Assignment 1

CSE 517: Natural Language Processing - University of Washington
Winter 2022

Part 1: Text Classification

In this assignment, we are provided with a dataset of positively and negatively categorized reviews, as well as a vocabulary of positive and negative words. To begin, we will construct two classifiers: a sentiment lexicon and a binary logistic regression classifier.

A. Sentiment Lexicon Based Classifier

I first created a data frame with one column for "Reviews" and a second column for positive and negative sentiments.

	Reviews	Sentiments
0	coyle takes a look at the young bruin star bob...	Positive
1	ugh .	Negative
2	the raja hurls forks at foes , the shoveler hi...	Negative
3	i love you also seems to be getting raving rev...	Positive

Following that, I built a dictionary of positive and negative words using Hu and Liu's opinion lexicon English. I tokenized the text in the review's column using the re package and then randomly picked 400 samples for the test data set. A sentiment classifier based on the lexicon is then constructed by counting the number of positive and negative words in each review. Each review document is assigned positive or negative sentiments based on the highest count.

To compute the accuracy, precision, recall, and F1 score, I used the numpy library. The performance metrics for the sentiment-based lexicon classifier are listed below in Table 1.

Table 1. Performance Metrics

Topic Modeling Method	Accuracy	Precision	Recall	F1 score
Sentiment Lexicon Based Classifier	0.545	0.567	0.646	0.604

B. Binary Logistic Regression Classifier

I first began with the whole data set and divided it into 20% test set and 80%, but the computation time in that case was too long, therefore, I decided to split the data into a 20% test set (4000 samples) and an 80% training set (20000 sample) because of my computer's slow processing.

Following that, I constructed a bag of words model by first establishing a dictionary of words in which the words in the reviews serve as the value and the sentiments serve as the key. Then, as indicated in the code snippet below, I constructed a list of common words by sorting the words in the dataset with a frequency higher than 10.

```
frequent_word_set = set()
for key, value in word_dict.items():
    if value > 10:
        frequent_word_set.add(key)

frequent_word_set
```

Next, I split the training data into two sections: train X and train Y. I pre-processed the data for train Y by assigning positive sentiments a value of 1 and negative sentiments a value of zero. I designed the binary logistic regression classifier our main goal is to find the coefficients values in a linear equation which give the least classification error for that we use sigmoid activation function as shown below to classify the data in the range of 0 and 1.

$$f(x) = \frac{1}{1 + e^{-(b0+b1x)}}$$

Fig1: Sigmoid function

I have used gradient descent to create the classifier, which is an optimization technique that minimizes error by modifying the coefficient values. We change the coefficient values by computing the derivative with respect to weight and bias, and then updating the weights and bias by multiplying the derivative by a given learning rate. I used 0.0001 as the learning rate, which determines the rate's speed. This is a small value that remains between 0 and 1. A low learning rate indicates a greater number of epochs. Following that, I used the sigmoid function, which returns 0 if the output is less than or equal to 0.5 and 1 if it is more than or equal to 0.5, where 0.5 was used as a threshold.
Below is the performance metrics in table 2, that shows how my test data performed after applying it to the logistic regression model.

Table 2. Performance Metrics

Topic Modeling Method	Accuracy	Precision	Recall	F1 score
Binary Logistic Regression Classifier	0.533	0.523	0.919	0.667

C. Breaking Good

Case 1: Positive by human English speakers, and your lexicon classifier predicts it as positive, whereas your logistic regression classifier predicts it as negative.

Document: *"I've recently noticed something I had not even considered before."*
Summary: In the above example, the sentiment lexicon classifier predicts it to be positive because the document contains a greater number of positive words. However, logistic regression predicts it as negative because I believe certain of the terms in the document have higher feature weights in the training data, indicating negative feelings. For example, the word "not" has a larger feature weight in this case.

Case 2: Human positive and your lexicon classifier predicts it as negative, whereas your logistic regression classifier predicts it as positive.

Document: *"You can bet that anybody who has ever seen Karan's madly deep performances in truly unsurprised profoundly and sense and sensibility will not be shocked"*
Summary: Since the nature of both classifiers is distinct, the Sentiment lexicon classifier in this example does not evaluate the context of the document, but rather conducts math based on positive and negative counts. A word count of positive and negative terms reveals that negative words exceed positive terms. While logistic regression expects it to be positive, some of the feature words

may have higher weights, indicating positive views. Another analogy would be that certain words never appeared in the training data throughout the training process, resulting in weights being assigned to terms that may result in positive sentiments.

Case 3: Both of your classifiers predict it as negative.

Document: *"They're all unhappy or marginally so about noodles in my hair"*
Summary: This text is either positive or funny to a human interpretation. However, in this case, both classifiers failed to recognize the context and classified the content as negative, regardless of the approach employed. After perusing the dictionary used for the lexical classifier, I noticed that the term "noodle" does not exist in the dictionary, indicating that it is neither a positive nor a negative word, which may account for the wrong prediction. Similarly, with logistic regression, it is possible that words such as noodles or hair never exist in the training data, resulting in a lower feature weight and a higher weight for words such as unhappy, and ultimately classifying the sentiments as negative.

Note: Solution 2 and Solution 3 Starts from the next page using paper and pen.

References:

[1] Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright © 2021. All rights reserved. Draft of December 29, 2021.
[2] Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.
[3] Roberts, Arthur Wayne. "Convex functions." In Handbook of Convex Geometry, pp. 1081-1104. North-Holland, 1993.
[4] McKinney, Wes. "Pandas, python data analysis library." URL http://pandas. pydata. org (2015).
[5] Hellmann, Doug. The Python standard library by example. Upper Saddle River, USA: Addison-Wesley, 2011.
[6] Link: https://math.stackexchange.com/questions/3983368/the-minimum-of-the-sum-of-two-convex-functions
[7] Ed discussion board for A1 Assignment.

Solution 2.

Maximum Conditional log likelihood for dataset D1.

$$l(\theta) = \ln \prod_i P(y^i | x^i, \theta) \qquad i = \text{sample}$$

Here, $l(\theta)$ is concave function of θ. There are no close form to maximize $l(\theta)$. However, concave function are easy to optimize.

→ Optimizing a convex function - Gradient ascent

Note: Max. log likelihood = Min. negative log likelihood, then we use gradient DEscent.

→ Gradient

$$\nabla_\theta l(\theta) = \left[\frac{\partial l(\theta)}{\partial \theta_0}, \ldots \frac{\partial l(\theta)}{\partial \theta_n} \right]$$

update rule:

$$\Delta\theta = \eta \nabla_\theta l(\theta)$$

$$\theta_j^{(t+1)} \Leftarrow \theta_j^{(t)} + \eta \frac{\partial l(\theta)}{\partial \theta_j}$$

Maximum Conditional log likelihood - Gradient ascent.

$$l(\theta) = \sum_i y^i \left(\theta_0 + \sum_j \theta_j x_j^i\right) - \ln\left(1 + \exp\left(\theta_0 + \sum_j \theta_j x_j^i\right)\right)$$

taking the derivative,

$$\frac{\partial l(\theta)}{\partial \theta_i} = \sum_i \left[\frac{\partial}{\partial \theta} y^i \left(\theta_0 + \sum_j \theta_j x_j^i\right) - \frac{\partial}{\partial \theta} \ln\left(1 + \exp\left(\theta_0 + \sum_j \theta_j x_j^i\right)\right)\right]$$

$$= \sum_i \left[y^i x_j^i - \frac{x_j^i \exp\left(\theta_0 + \sum_j \theta_j x_j^i\right)}{1 + \exp\left(\theta_0 + \sum_j \theta_j x_j^i\right)}\right]$$

$$= \sum_i x_j^i \left[y^i - \frac{\exp\left(\theta_0 + \sum_j \theta_j x_j^i\right)}{1 + \exp\left(\theta_0 + \sum_j \theta_j x_j^i\right)}\right]$$

$$\frac{\partial l(\theta)}{\partial \theta_i} = \sum_i x_j^i \left(y^i - P\left(Y^i = 1 \mid x^i, \theta\right)\right)$$

Here, x_j^i = j^{th} dim. of x^i i^{th} sample.

According to gradient ascent algorithm

$$\theta_0^{(t+1)} \leftarrow \theta_0^{(t)} + \eta \underbrace{\sum_i x_0^i \left[y^i - P(Y^i = 1 \mid x^i, \theta) \right]}_{\frac{\partial l(\theta)}{\partial \theta_j}}$$

For $j = 1 2 \ldots n$ (iterate over weights)

Here for D_1 dataset

$$\theta_j^{(1)} \leftarrow \theta_j^{(t)} + \eta \sum_i x_j^i \left[y^i - \hat{P}(Y^i = 1 \mid x^i, \theta) \right]$$

Similarly, for D_2 dataset

$$\theta_j^{(2)} \leftarrow \theta_j^{(t)} + \eta \sum_i x_j^i \left[y^i - P(Y^i = 1 \mid x^i, \theta) \right]$$

$\Rightarrow$ Coefficients from training on combined data set $D_1 \cup D_2$, Here feature and labels are same.

If D_1 and D_2 are mutually exclusive, then

$D_1 \cup D_2 = D_1 + D_2$ as $(D_1 \cap D_2 = 0)$

$\Rightarrow$ Also, considering the characterstic (property) of convex function, Here D_1 and D_2 are convex set

$\Rightarrow$ Here the min of D_1 and D_2 is θ_j^*

Hence we prove by convex property that

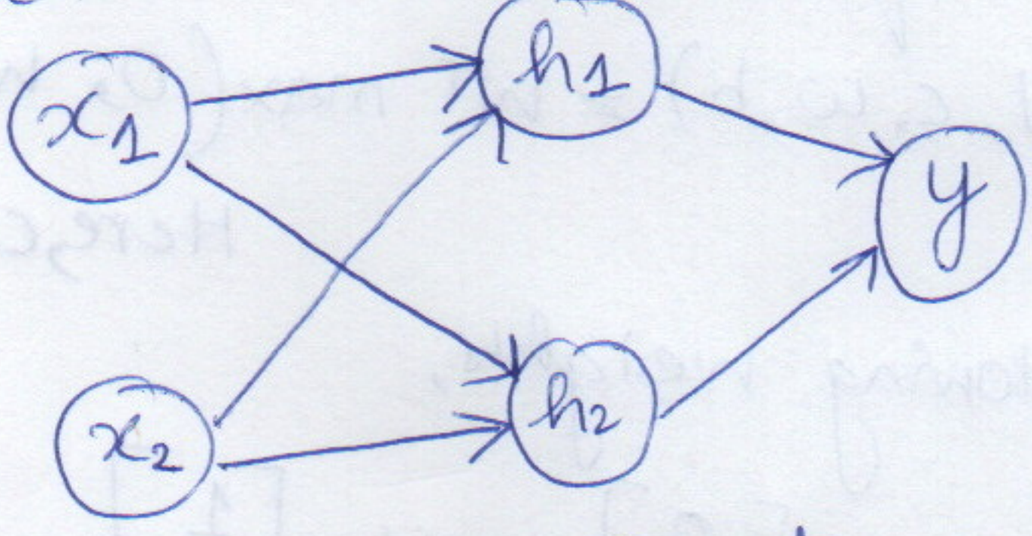
$$\boxed{\min(\theta_j^{(1)}, \theta_j^{(2)}) \leq \theta_j^* \leq \max(\theta_j^{(1)}, \theta_j^{(2)})}$$

Solution 3 XOR Problem

Feedforward network to compute

$$f(x_1, x_2) = \begin{cases} -1 & \text{if } x_1 = 1 \wedge x_2 = 1 \\ 1 & \text{if } x_1 = 1 \wedge x_2 = 0 \\ 1 & \text{if } x_1 = 0 \wedge x_2 = 1 \\ -1 & \text{if } x_1 = 0 \wedge x_2 = 0 \end{cases}$$

Nueral network with a single hidden layer is presented like below.



Here, x represents input
h represents hidden layer
y represents output

To linearly model an input vector (x) to output scalar (y), we commonly use a weight vector (w) and a scalar bias (b) parameter.

In ReLU, to calculate hidden layers (h), we do a linear transformation between our input vector and some learnt parameter (W) we add a bias and then apply a non-linear function to each element of our hidden layer. Hidden layer are expressed as $h = g(W^T x + c)$

activation function

In ReLU, $g(z) = \max(0, z)$

where z denotes the linear transformation from our input to our hidden layer. $Z = W^T x + C$

When we apply ReLU to each element of our linear transformation, we get a non-linear transformation.

The final network equation will be

$$f(x; W, c, w, b) = W^T \max(0, W^T x + C) + b$$

Here, C is a bias vector

Considering the following weights,

$$W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad c = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \quad w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

and $b = 0$, our ouput will be $X = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$

Here X represents the XOR input table

→ we will first find out $Z = W^T x + C$

$$W^T X = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}^T \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

now adding the bias vector C

$$W^T X + C = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

Now to determine h we will apply the ReLU function to every element in the matrix

$$\text{ReLU} \Rightarrow h = g(z) = \max(0, z)$$

where $z = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$

after applying ReLU we get, $h = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$

In order to get the output layer we will multiply the hidden layer (h) with weight vector (w) and will add a bias ($b = 0$)

we get, $w^T h + b$

$$\Rightarrow [1 \quad -2] \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} + 0$$

$$y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Now applying $\text{sign}(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x \leq 0 \end{cases}$ to above result we get $= \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$ as in top and bottom element of matrix were $x = 0$ therefore we place -1.