same K, but different random initializations is that K-means converges to local minima whereas we might miss the global minima. Random initializations help us decreasing probability of missing global minima.

(d) Yes, if we scale all dimensions in our initial centroids and data points by some factor, we are guaranteed to retrieve the same clusters after running K-means. The distances of centroids and points will be multiplied by the same term for all iterations for each point and centroid pair, therefore comparison for choosing the cluster centroid remains the same.
If we scale only certain dimensions, then we are not guaranteed to retrieve the same clusters. Consider two dimensional example, $\phi(x) = [1, 3]$. Let one of the centroids is $\mu_1 = [3, 3]$ and other one be $\mu_2 = [1, -1]$. Suppose we scale y axis with λ. Then

$$|\phi_x - \mu_1|^2 = 4$$
$$|\phi_x - \mu_2|^2 = 4\lambda^2$$

If λ is ¿ 1, ϕ_x belongs to μ_1 but if λ ¡ 1, ϕ_x belongs to μ_2.