PROJECT 2 REPORT

By Barry Li 260912069
Minzhe Feng 260886087
Sophia (Suofeiya) Man 260835947

Abstract

The project mainly concerns two parts, optimization, and text classification.

For the optimization part, we compare the performance of plain logistic regression, as well as stochastic gradient descent (SGD) with various mini-batch sizes and momentum coefficients, to gain some insights on their impacts on convergence speed and quality of the final solution.

For the text classification part, we set up a plain logistic regression model as our base prediction model. We then benchmark various text tokenization and preprocessing techniques to draw comparisons of the impact of these techniques on the final prediction quality (measured by accuracy).

Introduction

In the optimization part, the goal is to predict if a person will have diabetes or not. Plain logistic regression is used at first to set up a performance baseline for subsequent experiments. Then it SGD with various momentum coefficients and mini-batch sizes is added. We use training and validation losses to evaluate convergence, as the gradient norm does not always approach epsilon, and minimal validation loss to evaluate the model's performance. In general, the impact of momentum and mini-batch size on convergence speed is more obvious than on performance. The details can be found in the "Results" section.

in the text classification part, the goal is to classify the type of the text. We first build a prediction pipeline composed of a count vectorizer, a TF-IDF transformer and a logistic regression model. The raw text corpus is then processed with an assortment of techniques and fed into the pipeline. The decision of whether the text should be split into unigrams, bigrams, or a combination of both is also explored. Overall, we find that the most significant improvement comes from combining unigram and bigram, while more cosmetic changes, such as removing punctuations, can unfavorably affect the model's performance.

Datasets

The diabetes dataset (for part 1)

- 8 features:
  Pregnancies/Glucose/BloodPressure/SkinThickness/Insulin/BMI/DiabetesPedigreeFunction/Age
- Binary label: have diabetes = 1, without diabetes = 0
- Data normalization as the only preprocessing task.

The fake news dataset (for part 2)

- Text corpus: raw text corpus with varying contexts, such as movie reviews and recipes
- Binary label: emitted from the raw text corpus.
- A wide range of preprocessing decisions, such as n-gram and removal of punctuations, are employed.

Results

Part 1: Optimization

1.Find the optimal combination of learning rate and maximum iterations that leads to full convergence

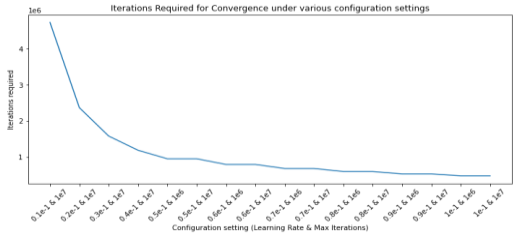


Figure 1

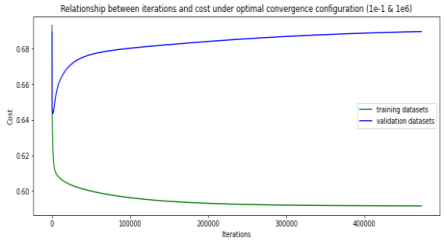


Figure 2

By figure 1, learning_rate=1e-1 and max_iters=1e6 is the optimal combination, since it converges faster with a relatively large learning rate. Figure 2 shows the implementation of the above setting on both the validation dataset and training dataset. It gives an accuracy score of 0.662 on the validation dataset. Clearly, the plot converges before the gradient norm approaches epsilon. Concerning hardware constraints, we choose learning_rate=1e-1 and max_iters=1e5 as our setting for the rest of the experiments.

2.Find the optimal configuration of gradient descent with various mini-batch sizes

We tried six values of mini-batch size and used training and validation losses to evaluate convergence. From the figures below, the model does not converge until the batch size increase to a certain threshold, in our case 50, and has the best convergence speed. As batch size keep increasing, convergence speed decreases. (See appendix for graph and tables)

3.Find the impact of momentum on convergence speed and performance with a fully-batched sample
From the figures in appendix, we found that, when momentum is smaller than 0.9, the minimum cost on the validation dataset generally occurs when epoch is between 1130 and 1150 and there is not a big difference graphically. However, when momentum coefficient increases from 0.988 to 0.999. The convergence speed decreases significantly and the graph noticeably, its oscillation increases dramatically, within the region where the minimum loss occurs. Furthermore, since the differences among validation losses with various values of momentum coefficients are trivial, there is no obvious relationship between momentum coefficient and performance of the model. In a word, the impact of momentum on the fully-batched dataset is not significant. (See appendix for graph and tables)3

4.Find the impact of batch size and momentum coefficient on convergence speed and performance
We choose small mini-batch size = 6, large mini-batch size=300. As the behavior with mini-bach size=600 shows a similar behavior to the one shown in part 1.3, and, hence, can be viewed as fully batched.
Group 1: the gradient descent with small mini-batch size = 6: (Some of the graph were in appendix for this part important ones are here)

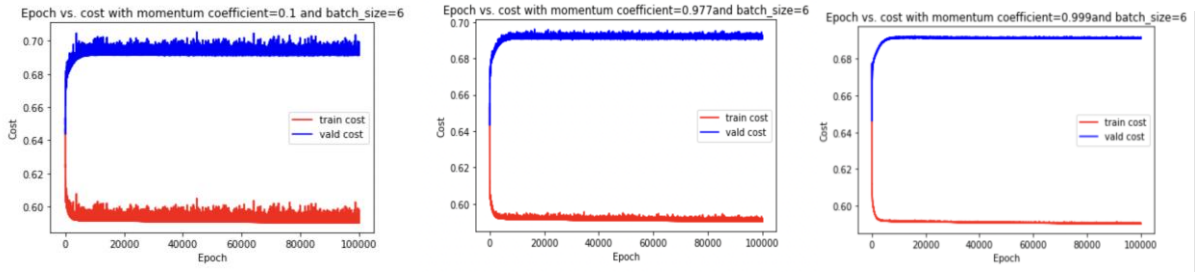


Table 3: Q1.4 The epoch that results in the minimum validation loss with mini-batch size=6:

momentum	0.1	0.5	0.9	0.955	0.977	0.988	0.999
Min loss	0.6437	0.6439	0.6438	0.6432	0.6432	0.6436	0.6461
epoch	11	12	12	10	9	12	21

When batch_size=6, the model maintains a large oscillation when momentum coefficient is less than 0.9, and then decreases significantly as momentum coefficient approaches 0.999. Furthermore, the convergence speed first increases slightly, then decreases when momentum coefficient changes from 0.977 to 0.999. From the table above, we can see that the performance, measured by min loss, is stable.

Group 2: the gradient descent with large mini-batch size = 300:

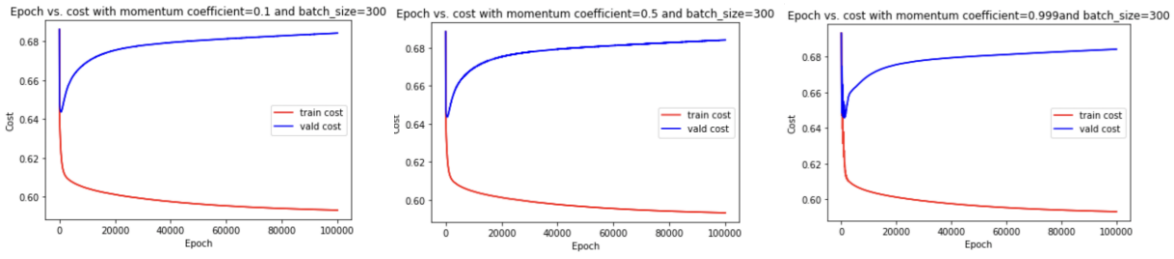


Table 4:Q1.4 The epoch that results in the minimum validation loss with large mini-batch size = 300:

Momentum	0.1	0.5	0.9	0.933	0.966	0.988	0.999
Min loss	0.6436	0.6436	0.6436	0.6436	0.6436	0.6436	0.6457
epoch	570	569	585	570	577	614	1122

Different from the case with a small mini-batch size, a large mini-batch size exhibits minimal oscillations. The convergence speed shows a similar pattern: it remains stable and relatively small when the momentum coefficient is smaller than 0.966, and then slows down significantly when approaching 0.999. There is no significant difference in performance. By combining the results with the fully-batch dataset from Part 1.3, the setting with a large mini-batch size is most effective with a higher convergence speed, while the setting with the full batch size is least effective as it requires a larger amount of epoch to converge.

Part 2: Text Classification

To setup, we build a pipeline consisting of a count vectorizer, a TF-IDF transformer (to scale down the impact of more common words and to increase convergence speed) and a logistic regression cv class (cv for automatic cross-validation) to find the optimal hyperparameters.

We then mainly experiment with two types of optimization techniques:

(1) Value of n (limited to 1 and 2 due to hardware constraints) when tokenizing text with n-gram

(2) Keep or remove special characters and punctuations

Table 5: Prediction accuracy of different configurations on the validation dataset: (P for punctuation)

Configuration	P & Unigram	P & Bigram	P & Unigram & Bigram	Bigram	Unigram + Bigram
Accuracy	0.7445	0.7471	0.7812	0.7359	0.7695

Prediction accuracy of the best-performing model on the test dataset: 0.7670

From the result, we can see that a combination of unigram and bigram performs better than using either of them alone, while keeping punctuations can negatively affect the model's performance (detailed discussion of possible causes below).

Discussions & Conclusions

Part 1: Optimization

Batch Size: We must consider batch size as a hyperparameter when training to find the optimal value that suits the data. Batch size does not necessarily affect the final accuracy of the model, but rather affects the rate of learning and convergence speed. Furthermore, a very small batch size always results in larger oscillations.

Momentum: There is no gold standard to select the best momentum coefficient. Counterintuitively, a larger momentum does not guarantee faster convergence.

Part 2: Text Classification

One possible reason that a combination of unigram and bigram performances better is that it combines advantages of both models: higher term frequency (hence lower chance of overfitting) of unigram models and learning more contextual information of bigram models. However, this mixed approach still can only improve the test set prediction accuracy by a tiny margin (when compared to plain unigram or bigram), while more peripheral changes, such as removing punctuations, can even lower the model's performance (probably because text files with specific contexts, such as recipes, rely on punctuations to be correctly identified), let alone bumping it up more significantly.

Therefore, alternative text tokenization methods and prediction models might be needed to further boost model performance. Also, although higher n in n-gram models is not experimented with due to hardware constraints, we doubt that it will contribute positively to the model's performance, as overfitting might become too huge a problem and overshadow the potential benefits of learning more contextual information that come with a higher n. Plain bigram models already underperform when compared to plain unigram ones, which can very well already be a result of more overfitting in bigram models.

Statement of Contributions

Barry Li: Mini-batch gradient descent implementation (part 1.2 & 1.4)

Minzhe Feng: Plain logistic regression (part 1.1) & text classification (part 2)

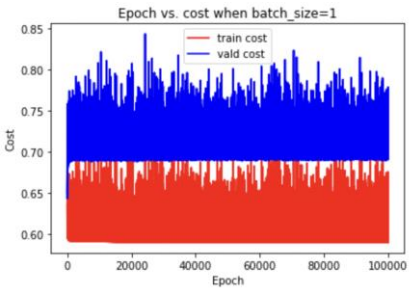
Sophia (Suofeiya) Man: Momentum implementation (part 1.3 & 1.4)

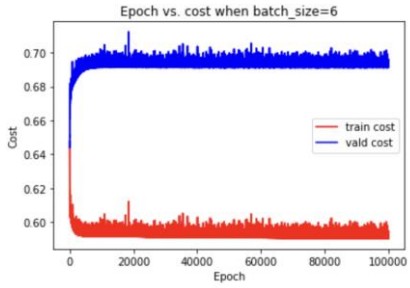
Appendix:

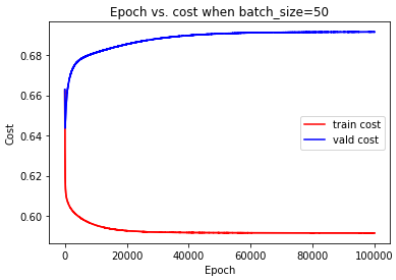
Part 1.2:

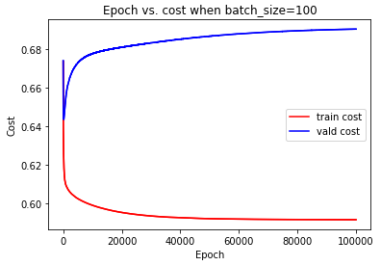
Table 1: Q1.2 The epoch that results in the minimum validation loss for each batch size

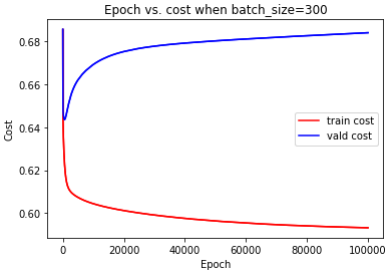
Batch size	1	6	50	100	300	600
Min cost	0.6428	0.6438	0.6435	0.6436	0.6436	0.6436
Epoch	2	10	90	195	561	1130

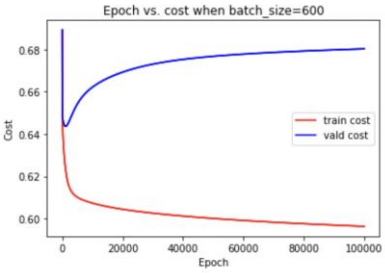








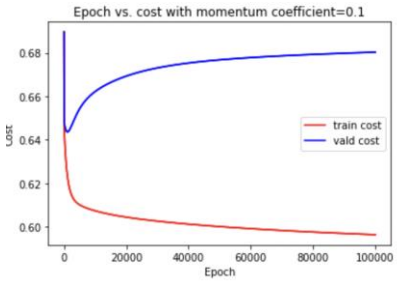


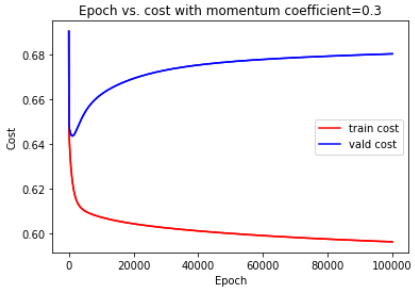


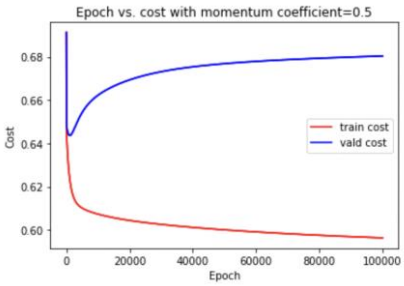
Part 1.3:

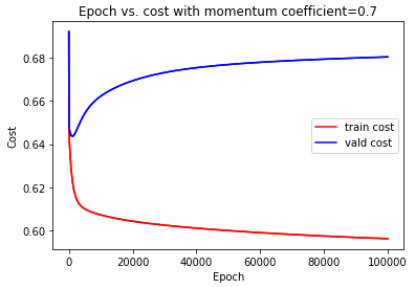
Table 2: Q1.3 The epoch that results in the minimum validation loss for each momentum coefficient

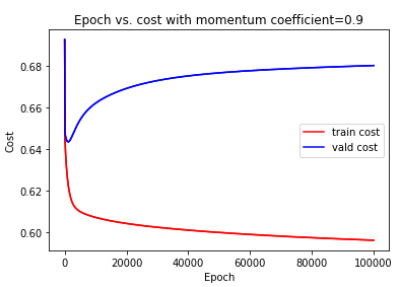
Momentum	0.1	0.5	0.9	0.933	0.966	0.999
Min cost	0.6436	0.6436	0.6436	0.6436	0.6436	0.6436
Epoch	1131	1131	1140	1145	1160	2245

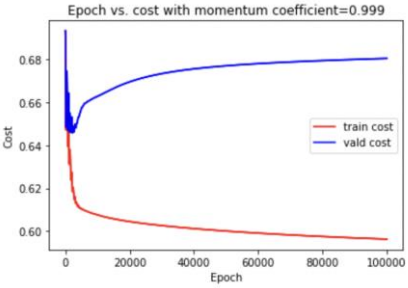






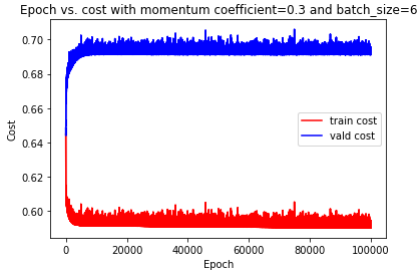


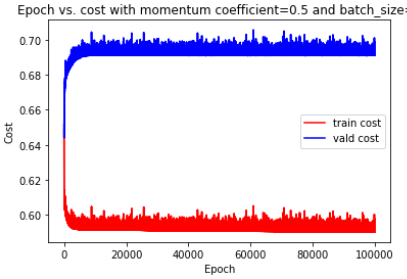


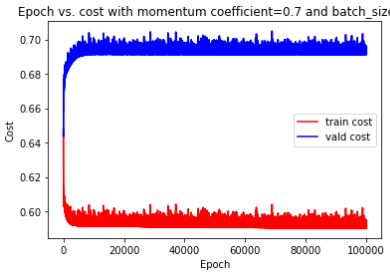


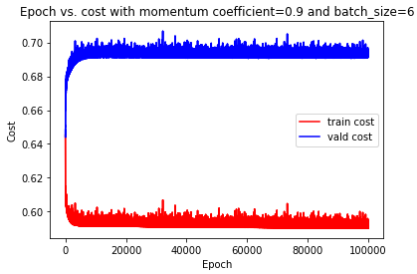
Part 1.4:

Group 1: (Graph that did not get presented due to lack of space)









Group 2: (Graph that did not get presented due to lack of space)