2/13/2020

SARTHAK MOHAPATRA

The University of texas at dallas, US

**Prediction of Average GPU run time & classifying the run type**

Algorithm from scratch using Gradient Descent to predict average GPU Run Time & classify it’s run type

**INTRODUCTION –**

In this project, we will learn and build Linear Regression and Logistic Regression method using the Gradient Descent approach from scratch. We will apply our algorithms on a dataset containing information about GPU kernel performance on matrix multiplication (A \* B = C) where, A, B and C are metrices. Our goals are highlighted below:

* Implementing a linear regression model using the Gradient Descent method to predict the GPU run time.
* Implementing a logistic regression model using the Gradient Descent method to classify a run as high or low time consuming.

In this project, we have performed various experimentations by varying the learning rate(α) and the threshold value for convergence and have observed the changes in the performance of the algorithm’s prediction and classification capabilities. Similarly, we have performed feature selection both by random selection and based on their importance as per our observations to experiment on the changes in the prediction and classification performance and choose the best model most significant features.

**DATASET & FEATURE DESCRIPTION –**

For this project, we have used the SGEMM GPU kernel performance Data Set available for download at [UCI ML Website](https://archive.ics.uci.edu/ml/datasets/SGEMM+GPU+kernel+performance). The dataset measures the running time of various matrix multiplication processes where each matrix is of shape 2048 × 2048. The total number of observations in the dataset is 241600.

For each test, four runs were performed, and their results were presented in the file. For our project, we have taken the average of four runs and have considered it as our target/dependent variable.

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No | Feature Name | Type | Values |
| 1 | MWG - per-matrix 2D tiling at workgroup level | int | 16, 32, 64, 128 |
| 2 | NWG - per-matrix 2D tiling at workgroup level | int | 16, 32, 64, 128 |
| 3 | KWG - inner dimension of 2D tiling at workgroup level | int | 16, 32 |
| 4 | MDIMC - local workgroup size | Int | 8, 16, 32 |
| 5 | NDIMC - local workgroup size | Int | 8, 16, 32 |
| 6 | MDIMA - local memory shape | Int | 8, 16, 32 |
| 7 | NDIMB - local memory shape | Int | 8, 16, 32 |
| 8 | KWI - kernel loop unrolling factor | Int | 2, 8 |
| 9 | VWM - per-matrix vector widths for loading and storing | Int | 1, 2, 4, 8 |
| 10 | VWN - per-matrix vector widths for loading and storing | Int | 1, 2, 4, 8 |
| 11 | STRM - enable stride - access off-chip memory in 1 thread | Cat | 0, 1 |
| 12 | STRN - enable stride - access off-chip memory in 1 thread | Cat | 0, 1 |
| 13 | SA - per-matrix manual caching of the 2D workgroup tile | Cat | 0, 1 |
| 14 | SB - per-matrix manual caching of the 2D workgroup tile | Cat | 0, 1 |

**DESCRIPTIVE STATISTICS AND EXPLORATORY DATA ANALYSIS -**

Since we are trying to accurately predict the GPU run time and correctly classify a run time as high or low time consuming, the features that we are going to use will play a major role in the effectiveness of the model while predicting or classifying various runs. So, to understand every variable and how their presence or their correlation impacts the run time, descriptive statistics and exploratory data analysis was performed.

**Correlation Among Variables -**

A screenshot of a computer

Description automatically generated

It explains the correlation among variables in the dataset. The key takeaways from the above correlation matrix/ heat-map is mentioned below:

* The dependent variable Average has the positive correlation with MWG and NWG. But the correlation coefficient is not high.
* The dependent variable Average has negative correlation with NDIMC and MDIMC. But the correlation coefficient is not high.
* The dependent variables have highest positive correlation of 0.35 with MWG and highest negative correlation of -0.22 MDIMC.
* MWG has a small positive correlation with VWM. Also, NWG has a small positive correlation with VWN.
* MDIMC is having a small negative correlation with NDIMC.

A screenshot of a cell phone

Description automatically generatedFor the correlated variables, the correlation coefficient is small enough and are not going to impact much. To further analyze the impact of independent variables on the dependent variable (Average Run Time), the following graphs was plotted:

The graph above explains us the effect of VWM on Average Run Time and all other ordinal variables/ features. From the graph we can observe that with increase in the order of VWM, the Average Run Time has increased significantly. Similarly, with increase in order of VWM, the order of MWG also seems to increase signifying the records having with higher order of VWM has higher order value of MWG. For all other variables/ features, there is not much significant impact of VWM as we can see that the changes are very minimal or unchanged.

A screenshot of a cell phone

Description automatically generatedTo understand the effect of VWN on other features and dependent variable, let’s analyze the below graph:

From the graph above, we can observe the effect of VWN on Average Run Time and all other ordinal variables/ features. It can be clearly seen that with increase in the order of VWN, the Average Run Time has increased significantly. It also explains the highest positive correlation of VWN and Average Run time in the Heat-Map. Similarly, with increase in order of VWN, the order of NWG also seems to increase signifying the records having with higher order of VWN, also has higher order value of NWG. For all other variables/ features, there is not much significant impact of VWN as we can see that the changes are very minimal or unchanged.

Since the categorical variables STRM, STRN, SA and SB are not having correlation with the ordinal variables, let’s analyze the graph below to understand the effect of the categorical variables on the Average Run Time:

A screenshot of a cell phone

Description automatically generated

(Please note that since the value of the categorical variables are very small compared to Avg Run Time, the bars for the categorical variables are not clear)

From the above graph, we can see that for STRM and STRN, there is not much variation in the Average Run Time. As compared to a run where the stride is not enabled to access off-chip memory in single thread (STRM = 0), the Average Run Time slightly decreases when stride is enabled to access off-chip memory in single thread (STRM = 1). Unlink STRM/ STRN, as compared to a run where per-matrix manual caching of the 2D workgroup tile is not there(SA=0/ SA=1), the Average Run Time increases when there is per-matrix manual caching of the 2D workgroup tile(SA=1/ SB=1) in a run process. Also, we can see that STRN has no impact on Average Run Time which is constant and has same mean value.

To understand the effect to the other ordinal features on Average Run Time, the below graphs explains the changes in the Average Run Time with variation in the order of the features:

A screenshot of a cell phone

Description automatically generatedTo understand the effect to the other ordinal features on Average Run Time, the below graphs explains the changes in the Average Run Time with variation in the order of the features:

In the figure to the right, we can observe that there are a greater number of observations in the dataset with higher order of the MWG & NWG. Also, with increase in order of MWG & NWG, there are good number of observations where the Average Run Time is very high. For KWG and KWI, although there are few records with high Average Run Time, but overall there is not much impact of KWI and KWG.

A screenshot of a cell phone

Description automatically generatedIn the figure to the right, we can see that MDIMA and NDIMB is not having impact on the Average Run Time. We have similar number of observations in each category MDIMA and NDIMB and the variation in records are almost same. But, for MDIMC and NDIMC, we can see that as the order of the MDIMC and NDIMC increases, the Average Run Time decreases very significantly, and we have very few transactions which have higher order value of MDIMC and NDIMC.

**Data Preparation –**

All the features in the dataset contains data in different ranges. Since we are going to use the Gradient Descent method to perform Linear Regression and Logistic Regression method, all the features in the dataset were scaled using the mean normalization method. Following the normalization process, each feature has mean 0 and standard deviation as 1.

For Logistic Regression with the gradient descent approach, the median of the Average Run Time was taken and observations below median were changed to class 0 (Low run) and above median to class 1 (High run) to convert the problem into a binary class problem.

**The GPU dataset was divided into training (70%) and validation (30%) dataset with 70:30 split ratio.**

**LINEAR REGRESSION USING GRADIENT DESCENT METHOD –**

For predicting the GPU’s Average Run Time, the Linear Regression with Gradient Descent method was built. In the algorithm, there are a total of 14 independent features/ Variables. Out of the 14 variables, there are 10 ordinal variables and 4 categorical variables. The dependent feature/ variable is Average Run Time which is calculated by taking the average of time of the 4 runs for each record/ combination. The Linear Regression Equation for the GPU’s run time prediction is mentioned below:

**YAvg Run Time = β0 + (βMWG \* MWG) + (βNWG \* NWG) + (βKWG \* KWG) + (βMDIMC \* MDIMC) + (βNDIMC \* NDIMC) + (βMDIMA \* MDIMA) + (βNDIMB \* NDIMB) + (βKWI \* KWI) + (βVWM \* VWM) + (βVWN \* VWN) + (βSTRM \* STRM) + (βSTRN \* STRN) + (βSA + SA) + (βSB \* SB) + е**

The Gradient Descent method is applied with the focus of minimizing the Cost Function which is defined Mean of as below:

J(β) = (1 ÷ 2m) × i=1∑m (Y^Predicted – YActual)2

The Linear Regression with Gradient Descent algorithm is used with the hyperparameter α and other parameters as mentioned below:

* α(Alpha) – Learning rate which helps in controlling the model’s convergence towards minimum.
* Convergence Threshold – The threshold value signifying convergence if change in cost function is below threshold.
* M – Total number of records in the dataset.
* Number of iterations – Number of iterations used to calculate the minimum cost.

After an initial run of the Linear Regression using Gradient Descent with an alpha of 0.0001, convergence threshold of 0.00001 and number of iterations as 10000, the following was the cost function values:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Alpha (Learning rate) | Threshold | Num of Records(m) | Min Cost Function Value |
| Training Data | 0.0001 | 0.00001 | 169120 | 0.3651264 |
| Validation Data | 0.0001 | 0.00001 | 72480 | 0.354251 |

**LOGISTIC REGRESSION USING GRADIENT DESCENT METHOD –**

For classifying a GPU run time as high time consuming or low time-consuming process, the Logistic Regression with Gradient Descent method is built and used. The independent features/ variables for the process is same as the Linear Regression process.

But, for converting the dependent variable into a binary class categorical variable, the median of the Average GPU Run time was taken. The Actual class of records with Average Run Time less than the median value was made to 0 and with higher than median value was made to 1.

The logistic regression equation is mentioned below:

**YAvg Run Time = 1 / ( 1 + е -( β0 + (βMWG \* MWG) + (βNWG \* NWG) + (βKWG \* KWG) + (βMDIMC \* MDIMC) + (βNDIMC \* NDIMC) + (βMDIMA \* MDIMA) + (βNDIMB \* NDIMB) + (βKWI \* KWI) + (βVWM \* VWM) + (βVWN \* VWN) + (βSTRM \* STRM) + (βSTRN \* STRN) + (βSA + SA) + (βSB \* SB) ) )**

For the Logistic Regression with Gradient Descent, same hyperparameter α and control parameters were used. After an initial run of the Logistic Regression using Gradient Descent with an alpha of 0.0001, convergence threshold of 0.00001 and number of iterations as 10000, the following was the cost function values:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Alpha (Learning rate) | Threshold | Num of Records(m) | Min Cost Function Value |
| Training Data | 0.0001 | 0.00001 | 169120 | 0.1084185 |
| Validation Data | 0.0001 | 0.00001 | 72480 | 0.5808993 |

**EXPERIMENTATIONS –**

1. **Experimenting and observing the changes in the both the models with hyperparameter tuning.**

The hyperparameter alpha is the learning rate that helps in controlling the model’s convergence towards the minimum. For the purpose of this experimentation, different values of alpha were taken and all the cost functions convergence towards the minimum and the minimum cost/error value with it’s corresponding alpha was plotted for both training and validation dataset as shown below:

A screenshot of a cell phone

Description automatically generated**Linear Regression plots**:

A screenshot of a map

Description automatically generated

The above graph on the left explains the convergence of the cost functions value towards the minimum. It can be observed that, with higher values of alpha (example: 0.01), the cost functions decrease very rapidly and convergence in very few iterations (example: ~800). Similarly, for very low value of alpha (example: 0.00001), it can be observed that the cost function decreases very slowly and couldn’t converge with 3000 number of iterations. When the value of alpha is set to 0.001, the cost function decreases at a decent rate and is close to the minimum convergence value.

But, when we set the value of alpha to 0.01, we can observe that the cost function decreases slowly and was close to converging to the minimum at the end of 3000th iteration.

The above figure on the right explains the minimum value of cost function for various alpha ranges in training and validation dataset. It can be observed that with increase in the value of alpha, the difference in training and validation dataset is increasing but at a very slow rate. As compared to cost function value when alpha was set to 0.0001, the cost function value when alpha was 0.01 seems to be higher.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**Logistic Regression plots**:

The figure on the left explains the convergence of the cost function with the minimum value for the logistic regression model. It can be observed that at alpha = 0.01, the cost function value decreased steadily and reached the minimum with approximately 3000 iterations. With alpha = 0.001, the cost function value decreased at a descent rate. Although it didn’t converge with the minimum at the end of the 3000th iteration but was close to the minimum value.

The figure to the right explains the minimum value of cost function for various alpha ranges both in train and validation dataset. For both training and validation dataset, after the value of alpha reaches 0.001, the change in the value of cost function with respect to itself is almost constant. But, unlike the training dataset, the validation dataset has greater cost function value at every level of alpha.

**Interpretation and key points** –

* For the Linear Regression model, keeping the convergence threshold value at 0.000001 the hyperparameter alpha value in between 0.00001 and 0.0001 will provide the best results and the cost function will most likely converge at a decent number of iterations.
* For the Logistic Regression model, keeping the convergence threshold value at 0.000001 the hyperparameter alpha value in between 0.01 and 0.001 will provide the best results and the cost function will most likely converge at a decent number of iterations.
* With very high value of hyperparameter alpha, the cost function will converge to the minimum very rapidly with least iterations and will result in incorrect independent feature coefficients.
* With very low value of hyperparameter alpha, the cost function will converge to the minimum very slowly. Although the independent feature coefficients will be accurate, we will need greater number of iterations to reach the minimum and the time consumed will be very high.

1. **- Experimenting and observing the changes in the both the models with convergence threshold tuning.**

The convergence threshold parameter/ value is used to control the continuation of iterations after the cost function has reached the minimal. If the difference in the previous run’s cost function value and the current cost function value is less than or equal to the threshold value, the iteration would stop, and we will assume that the cost function has reached the minimum.

A screenshot of a cell phone

Description automatically generated**Linear Regression:**

In the figure to the right, we can see that for a total 3000 iterations, there is no change in the value of the cost function, and it is constant in both training and validation dataset.

But we can observe that in case of linear regression, the minimum cost function value of training dataset is greater than the minimum cost function value of validation dataset.

A screenshot of a cell phone

Description automatically generated**Logistic Regression:**

The figure to the left explains the change in the cost function value in the training and validation dataset at various convergence threshold value. We can observe that, after 3000 iterations, there is no change in the value of the cost function with respect to the convergence threshold in both the datasets. Also, we can notice that, in case of logistic regression, the cost function value in validation dataset is significantly greater than the cost function value of training dataset.

A screenshot of a cell phone

Description automatically generatedSince there is no change in the value of the cost function value with respect to threshold value, let’s consider the minimum convergence threshold value of 0.000001 as an better option for finding the minimum of the cost function value and the effect of threshold on the training and validation dataset was plotted. In case of training dataset, there is very gradual decrease in the value of cost function. In validation dataset the cost function value changes even slower than the training dataset rate. The graph on the right explains the change in the training and validation dataset at each iteration over a total of 3000 iterations for an alpha value of 0.001 and the best-chosen convergence threshold value of 0.000001. We can observe that for validation dataset, the change in cost function value is very minimum and there is no significant change in the cost function value. But there is a gradual decrease in the cost function value for the training dataset throughout the iteration process.

1. **– Selecting 8 random features and building the model.**

As part of this experimentation, the following features were selected randomly, and the details of the features are mentioned below:

**KWG, MDIMC, MDIMA, NDIMB, KWI, VWN, STRN, SA**

For performing both the linear and logistic regression process, the datasets were modified accordingly so that we can perform both the prediction and classification algorithms and test the respective algorithms performance with the above 8 features. The results obtained from the algorithms are mentioned below:

**Linear Regression:**

**A close up of a map

Description automatically generated**The graph on the left explains the changes in the cost functions value while converging towards the minimum value. We can observe that with the 8 random features, the model’s performance in minimizing the cost function to find the minimum is bad as compared to the model with all the features. The trend is similar in both the training and the validation dataset. So, as compared to the model with the 8 random features, the model with all the features perform much better.

**Logistic Regression:**

**A screenshot of a cell phone

Description automatically generated**The figure on the right explains the change in the cost functions value while converging towards the minimum value. We can observe that with the 8 random features, the model’s performance in minimizing the cost function to find the minimum is almost the same in the both the cases of random features and the model with all the features. But, in the model with all the features, the cost function value in case of the validation dataset is minimum and hence can be considered as a better model in predicting the class of the GPU run than the model with the 8 random features. (The plots for both the models are exact and overlap each other)

1. **– Selecting 8 best features and building the model.**

As part of this experiment, we are trying to build a model that has 8 features but unlike the previous experiment, here we are selecting the features based our understanding of the data through the exploratory data analysis of the process and based on their correlation with Average Run Time.

Based on our analysis performed above, the following features are being chosen to get the best model:

**VWM, VWN, SB, SA, MWG, NWG, MDIMC and NDIMC**

For performing both the linear and logistic regression process, the datasets were modified accordingly so that we can perform both the prediction and classification algorithms and test the respective algorithms performance with the above 8 features. The results obtained from the algorithms are mentioned below:

**Linear Regression**:

**A close up of a map

Description automatically generated**The graph to the left shows the decrease in the cost functions value with an alpha value of 0.0001 over 3000 iterations. We can observe that the decrease in the cost function value is identical for both the training and validation dataset for both the model with the selected 8 features and the model with all the features. Hence, the model with least number of features is a better model.

(The plots for both the models are exact and overlap each other)

**Logistic Regression**:

**A screenshot of a cell phone

Description automatically generated**The figure to the right shows the decrease in the cost function value with an alpha value of 0.0001 and over 3000 iterations.

We can observe that for the validation dataset, with the model with all features, we get the cost function value which is least and the change in the value of the cost function is very less. For the model with 8 chosen features, the cost function’s change is identical. (The plots for both the models are exact and overlap each other)

**Interpretation & Discussion:**

Based on the results obtained from the project the following things can be inferred:

* An alpha/ learning rate which produces the best beta coefficients and at the same time doesn’t take much time to converge is to be chosen. As per our analysis and experimentations, the best value of alpha that we can consider is 0.0001.
* A convergence threshold value of 0.000001 can be considered as an ideal threshold value for both the models when the number of iterations are not beyond 5000. Since in experimentations that we have conducted as part of this project has limitations with hardware capabilities, the number of iterations that was opted for was 3000 and 0.000001 is the ideal threshold value.
* A model is considered as a good model that performs great with least number of features. So, from our analysis in experimentation no 4, for Linear regression, we can select the model with just the 8 features as it performs equally good as compared to the model with all the features in finding the minimum cost function value within 3000 iterations. In case of Logistic Regression, since the change in cost functions value is identical with both the models, we should choose the model with least features as the best model.
* Based on all the experimentation and exploratory data analysis, the features that matter most while predicting the Average GPU Run time are:
  + **VWM, VWN, NWG, NDIMC and MDIMC**

For predicting and classifying a GPU Run using the Gradient Descent method, the major challenges that was faced was with the hardware capabilities which led to limiting the number of iterations to 3000. For better experimentations and to test the capabilities of the algorithm better and even more, the following steps could be incorporated:

* With better hardware capabilities, the number of iterations can be increased (example 10000) which can give us better and more concise results. For example, with around 10000 iterations, the changes with respective to learning rate and convergence threshold could be better judged in case of Logistic Regression with Gradient Descent.
* Applying machine learning algorithms on top of domain knowledge has got its own advantages and provides better results as we know the importance of each feature on the dataset. But, in the GPU dataset, for majority of the fields, with the lack of domain knowledge, the feature selection process was not efficient.
* The features in the dataset is varies a lot in terms of scale. Especially for the dependent feature Average Run Time, the maximum GPU run time is 3341.5 millisecond and minimum is 13.13 millisecond. So, performing the analysis and considering the modeling process with log linear models can yield us better results and can be more significant.