Machine Learning	**Homework 1**
CSCI 5622 Fall 2017	**Due Time Sep 29, 2017**
Name: Payoj Jain	**CU identitykey: paja3104**

1 Logistic Regression (40pts)

1.1 Programming questions

1.2 Analysis

1.2.1 What is the role of the learning rate (eta) on the efficiency of convergence during training?

Solution. With increase in learning rate the efficiency of convergence first increases and then starts to decrease. With different etas the rate of convergence changes. As learning rate increases the gradient descents faster and almost achieve minima and thus, convergence occurs faster. But after a certain value of learning rate, convergence to the optimised solution is not achieved. Because of the high value of eta, gradient jumps the local minima and thus convergence rate slows down with increase in learning rate.

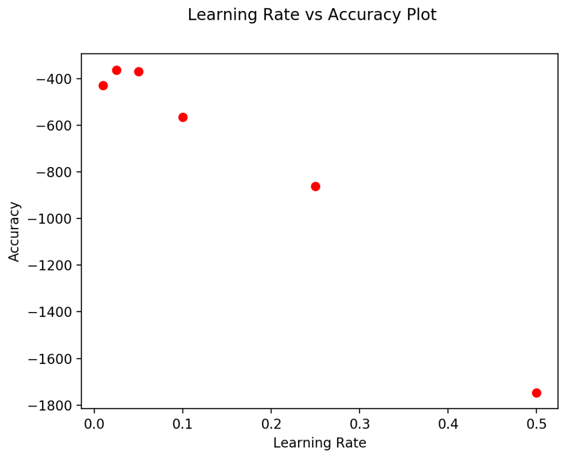


Figure 1: Graph shows that Loss function increases as eta increases till 0.025 and then starts to decrease.

1.2.2 What is the role of the number of epochs on test accuracy?

Solution. As we increase number of epochs the accuracy gradually becomes unchanged.

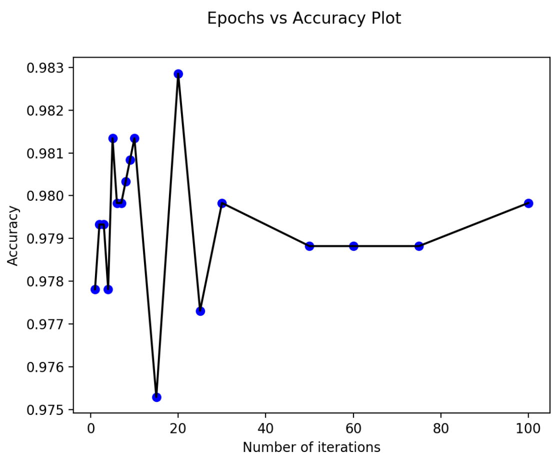


Figure 2: Best accuracy is achieved at epoch = 25. As can be seen that accuracy becomes less fluctuating as number of epochs increases.

2 Feature Engineering (30pts)

2.1 Programming questions

2.2 Analysis

2.2.1 What custom features did you add/try (other than n-grams)? How did those additional features affect the model performance? Why do you think those additional features helped/hurt the model performance?

Solution. I have added following custom features other than n-grams:

- Tfidf score for each word in the data,
- counts of exclamation (”!”),
- counts of (”?”),
- number of sentences,
- presence of frequently used positive and negative connotation words in movie reviews.

Tfidf score of every word signifies how important is that word for a particular document based on its frequency in the document and also how many documents have that word. Hence, adjectives will be given high score in reviews as adjectives are used in every review to describe how a particular is. Thus, helping the model in determining the sentiments of movie reviews.
As I went through the data, I realised that most positive reviews have more than two ”!”. So, just for experiment I added the count of ”!” as a feature to the data and adding this feature increased the accuracy proving that users use ”!” frequently when they are writing a positive review.
Same is the reason for adding count of ”?” in a review.
Users tend to write a long review when they find a movie really good. Thus, number of sentences may help the model determine whether reviews are positive or negative.
To train the model better I added some frequently used words in movie reviews and gave each word some weight based on how strongly that word is conveying the sentiment. Unfortunately,

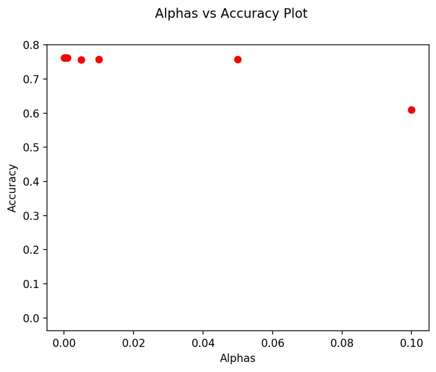
adding this feature reduced the accuracy. Reason could be most of these words can be used interchangeably. The word ”good” can be used as ”not good” in a negative review but my feature will add score towards positive.
The overall accuracy comes out to be around 81%.

2.2.2 What are unigrams, bigrams, and n-grams? When you added those features to the Feature- Union, what happened to the model performance? Why do these features help/hurt?

Solution. An n-gram is a collection of n tokens taken sequentially from a sentence or a document. For example, ”I am eating a banana.” is a sentence in which ”I am”, ”am eating”, ”eating a”, ”a banana”, ”banana .” are 5 bigrams and ”I”, ”am”, ”eating”, ”a”, ”banana”, ”.” are 6 unigrams. Unigrams are taking token one at a time from a sentence or a document and Bigrams are taking tokens two at a time sequentially (as appeared in the sentence).
Model performance increased substantially as I added ngram features to it.
ngram models store the information what kind of words occur together very often. Thus, helping the model to know what kind of words will occur together in positive reviews and what all could occur in negative reviews. For example, bigrams like ”not good”, ”not worth”, ”” tells about the negative connotation of a review and hence the model can classify better after

2.2.3 EXTRA CREDIT (extra 10 pts): Replace the code that fits the SGDClassifier on the training data with scikit-learn?s k-fold cross validation and use it to determine the best regularization constant alpha for the classifier. Why do we need cross validation to tune the alpha parameter? What did you determine was the best value for alpha? What experiments did you run and what were your cross validation results?

Solution. We need to avoid overfitting and we have limited amount of data. Also, we need our model to be trained in all types of possible training set available to us. That’s why we need to implement K-cross validation to tune the alpha parameter. Since, for a certain training set our model could overfit and give low accuracy but for remaining training set our model could perform better.
The best alpha is 0.0001 at which the . After adding features, I implemented K-cross validation method to tune the model. I implemented 5-cross validation i.e. split the given data into 5 equal sets and took 1 set as test and remaining set as training. I repeated this for each set. I took a set of alphas [0.0001,0.0005,0.001,0.005,0.01,0.05,0.1] and ran the code for 7000 epochs. The result is average accuracy for 5-cross validation for each alpha.



3 Gradient Descent Learning Rule for Multi-class Logistic Regression (20 pts)

3.1 Derive the negative log likelihood for multi-class logistic regression.

Solution.

$$p(y=c|x) = \frac{exp(\beta_c^T x)}{\sum_{c'=1}^{C} exp(\beta_{c'}^T x)}$$

The above probability is for a particular y. To calculate the likelihood we have to sum the probabilities of all ys.

$$l(\theta) = \sum_{i=1}^{N} [\frac{exp(\beta_{c,i}^T x_i)}{\sum_{c'=1}^{C} exp(\beta_{c'}^T x_i)}]$$

Taking log both sides,

$$log(p(y=c|x)) = \sum_{i=1}^{N} [log(exp(\beta_{c,i}^T x_i)/\sum_{c'=1}^{C} exp(\beta_{c'}^T x_i))]$$

$$= \sum_{i=1}^{N} [\beta_{c,i}^T x_i) - log(\sum_{c'=1}^{C} exp(\beta_{c'}^T x_i))]$$

Taking negative of this, so negative log likelihood is :

$$Lossfunction = -\sum_{i=1}^{N} [\beta_{c,i}^T x_i) - log(\sum_{c'=1}^{C} exp(\beta_{c'}^T x_i))]$$

3.2 The gradient descent learning rule for optimizing weight vectors generalizes to the following form: $\beta_j^{t+1} = \beta_j^t ? \eta \Delta \beta_j^t$ where η is the learning rate. Find the $\Delta\beta_{c,j}$ (the parameter for feature x_j in class c) for a multi-class logistic regression model.

Solution. *From the above we get the negative log likelihood for multi-class Logistic Regression.*

$$Lossfunction = -\sum_{i=1}^{N} [\beta_{c,i}^T x_i) - log(\sum_{c'=1}^{C} exp(\beta_{c'}^T x_i))]$$

Differentiating this w.r.t $\beta_{c,j}$,

$$\frac{\partial(-log(l(\theta)))}{\partial\beta_{c,j}} = -\sum_{i=1}^{N} [\partial\beta_{c,i}^T x_i) - \partial log(\sum_{c'=1}^{C} exp(\beta_{c'}^T x_i))]$$

$$= -\sum_{i=1}^{N} [1\{c_i = c_j\}] x_i^j + \sum_{i=1}^{N} [\frac{\sum_{c'=1}^{C} exp(\beta_{c'}^T x_i)[c_i = c_j] x_i^j}{\sum_{c'=1}^{C} exp(\beta_{c'}^T x_i)}]$$

$$= -\sum_{i=1}^{N} [1\{c_i = c_j\}] x_i^j + \sum_{i=1}^{N} [\frac{exp(\beta_{c'}^T x_i)}{\sum_{c'=1}^{C} exp(\beta_{c'}^T x_i)} 1\{c_i = c_j\} x_i^j]$$

$$= -\sum_{i=1}^{N} [1\{c_i = c_j\}] x_i^j + \sum_{i=1}^{N} [p(y=c|x_i) x 1\{c_i = c_j\} x_i^j]$$