COMP135 Machine Learning Project02

Yunke Zhu

1330327

Part One: Classifying Review Sentiment with Bag-of-Words Features

1. Construct Bag of Words

General Idea

According to the requirements of the outline, there are basically three ways to construct bag of words, which are:
A) **Traditional Approach:** This is the most simple way that is to record the frequency of each word in the dictionary.

B) **Exclude Common Words and Rare Words**: This approach is based on plan A, but ignore some common words such as "the", "and", "a", "an"..., Meanwhile will also ignore some words in low frequency.

C) **TF-IDF** Use "term frequency–inverse document frequency" policy as discussed in class to construct the words bag.

Methodology:

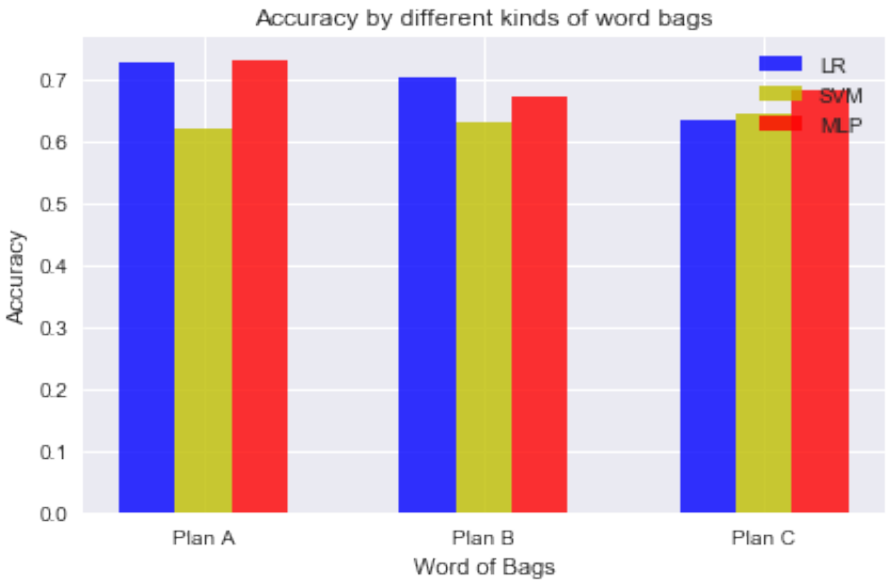
As for the first approach, I firstly read data from local and do some pre-processing, such as split them into single word, ignore lower/upper case, ignore useless symbols or numbers. After generated a clean text list, I can take the advantage of library `sklearn.feature_extraction.text.CountVectorizer` and run command `CountVectorizer.fit_transform()` After doing so, we can get a sparse matrix with the degree equal to the number of required words that need to be counted.

The second approach is similar to the first one, but doing more steps before applying the list into CountVectorizer. What I did in this section is to use stop words in libaray `nltk.corpus.stopwords`, this can help us ignore most common words, and I also ignore words with the frequency less than 5.

TF-IDF is a little bit different, for this approach we can directly take the advantage of libarary `sklearn.feature_extraction.text.TfidfVectorizer` instead of CountVectiruzer.

After construct three different bag of words, I will test them by applying them into logistic regression/SVM/MLP models, so let's see how the result differs.

Result:



Summary:

I tested the accuracy by using 3 different models, which are logistic regression in blue color, SVM in yellow color and standard MLP in red color. By computing the average accuracy, it is weird that the data without any processing gets the highest accuracy and tf-idf has the lowest accuracy. But on gradescope, tfidf has the highest accuracy among all the plans, so I decide to use **tf-idf** as my pre-processing.

2. Logistic Regression

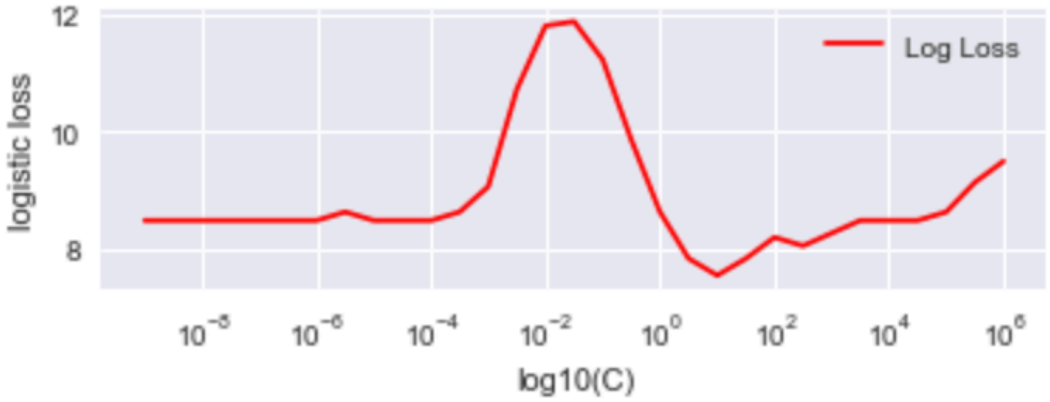
Model

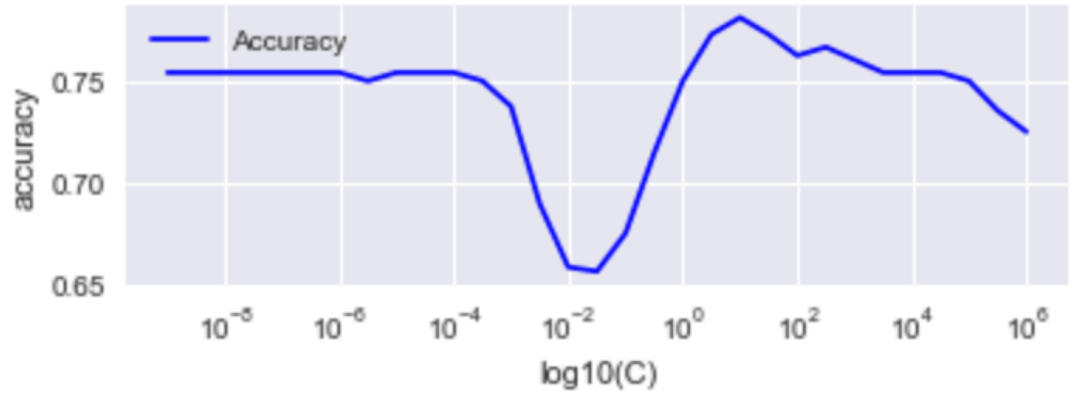
The first model is logistic regression model, which is implemented from sklearn libarary.

Parameter

The parameter of a logistic regression that I want to focus on are: **C**(penalty)

C(Penalty) Result






```
Best C-value for LR with 784-feature data: 10.0000000
Log-loss at best C-value: 7.5554
The accuracy under the best C value: 0.7812
```

The C value I select range from [10^-9, 10^6], and the best C value we get is 10; under such penalty value, we can gain the maximum accuracy as 78% and log loss reaches 7.56. What's more, we notice that the C value performs poorly when it approches to 0.01. And as C grows up, our accuracy will keep to drop based on the reason that a large penaly value will surely affect a standard model, and the accuracy may reaches 50% as it keep growing up, which means may have a half and half prediction.

3.Multilayer Perceptron(MLP)

Model

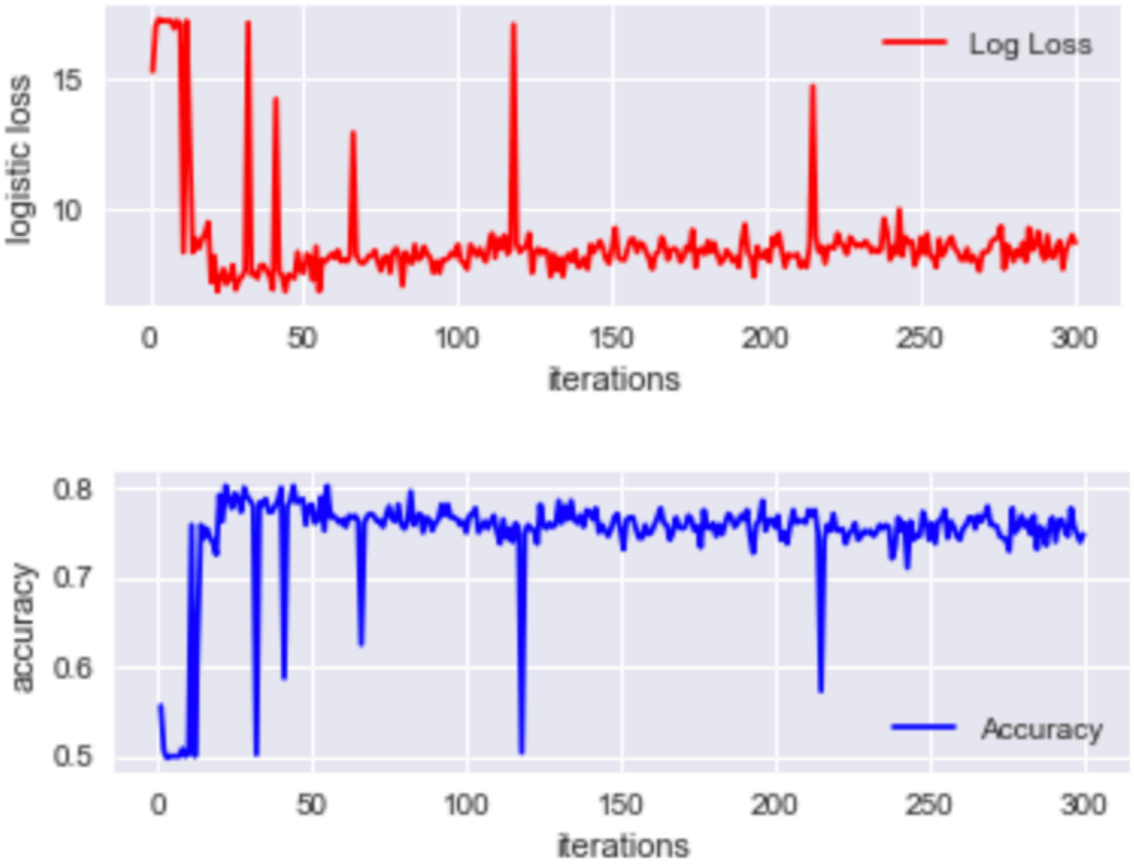
The model I implement the MLP model is built from assignment four with sklearn libarary.

Parameter

Since our data only contains about 2400 data, so adding more hidden layers or adjust the size of hidden layer

won't help a lot, in this situation, the parameter of a MLP model I would choose is **max_iter**, and the range I would test is between[1,300]

Max_iter Result



This is the result of both logloss and accuracy by prediction based on different iterations, as we can see from the two images, it seems that there are several peaks in the log loss result, but as the times of iteration goes by, it becomes much more flat than less iterations, but still there occurs a peak in about 220 iterations. From my guessing, the reason this happens may result from they are not converged. But if we have more iterations, it will have a higher posibility to converge.

4.Support Vector Classification(SVM)

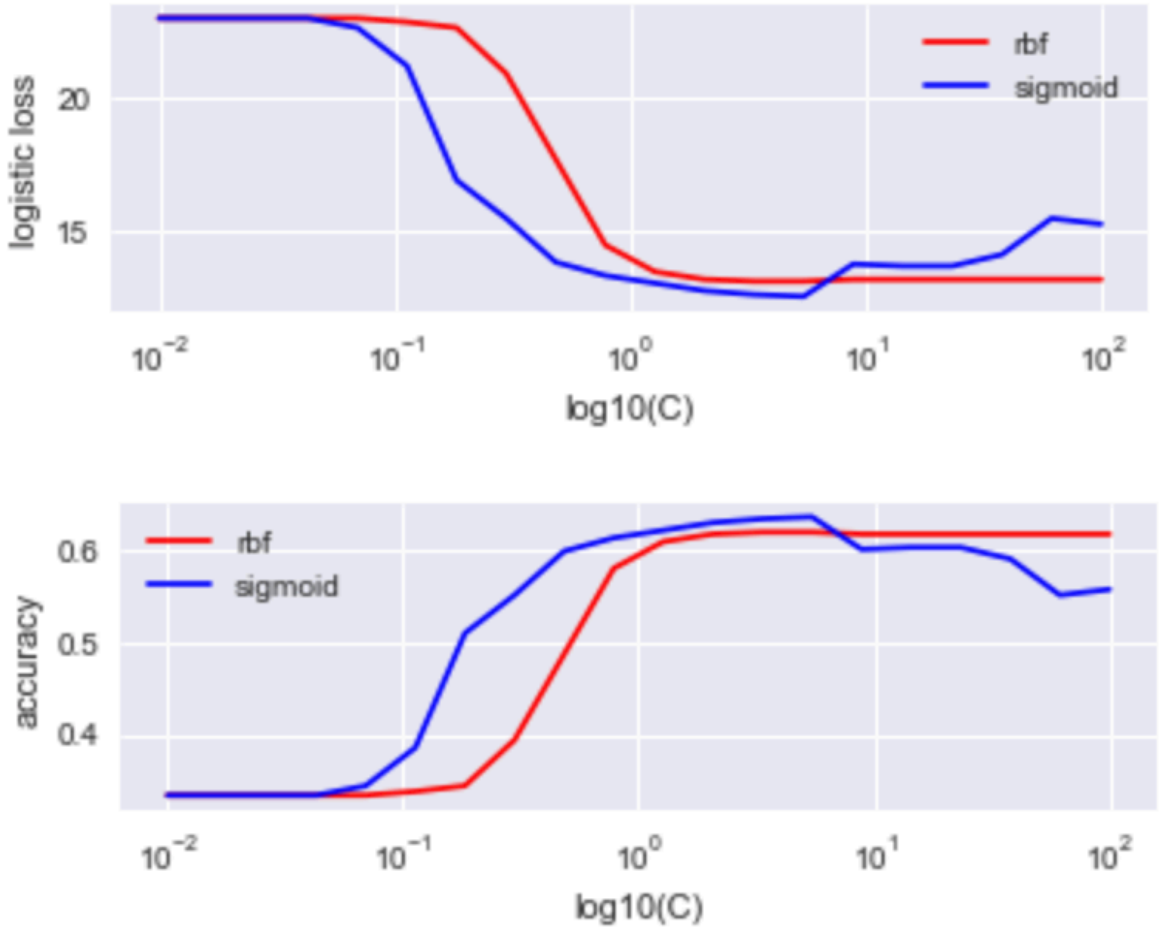
Model

The thrid model I choose is using SVM, basically I tested all the other possible options that I have learned with default parameters, and SVM gets the best accuracy, so it is encouraged to get a higher prediction accuracy in our testing dataset.

Parameter

For this model, I would choose two parameters, which are C(penalty) and kernel. And I would compare kernel with default rbf and sigmoid in penalty value ranges from [0.001, 100]

Kernel (Sigmoid vs RBF) Result



From the figure, we notice that the accuracy about this two kernels, sigmoid has a higher accuracy and lower log loss compared a normal rbf kernel. Meanwhile it is easy to tell that the best C value is about 1. So in the folliwing testing, I would choose SVM with C=1 as the best SVM model.

5.Comparison Between Three Classifiers

a) Logloss:

1. **SVM** has the lest log loss under the best kernel using sigmoid and C value as 1, the log loss is **13.17**.
2. **Logistic regression** has the second log loss under the best C value as 10, and its log loss is **13.46**.
3. **MLP**'s log loss is highest which is **15.83**, but this may get improved if we adjust the hidden layer.

b) Accuracy:

1. **SVM** has the highsest accuracy with **0.619**
2. **Logistic regression** gets rank two with the accuracy as **0.610**
3. **MLP** is the poorest, and its accuracy is only **0.542**

c) Performance:

Using `start_time_sec = time.time()` to test the time costs for three models. And we can get the result as follows:

1. **Logistic regression** has the best performance in time , only costs 0.008 seconds running a single model.
2. **SVM** takes 0.22215 seconds running the model using sigmoid as its kernel, which is 27 times slower than logistic regression.
3. **MLP** is similar to SVM, both of them are time consuming, it takes 0.305 seconds to run a single model, but this may result from the parameter ot max_iter.

d) Conclusion

With the evaluation on three aspects of these models, I would like to choose **SVM** as my prediction model. The main reason is simple because it has a higher performance and low loss. But you may ask about the time consuming problem, for this case, which we have a small datasets, using this model won't cause a big impact on detecting the final result. But if we have a much larger dataset, let's say over one million data, then we need to re-evaluate these models and make comparison again. But for this project, using an SVM model is my ideal option.

6. LeaderBoard Prediction

I used three models mentioned above to make predictions on the Testing datasets, and both logistic regression and MLP gives me an error rate about 0.17 and the SVM model performs best which gains 0.16 error rate.

From the results I notice that 2 points:

1. MLP is not that bad applying in testing dataset, the main reason may due to the size of our dataset grows. The basic training dataset I used is about 2000 sentences, and 400 sentences needs to be settled as cross validation dataset. When this turns out to test on the testing dataset, the more data we get, the more well-performed model we may have. So the result is much better, not only to MLP but logistic regression and SVM as well.

2. SVM and MLP performs very well compared to logistic regression with larger dataset while the change of data size does not have the same impact on logistic regression. So my guess is that as our dataset grows, the accuracy of both MLP and SVM will grows rapidly compared to logistic regression. As for MLP, we can construct more type of hidden layers to get a better performance perhaps.

Based on my observation, I donwload some extra data from kraggle dataset, and apply more new data to my models. the amount I add is 1000,5000,10000 new movie review with positive/negative results. And this turns out to be that by adding 1000 new data, I have the best performance using SVM with the accuracy as **0.14667**, as for the adding additonal 5000 or 10000 data, there will occur an overfitting problem, which means that my model is not able to handle so much feature. The future research may focused on adjusting the features of these models to cover this overfitting problem.

Part Two: Classifying Review Sentiment with Word Embeddings

1. Construct Bag of Words

General Idea

According to the requirements of the outline, we still can take the advantage of the former strategy(which also makes easier to construct and compare them), but instead using the default word vector from sklearn.vector, we are using the word embedding from `load_word_embeddings.py`, and the default dictionary is loaded from "'glove.6B.50d.txt"', same as last part, we will compare with two approaches which are:
A) **Traditional Approach:** This is the most simple way that is to record the frequency of each word in the dictionary.

B) **Exclude Common Words and Rare Words**: This approach is based on plan A, but ignore some common words such as "the", "and", "a", "an"..., Meanwhile will also ignore some words in low frequency.

Methodology:

As for the first approach, I firstly load data using part of `load_word_embeddings.py` from local, and do some pre-processing, such as split them into single word, ignore lower/upper case, ignore useless symbols or numbers. After generated a clean text list, Unlike previous processing using sklearn's vector, we will take the advantage of `analogy_lookup` to get the selected word's value.

The second approach is similar to the first one, but doing more steps before applying the list into CountVectorizer. What I did in this section is to use stop words in libaray `nltk.corpus.stopwords`, this can help us ignore most common words, and I also ignore words with the frequency less than 5.

After construct three different bag of words, I will test them by applying them into logistic regression/SVM/MLP models, so let's see how the result differs.

Result:

Summary:

I tested the accuracy by using 3 different models, which are logistic regression in blue color, SVM in yellow color and standard MLP in red color. By computing the average accuracy, it seems that using a traditional method and using a modified data are quite similar, but on gradescope, using plan B which means select some words that have a stable/normal frequency has a slightly lower error rate, so I will use plan B in the following research.

2. Logistic Regression

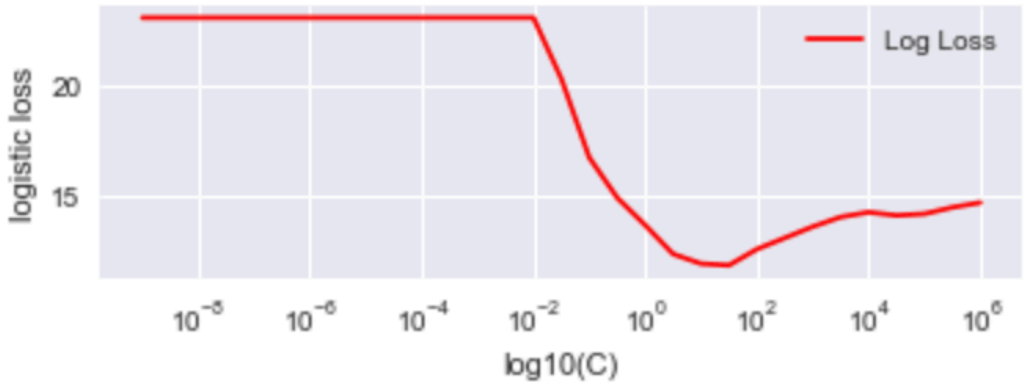
Model

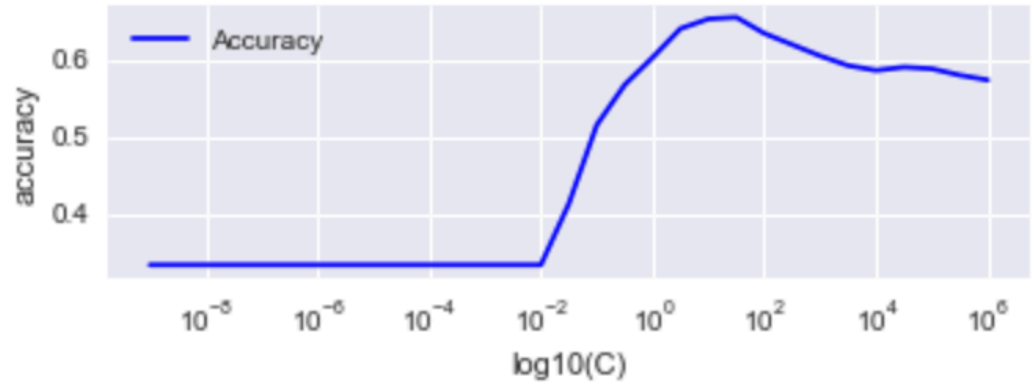
The first model is logistic regression model, which is implemented from sklearn libarary.

Parameter

The parameter of a logistic regression that I want to focus on is: **C**(penalty)

C(Penalty) Result






```
Best C-value for LR with 784-feature data: 31.6227766
Log-loss at best C-value: 11.9447
The accuracy under the best C value: 0.6542
```

The C value I select range from [10^-9, 10^6], and the best C value we get is 31.62; under such penalty value, we can gain the maximum accuracy as 78% and log loss reaches 11.9447. What's more, we notice that the C value performs poorly when it approches to 0.01. And as C grows up, our accuracy will keep to drop based on the reason that a large penaly value will surely affect a standard model, and the accuracy may reaches 50% as it keep growing up, which means may have a half and half prediction.

3.Multilayer Perceptron(MLP)

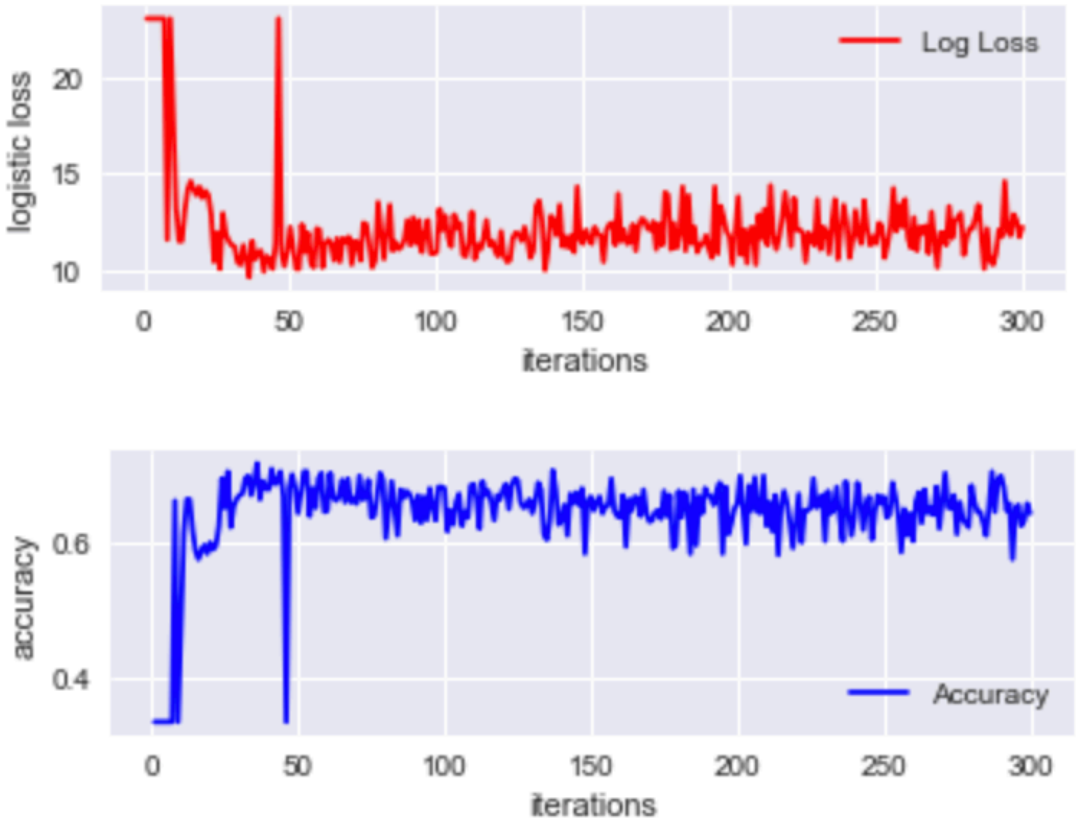
Model

The model I implement the MLP model is built from assignment four with sklearn libarary.

Parameter

Since our data only contains about 2400 data, so adding more hidden layers or adjust the size of hidden layer won't help a lot, in this situation, the parameter of a MLP model I would choose is **max_iter**, and the range I would test is between[1,300]

Max_iter Result



This is the result of both logloss and accuracy by prediction based on different iterations, as we can see from the two images, unlike from part one, this time the iterations seems coveraged after 50 times of iterations, so this time I would like to build the MLP model with 50 max_iter, which means this time we can have a better performed MLP model with less time cost.

4.Support Vector Classification(SVM)

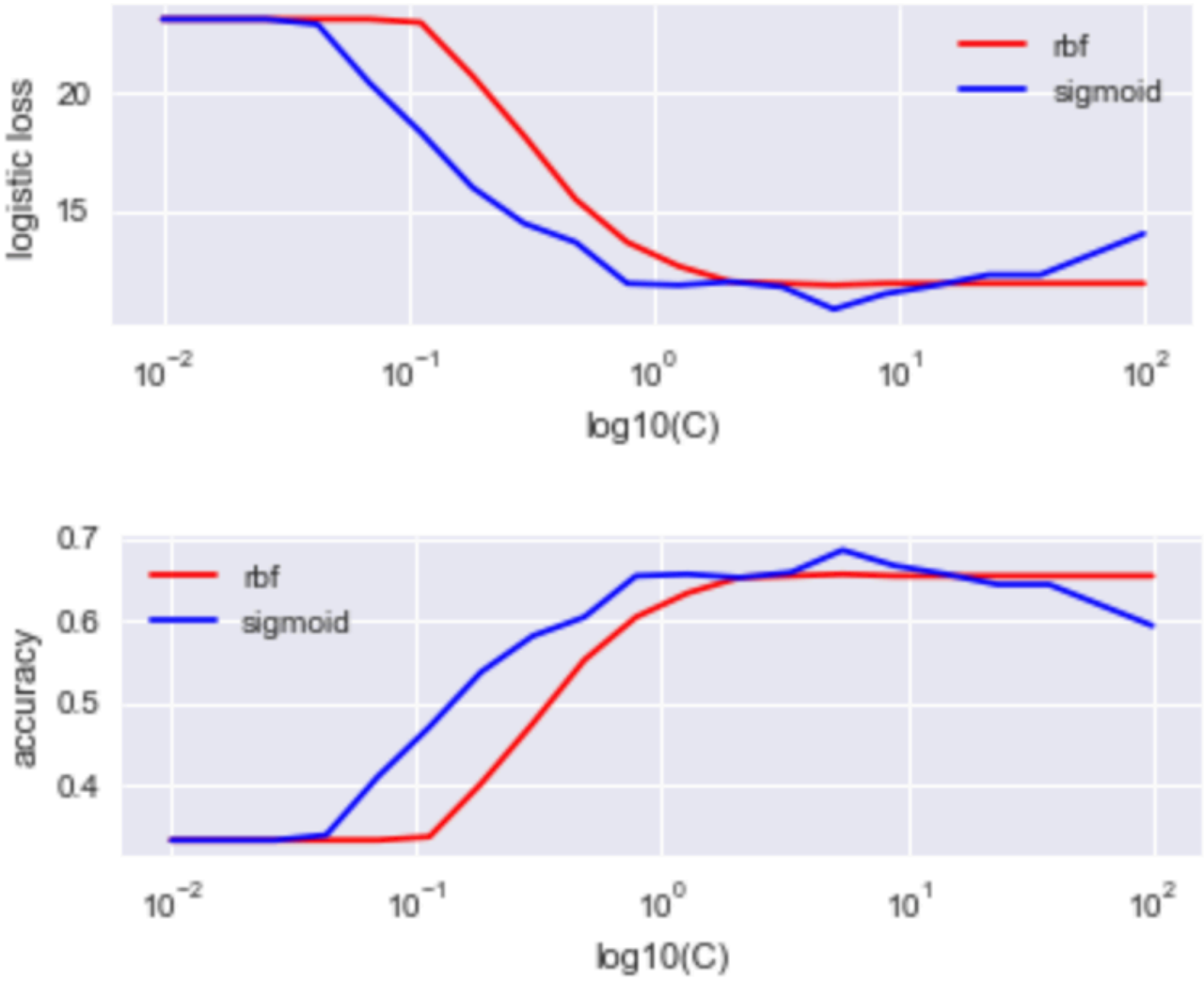
Model

The thrid model I choose is using SVM, basically I tested all the other possible options that I have learned with default parameters, and SVM gets the best accuracy, so it is encouraged to get a higher prediction accuracy in our testing dataset.

Parameter

For this model, I would choose two parameters, which are C(penalty) and kernel. And I would compare kernel with default rbf and sigmoid in penalty value ranges from [0.001, 100]

Kernel (Sigmoid vs RBF) Result



From the figure, we notice that the accuracy about this two kernels, sigmoid has a higher accuracy and lower log loss compared a normal rbf kernel. But unlike part one, this time the difference on advantage of using sigmoid is not that much. But still, I would still use sigmoid as our kernel. And it is easy to compute the best C value is about about 8 this time. So in the folliwing testing, I would choose SVM with C=8 as the best SVM model using sigmoid.

5.Comparison Between Three Classifiers

a) Logloss:

1. **SVM** has the lest log loss under the best kernel using sigmoid and C value as 1, the log loss is **11.37**.
2. **MLP**'s log loss is better in part one which is **11.44** this time, which is slightly lower than SVM.
3. **Logistic regression** has the highest log loss under its best C value as 31.667, and its log loss is **11.94**. But their log log this time doesn't differs a lot.

b) Accuracy:

1. **SVM** has the highsest accuracy with **0.671**
2. **MLP** gets rank two with the accuracy as **0.669**
3. **Logistic regression** is the poorest, and its accuracy is only **0.654**

c) Performance:

Using `start_time_sec = time.time()` to test the time costs for three models. And we can get the result as follows:

1. **Logistic regression** has the best performance in time , only costs 0.005 seconds running a single model.
2. **MLP** takes 0.05 seconds running the model using sigmoid as its kernel, which is 10 times slower than logistic regression. But this time its running performance gets greatly improved by shorten its max_iter.
3. **SVM** is poorest this time, it takes 0.251 seconds to run a single model, slightly better than the time in part one, but this time MLP seems more reliable.

d) Conclusion

With the evaluation on three aspects of these models, I would like to choose **Logistic Regression** as my prediction model in large dataset and use **MLP** when dealing with small dataset. The reason that I did not choose SVM this time is based on the Performance reason, unlike what I've done in part one, this time using SVM or MLP does not seems to have a worse accuracy, what's more, they are both good at getting the prediction in less time. And SVM still costs lots of time. Although SVM has a better accuracy and less logloss, but this slight advantage won't help a lot when dealing with large dataset. Another advantage of using an MLP is that there are more parameters that I can test unlike SVM, the MLP has a better potential in prediction especially adjustment of hidden layers. Since our test data is small, I would like to use MLP to do the prediction

6. LeaderBoard Prediction

I used three models mentioned above to make predictions on the Testing datasets, and both logistic regression

and MLP gives me an error rate about 0.17 and the SVM model performs best which gains 0.16 error rate (the error rate are almost the same as what I've done in part one).

From the results I notice that 2 points:

1. MLP is good this time applying in testing dataset, the main reason may due to the size of our dataset grows and the different strategy on data pre-processing. The basic training dataset I used is about 2000 sentences, and 400 sentences needs to be settled as cross validation dataset. When this turns out to test on the testing dataset, the more data we get, the more well-performed model we may have. So the result is much better, not only to MLP but logistic regression and SVM as well.
2. SVM and MLP performs very well compared to logistic regression with larger dataset while the change of data size does not have the same impact on logistic regression. So my guess is that as our dataset grows, the accuracy of both MLP and SVM will grows rapidly compared to logistic regression. As for MLP, we can construct more type of hidden layers to get a better performance perhaps.

Based on my observation, I donwload some extra data from kraggle dataset, and apply more new data to my models. the amount I add is 1000,5000,10000 new movie review with positive/negative results. And this turns out to be that by adding 1000 new data, I have the best performance using SVM, as for the adding additonal 5000 or 10000 data, there will occur an overfitting problem, which means that my model is not able to handle so much feature. The future research may focused on adjusting the features of these models to cover this overfitting problem.