# Ilyes Wassifi - 260779518

Daniel Gallant -

Cecilia Jiang -

# Abstract

In this project, we wanted to gain some insight into the optimization process of gradient based methods, by implementing different variations of gradient descent and analyzing their impact on the convergence speed. We adjusted hyperparameter learning rate for the regular implementation of gradient descent to get the best accuracy performance. As the next step, momentum was also added to the gradient descent implementation and we compared it to the regular gradient descent to see if there was any difference in the convergence speed or quality of the final solution. Finally, we investigated how different batch sizes can impact these two critical metrics. Out of all these different combinations of hyperparameters, we achieved the highest accuracy of 0.824**.**

Then, using the fake news dataset, we classified if the texts were written by computers or humans. We used sklearn SDGClassifier to complete this task. An accuracy of 0.717 is obtained as the final result.

# Introduction

In order to study the performance of gradient descent and its variations, we first had to run logistic regression on the dataset diabetes provided. This dataset was already cleaned and preprocessed, so we did not have to do this step. The first step was to use logistic regression to classify our dataset, and find a learning rate and a number of training iterations. Then, we implemented gradient descent and observed the accuracy obtained. To choose the learning rate, we plotted the learning rate vs accuracy, and we noticed that the maximum accuracy achievable was when the learning rate was equal to 1e5. The same process was used to determine the number of iterations needed, which ended up being 10^6. Once gradient descent was implemented, the following step was to program the mini-batch stochastic gradient descent, the main difference between both algorithms being the use of subsets to update the parameters in the mini-batch stochastic gradient descent and is, thus, faster. Finally, momentum was added to the mini-batch stochastic gradient descent, and we realized that it helped decrease the time it took the algorithm to converge to a final value.

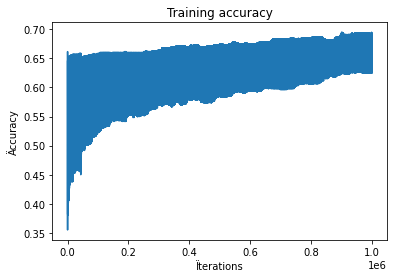
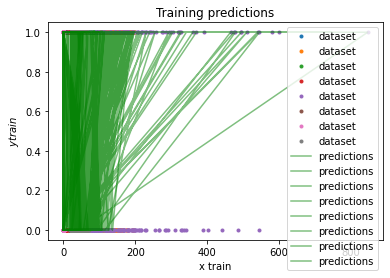
In the second part of this project, our goal is to make binary classification of a text dataset. We want to determine if an input text data is one piece of fake news or not. We applied the stochastic gradient descent algorithm in the sklearn package. As a result, we achieved 71% accuracy.

# Datasets

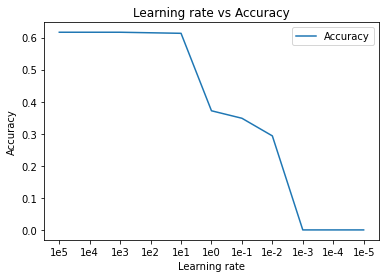
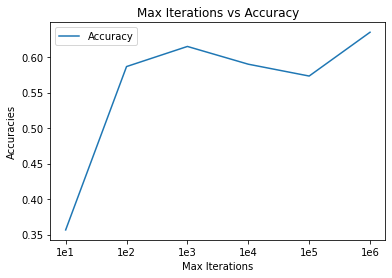
The dataset used for the first part of this project was the diabetes dataset. This dataset was divided in 8 different classes (Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function and Age) and a column displaying the different binary results (0 for non-diabetic, 1 for diabetic).

The dataset used for the second part of the project was “fake news”.

# Results

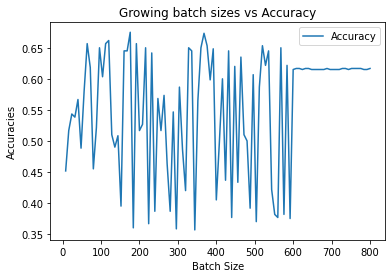
 

*Figure 1: Iterations vs accuracy Figure 2: Training predictions for the gradient descent*

*Figure 3: Learning rate vs Accuracy Figure 4: Max iterations vs accuracy*

As we can see on the figure 4 above, as the number of iterations increased, the accuracy increased accordingly. So we opted with a maximum number of iteration corresponding to the one giving us the best accuracy possible, which is 10^6 iterations. On figure 3 corresponding to the learning rate vs accuracy, we can notice that as the learning rate increased, the accuracy decreased, starting from a learning rate of 100. So we decided to choose the smallest learning rate possible while maximizing the accuracy to avoid it from overshooting, which was 100.



*Figure 5: Batch Size vs Accuracy*

For the mini-batch SGD, the best batch size found was 176, but a very good accuracy was found at a size of 64, which was the one chosen. The accuracies are resumed in the table below:

|  | **Accuracy (Training)** | **Accuracy (Test)** | **Time taken for the test (seconds)** |
| --- | --- | --- | --- |
| **Gradient Descent** | 0.635 | 0.588 | 77.89 |
| **Gradient Descent with momentum** | 0.597 | 0.529 | 82.72 |
| **Mini-batch SGD** | 0.562 | 0.706 | **53.50** |
| **Mini-batch SGD with momentum** | 0.733 | **0.824** | 56.78 |

The best accuracy was found with the mini-batch SGD with momentum as expected, with an accuracy of 0.824, and the worst accuracy was found with the gradient descent with momentum (accuracy of 0.529). For the time taken by the different algorithms, we can clearly see a difference between the regular gradient descent and the mini-batch SGD, the fastest one being the mini-batch SGD without the momentum (53.50 seconds).

For the second part of the project, we managed to get an accuracy of 71.7%

# Discussion and conclusion

This project helped us distinguish the differences in performance between the different gradient descent algorithms. We noticed that the mini-batch stochastic gradient descent with momentum helped us reach the best accuracy (82.4%), while gradient descent with momentum gave us the worst result amongst the 4 algorithms (52.9%). Dividing the dataset into multiple groups in mini-batch SGD helped us get better accuracy results, as well as a faster convergence time. Adding momentum to the mini-batch SGD increased the convergence time by 3.28 seconds but also increased the accuracy by 16.71%.

# Statement of contribution:

Daniel: Gradient Descent and Mini-batch SGD, Text Classification, Report

Ilyes: Momentum, Accuracy, Report

Cecilia: Logistic Regression, Report

# 

# Appendix

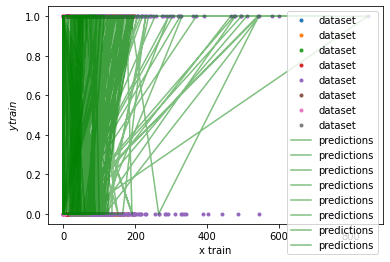
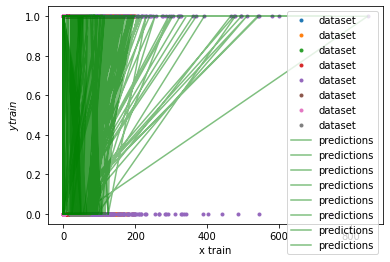


Figure 5: *Training predictions for the mini-batch SGD*



*Figure 6: Training predictions for the gradient descent with momentum*

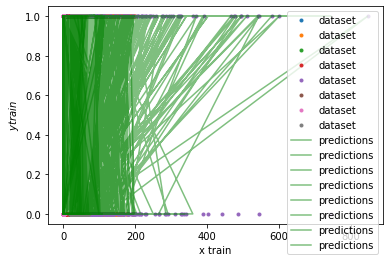


Figure 7: *Training predictions for the mini-batch SGD with momentum*