COMP 551 — Project 2

Group 66

Yuyan Chen, Dailun Li, Zijun Zhao

October, 2021

Abstract

In this project, we investigated the optimization process of gradient-based methods of logistic regression on the diabetes dataset. We implemented full-batch gradient descent and mini-batch stochastic gradient descent and added momentum to the gradient descent implementation. As a result, the model fully converged to a solution with a learning rate equal to 0.0003 and max iterations equal 800,000. Furthermore, we found that mini-batch stochastic gradient descent with mini-batch size (...) gave the most satisfying result in terms of convergence speed and accuracy. Besides, adding momentum increases the convergence speed and (the quality of the final solution???? definition????) We also explored text-specific preprocessing techniques with logistic regression on the fake-news dataset.

1 Introduction

This project investigates the optimization process of gradient-based methods (task1) and explores preprocessing techniques for fake news detection (task2). For task1, we investigated the speed of convergence and the performance of each logistic regression model on predicting if a person has diabetes based on eight numerical features. (+conclusion)

For task2, we investigated the impact of preprocessing methods on fake news detection. We explored different preprocessing methods, namely stemming, lemmatization, n-grams, count vectorization, term frequency vectorization, and term frequency-inverse document frequency vectorization. (+conclusion)

2 Datasets

The **diabetes** dataset contains 8 numerical features, namely, pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, and age. We used the dataset without performing cleaning or preprocessing.

The **fake news** dataset is used to detect if the article was generated by a computer or humans. We explored some combinations of the preprocessing methods in each column with or without stopwords.

Table 1: Preprocessing Decisions

Text preprocessing	Vectorization	N-grams
None	Count	unigram
Lemmatization	Tf	uni- and bigram
Stemming	Tf-idf	uni-, bi-, and trigram

3 Results

Baseline

We first running the basic logistic regression model (momentum = 0, full batch, $\varepsilon = 1e^{-4}$, learning rate $= 1e^{-2}$) with raw data for 10,000 epoches, the validation accuracy output is 0.62. (part 2 = ???) This serves as a baseline of the logistic regression model to evaluate the performance of other implemented methods.

Optimization

For part 1, model fitting stops when either $||\nabla|| < \varepsilon$ or max epochs have been reached ($\varepsilon = 0.005$). "Point of convergence" is defined as the number of epochs N at which the validation accuracy reaches the maximum and oscillates in a small range as the number of epochs increases.

1. We chose the **learning rate** to be **0.0003** and **max epochs** to be **800,000** for the following reasons. When the learning rate(lr) was 0.003, fitting stopped at 1,567,815 epochs. By inspecting the plot of training and validation accuracy (Figure 1), we found the model converged at 800K. The model did not converge to a solution when $lr = 0.0004$ (Figure 2). When $lr = 0.0002$, fitting stopped at 2,382,622 epochs and the model converged at around 1000K. (Figure 3). This indicated that the optimal learning rate that is neither time-consuming nor

causing inconsistent gradient should be between 0.0003 and 0.0004. 2. We used batch sizes of 8, 16, 32, 64, 128, and 256. Learning rate was fixed as 0.0003 and max iterations as 3,000,000. All models failed to converge as training and validation accuracies oscillated drastically (Figure 4, 5, 6, 7, 8, 9). We found that as batch size increased, training and validation accuracy reached the maximum more slowly. Besides, the range of accuracies decreased as number of epochs and batch size increased. In conclusion, the fully batched model outperformed all mini-batched models in terms of the convergence speed and the quality of the final solution.

3. As momentum increased from 0.1 to 0.9, the convergent point remained almost the same (600k epochs). Hence, our discussion focuses on the momentum values listed in Table 2. As shown in Figure 10, 11, 12, 13, 14, as momentum increased from 0.9 to 0.9995, the plots of validation accuracy remained almost the same. From 0.9999 to 0.99999, the validation accuracy began to oscillate in a larger range. For all momentum values, adding momentum increased the speed of convergence compared to the regular gradient. However, as momentum increased from 0.99995 to 0.99999, the speed of convergence began to decrease. Momentum had little to no impact on the quality of the final solution as the highest validation accuracies were all around 0.77.

Table 2: Momentum

momentum	Point of convergence
0.9	$\approx$ 600K
0.999	$\approx$ 600K
0.9995	$\approx$ 600K
0.9999	$\approx$ 600K
0.99995	$\approx$ 600K
0.99999	$\approx$ 700K

4. All models failed to converge with small batch size (4) and large batch size (256). However, adding momentum had a great impact for small batch size. The validation accuracies oscillated drastically when momentum equaled to 0.9, 0.999, 0.9995, and as momentum increased from 0.9999 to 0.99999, the range of validation accuracies decreased from 0.09 to 0.01. When momentum = 0.99999, the validation accuracy was bounded between 0.75 to 0.76. For large batch size, all models reached the maximum validation accuracy around 250K epochs. The range of validation accuracies decreased as momentum increased. Hence, we conclude that fully batched is the most effective setting and small mini-batched is the least effective.

Text Classification

Based on the result of part 1, we set **learning rate** to (...) and **max iterations** equal to (...). We first

4 Discussion and Conclusion

Optimization

1. As the variance in the gradient estimation for each iteration is large when the batch size is small, in general, models trained with mini-batch stochastic gradient descent(SGD) have less stable performance than the fully batched baseline. Though the smaller the batch size, the faster the model can reach the maximum accuracy, the performance of all models ocsillates drastically. None of the models converges to a final solution.

2. For mini-batched SGD, as momentum increases, the oscillation is reduced significantly. Since larger momentum places a smaller weight for the most recent gradient, larger momentum averages the impact of each mini batch on the gradient, and hence reduces the variation. However, for fully batched gradient descent, when momentum grows larger, it increases oscillation as the weight for the most recent gradient is so small that the weights are not updated in the right direction.

Statement of Contributions

References

Appendix

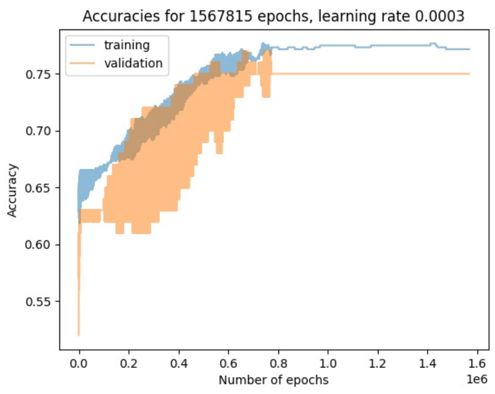

Figure 1: Training accuracy and validation accuracy versus number of epochs (lr = 0.0003)

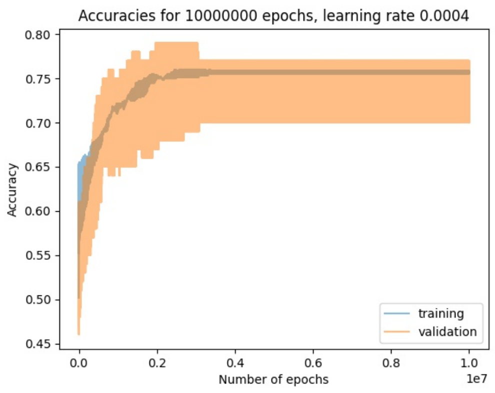

Figure 2: Training accuracy and validation accuracy versus number of epochs (lr = 0.0004)

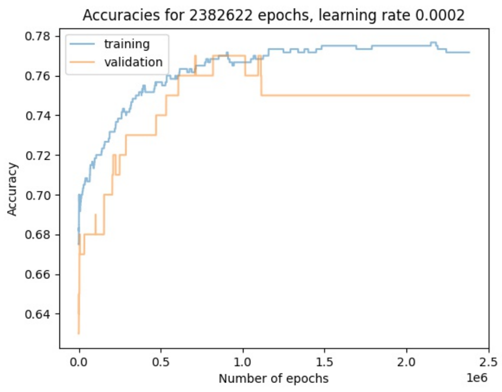

Figure 3: Training accuracy and validation accuracy versus number of epochs (lr = 0.0002)

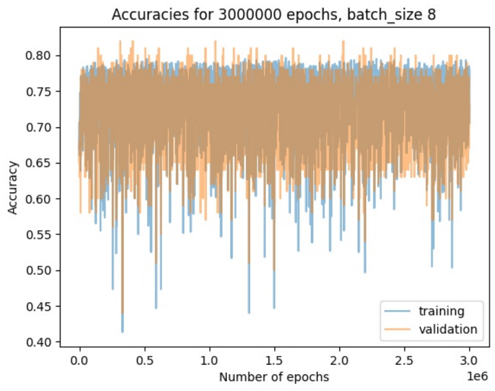

Figure 4: Training accuracy and validation accuracy versus number of epochs (batch size = 8)

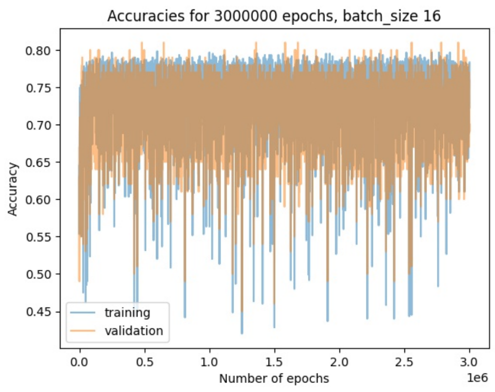

Figure 5: Training accuracy and validation accuracy versus number of epochs (batch size = 16)

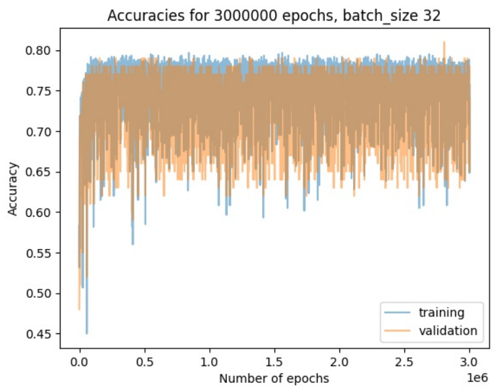

Figure 6: Training accuracy and validation accuracy versus number of epochs (batch size = 32)

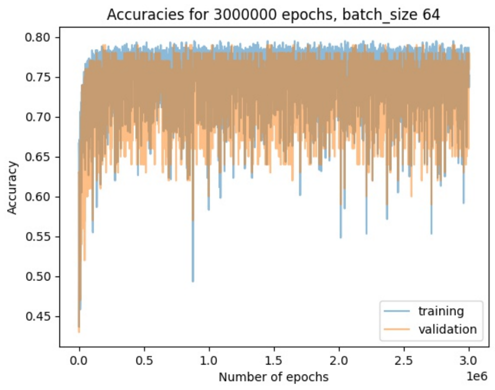

Figure 7: Training accuracy and validation accuracy versus number of epochs (batch size = 64)

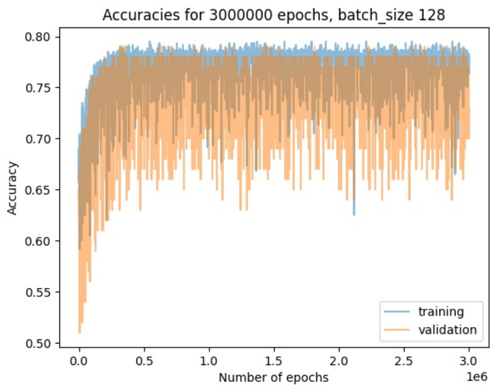

Figure 8: Training accuracy and validation accuracy versus number of epochs (batch size = 128)

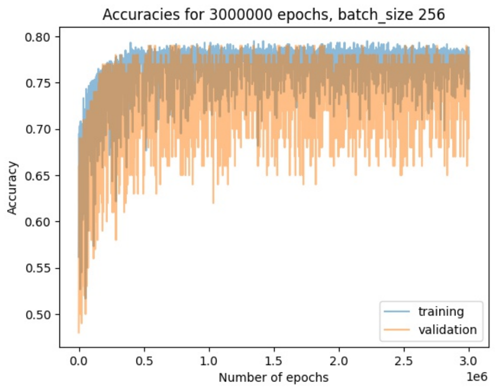

Figure 9: Training accuracy and validation accuracy versus number of epochs (batch size = 256)

Table 3: Momentum

momentum	stopping point (epochs)
0.1	1,588,414
0.2	1,588,414
0.3	1,588,414
0.4	1,588,414
0.5	1,588,414
0.6	1,588,413
0.7	1,588,412
0.8	1,588,411

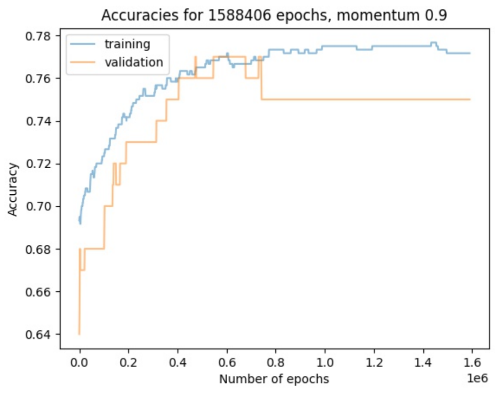

Figure 10: Training accuracy and validation accuracy versus number of epochs (momentum = 0.9)

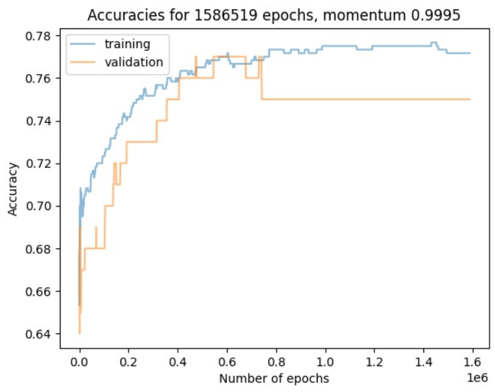

Figure 11: Training accuracy and validation accuracy versus number of epochs (momentum = 0.9995)

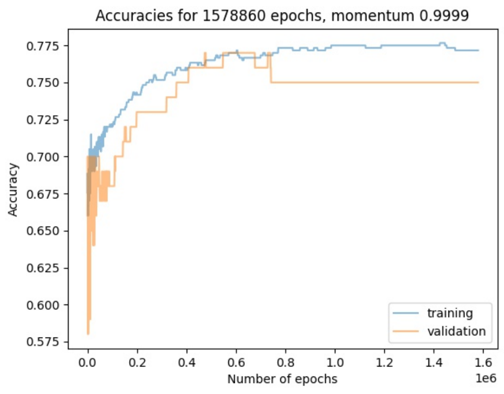

Figure 12: Training accuracy and validation accuracy versus number of epochs (momentum = 0.9999)

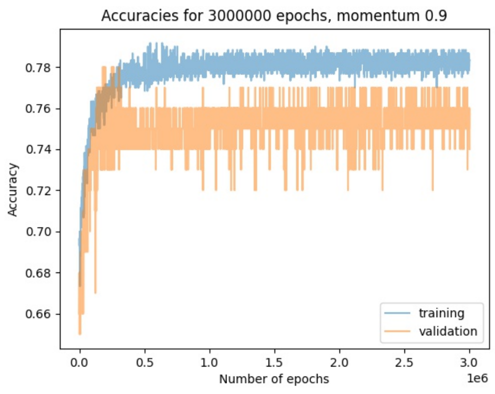

Figure 15: Training accuracy and validation accuracy versus number of epochs (batch size = 256, momentum = 0.99999)

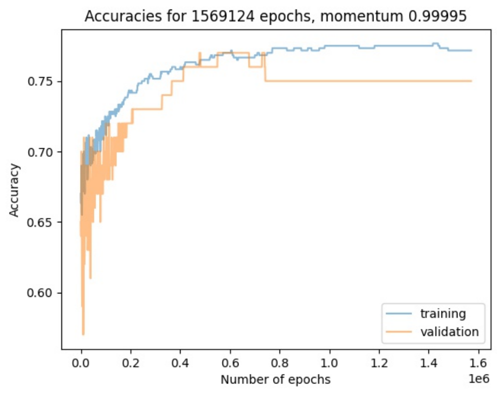

Figure 13: Training accuracy and validation accuracy versus number of epochs (momentum = 0.99995)

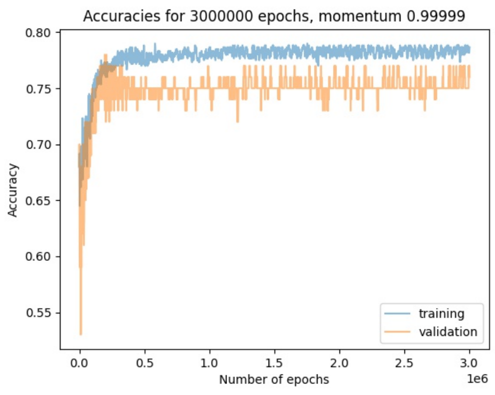

Figure 16: Training accuracy and validation accuracy versus number of epochs (momentum = 0.99999)

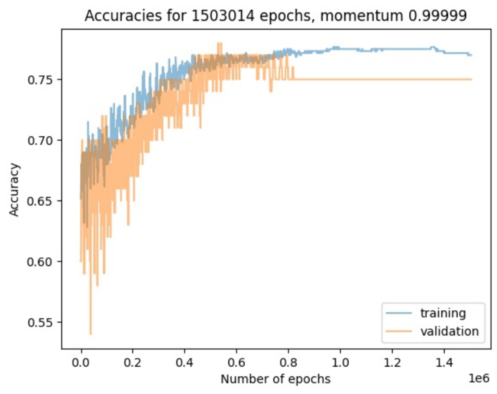

Figure 14: Training accuracy and validation accuracy versus number of epochs (momentum = 0.99999)

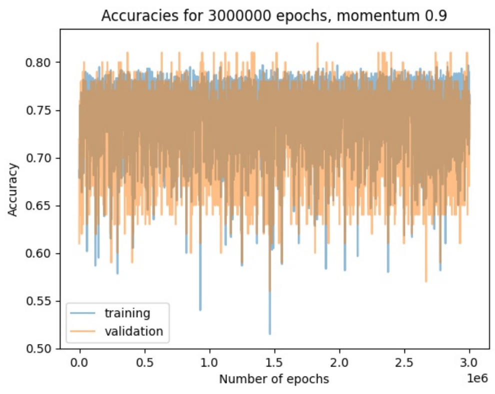

Figure 17: Training accuracy and validation accuracy versus number of epochs (batch size = 8, momentum = 0.9)

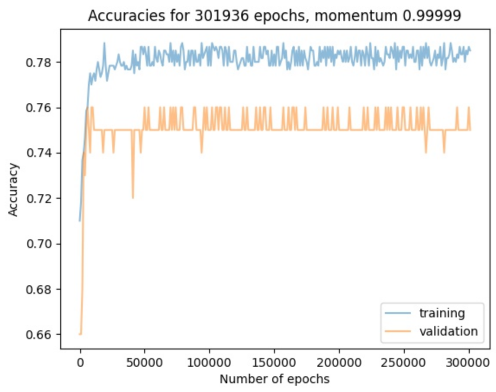

Figure 18: Training accuracy and validation accuracy versus number of epochs (batch size = 8, momentum = 0.99999)