COL774 Assignment 3

Mustafa Chasmai | 2019CS10341

October 2021

1. **Decision Trees (and Random Forests)**

(c) **Random Forests**

A random forest involves ensembling multiple decision trees together and using the predictions from each decision tree as votes to get a single final prediction. Even if the individual decision trees are weak learners, the ensembled model is expected to perform well, since each decision tree is treated independently.

For the assignment, the random forest implementation of sklearn was used. Each decision tree was allowed to grow to its maximum depth, and the number of such trees ensembled was taken as a hyperparameter, along with the number of samples to split and the max features.

After performing grid search over the given ranges of hyperparameters, the **optimal hyperparameters** obtained are:

i. number of estimators: 450
ii. max features: 0.7
iii. min samples split: 2

Some other parameters and options explored include the criterion, max depth and class weight. The criterion allows us to control how the individual decision trees chose best splits. Other than the commonly used entropy criterion, I explored the gini criterion, and its comparision with entropy. The max depth is the depth upto which each individual decision tree is allowed to grow. This can be useful to prevent overfitting and improve time to train. The class weight can be very important in the case of highly class-imbalanced data, and can allow mitigating the problem of learning a majority predictor.

The different evaluation metrics on these optimal parameters are:

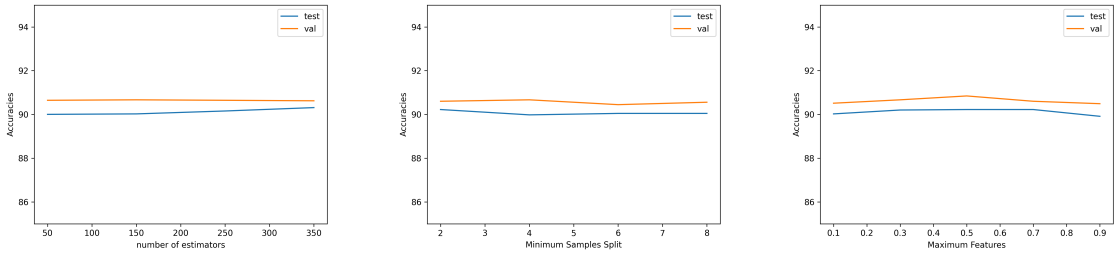
i. **Training Accuracy**: 100.00%
ii. **Out of Box Accuracy**: 90.60%
iii. **Validation Accuracy**: 90.60%
iv. **Testing Accuracy**: 90.22%

The performance with random forests is expected to be better than that of decision trees because of ensembling. Ensembling multiple classifiers together tends to provide more reliable predictions, giving performance better than each of the individual models.

One observation is that the training accuracy is 100% while the validation accuracy is only 90.6%. This indicates that the model is actually overfitting, and the low test accuracy supports this claim.

(d) **Parameter Sensitivity Analysis**

The plots for accuracies as each parameter is changes, keeping others constant are shown below.



(a) Varying number of estimators (b) Varying Min Samples (c) Varying Max Features

Figure 1: Test and validation accuracies after varying each individual hyperparameters

Observations about the sensitivities of the model to the different hyperparameters are:

i. **Number of Estimators** The accuracy increases as more decision trees are ensembled. The sensitivity to the number of estimators is relatively quite high compared to the othe parameters.

ii. **Min Samples** As the min samples increases, there is initially a small dip in the performance after which it seems to saturate. The model may be quite insensitive to the min samples compared to the other hyperparameters.

iii. **Max Features** As the max features increases, there is a clear increasing trend in the accuracies, after which they seem to saturate and dip a little. The changes are more pronounced here, and so the model is more sensitive to changes in this.

2. **Neural Networks**

(a) After parsing the training and test files, the features were converted into one hot encoding and stored in csv files "test_transformed.csv" and "train_transformed.csv". The features had a dimension of 85, and the labels had a dimension of 10.

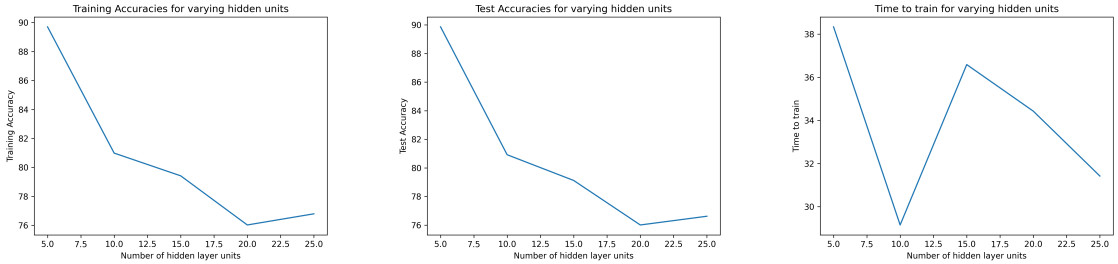
(b) A generic NeuralNetwork class was implemented that expects the input, output and hidden dimensions, along with the activation and batch size options. This model was trained with a trainer that handled data loading, SGD training and post processing.

(c) The asked values for the different number of hidden units are tabulated below.

Number of hidden units	Training Accuracy	Test Accuracy	Time to train
5	89.70%	89.86%	38.34s
10	80.99%	80.92%	29.15s
15	79.42%	79.12%	36.59s
20	76.03%	76.01%	34.43s
25	76.80%	76.62%	31.42s

Table 1: Variation with number of hidden units

And the trends can be seen in the plots below.



(a) Training Accuracy (b) Test Accuracy (c) Training Time

Figure 2: Variation of different metrics with changing number of hidden units

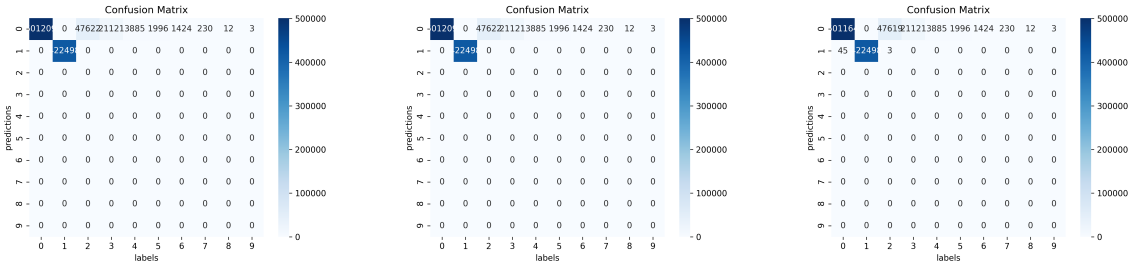
The stopping criteria used was simply a set number of epochs for which the training will run. With this, a clear decreasinng trend can be observed for both the training and test accuracies. From this, one may interpret that although the larger units in the hidden layer have greater representation capacity, they are also slower to train, and need more epochs to be fully trained.

The confusion matrices are reported next in Fir 3.

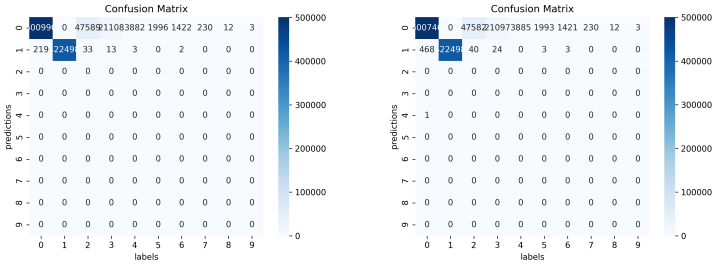
(d) The asked values for the different number of hidden units in the case of adaptive learning rate are tabulated in Table 2.

And the trends in the case of adaptive learning rate can be seen in the plots in Fig 4, while the confusion matrices are in Fig 5

The training and test accuracies are more uniform here, compared to the constant learning rate. Thus, using the adaptive learning rate makes the training more stable. Also the times here are comparatively lower than the ones observed in constant learning rate, implying that the adaptive learning rate actually allows the model to train faster.



(a) 5 hidden units (b) 10 hidden units (c) 15 hidden units



(d) 20 hidden units (e) 25 hidden units

Figure 3: Variation of different metrics with changing number of hidden units

Number of hidden units	Training Accuracy	Test Accuracy	Time to train
5	92.33%	92.37%	30.67s
10	92.33%	92.37%	26.89s
15	92.33%	92.37%	28.89s
20	91.96%	92.03%	29.62s
25	92.20%	92.24%	30.38s

Table 2: Variation with number of hidden units

(e) The training and test accuracies as well as time for both the activations are:

i. **ReLU**
   A. Training: 47.17%
   B. Test: 46.94%
   C. Training: 94.463s

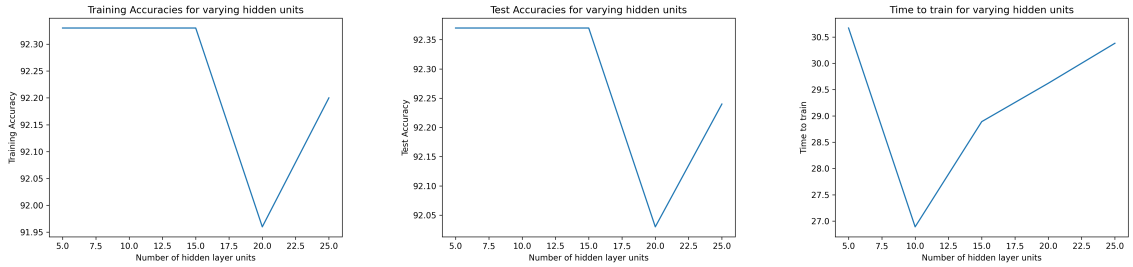
ii. **Sigmoid**
   A. Training: 73.12%
   B. Test: 72.63%
   C. Training: 130.96s

And the confusion matrices corresponding to these can be seen if Fig 6.

(f) The MLPClassifier from scikit-learn gave the results:

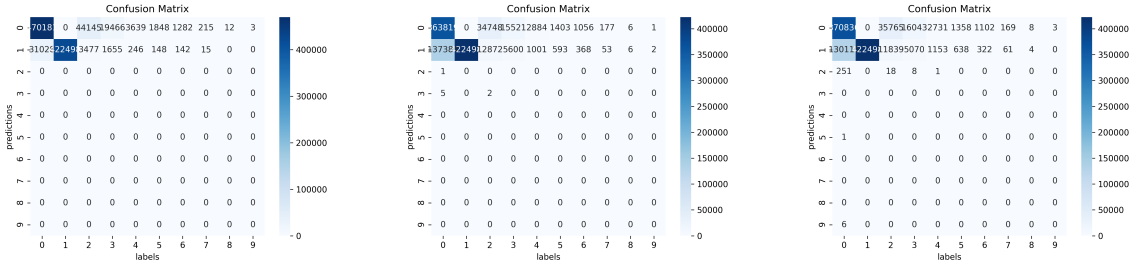
i. Training: 100%
ii. Test: 93.3%

These accuracies are quite better compared to custom implementations, and the training time is also lesser here. This is probably because of highly optimised and efficient implementations in these libraries.

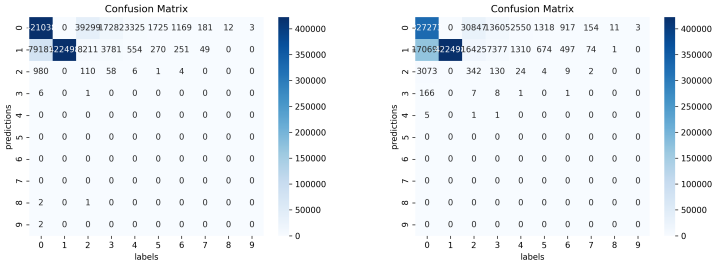


(a) Training Accuracy (b) Test Accuracy (c) Training Time

Figure 4: Confusion Matrices for changing number of hidden units

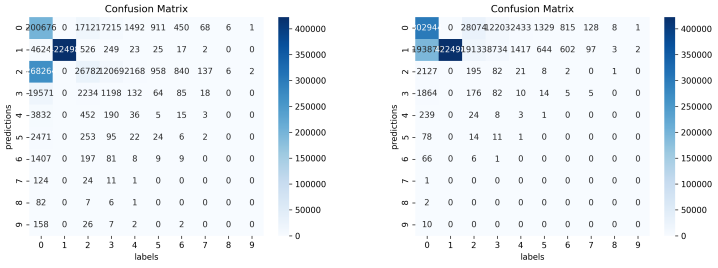


(a) 5 hidden units (b) 10 hidden units (c) 15 hidden units



(d) 20 hidden units (e) 25 hidden units

Figure 5: Confusion Matrices for changing number of hidden units in the case of adaptive learning rate



(a) With ReLU activation (b) With Sigmoid activation

Figure 6: Confusion Matrices for different activations