Shalini singh – sxs180240 Project Report 1

Executive Summary

The project aims at the implementation of gradient descent algorithm for Linear Regression and Logistic Regression incorporating multiple parameters such as Learning rate (alpha), threshold, number of iterations etc. Based on the implementation, various experiments are performed to understand the relationship between the parameters and the training and test cost/accuracy. Conclusion and discussion are provided at the end of the implementation and experiments.

Tasks

Task 1 – Data Set Description and partitioning

- Dataset consists of 241600 observations on 15 variables.
- The description of the variables can be found out by going to the dataset link .
- The dependent variable for the linear regression model is Avgcnt(y): Average GPU run time.
- None of the column contains any missing value, so no missing value imputation is required.

The features – 'Run 1', 'Run 2', and 'Run 3' and 'Run 4 are dropped as the average of these four parameter are calculated and used as target variable

Task 2 – Linear Regression Model Design

A linear regression model is designed as a function to model the Average GPU run time. The function is designed in such a way that it has input parameters – β_0, β and the dataset. The function implements the model using the input parameters and returns the output data. The regression equation is given below:

```
# Linear Regression Model Function
def linear_regression_model(x, beta_0, beta):
    y_hat = (x @ beta) + beta_0
    return y_hat
```

Task 3 – Gradient Descent Algorithm (batch update)

The gradient descent algorithm is implemented with batch update rule. The sum of squared error cost function used in the class is used as the basis for computing the error/cost for all the operations in gradient descent algorithm. The initial beta values are reported below:

```
Initial beta0 value:  [0.5]
Initial beta values(apart from beta0):  [[0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5]]
```

The gradient descent algorithm is implemented keeping in mind the various control parameters. All the parameters are taken as an input to the algorithm and the final β values, cost and No. of iterations are returned back for further analysis and processing. The cost function equation are shown below:

```
# COST function
# y_hat and y have same shape; They are matrices;
def cost(y_hat, y):
    m = len(y_hat)
    ct = (0.5/m) * np.sum((y_hat - y)**2)
    return ct
```

Task 4 – Conversion to binary classification – Logistic Regression

The target variable is transformed to a binary variable using 60 as the value for conversion. Using the gradient decent algorithm I calculate the training and test set accuracy.I have converted the target variable in to binary form by using the cut off parameter The accuracy is reported below:

Training Accuracy	0.7259
Test Accuracy	0.7385

Experimentation

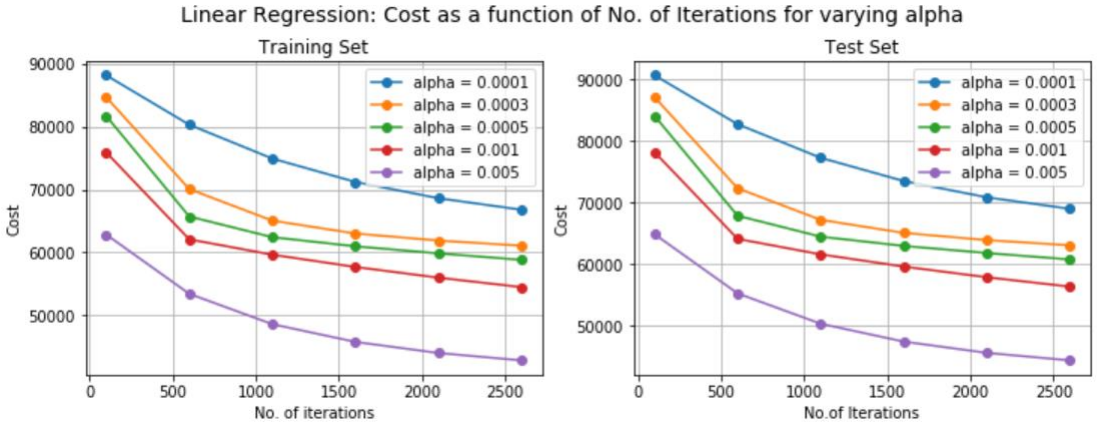
For all the experiments that are performed below, the initial β values are always 0.5. For each experiment, certain parameters are kept constant and some parameters are changed to decipher the relationship between COST and control parameters. The inferences are realized using plots and tables. The control parameters in research are explicitly mentioned for each experiment.

Experiment 1 – Varying control parameters for linear and logistic regression from a Cost perspective

Linear Regression: In this experiment, the *threshold* is fixed at a value of 0.00001. The reason such a value is chosen because for this value of *threshold*, we have the least cost. This experiment is performed to understand the relationship between Cost, No. of iterations and *alpha*. I have extracted the cost values for various value of no. of iterations and for varying *alpha* values.

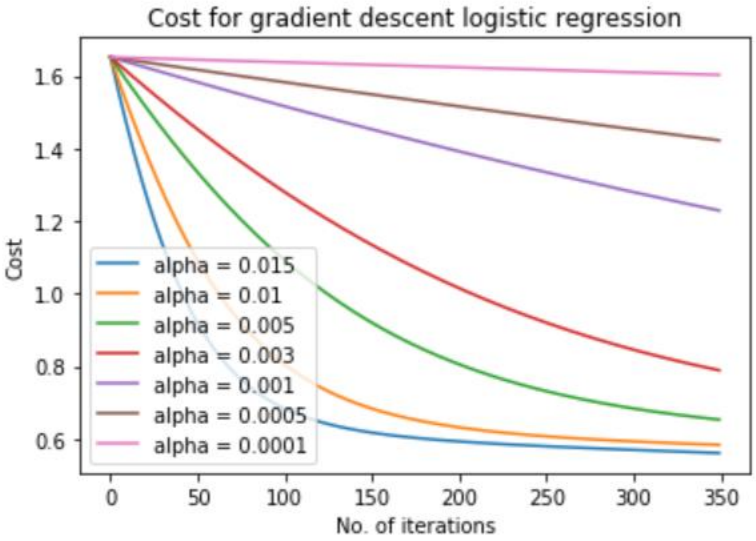
I have chosen a range of 100 to 3000 in steps of 500 for the 'No. of iterations' And, for the *alpha* values, I have chosen an array set of – [0.0001, 0.0003, 0.0005, 0.001, 0.005]. The prime reason for choosing this range is because for values lesser than 0.0001, the COST is high, almost equal to initial cost and for values higher than 0.005, the COST converges very quickly.

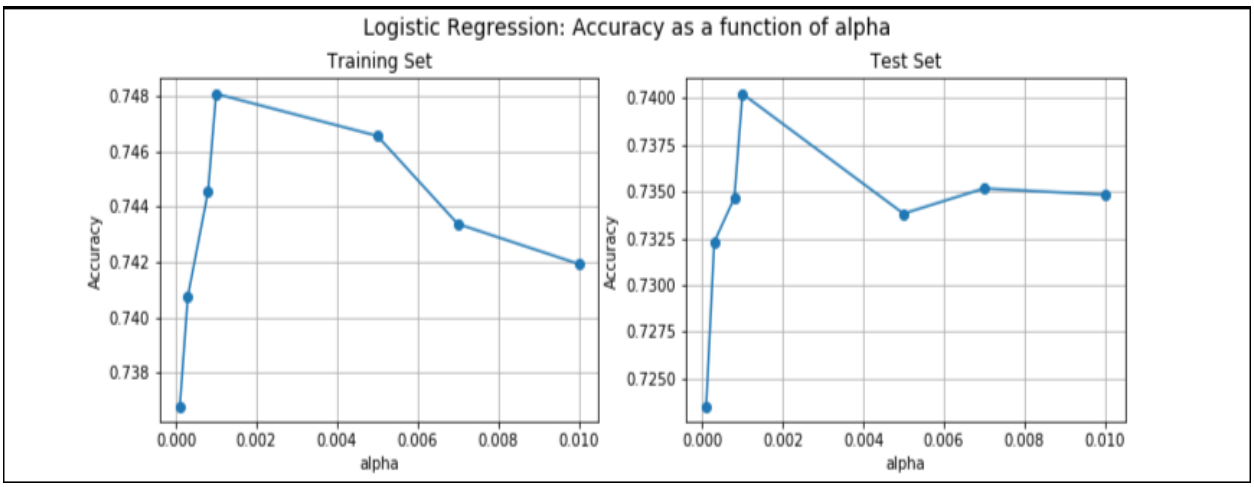
The resultant plots are reported below:



We can infer from the above plot that, as the 'No. of iterations' increase the COST decreases. Also, we can see that the convergence gets steeper/faster for increasing values of *alpha* also indicated by the lesser number of iterations for increasing values of *alpha*.

Logistic Regression: For logistic regression, this experiment is performed using the gradient descent algorithm I have used the almost the same range of alpha values for the reason mentioned earlier and I have also included extra few values to provide more visualization. plot below shows the relationship between COST and no of iteration.





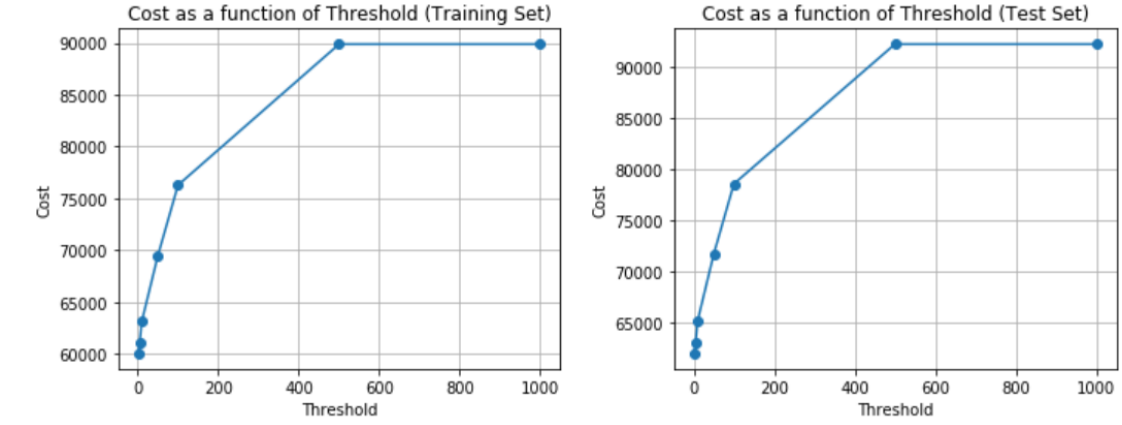
By looking at the plots, we can infer that the training and test accuracy spike at *alpha* = 0.001, he training and testing accuracy values decrease gradually for increasing values of *alpha.*

Experiment 2 – Varying thresholds for convergence for linear regression

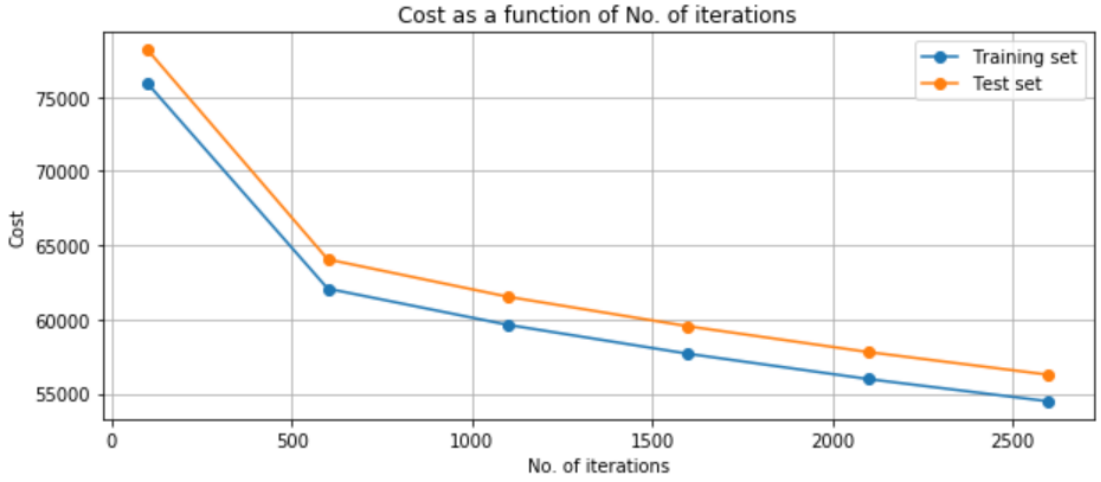
In this experiment, we try to establish the relationship between threshold and convergence for linear regression. I have inferred threshold as the parameter that sets an ideal difference value between successive costs as the stopping point or 'convergence' i.e. successive costs should differ by at least the 'threshold' value while running in the gradient descent algorithm, else we can safely say that convergence is reached.
In this experiment, I have chosen the range of values for *threshold* as given below:
thresh_set = [1,5,10,50,100,500,1000]

The maximum number of iterations is set fixed at 100,0(to be safe and accommodate all *threshold* alues). Implementing the gradient descent algorithm for varying threshold values, the below plot is obtained:

We can infer from the below plot that as *threshold* increases the COST also increases.



Proceeding with the experiment, after choosing the best *threshold*, the train and test errors are plotted as a function of gradient descent iterations. The plot is reported below:



It can be observed that as the *No. of iterations* increase the COST decreases which is expected as the Cost function is convex, and we are approaching the global minimum as gradient descent algorithm iterates.

Experiment 3 – Exploring Cost for a random set of 8 features for linear and logistic regression

In this experiment, 8 features are randomly selected, and their train and test errors are computed by running the gradient descent algorithm. The values of *alpha* and *threshold* are the best-chosen values which are obtained from previous experiments. The initial β values are 0.5 as always.

The 8 random features are chosen by the `random()` package in Python. This package produces a different output in every iteration. Hence, the results obtained in a future iteration may not be consistent with the results presented here. But the behaviour and trends are more or less preserved and remain intact. The resultant dataset is fitted with both linear regression and logistic regression models. The final results are tabulated and reported below (with original set's train and test metrics):

Linear Regression	Cost (Training Set)	Cost (Test Set)
Original Set (14 features)	0.6574	0.7081
Random set (8 features)	0.833	0.8309

Logistic Regression	Training Accuracy	Test Accuracy
Original Set (14 features)	0.751	0.7494
Random set (8 features)	0.6785	0.6804

From the above table, it can be inferred that for Linear Regression and Logistic Regression, the dataset with randomly selected features doesn't perform as good as the original dataset with 14 features. The COST of the random set is higher than the original set for both the Training and Test sets. Similarly, in Logistic Regression, the Train and Test accuracy is lower for the random set as compared to the original. Looks like not all the information is captured by the random set and hence, the models aren't able to predict better.
The 8 random features that were chosen are reported below:

```
['MWG', 'NWG', 'KWG', 'MDIMC', 'NDIMC', 'KWI', 'VWN', 'SB', 'y'],
```

Experiment 4 – Exploring Cost for a "chosen" set of 8 features for linear and logistic regression

In this experiment, I am choosing the 8 different features of my choice and performing Linear Regression and Logistic Regression on the resultant dataset. The training and test metrics are obtained and are tabulated to serve as a comparison medium for inference and interpretation of the obtained results. In my opinion, the 8 best features are reported below:

['MWG','KWG','MDIMC','NDIMB','KWI','VWN','STRN','SA']

After performing the Linear and Logistic Regression, the train and test metrics are captured and tabulated. The table is reported below, with all the values from original set, random set and chosen set:

Linear Regression	Cost (Training Set)	Cost (Test Set)
Original Set (14 features)	0.6574	0.7081
Random set (8 features)	0.833	0.8309
My set (8 features)	0.9613	0.973

Logistic Regression	Training Accuracy	Test Accuracy
Original Set (14 features)	0.751	0.7494
Random set (8features)	0.6785	0.6804
My set (8 features)	0.6905	0.6921

Looking at the table, we can make the following inferences-

MY choice of features has not provided better results when compared to the **original set** as is evident by **original set's** lower training and test costs for Linear Regression and higher training and test accuracy for Logistic Regression.

Conclusion / Discussion

This section summarises and discusses the interpretation of the results obtained as part of the experimentation. Some of the key inferences are:

- **Linear Regression:** While implementing gradient descent for Linear Regression, we realize that the cost function is indeed a convex function with a global minimum. The COST keeps decreasing as the No. of iteration keeps increasing. The control parameter *threshold* determines when the gradient descent algorithm should stop iterating, thus establishing the condition of convergence. As *threshold* increases, the *No. of iterations* decreases as the convergence condition is met earlier. Also, the control parameter *alpha* determines the step size for updating β values, which indirectly determines the rate of convergence of the cost values. There can be a situation when the global minimum is not met at all because of a high *alpha* value. As *alpha* increases, the rate of convergence increases.

- **Logistic Regression**: In logistic regression as *alpha* increases, we saw that accuracy increased rapidly to spike at a point and then decrease gradually as *alpha* increased. There is no definitive relationship between *alpha* and Accuracy.

- Zooming out and approaching from overall perspective of the project implementation, it can be inferred from Experiment 3 & 4 that, to model/predict the target variable, it is best to include as many features as possible to increase the predictive power of the resultant model.