CS583A: Course Project

Ronald Fernandes

December 1, 2019

1 Summary

I participated an inactive with late submission competition of predicting the wining percentage of a match of the famous game Players Unknown Battle Ground (PUBG). The final model I chose is Deep Learning Algorithm which takes about about 4.5 million rows and has about 28 features. I implemented the Deep Neural Network using Keras and ran the code on a HP Pavilion 2015 with one Intel i7 CPU and 8 GB memory initially. It did not provide enough computation hence I moved to google colab. Since, the data was huge along with feature engineering it took a lot of execution time. I ran the model for 50 epochs which gave a decent performance. Performance is evaluated on the Mean Absolute Error. In the public leaderboard, our score is 0.02061; I rank 322 among the 1529 teams. In the private leaderboard the rank is calculated as the same rows as the public leaderboard

2 Problem Description

Problem. The problem is to predict winning percentage for PUBG matches This is a regression problem. The competition is at `https://www.kaggle.com/c/pubg-finish-placement-prediction`. PUBG is game which has been gaining popularity recently. It has about 200 million users with about 30 million active users every day. There is a huge battleground where anything goes and hence, alot of data is collected. Its important to keep track of these data to look out for cheaters.

Data. The data has about 28 features like kills, walkdistance, numberOfWeaponsAcquired, matchDuration etc. The number of training samples is $n = 4.5$ million. The output is percentage hence it is between 0 and 1.

Challenges. The training set is huge with almost all of it as numerical features. Hence, I had add many more features to in order for my results to be competitive. Secondly, as there were about 4.5 million rows per round of execution took lot of time. I also faced memory issues when running on kaggle.

3 Solution

Model. I finally chose the deep Neural Network model, a standard model with 2 Layers having units 256, 128 and 64 each having a activation function of Leaky ReLU, Dropout Layers and a

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_pred,y_test)
```

```
0.08342486858348738
```

(a) The mean square error for Decision Tree

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_pred,y_test)
```

```
0.07442004981755747
```

(b) The mean square error for Neural Network

```
y_pred = gbm.predict(X_test, num_iteration=gbm.best_iteration)
mean_absolute_error(y_pred,y_test)
```

```
Starting predicting...
0.025807159395492454
```

(c) The mean square error for Light Gradient Boosting

Figure 1: Mean Squared Errors

Batch Normalization.

Implementation. I implemented the above model using Keras with TensorFlow as the backend. My code is available at `https://github.com/wangshusen/CS583A-2019Spring/`. I ran the code on Google Colab on its GPU core. It took about 5 hours about 50 epochs.

Settings. The loss function is mean squared error. The opitmizer is Adam Optimizer. Learning Rate was default and Batch Size was 64

Advanced tricks : For Kaggle submission I had to use advance techniques like memory management by scaling the data type to a possible lower memory data type

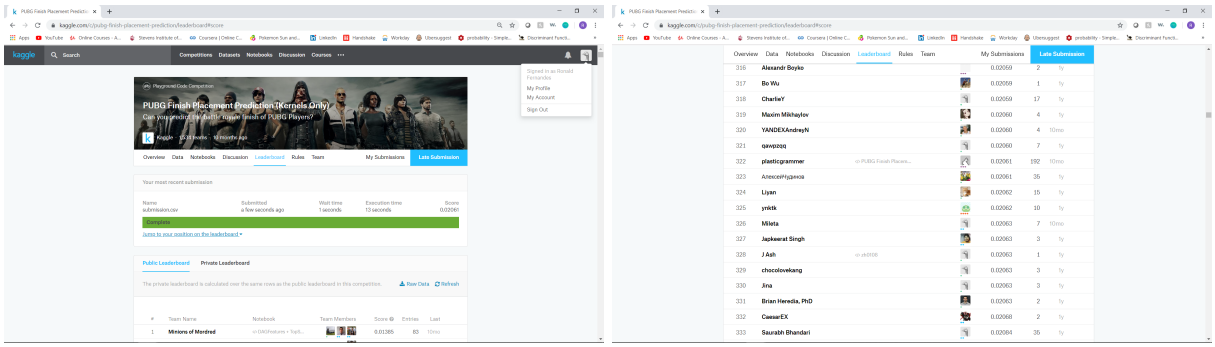
Cross-validation. I simply partition the training data to 80%-20%
Figure 1

4 Compared Methods

Light Gradient Boosting This was one of the advanced models apart from deep learning model I used. This proved the best of all the models as it is very light and executes much faster the other two models. It was a standout after 2000 rounds. It gave a mean absolute error of 0.02586 on validation set.

Fully-connected neural network. I implemented a 3-layer full connected network. The width of layers 256, 128 and 64. I also dropout and batch normalization to all layers due to size. It gave a mean absolute error of 0.0744 on the validation set.

Decision Tree Regressor I used Decision Tree Regressor provided by sklearn.This was the baseline model. I used a max depth of 100, It gave me a mean absolute error of 0.08343 on the validation set.



(a) My Score

(b) My Rank 322 Public Leader Board according to score (not updated due to inactive competition)

Figure 2: Kaggle Results

Advanced tricks. As most of the features were numerical we lot of feature engineering

- Derived Features: Many features were derived by finding out their rates and per-feature ratio
- Grouping Features: Match Ids, Group Ids and User Ids were grouped together to get aggregate features like min, max and mean
-

5 Outcome

I participated inactive competition with late submissions allowed. I achieved a score of 0.02061 on the public leaderboard. The public leaderboard was the final score and the private leaderboard was calculated on the same rows of public score. I rank 322 among 1529 teams. The screenshots are in Figure 2.