SGEMM GPU Kernel Performance prediction by using Supervised ML Regression model

By Saraswat Mukherjee

Abstract:

We are here trying to analyse the performance of SGEMM GPU Kernel on the basis of their run time. There are 14 parameters which provide different run times as outcome on putting them onto work with different combinations. For every combination we got four runtimes. We will be analyzing the parameters along with their runtimes due to different combinations. The experiment was run using ‘gemm_fast’ kernel from the automatic Open CL Kernel tuning library ‘CLTune’.

INTRODUCTION

We have got our dataset in the form of a 2048 x 2048 matrix. It contains 14 parameters or columns out of which the first 10 are ordinal which takes until 4 different powers of two values and the last 4 parameters are in binary format. They are combined into 1327104 ways out of which 241600 are feasible. We have used this dataset and put into different machine learning algorithms to create ML models. These models help us in predicting the runtimes of different combinations of those parameters.

The 14 parameters we have are independent variables. They are named as :-

- MWG
- NWG
- KWG
- MDMIC
- NDMIC
- MDIMA
- NDIMB
- KWI
- VWM
- VWN
- STRM
- STRN
- SA
- SB

The parameters are combined in different ways to form four runtimes which is the output after all the operations.

BUILD THE INITIAL PLATFORM

In order to start our work the first step will import all the modules that we will be required to use in different tasks. Here we will import the following libraries for our use:-

- Numpy
- Pandas
- Statsmodels.api
- Sklearn
- train_test_split
- Linear Regression
- Lasso
- Ridge
- Decision Tree Regressor
- R2_score
- Matplotlib
- Seaborn
- Plotly.express

Now we will be fetching the dataset from drive or Github account using Pandas library and save it in a variable called data. Now after getting the data we will first observe it properly and understand up to a profound level.

DATA WRANGLING

This is the time now to clean our dataset and present it in such a format so that we can make the best possible use of it. First we will try to find the basic information about each of the variables like their count and data types. We can see that the parameters or the independent variables are in integer data types while the dependent variables or the runtimes are in float data types.

Now as we have four runtimes in each row having four slightly different values, so for our ease we will take the mean of all the four and create a column in the dataset named runtime. We have a column with mean runtimes now so we don’t need four original runtimes. Hence we will drop all of them.

The null values may also create a lot of hassles while working with the data so we are required to find them and pop them out. Fortunately this dataset doesn’t have any null values. We also have all the data in numbers only no string or special characters, which consequently reduces the requirement to clean them. Just with these few efforts our data is now ready to work with.

EXPLORATORY DATA ANALYSIS

We will try to find now some valuable information about each variable like their count, minimum, maximum, mean, standard deviation etc. These basic statistical data will provide us a nice idea and a firm understanding of our dataset.

Check for outliers in runtime

We also need to check for the outliers as it causes problem while working with the data. So we will check for it by creating a box plot. But in our dataset there is no outlier present which makes our work much convenient. We can conclude that our entire data is useful.

Check for correlations

At times if we have a dataset with many correlated variables, then it may cause issues. So we should also check for correlation in our data. In order to get then we will create a heat map which prominently points out the correlation. Again fortunately we have no significant correlation in this data which further reduced our work. Also creating a histogram for every variable gives us an extensive and detailed overview of the dataset.

Check for skewness in runtime

Skewness also plays a vital role in understanding the biasness of our data. So we will check the skewness of the runtime data. On checking we found that our runtime data is right or positive skewed which can create problem later.

Rescaling the runtime data with log values for normalisation

So we will normalise the data by scaling it with log values. On creating a dist plot of this data values we can see that the plot has now been flattened. We will name these new rescaled data values as 'Truntime' now. But we should also check that the new data values are relevant to the original data values or not, otherwise it will be completely different dataset. For that we will check the correlation again using heat map. On doing so we can say that, they are indeed correlated. This concludes that the transformation is relevant and our data is good.

MACHINE LEARNING

To do the final work now we need the final ready to use data. As we now have the 'Truntime' with no skewness so the runtime column is not required. Hence we will only take the dataset without the runtime column as our relevant data.

Split the data into test and train

We need the data in the form of X and Y variables to work on with, so we will take the 'Truntime' as Y and the rest of the columns as X. After that we will split both the X and Y data as train and test respectively which we will further use with ML algorithm in order to make appropriate prediction.

Calculating R2 & adjusted R2 score

We will apply several ML algorithms on the train and test data we just got. Out of those used algorithms we compare their effectiveness on the basis of their respective R^2 and adjusted R^2 value. So we will create a function now called R2 cal to calculate those values and make our judgement on the effectiveness of the ML algorithms.

Linear Regression

The first algorithm will be linear regression which we be apply on our X and Y train data. By fitting the data into the algorithm we will create our first ML model. Then we will try to find the get the intercept and coefficient of the linear regression model. As mentioned earlier that we have split both the X and Y data into train and test separately. We have applied the algorithm on X and Y train data, so now will apply the algorithm along with predict method on X test data to predict the Y data which is our target variable. Now we will compare the Y test data with our predicted Y data. To get a visual understanding of the comparison we will create a scatter plot which will show us the relation between the predicted and test data. Now using the Y train and Y test data we will find the R^2 score and adjusted R^2 score value with help of the R2 cal function we created earlier.

Lasso Regression

The next ML algorithm which we will be using is the Lasso regression which will help us to eliminate less influential features. We will do the same thing now, fitting the X and Y train data into the algorithm and create the Lasso Regression ML model. Now by using the X test data we will predict the Y value. Then the Y test value will be compared with the predicted Y value. After that we will try to get the R^2 and adjusted R^2 score.

Ridge Regression

Our next algorithm will be Ridge Regression and we will again fit the X and Y train value in it to create our model. Then similarly like the last algorithm we will use the X test data with the Ridge Regression model to get our Y predicted data. Now using this Y predicted data and Y test value which we already have, R2 and adjusted R2 will be evaluated.

Decision Tree

Now we will go ahead and create a Decision Tree model by using X and Y train data with it. With the help of the model we will along with X test data we will predict the Y value like we did before and then taking Y test and predicted data into consideration we will get our R2 and adjusted R2 score.

Comparing the ML models

Now out of the above used ML algorithms we will compare them on the basis of R2 and adjusted R2 score. It is said that adjusted R2 is a better parameter to judge the optimality of the ML model. For linear Regression model we got adjusted R2 score around 0.55, for Lasso it was around 0.55, for Ridge it was around also 0.55. But on using the Decision Tree model we got the adjusted R2 value as 0.99 which looks very promising. On getting a Decision Tree Regression model we got a linear kind of graph which shows that the actual the predicted Y values are going hand in hand. In this model also try to find out which features and parameters are highly significant for getting those runtimes for the SGEMM GPU Kernel.

CONCLUSION

Hence we can conclude that Decision Tree Regression algorithm gives us the most optimal model out of all the used algorithms with the largest adjusted score. From this model itself we came to know what all features are important for getting the runtimes. With the information we created a bar graph showing their respective importance in calculating the runtime.