Comp 551 Mini-Project 1 Report

Simon Li (260984998)
Dept. of Electrical Engineering
McGill University
Montreal, Canada
xi.yang.li@mail.mcgill.ca

Yue Xin Li (260986580)
Dept. of Computer Science
McGill University
Montreal, Canada
yue.x.li@mail.mcgill.ca

Keyu Liang (260958571)
Dept. of Software Engineering
McGill University
Montreal, Canada
keyu.liang@mail.mcgill.ca

I. Abstract

In this project, we aimed to implement and compare the performance of two machine learning model, namely linear and logistic regression, using an analytical approach for the linear regression model and the gradient descent optimization algorithm for both linear and logistic regression. The logistic regression model was used to predict the class label of a given input data. Two benchmark datasets were used to train and evaluate the models: an energy efficiency dataset for linear regression and a qualitative bankruptcy dataset for logistic regression. The results showed that mini-batch gradient descent achieved a better performance compared to batch gradient descent and analytical methods in terms of convergence rate and overall loss. Significantly, the influence of batch size on the performance of mini-batch gradient descent was also analyzed, where it was found that the smaller batch sizes were more optimal. The findings of this project demonstrate the effectiveness of mini-batch gradient descent in optimizing the parameters of a machine learning model.

II. Introduction

The objective of this project was to compare the performance of linear and logistic regression models using two different optimization algorithms: an analytical approach for the linear regression model and the gradient descent optimization algorithm for both linear and logistic regression. The logistic regression model was implemented to predict the class label of a given input data. Two benchmark datasets were used in this study, the energy efficiency dataset for linear regression and the qualitative bankruptcy dataset for logistic regression. In terms of optimization algorithms, mini-batch gradient descent was found to be superior to batch gradient descent and the analytical methods in terms of convergence rate and overall loss. The effect of batch size on the performance of mini-batch gradient descent was also analyzed and it was found that an optimal batch size of 8 was most effective for linear regression, versus a size of 32 for logistic regression. The findings suggest that as the batch size increases, the performance and behavior of mini-batch gradient descent shift towards that of batch gradient descent, while for smaller batch sizes its behavior is similar to that of stochastic gradient descent. Furthermore, the effects of other hyperparameters such as the training set size and the training rate were also analyzed and discussed in the results section. The findings of this project demonstrate the effectiveness of mini-batch gradient descent in optimizing the parameters of a machine learning model, which can be of great significance in real-world applications, or so we hope.

III. Data preprocessing

The first dataset is the energy efficiency dataset, which consists of 8 features and 2 target variables, ”Heating Load” and ”Cooling Load.” The dataset was first loaded into a pandas dataframe and processed to remove missing or malformed features. The shape of the data was then determined and a summary of the data was obtained by calling df1.describe().T which gives a summary of each feature's distribution. The data was then visualized by creating scatter plots of each feature against the two target variables. The correlation between features was also calculated and showed that there were high correlations between some features. A high positive correlation (values close to 1) indicates a strong positive relationship between two features, while a high negative correlation (values close to -1) indicates a strong negative relationship. Features with low correlation (values close to 0) are considered to have little to no relationship. These relationships are important to consider when performing analyses and building models, as highly correlated features may suggest that they have similar information and that one of them might be redundant. To prepare the data for modeling, the data was normalized by dividing each feature by its L2 norm to ensure that all features are on the same scale, so that they can contribute equally to the results of the analyses performed. This way, all features are transformed to have the same magnitude and range, making it easier to compare the relationships between them.

Dataset 2 is a qualitative bankruptcy dataset that includes 249 samples with 7 attributes. The 7 attributes are: Industrial Risk, Management Risk, Financial Flexibility, Credibility, Competitiveness, Operating Risk and Class. Each attribute is described by one of three values: P (Positive), A (Average) or N (Negative). The Class attribute is described by two values: B (Bankruptcy) or NB (Non-Bankruptcy). The first step in processing the data was to remove any missing or malformed features by checking for '?' values in the data and removing the rows with these values. Next, the class distribution was analyzed by counting the number of instances for each class (NB and B). The class distribution was found to be 142 instances for class NB and 107 instances for class B. The data was then preprocessed by replacing values in the data with numerical values. The values ”P” and ”N” were replaced with 1 and 0 respectively and ”A” was replaced with 0.5. The values ”B” and ”NB” were replaced with 1 and 0 respectively. The data was then described to give more insights into the values and ranges of the features. Significantly, When working with this dataset, we think it is crucial to consider ethical concerns such as the protection of confidential information and the privacy of individuals and companies involved. Moreover, we try to thoroughly evaluate the accuracy and reliability of the data in order to ensure that it is not biased or misrepresented in any way. To do this, we carefully reviewed the parameters used for collecting the dataset as described in [3].

IV. Report the performance of linear regression and fully batched logistic regression. For both datasets use a 80/20 train/test split and report the performance on both training set and test set.

For the training set of the batched linear regression on the first dataset with a learning rate of 0.1, the initial value is 2.2154e-3 at t = 66 and the final value is 1.432e-3 at t = 990. The mean square error is 5.912e-4. For the test set, the initial value of 2.215e-3 at t = 66 and a final value of 1.431e-3 at t = 990, with a mean square error of 7.121e-4. As expected, the batched linear regression performs better on the training set, since it converges faster and it also has a smaller error. The performance does not decrease significantly for the test set: the error only increased by around 2e-4. Therefore, our model and the parameters chosen perform adequately for the batch linear regression.

For the training set of batched gradient descent on logistic regression on the second dataset with a learning rate of 0.01, we have found an initial value of 0.3813 at t = 10 and a final value of 0.2402 at t = 140, with a cross entropy loss of 0.3277. For the test set, we have found an initial value of 0.3813 at t = 10 and a final value of 0.2402 at t = 140, with a cross entropy loss of 0.3288. Just like for the batched set of linear regression, the training set performs slightly better than the test set. However, it is interesting to note that the variance of the output for logistic regression is smaller, and that the cross entropy errors are also close in value.

We also computed the values for batched linear regression on the second dataset, with the same learning rate as the first dataset, 0.1. For the training set, our initial value was 0.08133 at t = 10 and the final value was 0.01918 at t = 150, with a mean square error of 0.2905. Compared to the first dataset, this batched linear regression model performs more poorly, but is still better than the logistic regression model. Since the inputs for the two datasets are different, we could try to slightly adjust the learning rate to get better results.

Overall, when comparing the test performance for batched linear regression vs batched gradient descent, the linear regression model performs better, as it has a smaller error and it converges quicker to 0, but the logistic regression model has less variance between the training set and the test set.

V. Report the weights of each of features in your trained models and discuss how each feature could affect the performance of the models.

For Dataset 1, we have 8 features in total: Relative Compactness (X1), Surface Area (X2), Wall Area (X3), Roof Area (X4), Overall Height (X5), Orientation (X6), Glazing Area (X7), and Glazing Area Distribution (X8). In the linear regression mini-batch gradient descent model, the weights of each features for the first target variable Heating Load (Y1) are 0.213, -0.042, 0.243, -0.293, 0.600, 0.018, 0.197, 0.028. The weights of each features for the second target variable Cooling Road (Y2) are 0.222, -0.042, 0.243, -0.293, 0.600, 0.018, 0.197, and 0.028. While observing the weights on linear regression with mini-batch gradient descent, we noticed that features X1 (relative compactness), X3 (Wall Area), X4 (Roof Area), X5 (overall height), and X7 (the glazing area) have relatively higher impact on the performance of the model, while the others have relatively lower influences.

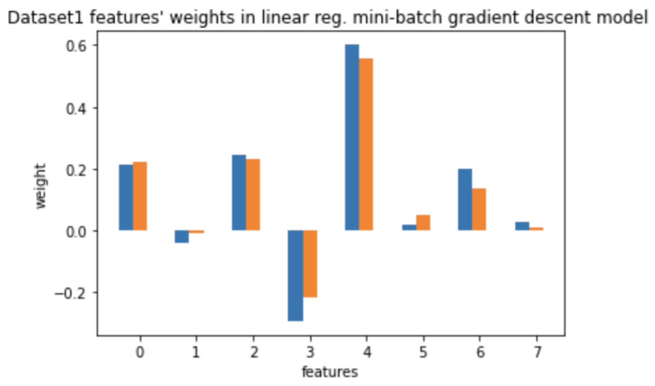

Fig. 1. Dataset 1 feature’s weights for both target variable Heating Load(blue) and Cooling Road(Orange) in linear regression mini-batch gradient descent model

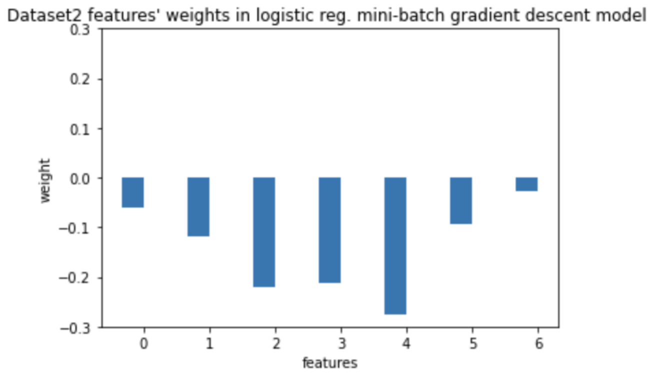

Fig. 2. Dataset 2 feature’s weight in logistic regression mini-batch gradient descent model)

For Dataset 2, we have an overall of 7 features, Industrial Risk, Management Risk, Financial Flexibility, Credibility, Competitiveness, and the Operating Risk, and Class. In the logistic regression gradient descent model, the weights of each attributes are -0.06, -0.117, -0.219, -0.213, -0.276, -0.093, and -0.026.For logistic regression model, we are getting negatives weights. We observed that Financial Flexibility, Credibility and Competitiveness has slightly higher impact on the performance compare to the other attributes.

Overall, the interpretation of weights creates lower influence on logistic regression models, compare to its influence on linear regression models.

VI. Sample growing subsets of the training data (20%,30%,...80%). Observe and explain how does size of training data affects the performance for both models. Plot two curves as a function of training size, one for performance in train and one for test.

By observing the sample growing subsets of the training data, we found out that the size of training data affects the performance for both models.

For both linear regression and logistic regression model, increasing the training data size leads to a better performance in train with smaller mean square errors and fast convergence, however, while the testing data size decreases, performance in test declines with larger mean square errors and slow convergence.

For example, for linear regression with mini-batch gradient descent, as shown on the following figure(Fig.3), with training data sample setting to 20% and testing sample to 80%, we are getting a mean square error of 8.926e-05 with an initial gradient of 1.361e-3 at t=66 and an final gradient of 1.581e-4 at t=990 in train, and a mean square error of 3.180e-05 with an initial gradient of 2.394e-4 at t=66 and an final gradient of 4.619e-5 at t=990 in test. With training data sample setting to 40% and testing data sample to 60% , we get a mean square error of 5.515e-05 with an initial gradient of 4.818e-4 at t=66 and an final gradient of 1.021e-4 at t=990 in train, and a mean square error of 3.699e-05 with an initial gradient of 2.783e-4 at t=66 and an final gradient of 7.574e-5 at t=990 in test. With training data sample setting to 60% and testing data sample to 40%, we get a mean square error of 3.749e-05 with an initial gradient of 2.703e-4 at t=66 and an final gradient of 6.780e-5 at t=990 in train, and a mean square error of 4.939e-05 with an initial gradient of 4.858e-4 at t=66 and an final gradient of 1.070e-4 at t=990 in test. Lastly, with training data sample setting to 80% and testing data sample setting to 20%, we get a mean square error of 3.020e-05 with an initial gradient of 2.359e-4 at t=66 and an final gradient of 4.561e-5 at t=990 in train, and a mean square error of 8.383e-05 with an initial gradient of 1.366e-3 at t=66 and an final gradient of 1.366e-4 at t=990 in test.

For logistic regression with mini-batch gradient descent, as shown on the next figure (Fig.4), with training data sample setting to 20% and testing sample to 80%, we are getting a cross entropy loss of 0.373 with an initial gradient of 0.325 at t=10 and an final gradient of 0.272 at t=140 in train, and a cross entropy loss of 0.325 with an initial gradient of 0.386 at t=10 and an final gradient of 0.240 at t=140 in test. With training data sample setting to 40% and testing data sample to 60% , we get a cross entropy loss of 0.363 with an initial gradient of 0.368 at t=10 and an final gradient of 0.288 at t=140 in train, and a cross entropy loss of 0.328 with an initial gradient of 0.379 at t=10 and an final gradient of 0.240 at t=140 in test. With training data sample setting to 60% and testing data sample to 40%, we get a cross entropy loss of 0.3375 with an initial gradient of 0.363 at t=10 and an final gradient of 0.245 at t=140 in train, and a cross entropy loss of 0.354 with an initial gradient of 0.398 at t=10 and an final gradient of 0.298 at t=140 in test. Lastly, with training data sample setting to 80% and testing data sample setting to 20%, we get a cross entropy loss of 0.334 with an initial gradient of 0.380 at t=10 and an final gradient of 0.241 at t=140 in train, and a mean square error of 0.366 with an initial gradient of 0.344 at t=10 and an final gradient of 0.274 at t=140 in test.

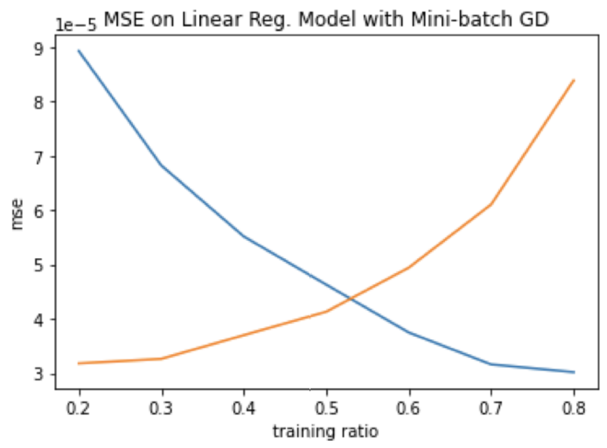

Fig. 3. Performance of linear regression with mini-batch GD model with two curves as function of training size, one for performance in train (blue) and one for performance in test (orange).

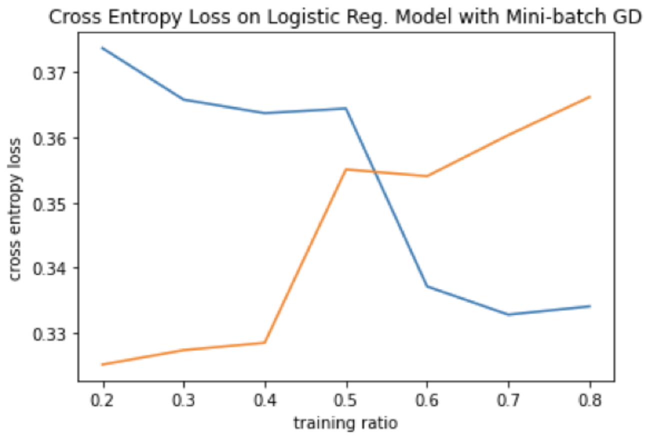

Fig. 4. Performance of logistic regression with mini-batch GD model with two curves as function of training size, one for performance in train (blue) and one for performance in test (orange).

VII. For both linear and logistic regression, try out growing minibatch sizes, e.g., 8, 16, 32, 64, and 128. Compare the convergence speed and final performance of different batch sizes to the fully batched baseline. Which configuration works the best among the ones you tried?

For linear regression, when increasing the number of samples, the gradient will start with a smaller value and will converge faster to 0. For example, a batch size of 8 will have an initial gradient of 2.357e-3 at t = 66 and end up with one of 4.5802e-5 at t = 990, whereas a batch size of 128 will have an original gradient of 3.394e-3 and a final gradient of 2.364e-4 for the same timestamps. If we compare the mean square error, we also notice that the smaller batch size have a smaller value: 3.2006e-5 for size = 8, compared to 1.4886e-4 for size = 128. We can therefore deduce that for linear regression, the smaller sample converges faster and has a better fit to the function.

For logistic regression, as the batch size increases, the gradient will also start with a bigger value and converge slower to 0. When comparing the cross entropy loss, a batch size of 32 had the lowest lost at 0.2983 and the batch size of 128 had the highest loss at 0.3346. Therefore, a batch size of 32 allows to find the best function to separate the data.

VIII. Present the performance of both linear and logistic regression with at least three different learning rates (your own choice).

When using batch gradient descent for linear regression in dataset 1, the gradient values tend converge faster when the learning rate is decreased. However, if the rate is set too high, the values may not converge. When testing out the rates of 0.01, 0.1 and 10, the learning rate of 10 made the values slightly converge towards 1 and then start diverging. Out of the 3 rates, alpha = 0.1 converged the fastest and also had the smallest mean square error. The rate of 0.01 also converged, but at a slower speed than when using the learning rate of 0.1.

For dataset 2, using logistic regression, we tested the rates 0.01, 10 and 100. We found that by increasing alpha, the gradient value at the first timestamp would be smaller and the function would converge faster towards 0. Out of the three learning rates, the rate of 100 converged the fastest, but had a slightly higher cross entropy loss than that of 10. None of the other two rates of 0.01 and 10 seemed to diverge, but they converted slower than that of 100.

When trying out a few range of learning rates, we noticed that batch linear regression seems more sensitive to a bigger learning rate, as the values would diverge quickly if the rate is too high. This does not seem to be the case for logistic regression, when tested with the same rates. On the contrary, smaller rates only seemed to affect the convergence speed to the final result. The errors for both linear regression and logistical regression do not seem to be greatly impacted by smaller rates.

IX. Compare analytical linear regression solution with mini-batch stochastic gradient descent based linear regression solution. What do you find?

While comparing analytical linear regression solution with mini-batch stochastic gradient descent based linear regression solution, we noticed that analytical linear regression converges faster than mini-batch stochastic gradient descent based linear regression with fewer number of features, and slower when the number of features increases. For example, while comparing the mean square errors by setting the batch number to 8, and the learning rate to 0.1, we get an mean square error of 6.003e-5 for mini-batch SGD. However, we get a smaller MSE for analytical linear regression model, which is 1.648e-05.

By looking at the formulas, the Analytical linear regression solution has unique solution, no iteration, and no requirement on data normalization. Nevertheless, with higher number of features, its computation complexity increases. While on the other hand, mini-batch gradient descent based linear regression solution calculates errors by splitting the dataset into small batches, thus works better with multidimensional input as well as with huge feature number.

X. Discussion and Conclusion

In conclusion, we implemented the models of linear regression and logistic regression, as well as some of their variations, like analytical linear regression, mini-batch and stochastic models. We have been able to observe their performance on the training dataset and compare it with that of the test dataset. We noticed that our gradient values and errors were similar for both the training and test dataset, which implies that our models can produce accurate outputs and that their performance is adequate. We have also been able to see the effect of different features, such as weight, the number of samples in the subset and learning rate. We concluded that some features, like relative compactness, wall area, roof area, overall height and glazing area affect the performance of the models more than the other parameters. By varying the weights, we have noticed that the weights have a higher influence on linear regression models, as opposed to logistic regression models. The two types of models would also perform better when given a larger subset for training data. Furthermore, when trying to determine the effect on the number of samples, we have found that linear regression models would converge faster and have a smaller cross entropy loss when using smaller mini batches. Logistic regression also tended to perform better with smaller batches, but a size of 32 had a smaller loss than the batches of size 8 and 16. When it came to learning rate, a learning rate of 0.1 was the best for linear regression, and for logistic regression, a larger rate of 100 gave the best performance. It was also noted that linear regression performance is more sensitive to bigger learning rates and would diverge if it is too large. Finally, we compared different variations of the same model, using fully batched and mini-batch models as well as analytical and stochastic linear regression. The analytical linear regression with a small number features would converge faster than the stochastic gradient descent linear regression. It is also simpler, as it has a unique solution and requires no iteration nor data normalization. However, when working with multidimensional input and a large number of features, it would be preferable to use mini-batch gradient descent. By observing the effect of using different variations of the linear and logistic regression models, we have been able to figure out the best parameters to use when building such models and observe patterns when increasing or decreasing those parameters. For future experiments, it would be interesting to try to implement the gradient descent with momentum, so that the oscillations between the values could be reduced. Moreover, we could try to implement more variations of the linear regression and logistic regression models, like Adam, RMSprop and Adagrad to compare their performance against the original models, and figure out how they could best be used.

XI. Statement of Contributions

Simon wrote the main parts of the code, and the abstract, introduction and data preprocessing in the report.

Yue Xin did some minor debugging and answered half the questions in the report.

Keyu also did some minor debugging and answered the other half of the questions in the report.

References

[1] D. Dua and C. Graff, "UCI Machine Learning Repository," University of California, Irvine, School of Information and Computer Sciences, 2017. [Online]. Available: http://archive.ics.uci.edu/ml.

[2] M. Kim and I. Han, "The discovery of experts' decision rules from qualitative bankruptcy data using genetic algorithms," Expert Systems with Applications, vol. 25, no. 4, pp. 637-646, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417403001027.

[3] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[4] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[5] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.