Marie Guertin (260870552)
Mohanna Shahrad (260972325)
Christian Martel (260867191)

Mini-Project #2: Optimization and Text Classification

Group 13

COMP 551 - Applied Machine Learning
Presented to Prof. Siamak Ravanbakhsh
Departement of Computer Science

October 27, 2021
McGill University

Abstract

In this project, we investigate the performance of logistic regression on two different predictive tasks for the following data sets: the Diabetes data set and the Fake News data set. First, we attempt to predict if someone has diabetes from various factors, such as age, number of pregnancies, BMI, blood pressure, etc. Our logistic regression model is optimized using gradient-based methods. We perform hyper-parameter tuning of the learning rate and the number of iterations and explore the impact of batch size and momentum on our model's accuracy and training time. We found that our model performs best when trained with a learning rate of 0.0003. It converges after 891096 training iterations and gives a test accuracy of 76.47%. Second, we try to detect whether an article is generated by a computer or written by a human being. From the text data, we extract the features such as character and word count, and word length average. We tested different preprocessing tools and our final logistic regression model has a test accuracy of 78.37% using Lancaster Stemmer.

1 Introduction

First, we explore the process of optimizing a logistic regression model with gradient-based methods. We use a preprocessed version of the Diabetes data set [1] which was provided to us. We attempt to predict whether someone has diabetes based on features such as age, number of pregnancies, BMI, blood pressure, etc. First, we determine the ideal learning rate through empirical testing. Next, we investigate the effect of batch size and momentum on our model accuracy and convergence speed. We found that our optimized fully-batched model. Our final solution performs fairly well with an accuracy of 76.47% on the test set. Our model's accuracy is very inferior when compared to other experiments who have achieved as much as 97.34% accuracy [2]. Our result diverges from this mainly due to the lack of data preprocessing.

Second, we investigate text classification on the Fake News data set which was provided to us. We attempted to classify the text news into fake and real categories by creating a proper text preprocessing pipeline. First, we did text preprocessing and cleaning to normalize the data. Afterward, we extracted features from text data using CountVectorizer and TfidfTransformer[3]. More specifically, we used text stemming methods to reduce the ambiguity for our algorithm to filter such words and reduce them to the base form. We tested three different stemmers namely Porter, SnowBall, and Lancaster, and compared their training and test results.

2 Datasets

2.1 Diabetes Data Set

The Diabetes Data Set contains a total of 768 data points. It has been split into training, validation, and test subsets with 600, 100, and 68 data points respectively. Each records contains a label "Outcome" indicating whether someone has diabetes ("Outcome" = 1) or not ("Outcome" = 0). The dataset contains 8 numerical features – pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, and age. 35.5% of records in the training set indicate a positive diabetes diagnostic. No preprocessing of the data set was performed.

2.2 Fake News Data Set

The Fake News Data Set is split into training, validation, and test data sets with 20,000, 2,000, and 3,000 data points respectively. The data set contains one feature (column 'text') and one target (column 'label') where the texts labeled as "1" are fake and the rest are real. As the first step after loading the data set, we checked data cleanliness by searching for duplicate rows and rows with missing values. The crucial step was to perform text preprocessing on the data set that was achieved by the following: expanding extractions, lowercasing the text, removing the punctuation

marks and extra spaces, and also stemming the sentences. After preprocessing, we generated the visualizations to get a better intuition of the data. (They are available in the provided notebook)

3 Results

3.1 Part 1: Optimization

First, we ran a basic gradient descent and did some empirical testing (see python notebook for graphs) to find an optimal learning rate. We found that a learning rate of 0.0003 yielded the greatest convergence speed with the best accuracy. The results are shown in table 1. Using a lower learning rate, we still had the same accuracy but the solution took more iterations to converge. This is expected because our algorithm still finds the same minima but its takes more iterations to do so. However, when the learning is too large, it doesn't converge because the algorithm keeps jumping over the minima never reaching it.

		Accuracy (%)		
learning rate	Convergence speed	Train	Validation	Test
0.00020	1243185	77.17	75.00	
0.00025	1094003	77.17	75.00	
0.00030	891096	77.17	75.00	76.47
0.00031	1011012	77.5	75.00	

Table 1: Summary of results for finding optimal learning rate

Next, we implemented mini-batch stochastic gradient descent and tested using batches of size 8, 32, 128 and 256. We found that the largest batch had the fastest convergence speed and had more stable graphs. We ran a million iterations for each batch size, and while none of them fully converged, it was clear that solutions with greater batch size converged faster and had smoother graphs and a better train and validation accuracy on average. It is clear that for all batch sizes, their convergence speed and the quality of the solution they yield is less optimal than that of the fully-batched baseline.

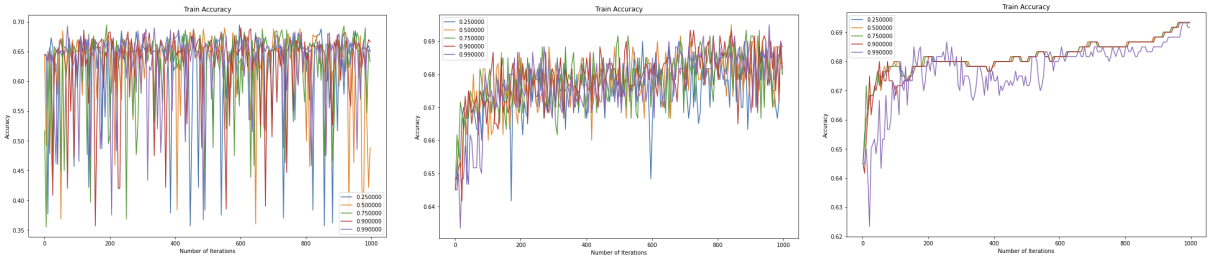

Figure 1: Train accuracy with regards to the number of training iterations for batch sizes of 8 (left), 256 (center) and fully batched (right)

Finally, we added momentum to our gradient descent. We found that training time increased exponentially when calculating momentum compared to the linear increase with normal gradient descent. After 5000 iterations, with a momentum coefficient of 0.9, the train and validation accuracy respectively increases from 63% to 69% and 61% to 67%. Furthermore, the gradient norm at 5000 iterations goes from 28.08 to 0.04 which indicates a great increase in convergence speed. It would have been interesting to see the fully converged solution, but unfortunately we lacked computing power and time. Finally, we ran the same experiment with a mini-batch and large-batch gradient descent. The result showed that momentum was most efficient in a fully batch setting and less

efficient in a mini-batch. From Fig. 1 we get a better intuition of why that is.

3.2 Part 2: Text Classification

The results of different experiments on validation data are shown in Table 2. We used LogisticRegressionCV model as it uses cross-validation to select the regularization coefficient C, unlike LogisticRegression where we need to specify a regularization coefficient each time.

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
Ngram Range Stopwords	(1,1) None	(1,1) English	(1,2) None	(1,2) English	(2,2) None	(2,2) English
Training Accuracy with Porter(%)	75.75	73.25	**78.65**	73.9	74.45	70.9
Training Accuracy with SnowBall(%)	75.7	72.45	**78.2**	73.75	74.4	71.1
Training Accuracy with Lancaster(%)	76.2	71.7	**78.15**	73.95	73.9	70.35

Table 2: Summary of text classification results for each validation experiment

In this part, we observed the impact of text preprocessing on the model's performance. Applying proper text preprocessing steps such as expanding contractions, lower-casing words, removing punctuations and extra spaces, increased performance significantly. In addition to the mentioned preprocessing steps, we also performed text stemming to reduce inflected words to their bases. To analyze more deeply, we compared the performance of three different stemmers (Porter, SnowBall, Lancaster). We found that in all cases the pipeline with Ngram value (1,2) and no stopwords had the highest accuracy. Finally, we fitted the test data with the best model's hyper-parameters for each stemmer which resulted in 78.13%, 78.33%, and 78.37% test accuracy respectively.

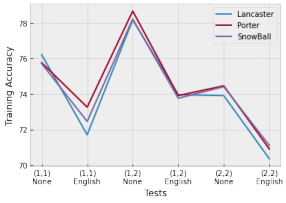

Figure 2: Training accuracies for each stemmer

Figure 3: Porter Classification report, Test Accuracy:78.13%

Figure 4: SnowBall Classification report, Test Accuracy:78.33%

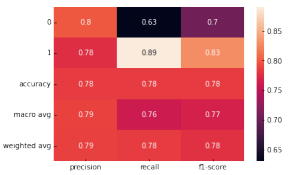

Figure 5: Lancaster Classification report, Test Accuracy:78.37%

4 Discussion and Conclusion

This project was a great first-hand experience with leveraging logistic regression models, optimization and preprocessing techniques to solve important real-life problems. In the first part, we explored learning rate, batch size and momentum in the context of gradient descent optimization. In the second part we examined the impact of proper text preprocessing on text classification. The preprocessing normalizes the text data and the application of preprocessing methods such as stopword and punctuation mark removal, and word stemming, can improve the performance of our model[4]. Due to the limited nature of this report, there are many plots and visuals that could not be included. These plots can be found in our Google Colab document.

5 Statement of Contributions

The workload was broken down equally across all team members as follows: Marie Guertin (Optimization, testing, debugging, report), Mohanna Shahrad (Text Classification, testing, debugging, report) and Christian Martel (project setup, Optimization, testing, debugging, report).

References

[1] Michael Kahn. *Diabetes Data Set*. 1994. URL: https://archive.ics.uci.edu/ml/datasets/Diabetes.

[2] Changsheng Zhu, Christian Uwa Idemudia, and Wenfang Feng. "Improved logistic regression model for diabetes prediction by integrating PCA and K-means techniques". In: *Informatics in Medicine Unlocked* 17 (2019), p. 100179. ISSN: 2352-9148. DOI: https://doi.org/10.1016/j.imu.2019.100179. URL: https://www.sciencedirect.com/science/article/pii/S2352914819300139.

[3] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[4] Yair Yigal Yaakov HaCohen-Kerner Daniel Miller. *The influence of preprocessing on text classification using a bag-of-words representation*. Version latest. May 2020. DOI: 10.1371/journal.pone.0232525.