Prepared by : Kamini Bokefode

Date : 02/13/2020

BUAN 6341 Applied Machine Learning

ASSIGNMENT 1 GPU Run time prediction

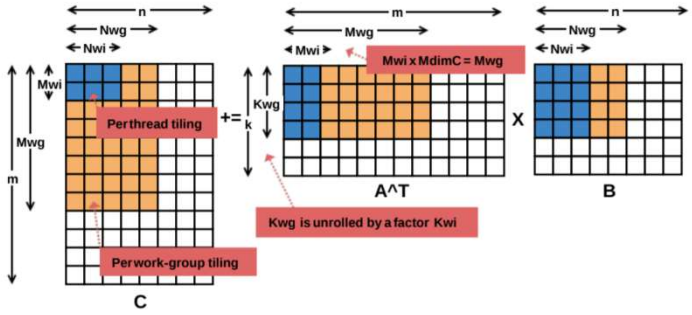
Goal

Implemented a linear and logistic regression model on the dataset to predict the GPU run time. Used the average of four runs as the target variable .Implemented the gradient descent algorithm with batch update (all training examples used at once). Use the sum of squared error normalized by 2*number of samples [J(β0, β1) = (1/2m)[∑(y^(i) – y(i))2] as cost and error measures, where m is number of samples, used all 14 features.

About the Data:

This data set measures the running time of a matrix-matrix product A*B = C, where all matrices have size 2048 x 2048, using a parameterizable SGEMM GPU kernel with 241600 possible parameter combinations. For each tested combination, 4 runs were performed, and their results are reported as the 4 last columns. All times are measured in milliseconds*. There are 14 parameters, the first 10 are ordinal and can only take up to 4 different powers of two values, and the 4 last variables are binary.

Following image below can be referred to get more information on matrices A,B,C and the parameters that define different tiling information.




```
// Parameters determined by the tuner
// MWG      : Tile-size in dimension M (e.g. 64, 128)
// NWG      : Tile-size in dimension N (e.g. 64, 128)
// KWG      : Tile-size in dimension K (e.g. 8, 16)
// MDIMC    : Threads per workgroup in M-dimension (e.g. 8, 16, 32)
// NDIMC    : Threads per workgroup in N-dimension (e.g. 8, 16, 32)
// MDIMA    : Re-shaped tile dimension of matrix A: KDIMA * MDIMA
// NDIMB    : Re-shaped tile dimension of matrix B: KDIMB * NDIMB
// KWI      : Unroll factor of the KWG loop (smaller or equal than KWG)
// VWM      : Vector width of matrices A and C (supported 1, 2, 4, and 8)
// VWN      : Vector width of matrix B (supported 1, 2, 4, and 8)
// STRM     : Use strided access within a thread in the M-dimension (1) or not (0)
// STRN     : Use strided access within a thread in the N-dimension (1) or not (0)
// SA       : Use local/shared memory to cache matrix A (1) or not (0)
// SB       : Use local/shared memory to cache matrix B (1) or not (0)
// PRECISION : Whether to use single (32) or double (64) precision data-types
```

The data consists of various combinations of tuning parameters which are dimensions of thread block, other parameters like tiling or unrolling factor and the resulting run time.

Algorithm Implementation

Gradient descent is a first-order iterative optimization algorithm for finding the local minimum of a differentiable function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point.

The algorithm is implemented for two different machine learning techniques, namely, Linear regression and Logistic regression. The algorithm is implemented with options to change the hyperparameters: Learning rate, convergence threshold, random restarts and max epoch.

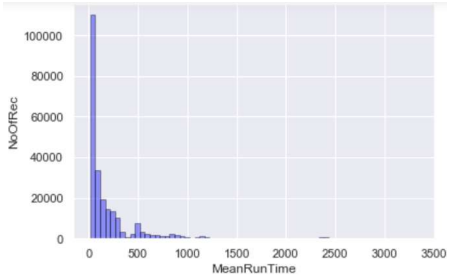
Data Preparation and Exploratory Data Analysis

As the first 10 features are ordinal in nature and has only powers of 2, these features have been scaled to values between 1 and 4. Ranking has been given based on the value for example, Higher power of 2 gets rank 1 in any column. This ranking was performed manually using map function of python. Key is the original value of columns and values is the mapped rank.

MWG and NWG: : MNWG_map = {16 :4,32 :3,64 :2,128:1}
KWG : KWG_map = {16 :2,32 :1}
MDIMC,NDIMC,MDIMA,NDIMB : MCMA_map = {8:3,16:2,32:1}
KWI :KWI_map = {2:2,8:1}
VWM,VWN :WMWN_map = {1:4,2:3,4:2,8:1}

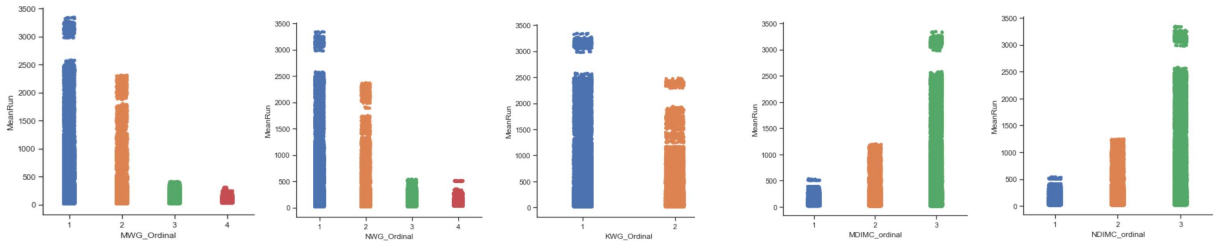
For Logistic Regression, the regression problem was converted into a classification problem by classifying all values more that 250ms as High run time and below 250 ms as low run time.

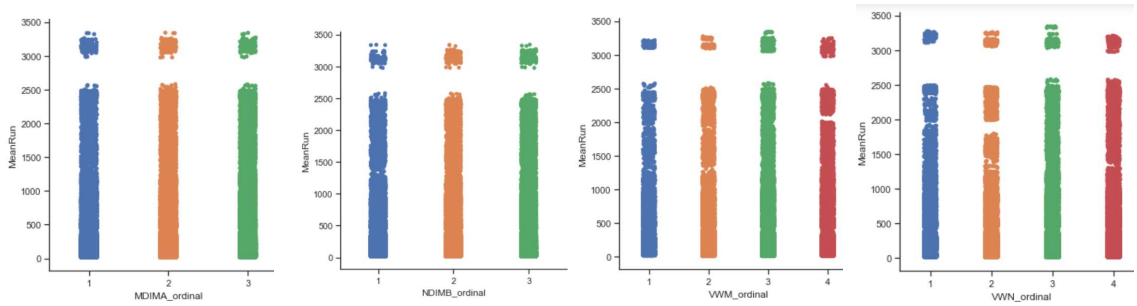
After plotting the histogram for target variable and it can be seen that the distribution is right skewed. Selecting 250 as threshold would result in creating two classes containing almost equal number of data records. The histogram shows that most of the combinations of the parameters result in run time between 0 and 500 ms.



Scatter plot of all the variables with target variable is shown below. It can be seen that all variables have only few possible values. Of all the variables, the variables in the first row contains at least one value which always results in low run time. Whereas variables in the second row have values which could result in all possible values of run time.

Based on this observation, the variables in first row have been selected for experiment 4.



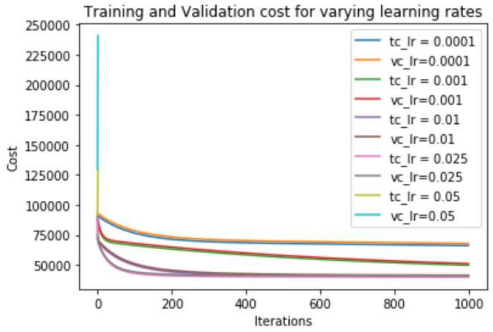


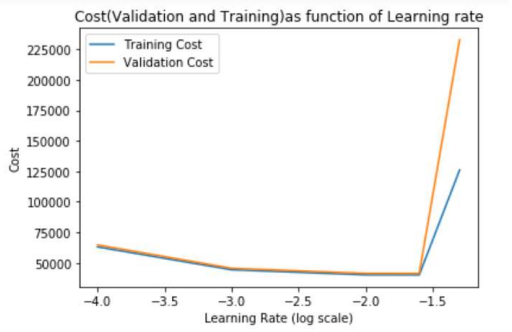
EXPERIMENTS

1. Hyperparameter tuning

a. LINEAR REGRESSION

- With each iteration, the cost is seen continuously decreasing and the algorithm is said to have reached convergence when the decrease in cost is within a defined threshold.
- We can see from the below plot of learning rate vs cost that the cost of training and validation decreases with the increase in learning rate from .0001 to .025, however on further increasing the learning rate the cost overshoots.
- Best parameters are as follows:
  MeanRun = -67.06*MWG_Ordinal - 59.29*NWG_Ordinal + 8.06*KWG_Ordinal + 72.42*MDIMC_ordinal + 71.58* NDIMC_ordinal + 16*MDIMA_ordinal + 16.454*NDIMA_ordinal + 9.28*KWI_ordinal - 3.22*VWM_ordinal -3.22*VWN_Ordinal - 0.26*VWN_Ordinal + 3.80*STRM_1 + 4.461*STRN_1 + 14.6354*SA_1 + 14.44*SB_1

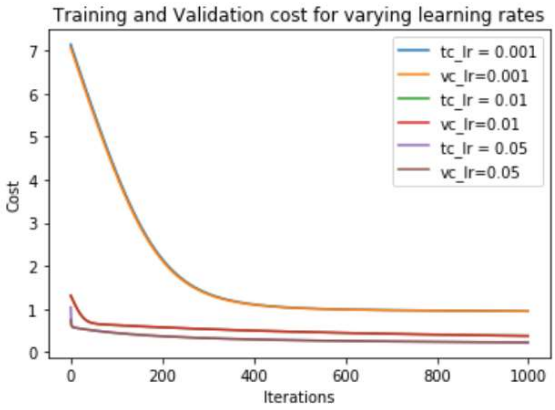


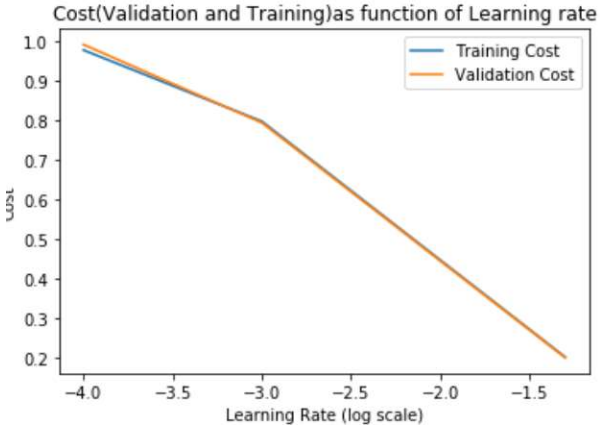


b. LOGISTIC REGRESSION

- With each epoch, the cost is seen continuously decreasing and the algorithm is said to have reached convergence when the decrease in cost is within a defined threshold.
- When the learning rate is low, the algorithm takes a large number of iterations to converge When learning rate is high, the algorithm converges faster and reaches the minima in close to 600 iterations.
- It can be seen that the difference between training and validation cost is significantly different at learning rate of 10e-4 however with the increase in learning rate this difference is insignificant.
- Best Parameters are as follows:

MeanRun =1.90 MWG_Ordina+ 1.62NWG_Ordinal -0.13 KWG_Ordinal-1.315 MDIMC_ordinal-1.342 NDIMC_ordinal+ 0.085MDIMA_ordinal+0.043 NDIMA_ordinal-0.156 KWI_ordinal +0.1329 VWM_ordinal + 0.197STRM_1 + 0.3652STRN_1 + 0.3777SA_1 + 0.67411SB_1





Summary:

- In logistic regression when the learning rate is low (10e-3), the algorithm takes a large number of iterations to converge but does not completely converges to minima. However at high learning rate(.05) the the algorithm converges very fast for both validation and training cost.
- Cost of training and validation decreases with the increase in learning rate from .001 to .05

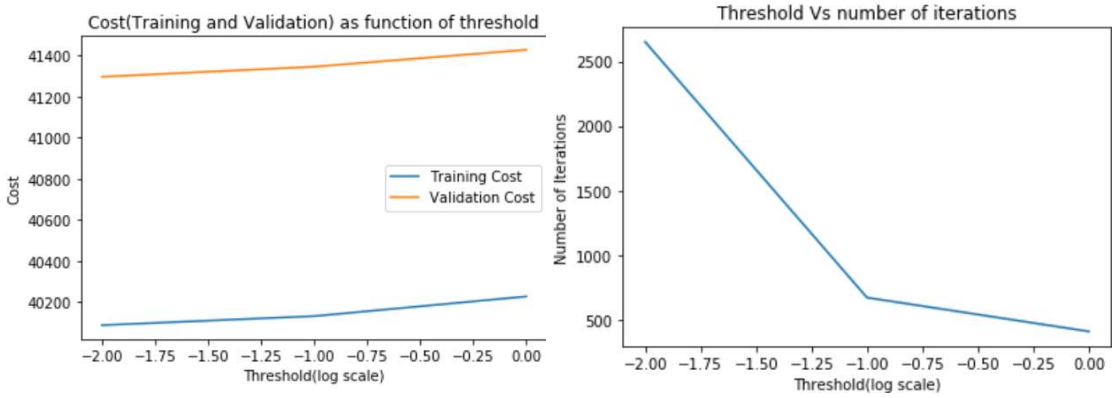
	Minimum Cost(training)	**Minimum Cost(validation)**	**Best Learning Rate**	**High Learning rate behavior**	**Low Learning rate behavior**
Linear Regression	40094.28	41308.25	0.025	Overshoots the cost	Takes more iterations to converge
Logistic Regression	0.2155	0.21463	.05	Takes more iterations to converge	Takes less iterations to converge

2. Changing convergence threshold

a. LINEAR REGRESSION

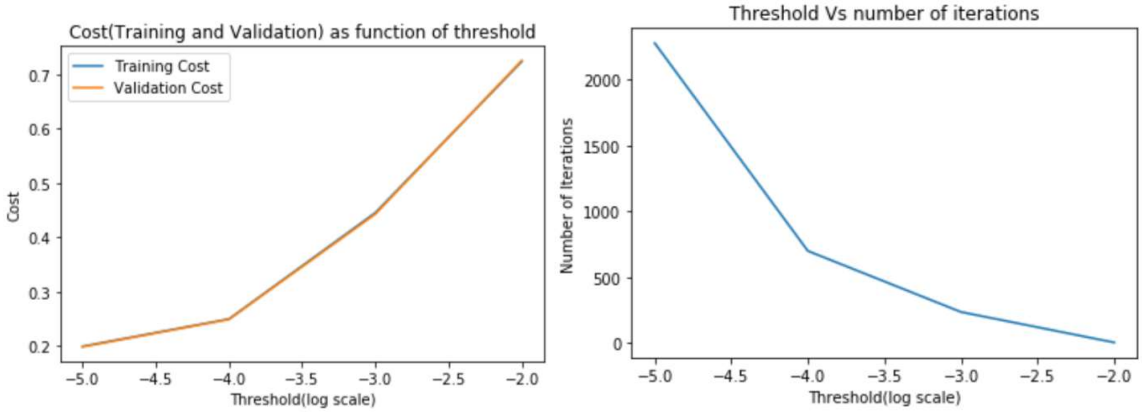
- Upon trying various values of threshold for the experiment it was feasible to play around with the following three values of threshold .01,.1 and 1. Values less than .01 took unusually longer to reach the minimum costs.
- If can be inferred from the following plots that as we increase the threshold, the number of iterations required to decreases significantly however it does not help us converging to minima.
- Here are output of experiment performed.

Minimum cost – Training : 40086.45
Minimum cost – Validation: 41297.28
Number of Iteration: 2651
Best threshold : .001



b. LOGISTIC REGRESSION

When the Change in Cost is within the Convergence threshold, the gradient descent algorithm is said to be converged. So, the algorithm converges faster at higher threshold value but at a certain point increasing threshold is found to increase the cost in both train and test dataset. We find the optimal value of threshold by plotting cost as a function of convergence threshold and find the point at which the test error is minimum.



- It can be seen that as we increase the threshold from 10e-13 to 10e-12 the cost value for both the train and validation decreases however when we further increase the threshold value it seems to increase the cost for both of them.
- Plotting the cost of validation and training for threshold of 10e-12 and learning rate of .05 gives below plot. Although from the plot it seems that the validation and training cost are same for the given threshold, there is difference of around 0.0005 between them.
- Here are output of experiment performed.
  Minimum cost – Training: 0.1991
  Minimum cost – Validation: 0.1981
  Number of Iteration: 2276
  Best threshold : 10e-5

3. Random Feature Selection

In this experiment, 10 features are selected at random using the df.columns.to_series().sample(8) on the dataframe df. function and the train and test errors are compared to the errors from the original set of features.

For the experiment following features were selected randomly by the sample method:

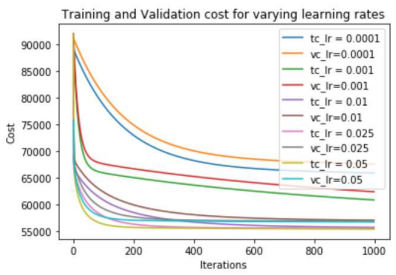
```
4  X = df[df.columns.to_series().sample(8)]
5  X.columns
```

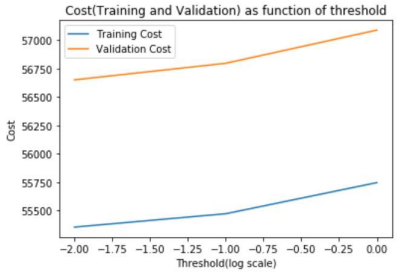
```
Out[78]: Index(['SB_1', 'NWG_Ordinal', 'KWI_ordinal', 'VWM_ordinal', 'STRM_1',
                'MWG_Ordinal', 'NDIMB_ordinal', 'VWN_ordinal'],
               dtype='object')
```

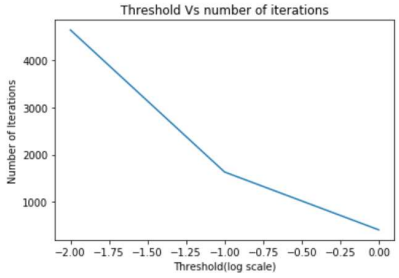
As expected, the errors of random feature selection are found to be more than the respective errors from the original set of 14 features.

a. LINEAR REGRESSION

- Upon testing the cost for learning rates ranging from 10e-4 to .05 it can be seen clearly that the higher learning converges slowly, and lower learning rate converges faster(refer plot 1 below). However only the higher learning rate of .05 resulted in lowest cost for both test and validation. Expectedly validation cost is higher that training cost for given learning rate of .05.
- Tested threshold values from .01 to 1 as the cost value was very high and any threshold value more than .01 was taking unusually longer time to learn and converge. As shown in the second plot higher threshold values give minimum cost compared to lower threshold values. However, they take more number of iterations to converge compared to lower threshold values which take less iterations to converge(refer 3rd plot below).



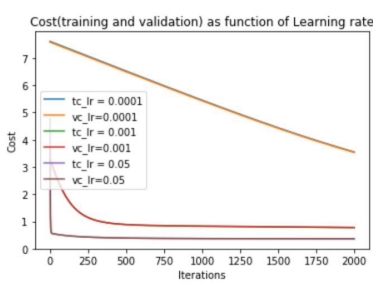


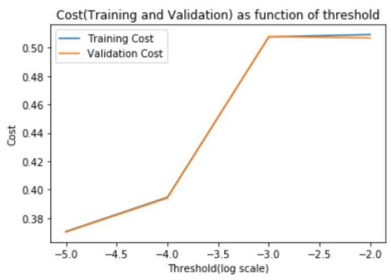


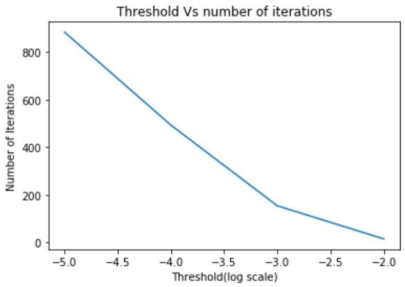
b. LOGISTIC REGRESSION

The outcome of experiments performed for logistic are similar to that of linear regression in terms of relationship of learning rates and threshold with iterations and cost for training and validation set.

- We can see from the below graph that the best learning rate is same as that for regression with 14 features which is 0.05.
- However, the converged cost value with 8 features is still 70% more than the converged cost with 14 features.
- The validation cost seems to be less than that of training cost. This is not a usual behavior and may point to some anomaly in the algorithm. However, as the other values are inline to expectation we can say that it has probably occurred by chance.







The summary of comparison between logistic and linear regression with randomly selected 8 features and all the features is as follows:

	Best Threshold	Minimum Cost(training) For best threshold	Minimum Cost(validation) For best threshold	Best Learning Rate	Minimum Cost(training) For best Learning Rate	Minimum Cost(training) For best Learning Rate
Logistic Regression(8)	10e-5	0.3705	0.3701	.05	0.3705	0.3701
Logistic Regression(14)	10e-5	.1991	.1981	.05	0.2155	0.21463
Linear Regression(8)	.001	55353.83	56650.15	.05	55439.53	56758.53
Linear Regression(14)	.001	40086.45	41297.28	.025	40094.28	41308.25

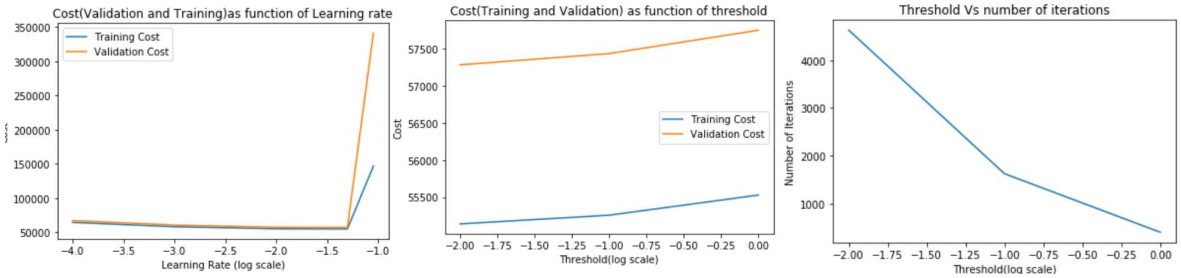
We can clearly infer that the randomly selected features do not contribute much prediction as the error will be more than compared to the regressions with all the features. But using all the features can cause problem of over fitting. So having some domain knowledge and then selecting features can turn out to be helpful but does not guarantee best results.

4. Manual Feature Selection

Based on the exploratory data analysis it appears that variables like VWN, VWM,NDIMB,MDIMB, and binary variables, all other variables have some values which always result in low run time. Based on this observation, selected following 8 features:
MWG_Ordinal,NWG_Ordinal,KWG_Ordinal,MDIMC_Ordinal,NDIMC_Ordinal,MDIMA_Ordinal,NDIMA_Ordinal,VWM_Ordinal

a. LINEAR REGRESSION

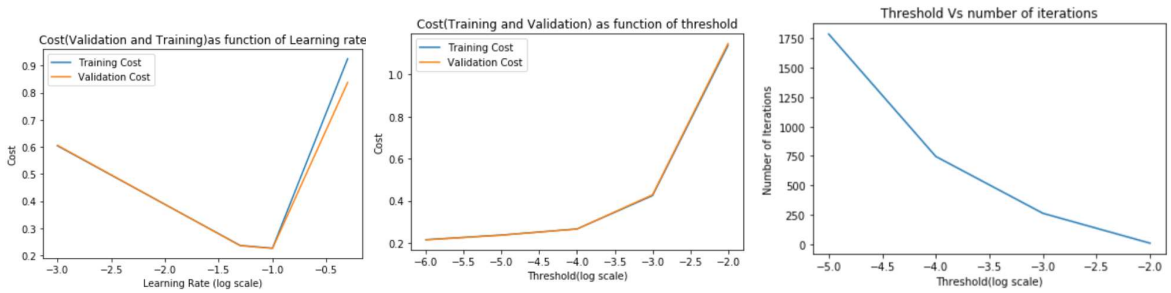
- In this experiment we can see that as learning rate is increased from 10e-4 to the .05 the training and validation cost decreases however on further increasing the learning rate, both the costs overshoot.
- Thresholds and cost relation are same i.e. as threshold decrease training and validation cost decreases and the number of iterations to reach this cost increases.
- We can see in 2nd plot that validation cost is quite high compared to the training cost for given threshold.



b. LOGISTIC REGRESSION

The outcome of experiments performed for logistic are similar to that of linear regression in terms of relationship of learning rates and threshold with iterations and cost for training and validation set.

Following are the outcomes of the experiments performed to check the relation of learning rates and threshold on cost and number of iterations for training and validation cost.



The choice of features based on exploratory data analysis did provide better results that selecting the features randomly. We can clearly see from the summary table below that cost for both training and validation is less in this case. However, dropping other features did not help in improving the cost further. We can see that the cost with all the features is the lowest in both the logistic and linear regression.

Summary

	Best Threshold	Minimum Cost(training) For best threshold	Minimum Cost(validation) For best threshold	Best Learning Rate	Minimum Cost(training) For best Learning Rate	Minimum Cost(validation) For best Learning Rate
Logistic Regression(8)	10e-6	0.2155	0.2156	.05	0.2625	0.2615
Logistic Regression(8)R	10e-5	0.3705	0.3701	.05	0.3705	0.3701
Logistic Regression(14)	10e-12	0.195	0.196	.1	0.2155	0.21463
Linear Regression(8)	.01	55138.70	57285.01	.05	55146.91	57297.67
Linear Regression(8)R	.001	55353.83	56650.15	.05	55439.53	56758.53
Linear Regression(14)	.001	40086.45	41297.28	.025	40094.28	41308.25

RESULTS

Based on the experiments performed above it can be said that the least test and validation error is received in regression with all the 14 features. One of the major reasons for such outcomes is the fact that these variables are ordinal in nature and a combination of these features decides the outcome. Independently these features may not have any significant impact on the run time as such.

Upon reviewing the heat map of correlation for all the variables, no significant relation can be obtained. This is expected as the data is set of all possible combinations of values of the 14 features. However as mentioned in exploratory data analysis some variables have value that always result in low run time. But when we ran the regression with only those features in experiment 4, the results were still not better that the regression with all features.

Rank based mapping was performed to normalize the values of the features, maintaining the order of the values. As explained in data preparation section this was done to scale down the range from 2 to 128 to 1 to 4.

There were few shortcomings of the model which can be improvised to make it better fit. First recommendation is to use the log scale for target variable in linear regression model. Another suggestive step would be to encode the ordinal variables as dummy variable. This would increase the number of features in the model however on categorizing the individual values as one column but can provide more information in terms of significance of each value with reference to base value in predicting run time. For example, for first column MWG which has possible values of 16,32,64 and 128, we can make 16 as base and create three columns for 32, 64 and 128. This will help us to understand which value of this feature can provide more information that others in predicting the run time.