2/13/2020

Applied Machine Learning

Assignment 1

Rupal Shrivastava
RXS190017

Table of Contents

Introduction

In this assignment, we will implement linear and logistic regression on the GPU runtime dataset. We are trying to implement gradient descent algorithm with batch update to predict GPU computation time. The prediction is done on the basis of different combination of the processor configurations.

Dataset Description

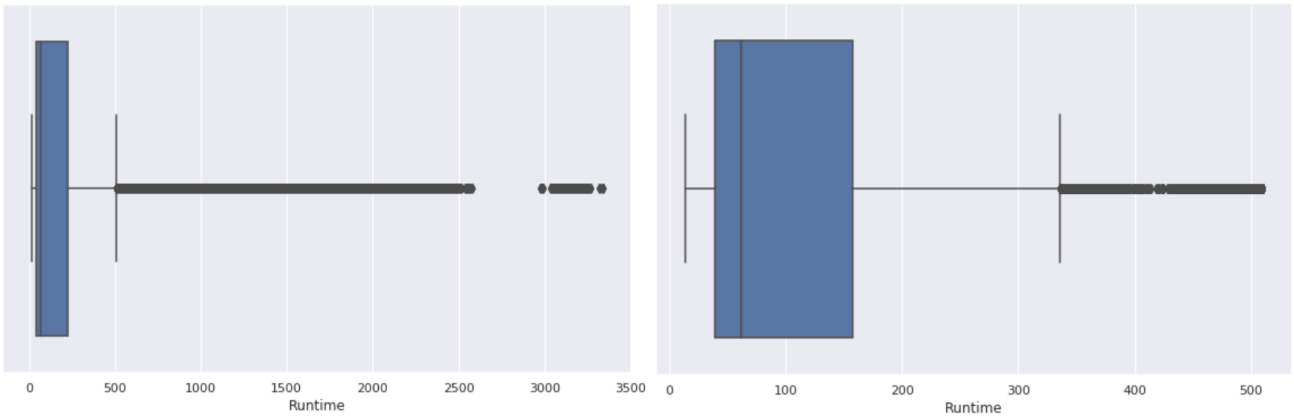
The dataset used for this project can be found at:
https://archive.ics.uci.edu/ml/datasets/SGEMM+GPU+kernel+performance
The dataset consists of 14 features (independent variables) and 241,600 rows. First 10 features are ordinal and last 4 as binary variables. There were 4 runs performed on this dataset, which corresponds to 4 runtime variables in the dataset. More details on the dataset can be found in the link mentioned above.

The assignment is divided in three parts: Data Preprocessing, Algorithms and Experiments.

Data Preprocessing

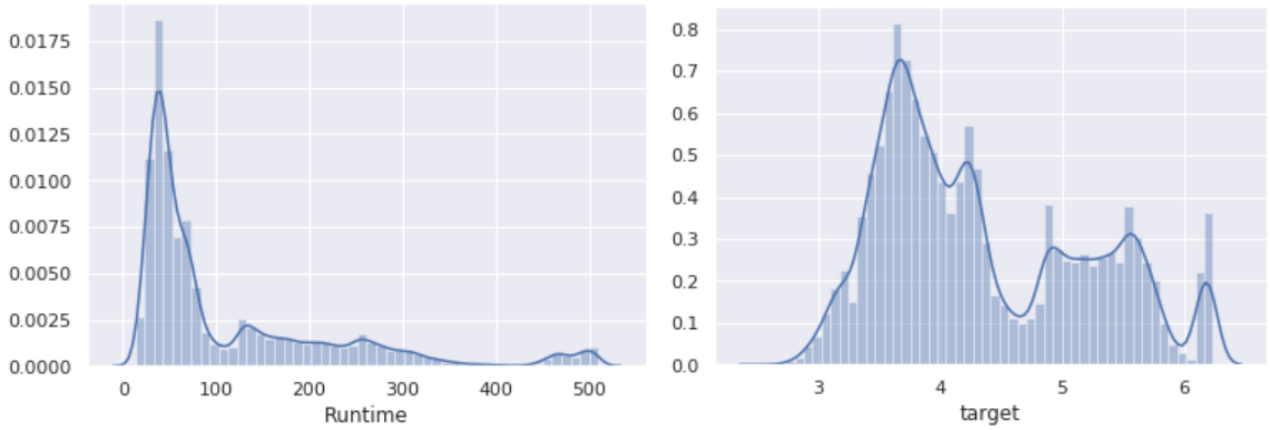
Since the data has 4 variables containing the GPU runtime, a single runtime variable was created by taking average of these variables. Then we check for the following:

1. *Null Values*: The data has no null datapoints
2. *Outliers*: The average runtime variable range in (13.3175, 3341.5075) with 3rd Quartile at 228.3875. Clearly, the data has outliers. These were then removed using interquartile range calculation, removing approximately 10% of the data. Here is the target variable distribution before and after removing the outliers.



Before and after outlier removal

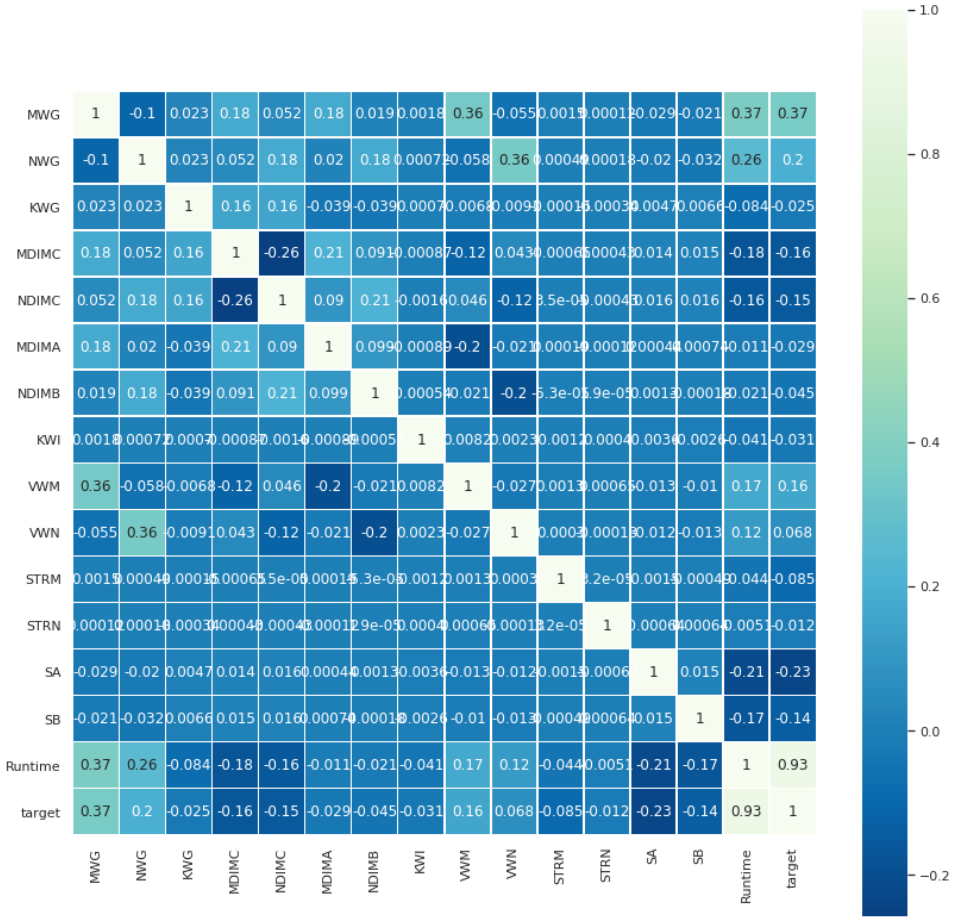
3. *Checking Target variable distribution*: The average runtime distribution was left skewed. So, taking the log of the average runtime which normalize the data distribution, which will be used as the target in all the processing ahead. This can be seen by the plots below.



GPU runtime before and after taking log

4. *Feature Scaling*: For the purpose of making bringing all the feature on the same scale, we perform feature scaling on all the 14 features. Features are scaled using: $$z = \frac{x - \mu}{\sigma}$$

   This also helps gradient decent to quickly reach the minimum cost.
5. *Correlation Matrix*: Checked the correlation among features. None of the features are highly correlated with each other, as can be seen by the heatmap below.



6. *Intercept*: Created a dummy variable with all 1s to ease the intercept calculation while running the model.

Algorithms

For all the algorithms, we have used a train/test split of 80/20 percent. The coefficient matrix of size equal to the dataset initiated with zero for each algorithm.

Linear Regression

All the algorithms are broken into different functions for: cost calculation, gradient decent, predicting target variable, calculating root mean squared error (RMSE) and one main Linear Regression function to perform all this on the dataset passed.

$$\text{Average Runtime} = \beta_0 + \beta_1 * \text{MWG} + \beta_2 * \text{NWG} + \beta_3 * \text{KWG} + \beta_4 * \text{MDIMC} + \beta_5 * \text{NDIMC} + \beta_6 * \text{MDIMA} + \beta_7 * \text{NDIMB} + \beta_8 * \text{KWI} + \beta_9 * \text{VWM} + \beta_{10} * \text{VWN} + \beta_{11} * \text{STRM} + \beta_{12} * \text{STRN} + \beta_{13} * \text{SA} + \beta_{14} * \text{SB}$$

Logistic Regression

Converted the target variable in binary by using the mean as the threshold, that is, it the target is greater than mean then the value is 1 else 0. All the algorithms are broken into different functions for: cost calculation, gradient decent, predicting target variable, calculating accuracy and one main Logistic Regression function to perform all this on the dataset passed.

Experiments

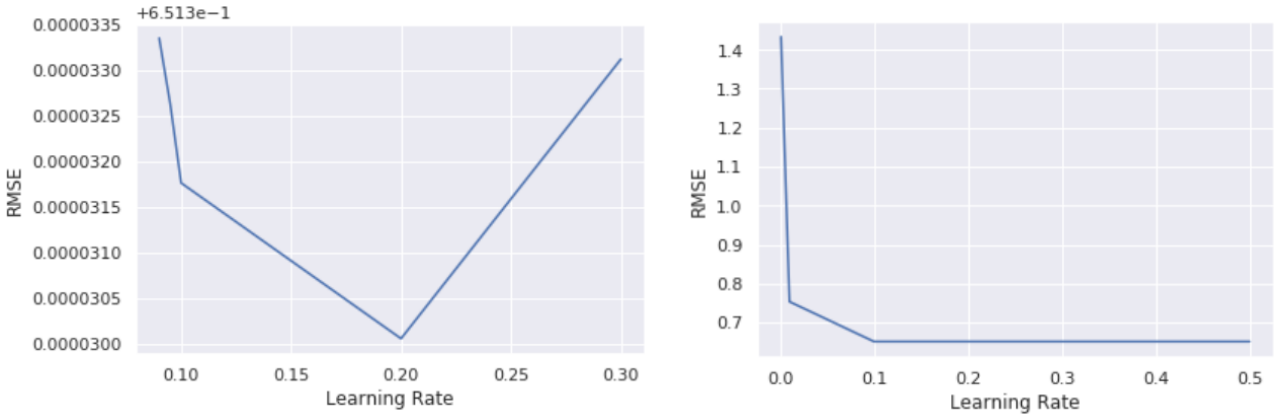
Experiment 1

Linear Regression

Experimenting with various values of learning rate, to find the optimum value with minimum RMSE.

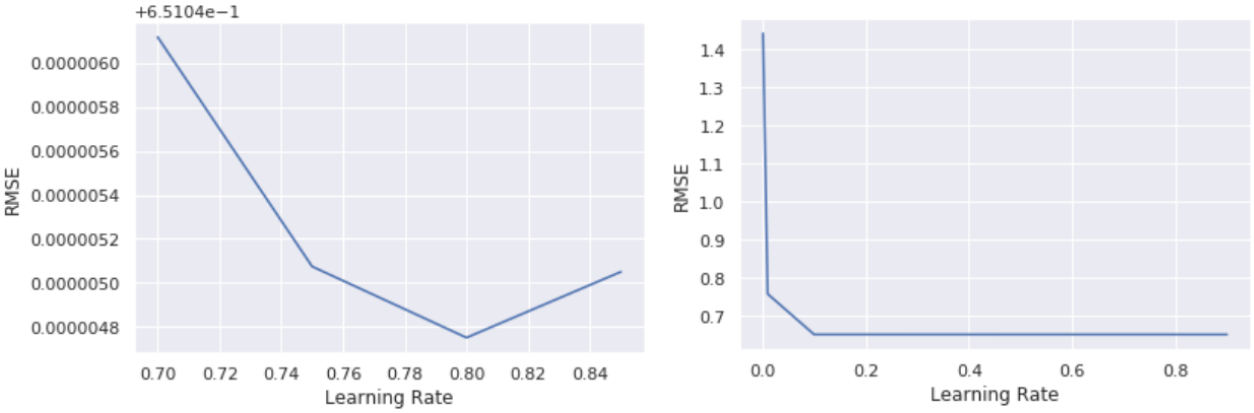
Using Train and Test sets

As it can be seen from the graph on the left, we get minimum RMSE = 0.6513300 at learning rate=0.2. The optimum learning rate is small to quickly reach the minimum cost in gradient decent without missing the global minima.



Within training dataset

As it can be seen from the graph on the left, we get minimum RMSE = 0.6510447 at learning rate=0.8. The optimum learning rate is so close to 1, as we are calculating RMSE using the same dataset what we trained the model on. This is the same reason why the RMSE is lower in this case than train and test sets.

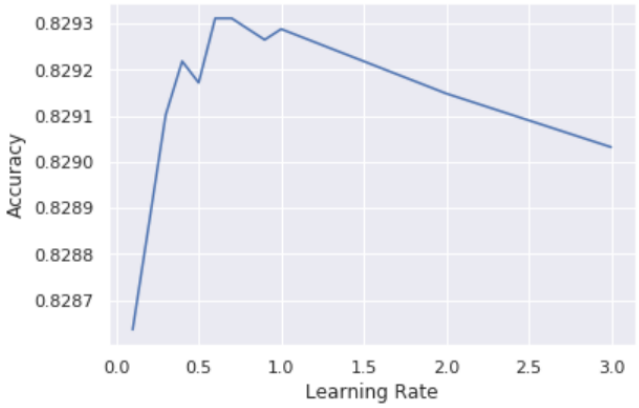


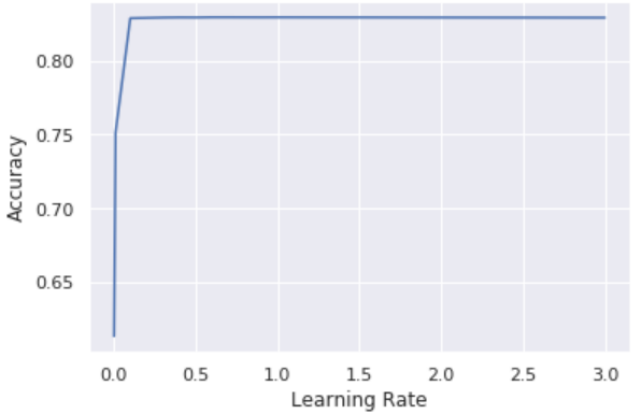
Logistic Regression

Experimenting with various values of learning rate, to find the optimum value with maximum accuracy.

Using Train and Test sets

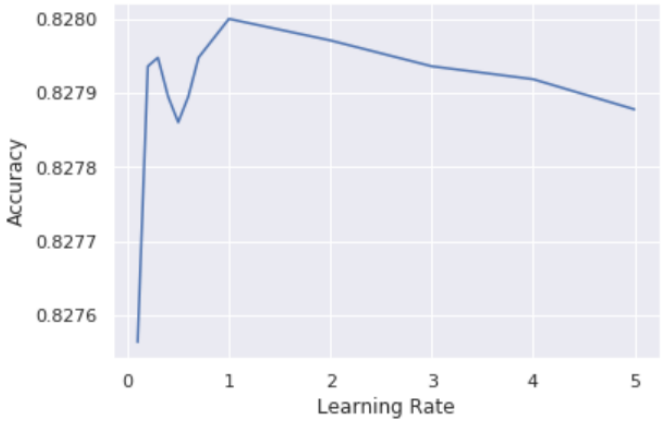
As it can be seen from the graph on the left, we get maximum accuracy = 0.8293108 at learning rate=0.6. After reaching the optimum accuracy, the accuracy does not change much as it can be seen on the graph on the right.

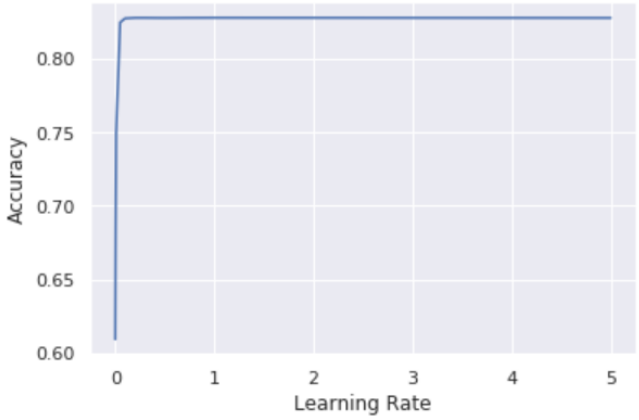




Within training dataset

As it can be seen from the graph on the left below, we get maximum accuracy = 0.8279997 at learning rate=1. The optimum learning rate is 1 as we are using the same dataset to train the model.





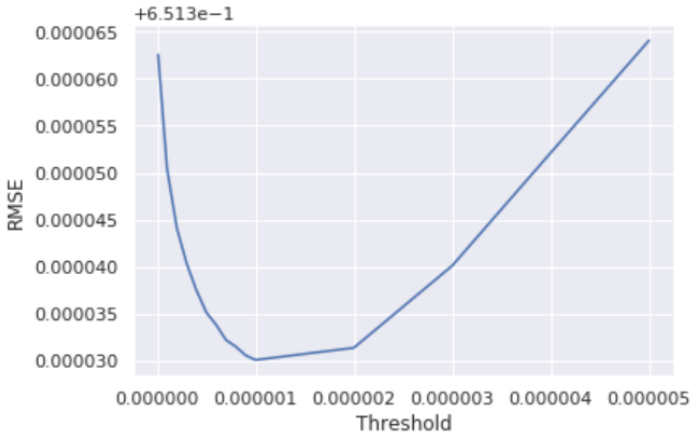
Experiment 2

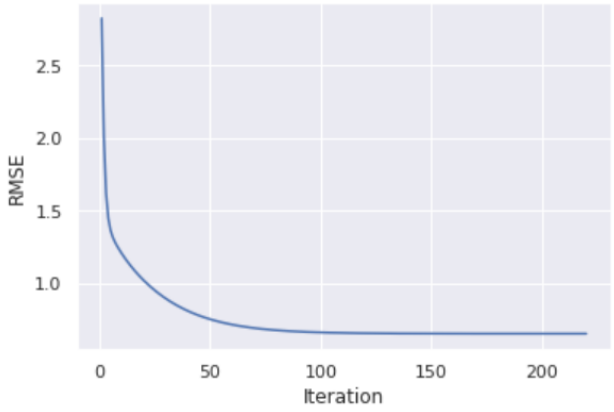
Linear Regression

Picking the best value of learning rate, experimenting with various threshold for convergence, to find minimum RMSE.

Using Test and Train sets

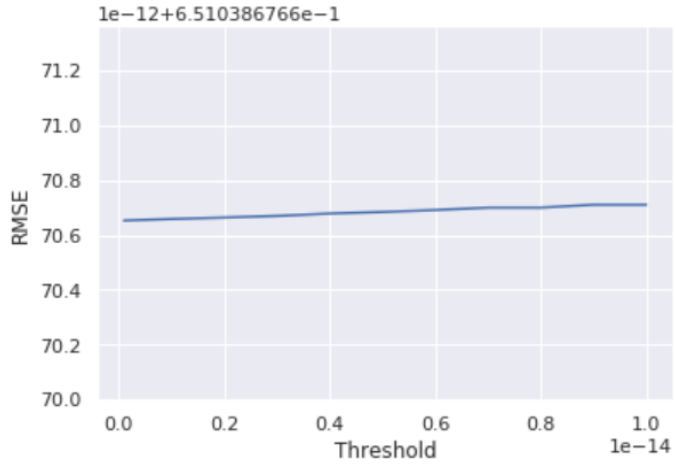
For learning rate = 0.2, we are getting the minimum RSME = 0.6513300 at threshold = 10^{-6} as can be seen by the graph on the left. On the right, graph show RMSE as a function of number of iterations for best values of learning rate and threshold, that is learning rate = 0.2 and threshold = 10^{-6}. We can also interpret that increasing the number of iterations will not reduce RMSE on optimum values of learning rate and threshold.

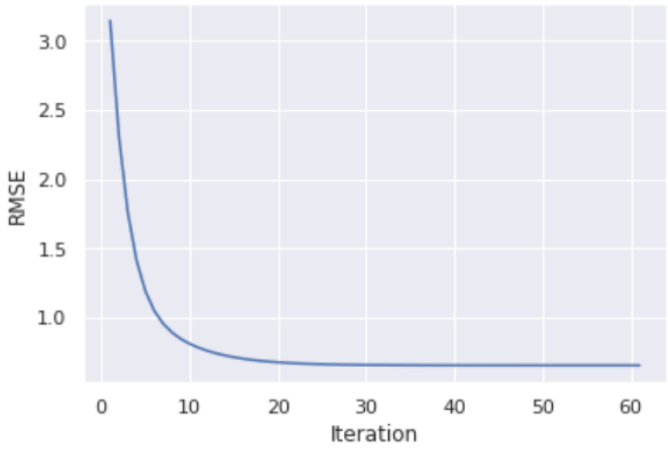




Within Training set

For learning rate = 0.8, the RMSE values keeps on reducing as the convergence threshold tend to reach 0. This can be seen by the graph on the left, where the threshold is as low as 10^{-14}. Since optimum threshold is at 0, we are using the threshold used in train and test set as the optimum threshold. On the right, graph show RMSE as a function of number of iterations for best values of learning rate and threshold, that is learning rate = 0.8 and threshold = 10^{-6}. We can also interpret that increasing the number of iterations will not reduce RMSE on optimum values of learning rate and threshold.



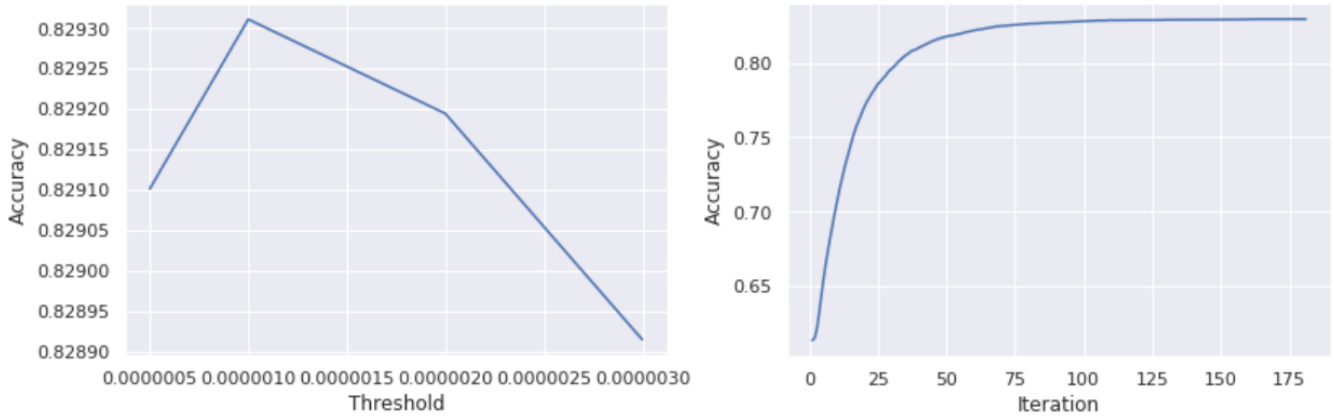


Logistic Regression

Picking the best value of learning rate, experimenting with various threshold for convergence, to find maximum accuracy.

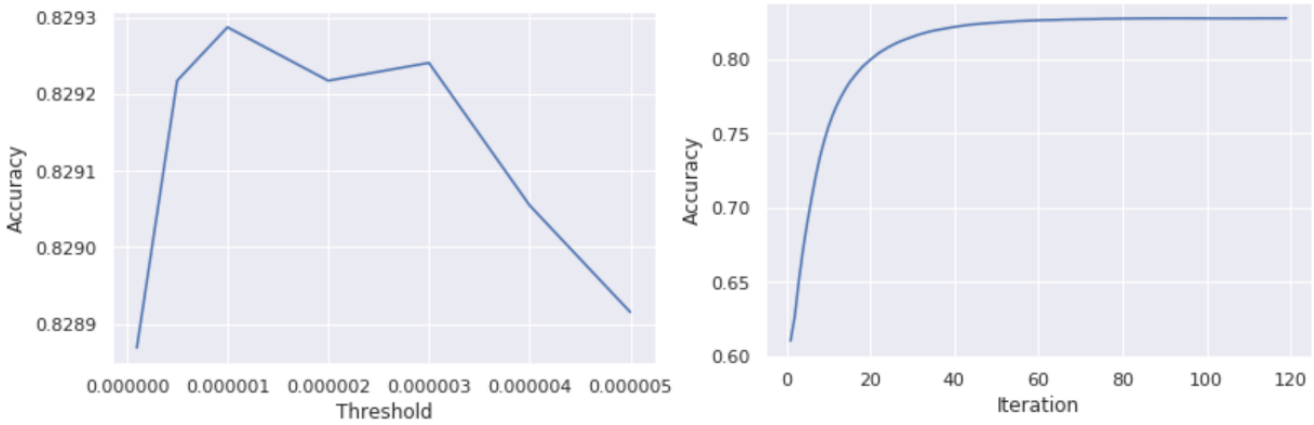
Using Test and Train sets

For learning rate = 0.6, we are getting the maximum accuracy = 0.8293108 at threshold = 10^{-6} as can be seen by the graph on the left. On the right, graph show accuracy as a function of number of iterations for best values of learning rate and threshold, that is, learning rate = 0.6 and threshold = 10^{-6}. We can also interpret that increasing the number of iterations will not increase accuracy on optimum values of learning rate and threshold.



Within Training set

For learning rate = 1, we are getting the maximum accuracy = 0.8292875 at threshold = 10^{-6} as can be seen by the graph on the left. On the right, graph show accuracy as a function of number of iterations for best values of learning rate and threshold, that is, learning rate = 1 and threshold = 10^{-6}. We can also interpret that increasing the number of iterations will not increase accuracy on optimum values of learning rate and threshold.



Experiment 3

Picking 8 random features and retraining the model, at optimal value of learning rate and convergence threshold. Then comparing the results with the model with all 14 features to check model performance.

Linear Regression

Using Train and Test set

Randomly selected features: 'VWM', 'NDIMB', 'NDIMC', 'STRN', 'KWG', 'SB', 'VWN', 'NWG'

Optimal learning rate = 0.2, threshold = 10^{-6}

RMSE for 8 features: 0.7914725 and RMSE for 14 features: 0.6513300

RMSE is increasing when selecting 8 random features, showing that having more features improves the model.

Within Training set

Randomly selected features: 'VWM', 'NDIMB', 'NDIMC', 'STRN', 'KWG', 'SB', 'VWN', 'NWG'

Optimal learning rate = 0.8, threshold = 10^{-6}

RMSE for 8 features: 0.7937721 and RMSE for 14 features: 0.6510447

RMSE is increasing when selecting 8 random features, even with the same training data, showing that having more features improves the model.

Logistic Regression

Using Train and Test set

Randomly selected features: 'VWM', 'NDIMB', 'NDIMC', 'STRN', 'KWG', 'SB', 'VWN', 'NWG'

Optimal learning rate = 0.6, threshold = 10^{-6}

Accuracy for 8 features: 0.6655107 and accuracy for 14 features: = 0.8293108

Accuracy is decreasing when selecting 8 random features, showing that having more features improves the model.

Within Training set

Randomly selected features: 'VWM', 'NDIMB', 'NDIMC', 'STRN', 'KWG', 'SB', 'VWN', 'NWG'

Optimal learning rate = 1, threshold = 10^{-6}

Accuracy for 8 features: 0.6617306 and accuracy for 14 features: 0.8292875

Accuracy is decreasing when selecting 8 random features, even with the same training data, showing that having more features improves the model.

Experiment 4

Linear Regression

Picking 8 relevant features and retraining the model, at optimal value of learning rate and convergence threshold. Then comparing the results with the model with all 14 features to check model performance.

Using Train and Test set

Selected features: 'MWG', 'NWG', 'KWG', 'MDIMC', 'NDIMC', 'STRM', 'SA', 'SB'

Optimal learning rate = 0.2, threshold = 10^{-6}

RMSE for 8 features: 0.6541226 and RMSE for 14 features: 0.6513300

RMSE is way better in this case as compared to random selection, because we are selecting most relevant features that affect the GPU runtime. RMSE is almost equivalent when selecting 8 relevant features, showing that having more features would not improve the model, if the features are not relevant. This means the remaining 6 features do not affect the GPU runtime.

Within Training set

Selected features: 'MWG', 'NWG', 'KWG', 'MDIMC', 'NDIMC', 'STRM', 'SA', 'SB'

Optimal learning rate = 0.8, threshold = 10^{-6}

RMSE for 8 features: 0.6541306 and RMSE for 14 features: 0.6510447

RMSE is way better in this case as compared to random selection, because we are selecting most relevant features that affect the GPU runtime. RMSE is almost equivalent when selecting 8 relevant features, showing that having more features would not improve the model, if the features are not relevant. This means the remaining 6 features do not affect the GPU runtime.

Logistic Regression

Using Train and Test set

Selected features: 'MWG', 'NWG', 'KWG', 'MDIMC', 'NDIMC', 'STRM', 'SA', 'SB'

Optimal learning rate = 0.6, threshold = 10^{-6}

Accuracy for 8 features: 0.8364093 and accuracy for 14 features: = 0.8293108

Accuracy is way better in this case as compared to random selection, because we are selecting most relevant features that affect the GPU runtime. Accuracy is almost equivalent when selecting 8 relevant features, showing that having more features would not improve the model, if the features are not relevant. This means the remaining 6 features do not affect the GPU runtime.

Within Training set

Selected features: 'MWG', 'NWG', 'KWG', 'MDIMC', 'NDIMC', 'STRM', 'SA', 'SB'

Optimal learning rate = 1, threshold = 10^{-6}

Accuracy for 8 features: 0.8368748 and accuracy for 14 features: 0.8292875

Accuracy is way better in this case as compared to random selection, because we are selecting most relevant features that affect the GPU runtime. Accuracy is almost equivalent when selecting 8 relevant features, showing that having more features would not improve the model, if the features are not relevant. This means the remaining 6 features do not affect the GPU runtime.

Conclusion

The best model depends on a lot of factors like optimum learning rate and threshold. The model accuracy also depends on the selecting the most relevant features for the target variable. Here, GPU runtime majorly depends on 'MWG', 'NWG', 'KWG', 'MDIMC', 'NDIMC', 'STRM', 'SA', 'SB' features.

Citation

- Rafael Ballester-Ripoll, Enrique G. Paredes, Renato Pajarola.
Sobol Tensor Trains for Global Sensitivity Analysis.
In arXiv Computer Science / Numerical Analysis e-prints, 2017 ([Web Link])

- Cedric Nugteren and Valeriu Codreanu.
CLTune: A Generic Auto-Tuner for OpenCL Kernels.
In: MCSoC: 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip. IEEE, 2015 ([Web Link])