Deep Learning

Assignment 1

Task 3 Report

Abstract

To develop a connected neural network to process the spam data from the dataset communities, split the train and test sets and split the train sets again into validation and test sets, train the developed model and finally to measure the accuracy by regression classification.

Problem Statement

Communities is a dataset collection with 1994 instances of socio economic data and 128 attributes with some missing data. The data is loaded into the model and is split into train and test sets. Then the train set is again split into train and validation set. Finally the accuracy of model across the validation and the test sets is obtained.

Proposed Solution

The purpose of the classification is to determine crime rate from the given information. In case of model design, the activation functions are relu for all the layers. Finally the optimizer used is rmsprop.

Loading the data:

The data is obtained in the .DATA format. The attributes were found the names file with certain metadata. The attributes were then popped from the file by removing the unnecessary data. In case of the data file, there were certain missing values that were replaced and the corresponding columns were removed.

Regression classification:

Certain amount of data is fed into the model and it is used to determine the crime rate. Thus the regression model is used for the purpose of classification and for finding.

Model design:

Activation = relu

RELU function is used for activation. The purpose of this is to obtain the maximum value when the given input is greater than zero and to substitute zero when the given input is negative. RELU activation gives better performance than other activation functions such as sigmoid and tanh in the hidden layers.

Optimizer = rmsprop

Rmsprop is used in this model. It converts data to mini batches and iterates through them to learn from it. It is similar to the gradient descent algorithm. By this way it is easy to tune the model more accurately. This optimizer is more suitable for huge datasets.

All the above functions were implemented for a k fold times. Here k = 6.

Implementation Details

Implemented using anaconda and Jupyter.

Implementation issues and processes includes:

1. Since the datasets were huge, it was necessary to increase the number of hidden layers to give better results. It was also necessary to increase the number of iterations through the training set to make the model better.
2. The task was implemented in Jupyter notebook. It was found that Jupyter stored the values of the models and at each hyper tuning the kernel was restarted.
3. The datasets were found to be huge and thus processing the data with so many epochs and hidden layers were difficult.
4. The process of running such an entire data on the local CPU was difficult.

Evaluation metrics

Initially the validation data obtained from the train data was ran across the model and finally the test data was ran across the model which was tuned with the test label.

Out of 1395 train data 1163 were considered as train set and 232 were taken as validation data.

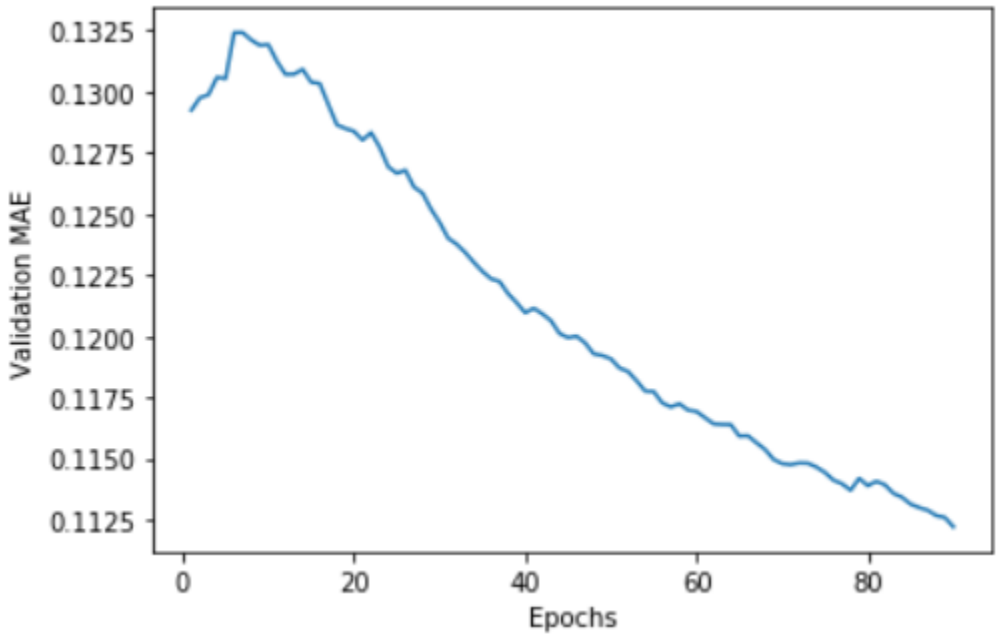
The accuracy obtained were as follows:

Validation set accuracy at each epoch:

```
processing fold # 0
Train on 1163 samples, validate on 232 samples
Epoch 1/100
1163/1163 [==============================] - 1s 649us/step - loss: 0.0919
- mae: 0.2196 - val_loss: 0.0688 - val_mae: 0.1927
Epoch 2/100
1163/1163 [==============================] - 0s 220us/step - loss: 0.0434
- mae: 0.1528 - val_loss: 0.0477 - val_mae: 0.1582
Epoch 3/100
1163/1163 [==============================] - 0s 219us/step - loss: 0.0288
- mae: 0.1261 - val_loss: 0.0873 - val_mae: 0.2361
Epoch 4/100
1163/1163 [==============================] - 0s 234us/step - loss: 0.0286
- mae: 0.1214 - val_loss: 0.0497 - val_mae: 0.1559
Epoch 5/100
1163/1163 [==============================] - 0s 208us/step - loss: 0.0203
- mae: 0.1055 - val_loss: 0.0354 - val_mae: 0.1425
Epoch 6/100
1163/1163 [==============================] - 0s 222us/step - loss: 0.0194
- mae: 0.0984 - val_loss: 0.0408 - val_mae: 0.1513
Epoch 7/100
1163/1163 [==============================] - 0s 224us/step - loss: 0.0156
- mae: 0.0936 - val_loss: 0.0348 - val_mae: 0.1393
```

……

The training loss and the validation loss across the epochs were plotted as follows:



Observations from the graph:

The mean absolute error reduces in the validation set in accordance with the iteration. Thus the model tunes itself better at each iteration.

Overall mae is:

```
0.12323356628417968
```

Test set accuracy at the epoch:

599 test samples of the three classes were taken and the result obtained from the model were as follows:

```
599/599 [==============================] - 0s 64us/step
0.11493252217769623
```

Finally the accuracy of the test sample is obtained to be 90%. This accuracy is obtained after man y hyper tunings by

1. Changing the number of layers of the model
2. Changing the epochs
3. Changing the batch size
4. Choosing the correct optimizer.

References:

1. Keras code given by the professor.
2. https://github.com/vbordalo/Communities-Crime/blob/master/Crime_v1.ipynb