COMP551 MiniProject 2: Optimization and Text Classification

Nianzhen Gu[261044523], Isabella Hao[260843097], Yiyuan Shang[260833831]

October 26, 2021

ABSTRACT

In this project, we implemented the two different programs of logistic regression to obtain a better understanding of gradient-dependent functions, as well as consolidate our cross-validation knowledge for future machine learning inquiries. In the larger and more complicated part of the project, we trained our untouched diabetes dataset with given logistic regression implementations including cost and gradient functions to attain the best relative learning rate and best number of iterations, from there we implemented a mini-batch stochastic gradient descent function with momentum. With the successfully built system, we tested the influence of different batch sizes and momentum coefficients on the performance of the program. In the test classification part, with the help of logistic functions, we wrote a program to distinguish the computer-generated fake news and human-written real news from the given datasets.

INTRODUCTION

Two programs were implemented in the project, which are both based on the binary classification performed by logistic regression. The program in the optimization part of the project showed that the mini-batch stochastic gradient descent was more accurate and had a higher convergence speed compared to the regular gradient descent model. When introduced momentum, the superiority is mostly shown on the batches that are larger in size. The program for the text classification part of the project was trained to successfully label 74.87% of the news.

DATASETS

Both parts of this project did not require any modification or processing of the data, both datasets were sorted into testing, training and validation sets when received. The only mandatory processing was to separate the data of each set of data into the inputs and outputs, or x's and y's. Upon the agreement made among group members, we decided to directly use the x's and y's of the diabetes datasets for part 1. However, to achieve higher accuracy we cleaned the data for part 2.

The diabetes datasets consisted of eight health indicators and one outcome. The health indicators were represented as decimal numbers meanwhile the outcome was binary.

The fake news dataset contains the texts and labels. Texts are the news that is written by humans or computers. The labels are either 0 or 1 that represent the text written by computer (fake news) and human (real news) respectively. To obtain better results, we started with the data cleaning process. All the capital letters were changed to lowercase, to reduce the complexity of the model trying. Specifically, to count the frequency of words used. Then we remove all the punctuation because we don't think it can benefit the model. After the data cleaning, we visualize the data in the training dataset. For the data information, please see the Appendix.

RESULT

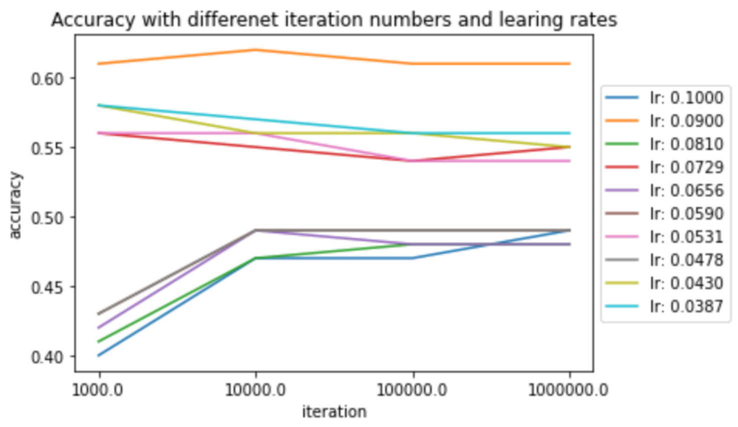
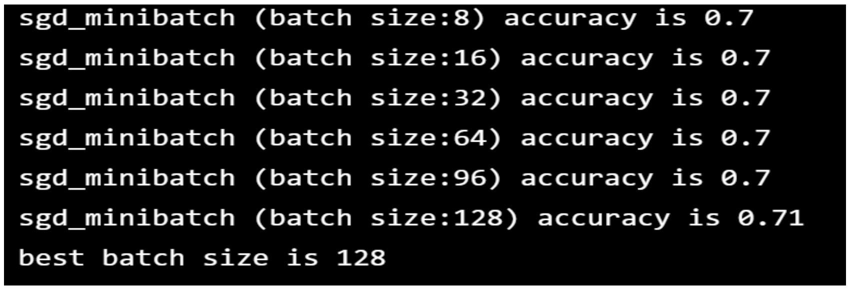


Figure 1: Accuracy of different learning rates vs iterations.

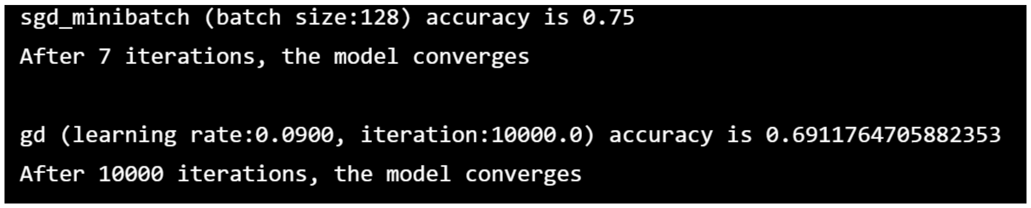
The first goal of the optimization part of the project was to find a pair of learning rate and iteration that will allow the logistic regression model to produce a fully converged solution. We wrote a function called best_lr_iter(learning_rate, iteration) to help us find a relatively better pair. The function returns the pair of learning rate and iteration that has the highest accuracy, along with the accuracy they obtained, as well as three arrays containing the lists of iterations, learning rates and the accuracies produced by the function. The output of best_lr_iter(0.1,1000) is shown in Figure 1, as the orange line shown in the graph, the highest accuracy, 0.62, was produced when learning rate = 0.09 and iteration =1e4.



```
sgd_minibatch (batch size:8) accuracy is 0.7
sgd_minibatch (batch size:16) accuracy is 0.7
sgd_minibatch (batch size:32) accuracy is 0.7
sgd_minibatch (batch size:64) accuracy is 0.7
sgd_minibatch (batch size:96) accuracy is 0.7
sgd_minibatch (batch size:128) accuracy is 0.71
best batch size is 128
```

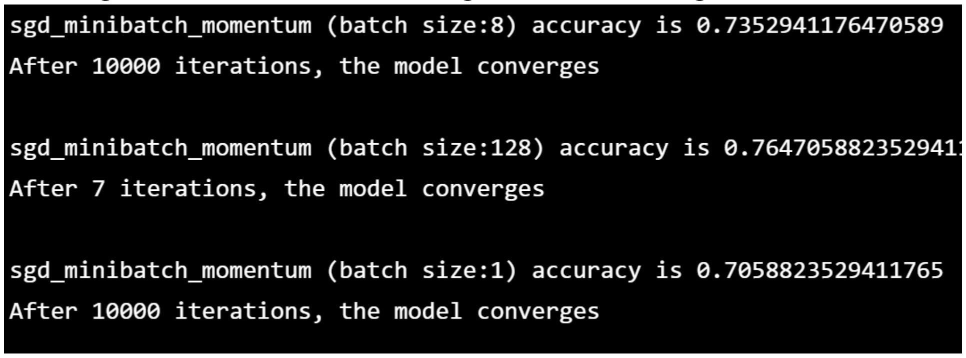
Figure2: Mini-batch stochastic gradient descent result printed.

With the 0.09 learning rate and 1e4 iteration, minibatch stochastic gradient descent was run along with batch sizes of 8, 16, 32, 64, 96, and 128. The function was run a few times since the data batch was randomly selected. However, with the constantly changing data input, the accuracy remained stable. As shown in Figure 2, the accuracy produced by different batch sizes was at most different in the second decimal place. In this instance, the best configuration was batch size = 128.



```
sgd_minibatch (batch size:128) accuracy is 0.75
After 7 iterations, the model converges

gd (learning rate:0.0900, iteration:10000.0) accuracy is 0.6911764705882353
After 10000 iterations, the model converges
```

Figure 3.1: Minibatch stochastic gradient descent vs gradient descent.

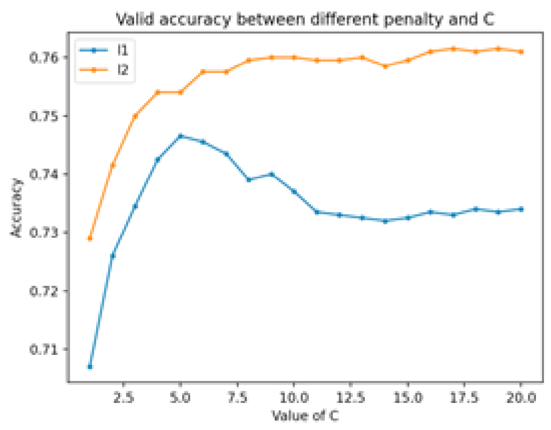


```
sgd_minibatch_momentum (batch size:8) accuracy is 0.7352941176470589
After 10000 iterations, the model converges

sgd_minibatch_momentum (batch size:128) accuracy is 0.764705882352941
After 7 iterations, the model converges

sgd_minibatch_momentum (batch size:1) accuracy is 0.7058823529411765
After 10000 iterations, the model converges
```

Figure 4: Different batch sizes with momentum.

As shown in Figure 3, when mini-batch stochastic gradient descent is put in comparison with regular gradient descent, it is obvious that it takes a significantly less amount of time to converge the model. In addition, the accuracy was also higher than that of the regular gradient descent.

Furthermore, Figure 4 shows after adding momentum to the implementation of the minibatch stochastic gradient descent model, the accuracies are still higher than the regular one. Figure 5 shows that when momentum was set to 1.05 the accuracy was relatively the highest. In addition, knowing momentum was 1.05 for all instances, we have also observed in Figure 4 that the accuracy grows with the batch size, so does the speed of convergence, the largest batch size was the most effective and most accurate.



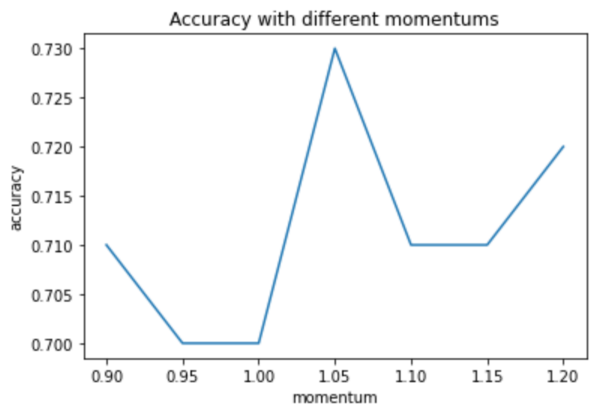


Figure 5: Resulted momentum vs accuracy(right).
Figure 6. The validation accuracy between different penalties and C (left).

In part2, we mainly used the LogisticRegression() function to train our model. Before fitting the data into this function, CountVectorizer() and TfidfTransformer() functions were called to convert the collection of text documents to a matrix of token counts[1] and convert the count matrix to a normalized tf or tf-idf representation[2]. Then we use a for loop to do the validation using the fake_news_val dataset. We mainly focus on the parameters in LogisticRegression() function. We set the solver = "liblinear" to see which of l1 and l2 penalty is better and we change C from 1 to 20 to find the best number for the model.

After the validation, as shown in Figure 6 we find that the l2 penalty with C = 17 can give us the highest validation score, which is 0.76. Then we use these best parameters to test our model using the fake_news_test dataset and get a 74.87% accuracy result.

DISCUSSION & CONCLUSION

In conclusion, the two parts of the project both required binary classification, data fitting and processing performed by the logistic regression model. The first part of the project showed that the mini-batch stochastic gradient descent with a momentum of 1.05 is 0.11 higher in accuracy compared to the regular gradient descent. It was observed that among different sizes of batches with the same learning rate and iterations, the accuracy is fairly consistent. Although the accuracies tend to change due to the randomized data selected for each execution, the difference was only varying in the range of 0.70±0.02. In addition, when momentum remains the same, the larger the batch size, the faster the convergence speed. Although it was Part 1 could have been improved by processing the data with normalization every step of the way to ease the optimization. On the other hand, the text classification algorithm built did not manage to succeed at 80% accuracy, but 74.87% was very close to that.

STATEMENT OF CONTRIBUTION

The team was gathered and led by Isabella, who also planned for our group meetings. Our team split the work evenly yet with our individual focuses. Specifically, Isabella and Yiyuan worked on part1, optimization, of the lab. Isabella wrote the first 2 steps of part one, Yihuan handled steps 3 and 4. Nianzhen worked on the second part, text classification, of the lab. Then the three of us all contributed to the report.

References:

[1]*Sklearn.feature_extraction.text.CountVectorizer*. scikit. (n.d.). Retrieved October 26, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html.

[2]*Sklearn.feature_extraction.text.TfidfTransformer.* scikit. (n.d.). Retrieved October 26, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html.

Appendix

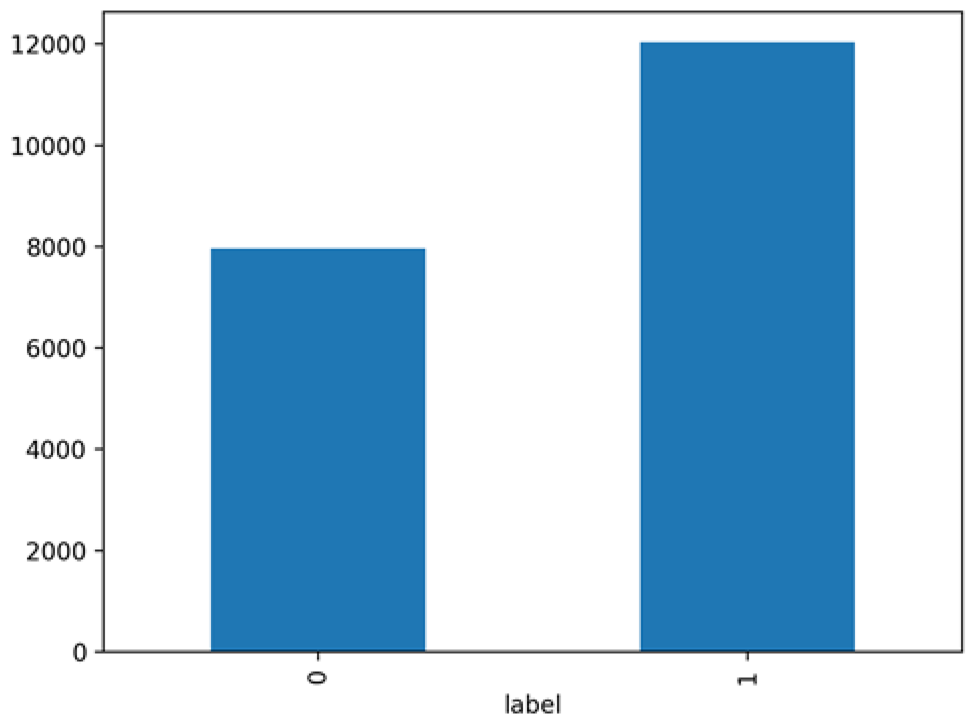


Figure A.1. visualized result of real news and fake news in fake_news_train dataset

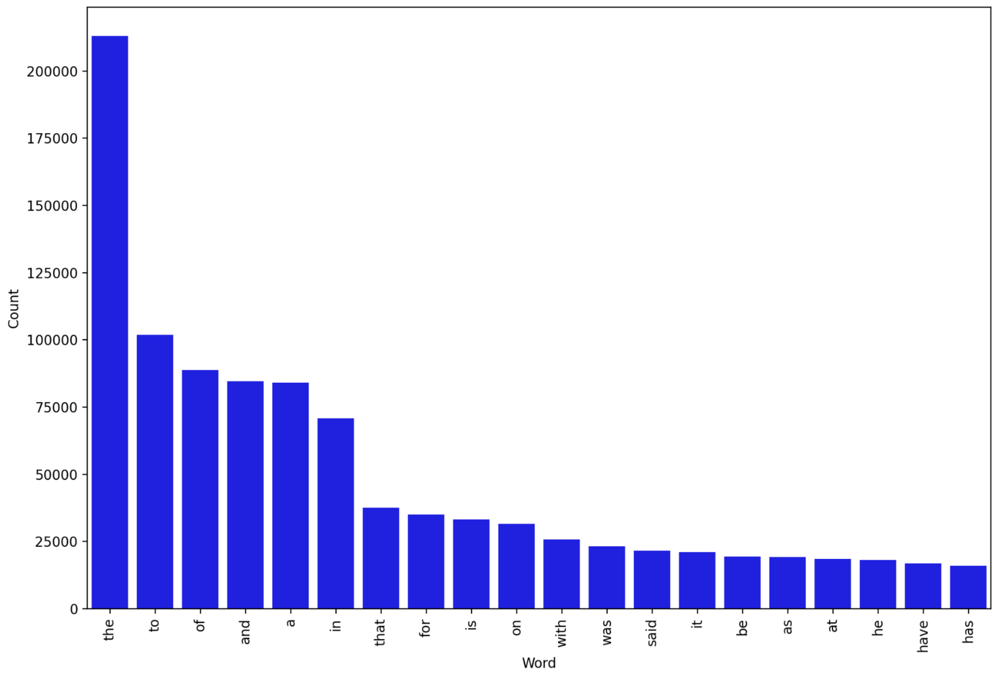


Figure A.2. Top 20 words appeared in the fake news

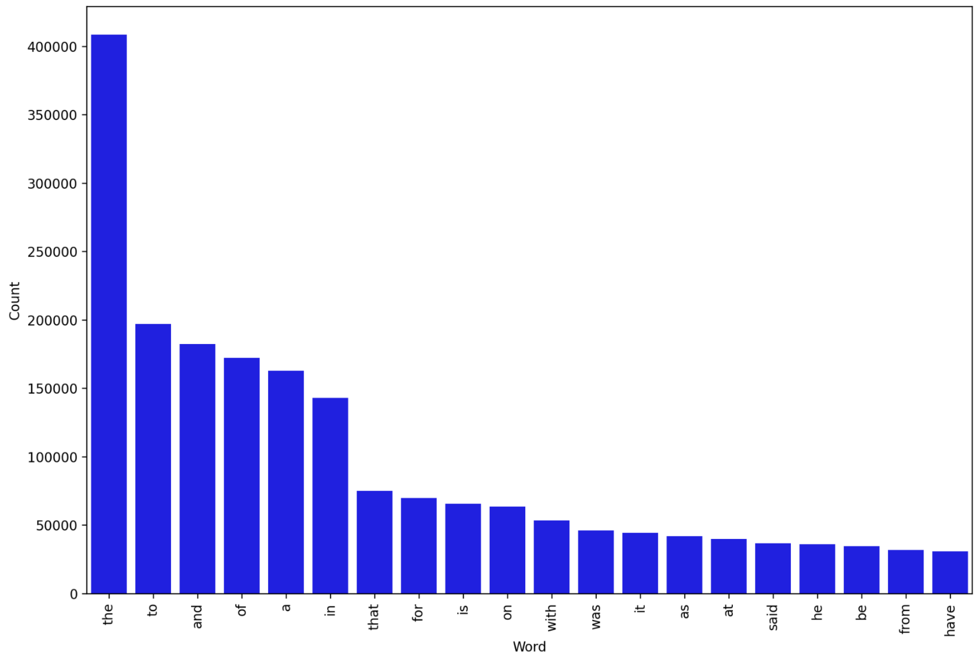


Figure A.3. Top 20 words appeared in the real news