ITU AI/ML IN 5G CHALLENGE

05 Nov 2020

ID	Title	Team
ITU-ML5G-PS-013	Improving the capacity of IEEE 802.11 WLANs through Machine Learning	- M Rajasekar - Megha G Kulkarni - K Venkat Ramnan - Vishalsagar Udupi

1. Introduction

Based on the details of the challenge, problem statement and the data sets provided, several alternatives for arriving at possible solutions were debated, deliberated and finalized on trial-and-error basis as the complexity of the problem was quite high. Multiple approaches were considered initially and few were short-listed for carrying out the experimentation. The following are the approaches implemented:

(a) Artificial Neural Networks (ANN) approach
(b) K-Nearest Neighbor (KNN) approach
(c) Random Forest method

2. Data cleaning and Preprocessing

The data given has many features. For the below approaches, only certain features are selected for training. The features that are static are not considered for training. In order to select the best features, the P-values of each feature was considered. To make final conclusions on which features to be selected for training, heat maps are implemented. The features considered from the dataset for data cleaning are x(m), y(m), primary_channel, min_channel_allowed, max_channel_allowed, RSSI, and SINR. The heat map for features listed above is shown in Figure 1. We can infer that many of the features are independent (desirable for training ML models) and even though RSSI and SINR are highly correlated, they are considered as they affect the label Throughput to a great extent.

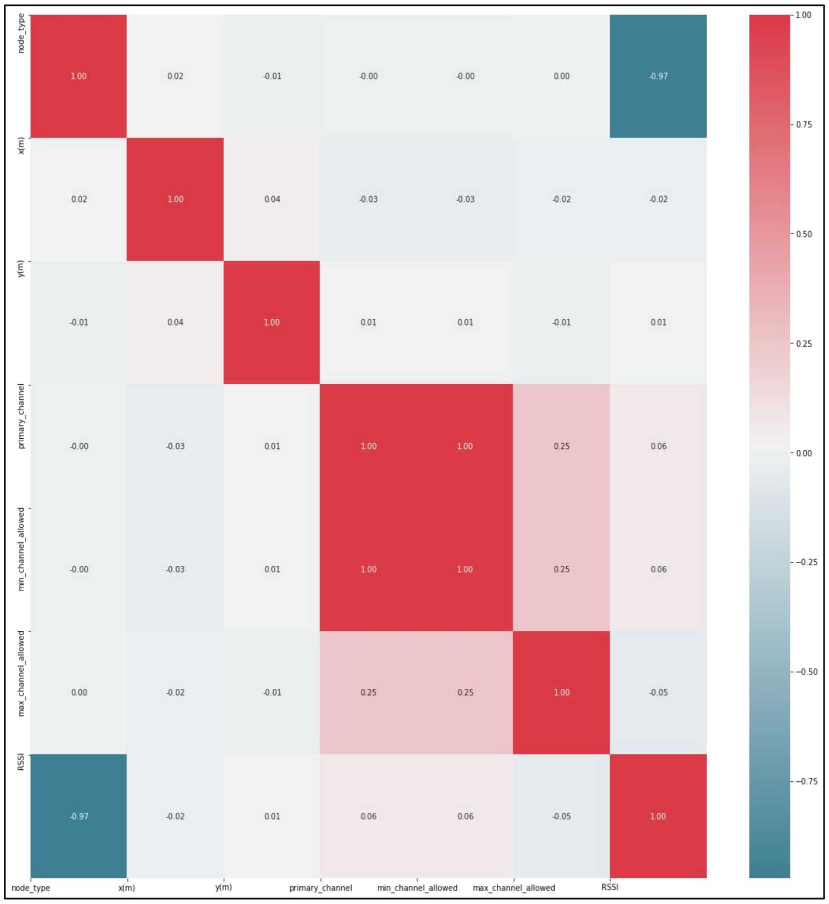


Figure 1: Heat map of data

3. The given approaches have been implemented with Python language. The libraries used for building the neural network were TensorFlow and Keras. For the other methods, scikit-learn has been used. The preprocessed data is used for the training process. For training, the features were: **x coordinate, y coordinate, primary channel, min channel allowed, max channel allowed, SINR, RSSI**. The label to be predicted was throughput. The given approaches had the model designed in order to take in only STA values both for training and testing. To get the AP values after predicting the STA values of the test data, a separate function is used in order to sum the values of throughputs of STAs to get the value of the respective AP. To the given approach, the data is fed directly as CSV files using the pandas library.

4. The data is transformed into a standard scaled form before fed into the network (model). This transforms the data such that its distribution will have a mean value of 0 and a standard deviation of 1. The StandardScaler function provided by python is used. The main reason for using this is to normalize the features individually so that each

column(feature) has a mean of 0 and a standard deviation of 1. This standard scaled data is used for all the given approaches.

5. Approach #1 : Artificial Neural Network Approach(ANN)

(a) Motivation

From the problem statement given, as we are required to predict values (throughput in this case), this problem statement falls under the category of regression. Usually, for regression type problems, linear regression, or multilinear regression is implemented. Linear regression is very powerful when there exists a linear equation relating to the features and the labels. Neural networks can in principle model these nonlinearities automatically. In neural networks, understanding the underlying physics of the phenomenon is not required, and forming a physical model (like in linear regression) is not required. When the performance required is on the lines of 'goodness of fit' , using neural networks is always the better option. Neural networks use the concepts of gradient descent and backpropagation in order to minimize the errors obtained after each epoch. The artificial neural network approach was considered mainly for the following reasons as given below:

i. ANN can find the hidden relation between the features(correlations).In the given problem statement, this proves to be very useful because one feature may not be direct to the target variable to be predicted, but it might be related to another feature. This interrelation between variables plays a very important role in determining the target variable. This hidden relation is found by the ANN.
ii. ANN has high dimensionality, thus allowing us to create better models by adding more layers. The trials at the beginning for the given statement used only one hidden layer, but as the number of layers increased, the error in validation data reduced. But it must be noted that beyond a certain number of layers, the model will give the same error.
iii. There is no need for feature engineering. Although the data being fed to all the approaches here is filtered data, the ANN is capable of deciding after certain iterations, which features really contribute to the prediction of the throughput. Thus instead of using all the features provided in the dataset, it will be able to

predict with much fewer features thus making the overall model faster and computationally less expensive.

iv. It is easier to play around with the hidden layers and the hyperparameters, thus it can be tuned to give better results. The number of nodes in each of the hidden layers was changed and experimented with to get better results.

v. ANN has the ability to learn by reducing errors in every iteration, thus making it the best model to obtain high-quality results efficiently. As it uses the concept of backpropagation, it reduces the error after every epoch. After a given set of epochs and batch sizes, the results improved each time after every epoch.

(b) Description of Approach

The model for this approach is built using Tensorflow and Keras library. The model mainly consists of one input layer, 7 hidden layers, and 1 output layer. As the following problem statement is a regression type of problem, the two activation functions preferred are Linear activation function and Rectified Linear (ReLU) function. In this approach, the hidden layers are all ReLU activated. In each of the first six hidden layers, there are 1024 nodes. For the seventh hidden layer, there are 512 nodes. The standard scaled data is fed for training purposes. The pictorial representation of the model is as below:

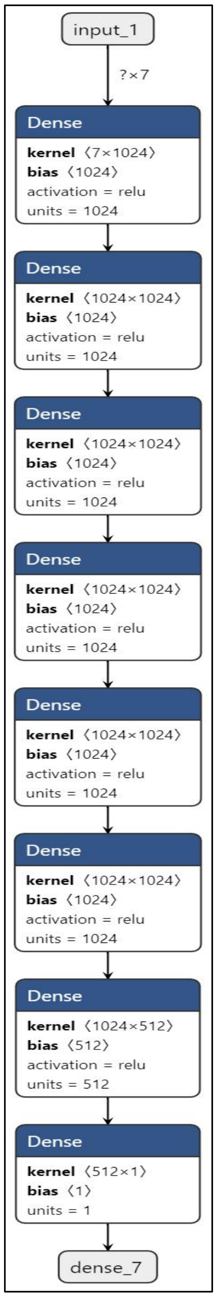


Figure 2. Flowchart of the ANN model
Note: The '?' symbol in the above diagram indicates 'None'.

All the hidden layers are Dense layers which are ReLU activated and were placed in successive order as shown in Figure 2. In the figure, the kernel and bias for each of the layers have also been mentioned. The optimizer that is used is the Adam optimizer as it uses the best properties of the RMSProp optimizer as well as the

AdaGrad optimizer. It also is used as it outperforms as the gradients become sparser. For training the model, the batch size considered was 250 and the number of epochs was 1000. The metrics for error calculation were the mean square error(MSE) and the mean absolute power(MAE). The root of the MSE values was taken in order to get the RMSE value.

(c) **Visualizations of training**

Before the testing data was available, the above model was trained by splitting the preprocessed training data into 80% for training and 20% for validation. The model structure is as mentioned above. The following figures 3 and 4 show the MSE and MAE for the validation and training of the data. The X-axis represents the number of epochs and the Y-axis represents the value of MSE or MAE.

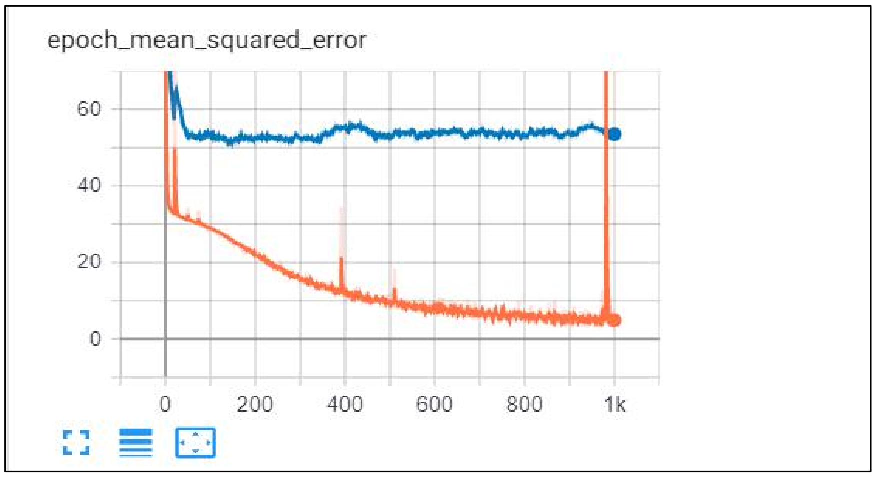


Figure 3. The plot of the number of epochs vs MSE (80-20 split)

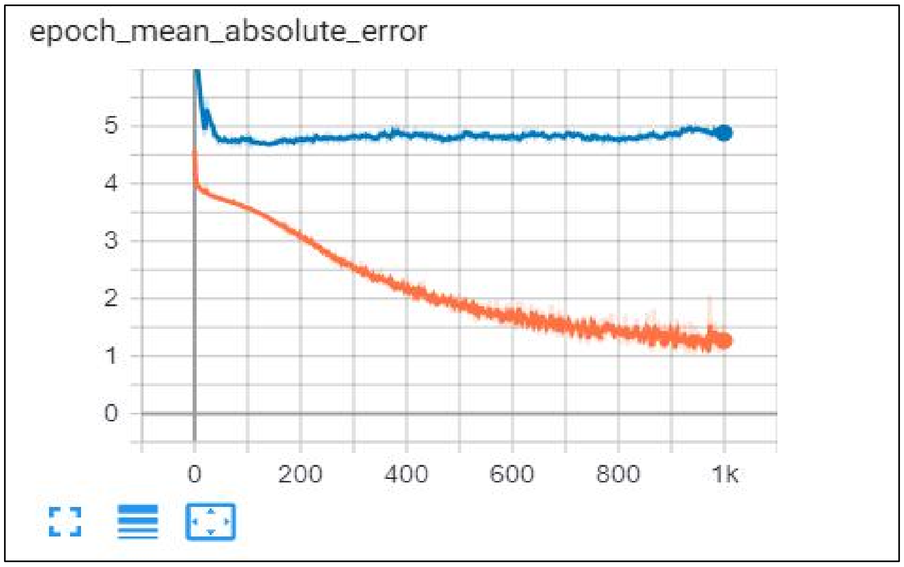


Figure 4. The plot of the number of epochs vs MAE (80-20 split)

For the above 80 - 20 split, the RMSE for validation data obtained is 5.5 and the MAE is close to 4.8. The RMSE for training data obtained is 1.42 and the MAE is close to 1. The R2 score was 0.78. These results varied by + or - 0.5 in the worst case.

After the testing data is provided, the complete preprocessed dataset is used to train the model. The structure of the model is the same. The following are the graphs for MSE and MAE for training data. The X-axis represents the number of epochs and the Y-axis represents the value of MSE or MAE.

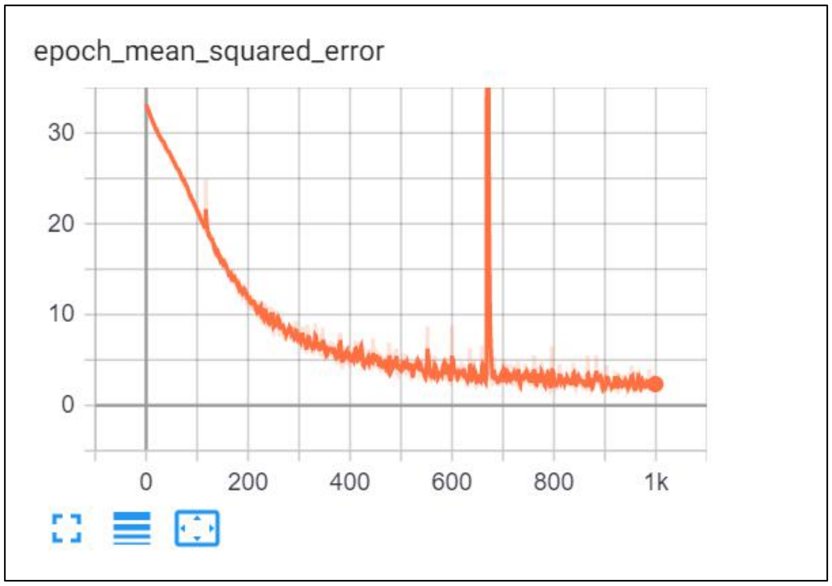


Figure 5. The plot of the number of epochs vs MSE

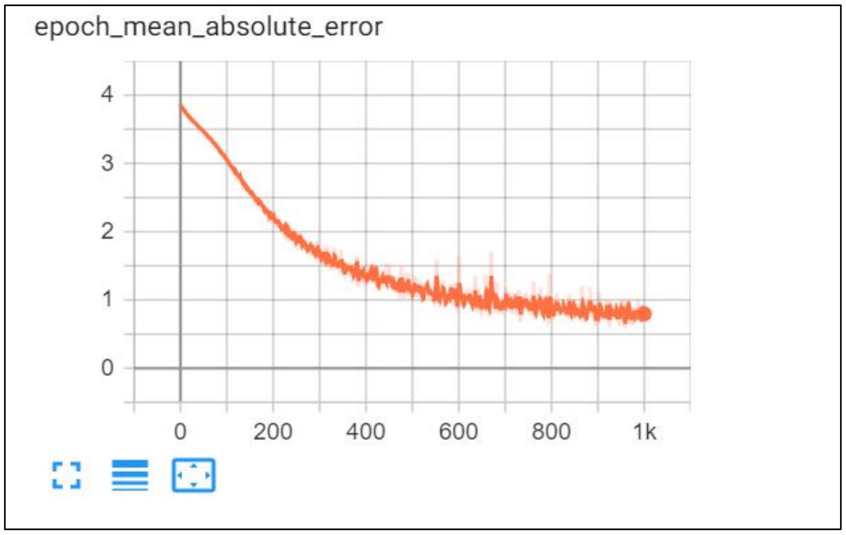


Figure 6. The plot of the number of epochs vs MAE

After training the RMSE is close to 3.12 and the MAE value is close to 2.8. The R2 score is close to 0.9. Once again during different trials the values varied by + or - 0.4.

6. Approach #2 K Nearest Neighbors Approach

(a) Motivation:

Neighbors-based regression can be used in cases where the data labels are continuous rather than discrete variables. In the given problem statement Throughput (label) takes continuous values, it's appropriate to use Neighbors-based regression. The label assigned to a query point is computed based on the mean of the labels of its nearest neighbors.

(b) KNeighborsRegressor (from scikit learn)

It implements learning based on the k nearest neighbors of each query point, where k is an integer value specified by the user. In the given problem statement, if we consider each BSS as a cluster and STA or AP as a datapoint it's appropriate to take into consideration the features of its nearest neighbors as neighbors affect the throughput of the considered datapoint. The KNeighborsRegressor is used to do so.

(c) Brief Description About KNN Approach

The following approach is implemented using Python. Necessary libraries include-Pandas, Scikit Learn, Numpy, Regex, Glob. The number of neighbors selected is 10, as the number of STAs present in each of the deployment ranges from 5 to 20. Also, this method was experimented with different numbers of neighbors, but with 10 as neighbors count the RMSE and MAE turned out to be the best among the rest. Then the features and labels from the large training CSV are fed to this model. Initially, the dataset was split into an 80:20 ratio for training and testing respectively. Once the test dataset was provided, we used the initial dataset completely for training and the test dataset was used for testing.

RMSE and MAE achieved for the above approach:

RMSE = 6.88 and MAE = 4.67

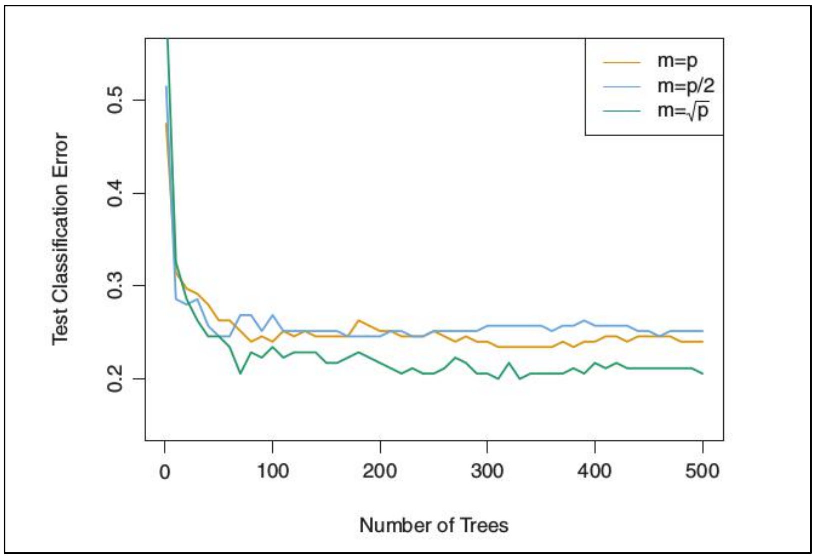
Once the model has been trained, the test data is fed, one CSV after the other to get Throughput values. These values are stored as separate CSV files for each of the 400 test deployments.

7. **Approach #3 Random Forest Method**

(a) Motivation. Random Forest uses the bootstrap method in building decision trees, which provide an improvement over bagged trees by way of a small tweak that decorates the trees. When building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. A fresh sample of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$ — that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors.

In splitting process, random forest is not even allowed to consider a majority of the available predictors. Suppose that splitting process consider a strong predictor, consequently all of the bagged trees will look quite similar to each other and the predictors from the bagged trees will be highly correlated. This state is not lead to reduction variance.

Of 2 paragraphs above we can analyze, random forest forcing each split to consider only a subset of the predictors therefore, on average $(p - m)/p$ of the splits will not even consider the strong predictor, and so other predictors will have more of a chance. Using a small value of *m* in building a random forest will typically be helpful when we have a large number of correlated predictors.



Results from random forests for the 15-class gene expression data set with *p*=500 predictors. The test error is displayed as a function of the number of trees. Each colored line corresponds to a different value of *m*, the number of predictors available for splitting at each interior tree node. Random forests (*m*<*p*) lead to a slight improvement over bagging (*m*=*p*). A single classification tree has an error rate of 45.7 %.

(b) How Does Random Forest Work?

The random forest algorithm can be summarized in four simple steps:

1. Draw a random bootstrap sample of size n (randomly choose n samples from the training set with replacement).
2. Grow a decision tree from the bootstrap sample. At each node:
    a. Randomly select d features without replacement.
    b. Split the node using the feature that provides the best split according to the objective function, for instance, by maximizing the information gain.

3. Repeat the steps 1 to 2 ‘k’ times
4. Aggregate the prediction by each tree to assign the class label by majority vote or average

(c) Description of the approach:

This method is mainly implemented using scikit-learn library. The training data is transformed using Standard Scaler. And it is the CSV obtained from combining all the pre-processed training CSVs (from different deployments in different scenarios). The testing data is pre-processed in the same way as training data. In this approach, we have used max depth of 10 as one of the parameters for the random forest regressor. Next the model is trained using the transformed data, and the predictions are made. Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are used as metrics to find the accuracy of the Regressor. In the *predictor* function all the test CSV files are fed to the model for predicting the throughput. The output files are saved in a zip file and stored as per the given format. After the compilation of the program is completed, the directory of the test files is filled with new CSV files with the same name as each corresponding test file, but with *'Ap_throughput'* as an extension to their names. These are the output throughput files. In this approach we have used 2 different ways of training the model with our data (pre-processed version). 1st approach is a 80 – 20 based division of the dataset where 80% of our data is used for training and 20% is used for testing. 2nd approach is based on a complete 100 % of our data is fed as a training data.

(d) Final results of Random Forest Approach

RF - Approach 1	RF - Approach 2
Random forest for 100 percent training data: **RMSE= 5.39** **MAE=3.65**	Random forest for 20% of testing data: **RMSE of 6.91** **MAE of 4.88**