Amir Kashipazha
CSCI 5622 HW 2 Analysis

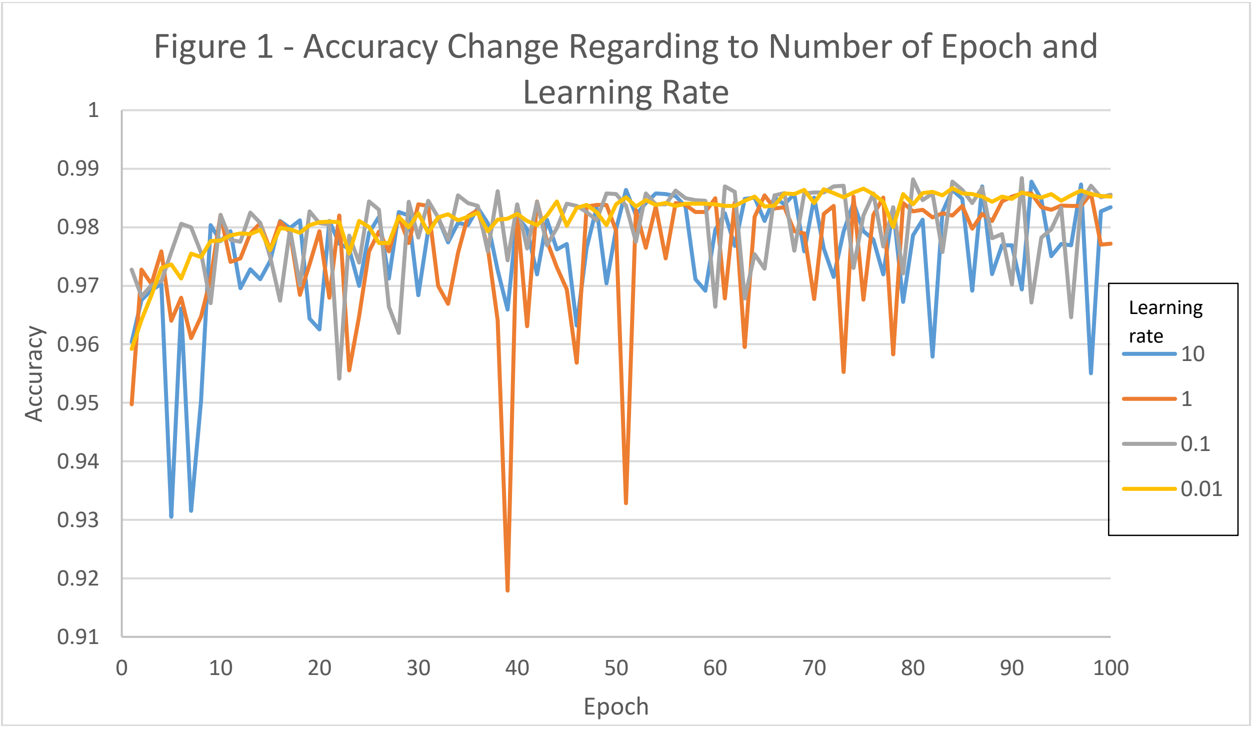
1.2 Analysis

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

For logistic regression, we need to create a mode and estimate parameters of the model regarding the given training dataset. Gradient Descent estimates this parameter by minimizing the model error. The learning rate is a user defined value that determines how fast or slow the model comes close to the optimal solution. So by using a larger learning rate we are able to find an optimal solution faster than a smaller learning rate. By using a very large learning rate we may accidentally skip the optimal solution, but by using a very small learning rate we would need to do many iterations to find the optimal solution. Therefore, selecting a suitable learning rate is important to reach an optimal solution. As we can see in Figure 1, by using large learning rate such as 10 or 1, we can get reliability but the accuracy decreases suddenly in some epochs. On the other hand, this does not happen for a small learning rate like 0.01. Accuracy starts to increase and then comes to a constant value or with minimal fluctuation.

2. What is the role of the number of epochs on test accuracy?

The large number of epochs caused higher accuracy for all learning rates as we can see in Figure 1. Regardless of the accuracy fluctuation, the accuracy of this mode is increased from epoch 0 to epoch 20 then becomes constant. I recommend using a learning rate of 0.01 with an epoch of 20 for this model.



2.2 Analysis

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?

At the first step, I added features one by one which were unigrams, bigrams, and trigrams. These three features together increased accuracy to 99% and 79% for training and testing data sets, respectively. Figure 2 shows these changes. Then I added all capital words or first capital words, negative, and positive words respectively. As Figure 2 shows, adding new features did not change the accuracy for either the training or testing data sets. This make sense for the training data set since we cannot have an accuracy of more than 100%. I believe the three added features did not increase accuracy for several reasons. One reason could be that the given negative or positive words are not in the dataset. Or numbers of the testing data is not enough, since we did not have this problem for the testing data.

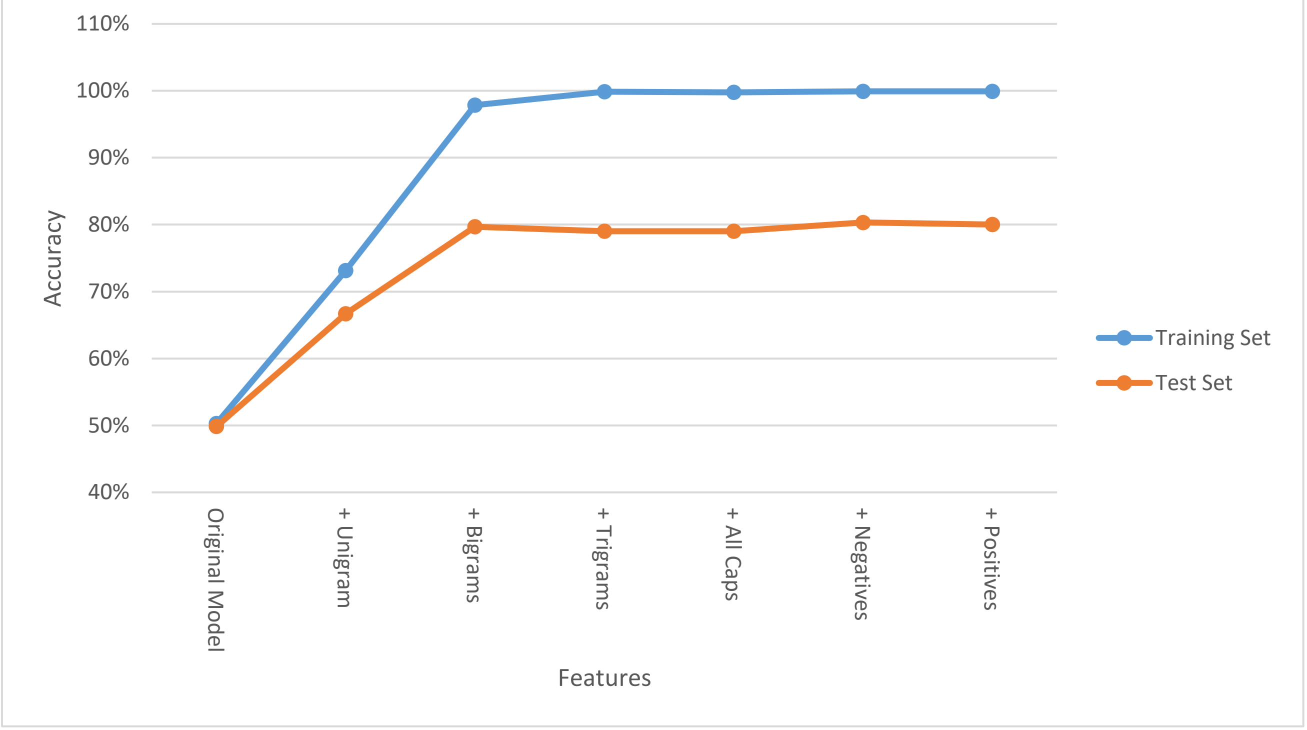
In order to find which feature provides the best classification, I used each of them individually. The result can be found in Figure 3, which shows that the trigram has the highest accuracy for training dataset but low accuracy for training data set. I believe the bigram did the best job for both training and testing data and then the all caps data set did the best. Negative and positive did increase the accuracy even when I combined them. I combined the bigram and all caps in one model, but interestingly, the accuracy did not improve. I believe adding features could increase accuracy as we can see in Figure 2. But the accuracy augmentation was not considerable from adding all caps. So there is no reason to add a feature when it does not lead to a higher accuracy. It is crucial to use a combination of feature since, as Figure 3 shows, using features that have the highest accuracy reduces it considerably. Therefore, using a combination of features is a key for getting the highest accuracy.

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?

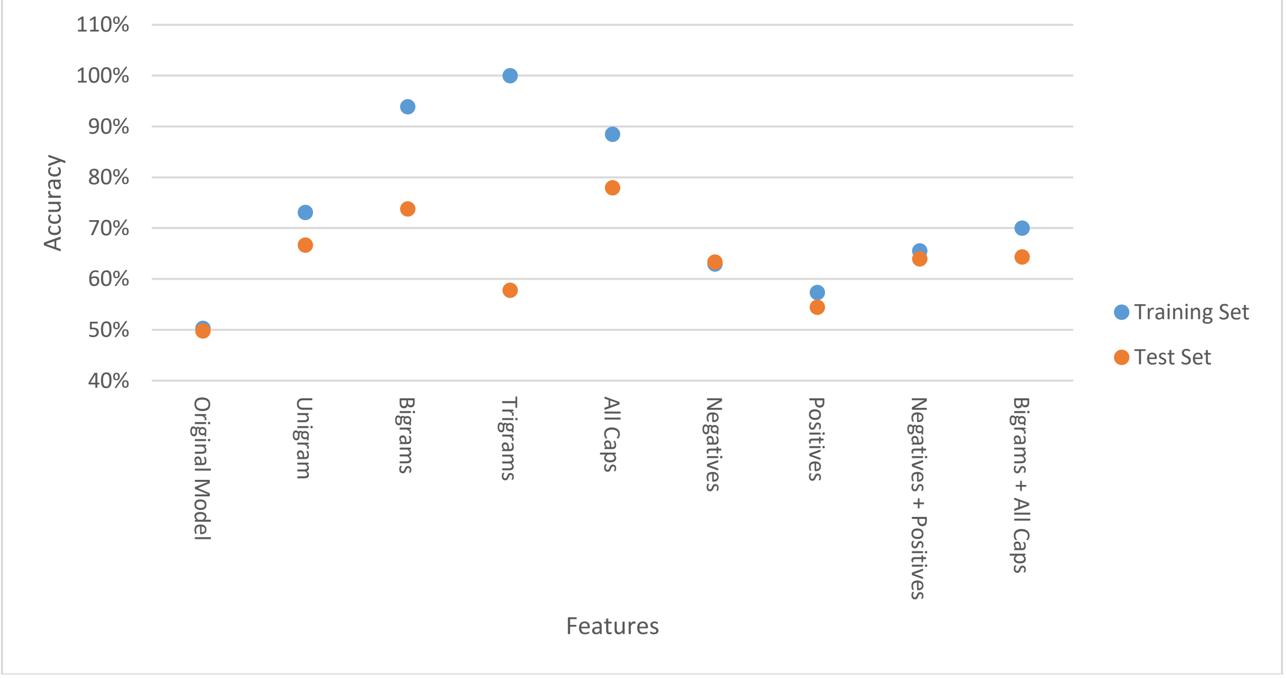
Unigrams are counts of each individual words of a document. Bigrams are counts of two consecutive words in a document. N-grams are counts of n consecutive words in document.

When I added n-grams features to the model, the accuracy and running time increased. N-grams contain orders of words which is an important feature for classification. As we can see in Figure 3, bigrams and trigram did a better job than unigram since they know more about the order of words than unigram. Except trigram did not do a good job for test data set, which could be because of length of it. So adding these features alone improved the accuracy considerably without adding new features.

Figure 2 - Accuracy Change Regarding to Fetures



Figue 3 - Accuracy of Features



3 Gradient Descent Learning Rule for Multi-class Logistic Regression

$$P(y = C|x) = \frac{exp(\beta_C^T x)}{\sum_{C'}^{C} \exp(\beta_{C'}^T x)}$$

$$\Pi_1^n\left(\frac{exp(\beta_C^T x)}{\sum_{C'}^{C} \exp(\beta_{C'}^T x)}\right)$$

$$-log(\Pi_1^n(\frac{exp(\beta_{Cn}^T x_n)}{\sum_{C'}^{C} \exp(\beta_{C'}^T\, x_n)})) =$$

$$\sum_{n=1}^{n}(\ -\log(\ \exp(\beta_{Cn}^T x_n)\ ) + \log\left(\sum_{C'}^{C} exp\left(\beta_{C'}^T x_n\right)\right)\ ) =$$

$$\boldsymbol{\sum_{i=1}^{n}(\ -\ \beta_C^T x_i\ + \log\left(\sum_{C'}^{C} exp\left(\beta_{C'}^T x_i\right)\right)\ )}$$

Now we need to get partial derivative for each term of the last equation:

First term:

$$- \frac{\partial}{\partial\beta_{c'}^{j}} \beta_C^T x_i = \boldsymbol{-x_{i,j}}$$

Second term:

$$- \frac{\partial}{\partial\beta_{c'}^{j}} \log\left(\sum_{C'}^{C} exp\left(\beta_{C',j}^T x_{i,j}\right)\right) =$$

$$\frac{x_i exp\left(\beta_{C',j}^T x_{i,j}\right)}{\sum_{C'}^{C} exp\left(\beta_{c',j}^T x_{i,j}\right)}$$

We will have:

$$\sum_{i=1}^{n}\left(\frac{x_{i,j} exp\left(\beta_{C'}^T\, x_{i,j}\right) I(y=c')}{\sum_{C'}^{C} exp\left(\beta_{c',j}^T x_{i,j}\right)} - x_{i,j}\ I(y = c')\ \right)$$