CS221 Fall 2021 Homework 2 Sentiment

SUNet ID: jxiangyu
Name: Xiangyu Liu

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) After run stochastic gradient descent once for each of the 4 samples, the weights are: [0, 0, -0.1, 0.1, -0.1, 0.1].

name	Weights					Features			
	Init	#1	#2	#3	#4	pretty bad	good plot	not good	pretty scenery
pretty	0	-.01	-.01	-.01	0	1	0	0	1
good	0	0	0.1	0	0	0	1	1	0
bad	0	-0.1	-0.1	-0.1	-0.1	1	0	0	0
plat	0	0	0.1	0.1	0.1	0	1	0	0
not	0	0	0	-0.1	-0.1	0	0	1	0
scenery	0	0	0	0	0.1	0	0	0	1

(b) Assume the weights for "bad", "good" and "not" are w_b, w_g, w_n respectively.

$$\begin{cases} f(\text{"bad"}) = w_b, \\ f(\text{"good"}) = w_g, \\ f(\text{"not bad"}) = w_n + w_d, \\ f(\text{"not good"}) = w_n + w_g, \end{cases} \tag{1}$$

To get zero error, we need the following:

$$\begin{cases} f(\text{"bad"}) = -1, \\ f(\text{"good"}) = +1, \\ f(\text{"not bad"}) = +1, \\ f(\text{"not good"}) = -1, \end{cases} \tag{2}$$

which would require:

$$\begin{cases} f(\text{"bad"}) = -1 & => w_b = -1, \\ f(\text{"good"}) = +1 & => w_g = 1, \\ f(\text{"not bad"}) = +1 & => w_n = +2, \\ f(\text{"not good"}) = -1 & => w_n = -2, \end{cases} \tag{3}$$

However, the $w_n = +2$ and $w_n = -2$ contradict each other. Therefore, no linear classifier using word features can get zero error on this dataset.

To fix this, we need to consider bi-gram features, which use pair of adjacent words as features.

Problem 2

(a)

$$\begin{aligned} Loss_{square}(x, y, \mathbf{w}) &= residual^2 \\ &= (f_w(x) - y)^2 \\ &= (\sigma(\mathbf{w} \cdot \phi(x)) - y)^2 \end{aligned} \tag{4}$$

(b)

$$\begin{aligned} \nabla_{\mathbf{w}} Loss(x, y, \mathbf{w}) &= \nabla_{\mathbf{w}}(\sigma(\mathbf{w} \cdot \phi(x)) - y)^2 \\ &= 2(\sigma(\mathbf{w} \cdot \phi(x)) - y)\nabla_{\mathbf{w}}\sigma(\mathbf{w} \cdot \phi(x)) \\ &= 2(p - y) * \nabla_{\mathbf{w}} p \end{aligned} \tag{5}$$

(c) Having a small magnitude of gradient of the loss means that the prediction $f(x)$is very close to the actual value y. and when $f(x) == y$, the magnitude of gradient of the loss is 0.

For this data point where $y = 1$, to have a 0 magnitude of gradient of the loss, we will need:

$$\begin{aligned} p = y &= 1 \\ \sigma(\mathbf{w} \cdot \phi(x) &= 1 \\ \frac{1}{1 + e^{-\mathbf{w}\phi(x)}} &= 1 \\ e^{-\mathbf{w}\phi(x)} &= 0 \\ \mathbf{w}\phi(x) &= +inf \end{aligned} \tag{6}$$

Since $\mathbf{w}$ is not infinity, and $\phi(x)$ which is the feature vector, is not infinity, then it is not possible for the magnitude of the gradient to be exactly 0.

Problem 3

(e)

When n is 7, it produces the smallest validation error. Most English words are less than 7 letters, so 7-gram can capture most English words. Furthermore, for an n-gram that is smaller than 7, e.g. 2-gram or 3-gram, it doesn't have enough letters to capture words. While a larger n-gram, e.g. 10 or 20-gram, will capture entire sentence and lose the granularity of words.

An 7-gram can not only capture words, but also capture the context around the words (e.g. the words/letters before and after the n-gram). But the word features can only use words and don't know the context.

Consider this example: ("Good movies with bad ending", -1) and ("Bad movies with good endings", 1). The word feature will always produce the same feature vector for both sentences, and thus have the same prediction for both sentences. It can never predict the correct truth for at least one of these two cases. On the other hand, the n-gram feature will be able to capture the difference between the two sentences and thus outperform the word feature.

Problem 4

(a) Classifier D will output 1 (toxicity) if and only if there is a presence of demographic mentions.

Classifier T will output 1 (toxicity) if and only if there is a presence of toxic words.

(b) For Classifier D:

y	d	t	# data points	$f_{\mathbf{w_D}}(x)$
-1	0	0	63	-1
-1	0	1	27	-1
-1	1	0	7	1
-1	1	1	3	1
1	0	0	3	-1
1	0	1	7	-1
1	1	0	27	1
1	1	1	63	1

1. Classifier D's average loss:

$$\begin{aligned}&(63*0+27*0+7*1+3*1+3*1+7*1+27*0+63*0)/200\\&=20/200\\&=0.1\end{aligned} \tag{7}$$

2. Classifier D's average loss for each group:

	y = 1	y = -1
d = 1	0	1
d = 0	1	0

3. Classifier D's maximum group loss: 1

(c) For Classifier T:

y	d	t	# data points	$f_{\mathbf{w_T}}(x)$
-1	0	0	63	-1
-1	0	1	27	1
-1	1	0	7	-1
-1	1	1	3	1
1	0	0	3	-1
1	0	1	7	1
1	1	0	27	-1
1	1	1	63	1

1. Classifier T's average loss:

$$\begin{aligned}&(63*0+27*1+7*0+3*1+3*1+7*0+27*1+63*0)/200\\&=60/200\\&=0.3\end{aligned} \tag{8}$$

2. Classifier T's average loss for each group:

	y = 1	y = -1
d = 1	0.27	0.3
d = 0	0.3	0.27

3. Classifier T's maximum group loss: 0.3

(d) Classifier D has a lower average loss. Classifier T has a lower maximum group loss.

(e) I would deploy the Classifier T, which has the lower maximum group loss, because of the **utilitarianism** principle. Because this classifier ignores the demographic feature and only focuses on the toxic words, it creates a fair prediction regardless the demographic, which in turns creates the greatest average well-being.

(f) I would chose to #2. Hire social scientists to establish criteria for toxicity and annotate each comment. and #4. Ask users of the platform to deliberate about and decide on community standards and criteria for toxicity, perhaps using a process of participatory design. Because each individual might have a different definition of toxicity, it requires a mutual understanding and agreement among the users to establish the definition and criteria for toxicity, in order to make a comment as toxic or not.

Therefore, the #2 and #4 methods would allow the platform to achieve such goal.

On the contrast, the #1 and #3 methods risk creating a biased definition of toxicity, because individual user can mark the comment as toxic or not based on individual interpretation without any mutual agreement or criteria on the definition among other users. This will create biasing in the data.

Problem 5

(a)

1.

	#1	#2 (converged)
$\phi(x_1) = [10, 0]$	μ_2	μ_2
$\phi(x_2) = [30, 0]$	μ_2	μ_2
$\phi(x_3) = [10, 20]$	μ_1	μ_1
$\phi(x_4) = [20, 20]$	μ_1	μ_1
$\mu_1 = [20, 30]$	$[15, 20]$	$[15, 20]$
$\mu_2 = [20, -10]$	$[20, 0]$	$[20, 0]$

2.

	#1	#2 (converged)
$\phi(x_1) = [10, 0]$	μ_1	μ_1
$\phi(x_2) = [30, 0]$	μ_2	μ_2
$\phi(x_3) = [10, 20]$	μ_1	μ_1
$\phi(x_4) = [20, 20]$	μ_2	μ_2
$\mu_1 = [0, 10]$	$[10, 10]$	$[10, 10]$
$\mu_2 = [30, 20]$	$[20, 10]$	$[20, 10]$

(c)

1. 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?

   Answer :Yes. Because it is a uniform scaling on all dimensions, this factor can be extracted out when the assignments and centroid locations are being calculated. Therefore, it doesn't change the final output and clusters.

2. What if we scale only certain dimensions?

   Answer :No, the cluster will change.

   Using the example in the question 5.a.2. If we scale the x value by 0.1, we will get:

	#1	#2 (converged)
$\phi(x_1) = [1, 0]$	μ_1	μ_1
$\phi(x_2) = [3, 0]$	μ_1	μ_1
$\phi(x_3) = [1, 20]$	μ_2	μ_2
$\phi(x_4) = [2, 20]$	μ_2	μ_2
$\mu_1 = [0, 10]$	$[2, 0]$	$[2, 0]$
$\mu_2 = [3, 20]$	$[1.5, 20]$	$[1.5, 20]$

   Which is different than the original clusters.