CS221 Fall 2018 Homework [sentiment]

SUNet ID: [chiwang]
Name: [Chi Wang]

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

Problem 1

1.a:

Answer: Let's go thru 4 data points one by one, then we will have answer at the end. First, let's set w = [0, 0, 0, 0, 0, 0], which maps the weight to each words here ["pretty", "good", "bad", "plot", "not", "scenery"], and calculate hinge loss function's derivative function below.

$$\nabla_w Loss_{hinge}(x,y,z) = \begin{cases} -y\phi(x) & if \quad y\phi(x)\cdot w < 1 \quad (1) \\ 0 & if \quad y\phi(x)\cdot w \geq 1 \quad (2) \end{cases}$$

Then let's process the first training data: ("pretty bad", -1), $\phi(x)$={pretty:1,bad:1}.

$$\begin{aligned} w &= [0,0,0,0,0,0] - 0.5 \times 1 \cdot [1,0,1,0,0,0] & (3) \\ w &= [-0.5,0,-0.5,0,0,0] & (4) \end{aligned}$$

Second training data: ("good plot", +1), $\phi(x)$={good:1,plot:1}.

$$\begin{aligned} w &= [-0.5,0,-0.5,0,0,0] - 0.5 \times (-1) \cdot [0,1,0,1,0,0] & (5) \\ w &= [-0.5,0.5,-0.5,0.5,0,0] & (6) \end{aligned}$$

Third training data: ("not good", -1), $\phi(x)$={good:1,not:1}.

$$\begin{aligned} w &= [-0.5,0.5,-0.5,0.5,0,0] - 0.5 \times (+1) \cdot [0,1,0,0,1,0] & (7) \\ w &= [-0.5,0,-0.5,0.5,-0.5,0] & (8) \end{aligned}$$

Second training data: ("pretty scenery", +1), $\phi(x)$={pretty:1,scenery:1}.

$$\begin{aligned} w &= [-0.5,0,-0.5,0.5,-0.5,0] - 0.5 \times (-1) \cdot [1,0,0,0,0,1] & (9) \\ w &= [0,0,-0.5,0.5,-0.5,0.5] & (10) \end{aligned}$$

Thurs we could find weights of the six words after training are [0, 0, 0.5, -0.5, 0.5, -0.5].

1.b:

Answer: Let's create a small labeled dataset of four mini-reviews using the words "not", "good", and "bad" below:

- (+1) not bad
- (-1) not good
- (+1) good
- (+1) good bad

In order to get a linear classifier using word features which can get zero error on above dataset, we need to solve below equation.

$$M = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \quad M \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

Since $|M| = 0$, so we couldn't find an answer vector w for above equation, there is no linear classifier could have zero error for above data set.

Let's add one more feature - "word count" to solve this problem, "word count" means the number of word in the comments, then we have a new equation as below:

$$M = \begin{pmatrix} 1 & 0 & 1 & 2 \\ 1 & 1 & 0 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}, \quad M \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

Afer calculation, we could find a classifier with $w = [-5, -2, 0, 3]$ mapping to features ["not", "good", "bad", word_count] has zero error for the dataset we built here.

Problem 2

2.a:

Answer: $Loss(x, y, w) = ((1 + e^{-w\cdot\phi(x)})^{-1} - y)^2$

2.b:

Answer: Let's calculate gradient of the LOSS function with chain rule.

$$h(p) = (p - y)^2 \qquad p(q) = (1 + q)^{-1} \qquad q(x) = e^{-w\cdot\phi(x)} \tag{11}$$

$$\frac{dh}{dp} = 2(p - y) \qquad \tfrac{dp}{dq} = -(1 + q)^{-2} \qquad \frac{dq}{dw} = e^{-w\cdot\phi(x)}(-\phi(x)) \tag{12}$$

Then we have $\nabla_w LOSS =$

$$\frac{dh}{dp} \times \frac{dp}{dq} \times \frac{dq}{dw} = -2((1+e^{-w\cdot\phi(x)})^{-1} - 1)(1+e^{-w\cdot\phi(x)})^{-2} e^{-w\cdot\phi(x)} \phi(x) \tag{13}$$

2.c:

Answer: Any positive w would make gradient small. By looking at above the gradient equation (13), we could see when $w \longrightarrow \infty$ while $\phi(x)$ is fixed, $e^{-w\cdot\phi(x)} \longrightarrow 0$ and $e^{w\cdot\phi(x)} \longrightarrow \infty$, which will make the magnitude of the gradient of the loss arbitrarily close to zero.

Also The gradient will close to zero, but never zero, as the logistic function's range is (0,1).

2.d:

Answer: when $w\phi(x) = 0$, the largest gradient can be $\frac{1}{2} \parallel \phi(x) \parallel$.
When $w\phi(x) = 0$ we could see $e^{-w\cdot\phi(x)} = 1$ and $(1+e^{-w\cdot\phi(x)})^{-1} = 0.5$, get these into gradient equation (13), we could have $\nabla_w LOSS = \frac{\phi(x)}{2}$.

2.e:

Answer: Let's transform y to y' with expression $y' = y^2$. If the predictor of y is $y = w\phi(x)$, then the predict function for y' is $y' = (w\phi(x))^2$.
In this case, we could have loss function that is always converge.

Problem 3

3.d:

Answer: By looking at weights and error-analysis file, there are quite some unmeaningful words like "and", "to", ".", "'it's". I think we would have better result if we sanitize training data.

3.f:

Answer:From machine perspective, both feature extract function: "extractCharacterFeatures" and "extractWordFeatures" are extracting sequences of chars as features from input review string, they are basically the same. It's hard to tell breaking string by whitespace into features are better than taking features by N-grams. In both method, we are training the algorithm to reduce the loss by tuning weight vector for set of chars, it just how those char sets been generated are different, so they should have similar result.

Problem 4

4.a:

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

$$z_1 = argmin\{(1-2)^2 + (0-3)^2, (1-2)^2 + (0+1)^2\} = 2 \quad (14)$$
$$z_2 = argmin\{(1-2)^2 + (2-3)^2, (1-2)^2 + (2+1)^2\} = 1 \quad (15)$$
$$z_3 = argmin\{(3-2)^2 + (0-3)^2, (3-2)^2 + (0+1)^2\} = 2 \quad (16)$$
$$z_4 = argmin\{(2-2)^2 + (2-3)^2, (2-2)^2 + (2+1)^2\} = 1 \quad (17)$$

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

$$z_1 = argmin\{(1-1.5)^2 + (0-2)^2, (1-2)^2 + (0-0)^2\} = 2 \quad (18)$$
$$z_2 = argmin\{(1-1.5)^2 + (2-2)^2, (1-2)^2 + (2-0)^2\} = 1 \quad (19)$$
$$z_3 = argmin\{(3-1.5)^2 + (0-2)^2, (3-2)^2 + (0-0)^2\} = 2 \quad (20)$$
$$z_4 = argmin\{(2-1.5)^2 + (2-2)^2, (2-2)^2 + (2-0)^2\} = 1 \quad (21)$$

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

$$z_1 = argmin\{(1-0)^2 + (0-1)^2, (1-3)^2 + (0-2)^2\} = 1 \quad (22)$$
$$z_2 = argmin\{(1-0)^2 + (2-1)^2, (1-3)^2 + (2-2)^2\} = 1 \quad (23)$$
$$z_3 = argmin\{(3-0)^2 + (0-1)^2, (3-3)^2 + (0-2)^2\} = 2 \quad (24)$$
$$z_4 = argmin\{(2-0)^2 + (2-1)^2, (2-3)^2 + (2-2)^2\} = 2 \quad (25)$$

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

$$z_1 = argmin\{(1-1)^2 + (0-1)^2, (1-2.5)^2 + (0-1)^2\} = 1 \quad (26)$$
$$z_2 = argmin\{(1-1)^2 + (2-1)^2, (1-2.5)^2 + (2-1)^2\} = 1 \quad (27)$$
$$z_3 = argmin\{(3-1)^2 + (0-1)^2, (3-2.5)^2 + (0-1)^2\} = 2 \quad (28)$$
$$z_4 = argmin\{(2-1)^2 + (2-1)^2, (2-2.5)^2 + (2-1)^2\} = 2 \quad (29)$$

4.c:

Answer: Upgraded K-means algorithm.
First, construct graph $g_1, g_2, ..., g_m$ where $g_i = \{d|(d_i, d_j) \in S\}$ from pairs $p = (d_i, d_j) \in S$, $g_i \cap g_j = \emptyset$. Define a new set $K = \{k_i | k_i \in g_i \quad and \quad k_i \notin (g_1 \cup g_2 ... \cup g_n)\}$, that data example k_i represent all examples in g_i

Then proceed with normal KMean algorithm, only difference is at the centroid assignment calculation step, if node $i \in g_j$ then we replace node i with node k_j which is the node we selected to represent all nodes in g_i. This will make sure all node in same graph been assign to same centroid/cluster all the time.

4.d:

Answer: K-means is guaranteed to decrease the loss function each iteration and will converge to a local minimum, but it is not guaranteed to find the global minimum.
So, simply run K-means several times from multiple random initializations and then choose the solution that has the lowest loss may help us to find the global minimum.

4.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)? – **Yes**
If we scale all dimensions for initial centroids and data points, it won't impact centroid assignment since the distance between centroids and all datapoint are all scaled with same factor.

What if we scale only certain dimensions? If your answer is yes, provide a short explanation. If it is no, provide a counterexample. – **No**
Let's find a counterexample, given four sample data points, $k = 2$, $d_1 = (2, 100), d_2 = (-2, 100), d_3 = (2, 0), d_4 = (-2, 0)$, clearly d_1, d_2 and d_3, d_4 are in two cluster. When we only scale y axis with $\frac{1}{100}$, then we have $d_1 = (2, 1), d_2 = (-2, 1), d_3 = (2, 0), d_4 = (-2, 0)$, we could see d_1, d_3 are in one cluster and d_2, d_4 are in another cluster, cluster result changed.