COL774: Assignment-3 Report

Vatsal Varshney (2021MT10700)

October 31, 2023

Contents

1 Decision Trees and Random Forests **2**
- (a) Decision Tree Construction 2
- (b) Decision Tree One Hot Encoding 2
- (c) Decision Tree Post Pruning 3
- (d) Decision Tree sci-kit learn 4
- (e) Random Forests 5

2 Neural Networks **6**
- (a) Implementation 6
- (b) Single layer of varying size 6
- (c) Varying depth of network 7
- (d) Adaptive Learning 7
- (e) ReLU Activation 8
- (f) Sklearn Neural Network (`MLPClassifier`) 9

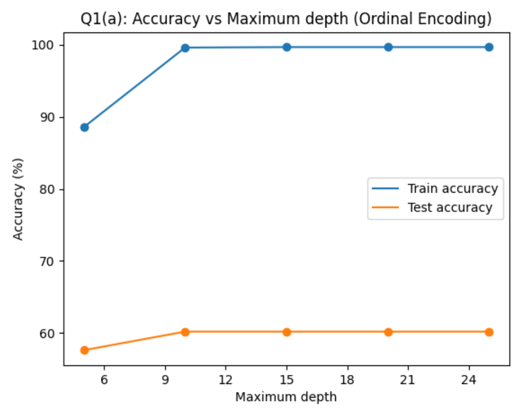
1 Decision Trees and Random Forests

(a) Decision Tree Construction

Decision tree classifier was implemented as required.

Maximum Depth	Training accuracy (%)	Test accuracy(%)
5	88.5652	57.6008
10	99.6295	60.1861
15	99.6934	60.1861
20	99.6934	60.1861
25	99.6934	60.1861

Table 1: Accuracy vs maximum depth in part (a)



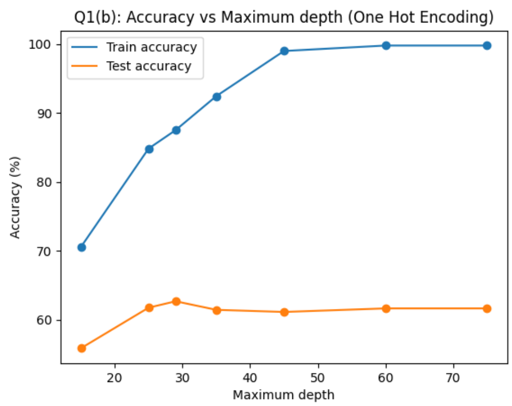
- Training time for all 5 trees combined was 1m44s.
- Testing accuracy was found to be increasing with increasing depth, and maximum testing accuracy was found to settle at only 60.19%.
- Only win accuracy was 49.64% and only lose accuracy was 50.36%
- Since the training accuracy was very high and testing accuracy was not very far from baseline accuracy, the models were highly overfitted.

(b) Decision Tree One Hot Encoding

- Training time for all 7 trees combined was 3m36s.
- As max depth increased, testing accuracy first increased, then achieved a maximum of 62.67% at depth 29, then slightly decreased and settled at 61.63%.
- As compared to (a), the testing accuracy improved slightly, and the maximum training accuracy was achieved at a higher depth.

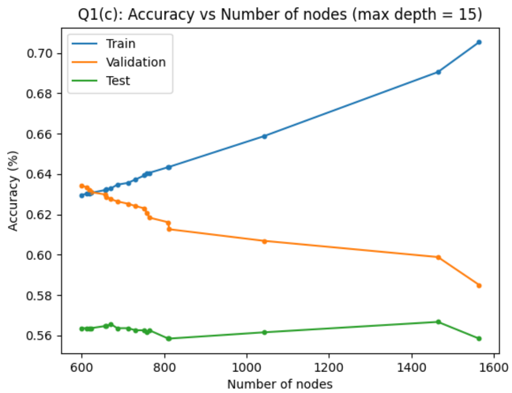
Maximum Depth	Training accuracy (%)	Test accuracy(%)
15	70.5379	55.8428
25	84.8345	61.7373
29	87.5176	62.6680
35	92.4492	61.4271
45	99.0034	61.1169
60	99.7956	61.6339
75	99.7956	61.6339

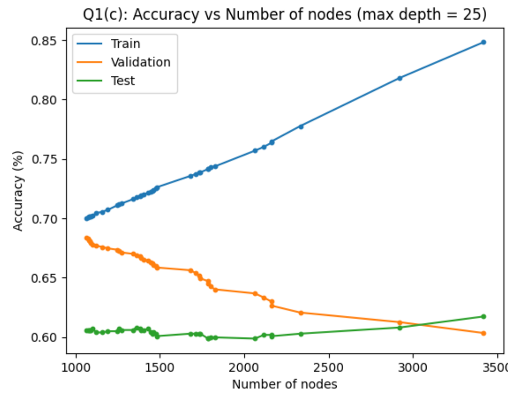
Table 2: Accuracy vs maximum depth in part (b)

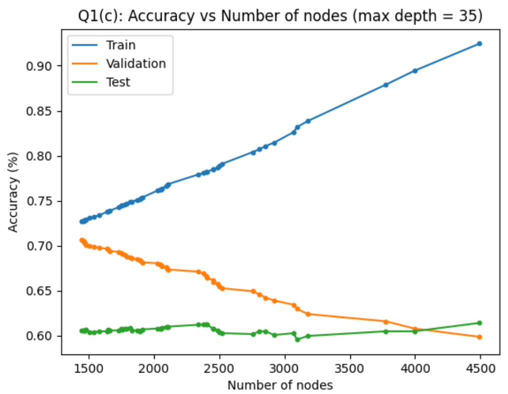


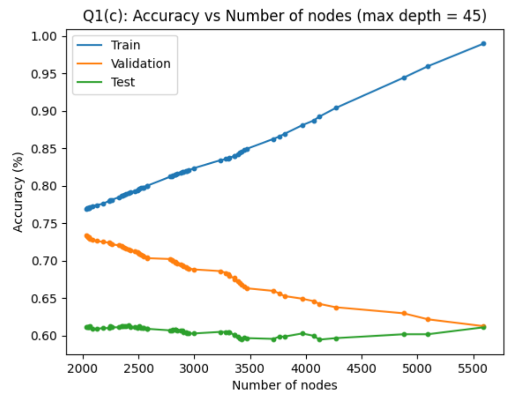
(c) Decision Tree Post Pruning

- Pruning the 4 trees took close to 1hr combined.
- As the tree was successively pruned, the training accuracy decreased as expected. However, contrary to our expectation, the testing accuracy didn't increase, it fluctuated around the initial level only.
- The reason for this behaviour could be small sizes of the data (training, validation and test). It could also be that the validation data and test data are too different from each other.









(d) Decision Tree sci-kit learn

- DT with `criterion='entropy'` and rest arguments default gave training accuracy of 100% and test accuracy of 63.39%, which is already better than our implementation.

- Varying the `max_depth` parameter, it was found that test accuracy was highest for `max_depth=35` but validation accuracy was highest for `max_depth=45`

`max_depth`	Train accuracy (%)	Validation accuracy (%)	Test accuracy (%)
15	71.3045	58.5057	60.9100
25	85.4606	60.1149	63.5988
35	94.5190	61.7241	64.9431
45	**99.5401**	**63.1034**	**63.5988**

Table 3: `max_depth` vs Accuracy in part (d)

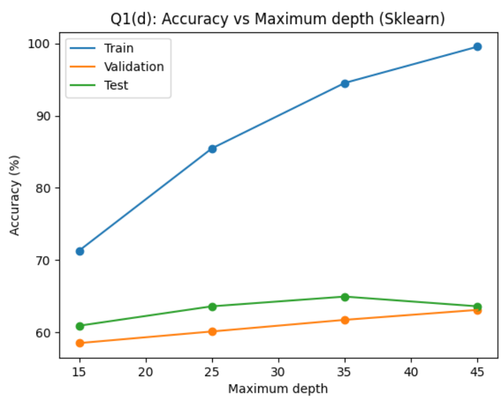
- Comparing the model of `max_depth=45` of this part with that of part (b), we observe that this one performs better, with test accuracy higher by about 2% and train accuracy higher by 0.5%.

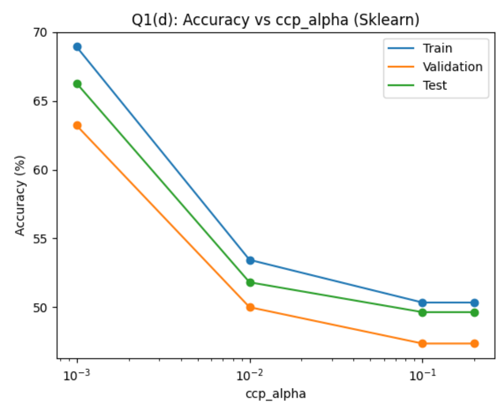
- Varying the `ccp_alpha` parameter, it was found that all the accuracies were highest for `ccp_alpha=0.001`

ccp_alpha	Train accuracy (%)	Validation accuracy (%)	Test accuracy (%)
0.001	**68.9408**	**63.2184**	**66.2875**
0.01	53.4432	50.0000	51.8097
0.1	50.3386	47.3563	49.6381
0.2	50.3386	47.3563	49.6381

Table 4: `ccp_alpha` vs Accuracy in part (d)

- Comparing the model of `ccp_alpha=0.001` of this part with the model of `max_depth=45` of part (b), we observe that although the model of (b) has higher training accuracy, but the model of (d) has higher test accuracy, which means model of (b) overfits.





(e) Random Forests

- Parameter space for grid search:
  - `'n_estimators': [50, 150, 250, 350]`
  - `'max_features': [0.1, 0.3, 0.5, 0.7, 0.9]`
  - `'min_samples_split': [2, 4, 6, 8, 10]`
- Time taken for grid search: 13m11s
- Best parameters:
  - `'n_estimators': 150`
  - `'max_features': 0.7`
  - `'min_samples_split': 6`
- For the optimal parameters:
  - Training accuracy: 99.46 %
  - Out-of-bag accuracy: 71.85 %

- Validation accuracy: 98.85 %
- Test accuracy: 72.60 %

- As compared to (b) and (c), there is a significant improvement in validation accuracy. which is understandable as the validation set was also used now. More importantly, there was an increase of about 10% in test accuracy too.

2 Neural Networks

(a) Implementation

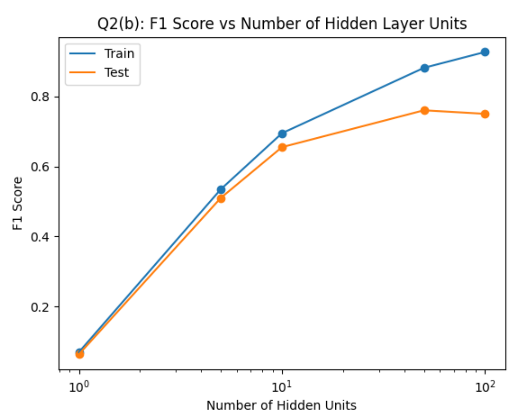
The Neural networks classifier was implemented as specified

(b) Single layer of varying size

- Stopping criteria: absolute difference of loss of consecutive epochs less than 5e-3, or 1000 epochs, whichever comes first.
- Training time for all 5 models combined: 25m35s
- For size=1, the model predicted all examples to the same class, which resulted in very low F1 score.
- As the size of hidden layer increased, all parameters were observed to be increasing, except for size=100 in which the parameters for test data slightly decreased. So the best performance was observed for **size=50**

#Hidden units	Train precision	Test precision	Train recall	Test recall	Train F1	Test F1
1	0.2000	0.2000	0.0418	0.0374	0.0692	0.0630
5	0.5614	0.5423	0.5613	0.5365	0.5348	0.5101
10	0.7078	0.6694	0.7119	0.6706	0.6948	0.6551
50	0.8818	0.7612	0.8828	0.7661	0.8817	0.7603
100	0.9276	0.7517	0.9302	0.7599	0.9270	0.7502

Table 5: Parameters obtained while varying the size of the single hidden layer

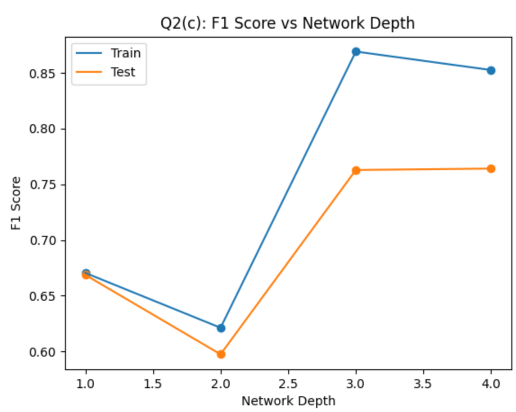


(c) Varying depth of network

- Stopping criteria: absolute difference of loss of consecutive epochs less than 1e-3, or 200 epochs, whichever comes first.
- Training time for all 4 models combined: 53m40s
- Comparing with part (b), it didn't perform as good. This is because of the large size of the hidden architecture, which significantly increased the running time and restricted us to keep the max number of epochs to be lower. If it did run till 1000 epochs, maybe it would have performed well too.
- The variation of F1 score with depth was also not monotonic, though larger networks tend to show better performance.

Architecture	Train precision	Test precision	Train recall	Test recall	Train F1	Test F1
[512]	0.6776	0.6783	0.6824	0.6868	0.6705	0.6688
[512, 256]	0.6908	0.6619	0.5907	0.5743	0.6213	0.5975
[512, 256, 128]	0.8685	0.7609	0.8787	0.7808	0.8691	0.7629
[512, 256, 128, 64]	0.8530	0.7624	0.8612	0.7825	0.8525	0.7641

Table 6: Parameters obtained while varying depths of the network



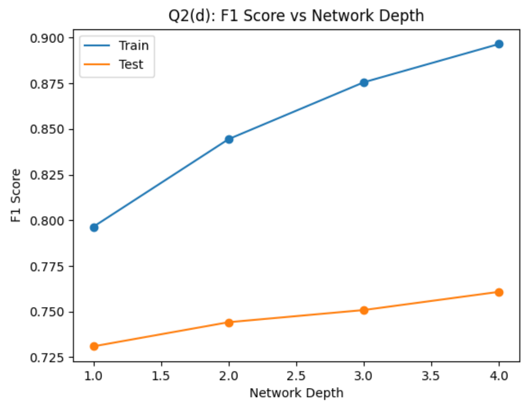
(d) Adaptive Learning

- Stopping criteria: absolute difference of loss of consecutive epochs less than 1e-4, or 200 epochs, whichever comes first.
- Tolerance for stopping had to be reduced because the step sizes became smaller as the training progressed.
- Training time for all 4 models combined: 53m48s
- Comparing with part (c), we see an improvement for smaller architectures (depth 1 and 2), but there was not much of a difference for larger architectures (depth 3 and 4). However, the results now look consistent unlike part (c).

- Adaptive learning did not make the training time per epoch less, but there was improvement of performance in some cases, so we can say that it takes less number of epochs to learn the same amount of information, so the learning may have become faster in some sense.

Architecture	Train precision	Test precision	Train recall	Test recall	Train F1	Test F1
[512]	0.8027	0.7352	0.8009	0.7308	0.7964	0.7309
[512, 256]	0.8451	0.7425	0.8438	0.7467	0.8444	0.7442
[512, 256, 128]	0.8762	0.7510	0.8781	0.7526	0.8756	0.7508
[512, 256, 128, 64]	0.8974	0.7613	0.8967	0.7618	0.8965	0.7608

Table 7: Parameters obtained with Adaptive learning while varying depths of the network

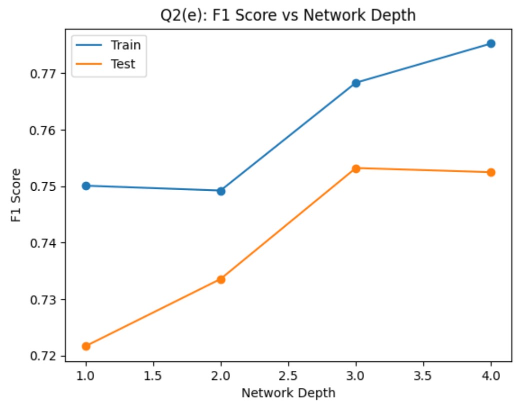


(e) ReLU Activation

- Stopping criteria: 200 epochs
- Training time of all 4 models combined: 51m16s
- Simple ReLU activation function gave rise to dying ReLU problem, in which a lot of activation units became 0 due to the nature of ReLU function. This caused the model to fail and output the same class for all inputs. To avoid this, I have used Leaky-ReLU, which is max(1e-4*x,x). This is not a bad approximation as the value in negative side of x is very close to 0.
- As compared to part (d), the performance was more or less the same.

Architecture	Train precision	Test precision	Train recall	Test recall	Train F1	Test F1
[512]	0.7463	0.7195	0.7771	0.7540	0.7501	0.7217
[512, 256]	0.7561	0.7400	0.7576	0.7404	0.7492	0.7336
[512, 256, 128]	0.7707	0.7529	0.7697	0.7624	0.7683	0.7532
[512, 256, 128, 64]	0.7766	0.7535	0.7789	0.7543	0.7753	0.7525

Table 8: Parameters obtained with ReLU activation and Adaptive learning while varying depths of the network



(f) Sklearn Neural Network (`MLPClassifier`)

- Stopping criteria: Default (loss not changing for 10 epochs by at least 1e-4, or max epochs 200)
- Training time of all 4 models combined: 2h17m.
- As compared to part (e), the performance was found to be much worse, even though it took significantly more training time than (e). This difference could have been caused by the different loss function used by `MLPClassifier`

Architecture	Train precision	Test precision	Train recall	Test recall	Train F1	Test F1
[512]	0.5758	0.5668	0.5566	0.5525	0.5562	0.5490
[512, 256]	0.5977	0.5634	0.5846	0.5520	0.5878	0.5543
[512, 256, 128]	0.6093	0.5868	0.6001	0.5817	0.6034	0.5830
[512, 256, 128, 64]	0.6219	0.6191	0.6129	0.6150	0.6158	0.6162

Table 9: Parameters obtained using `MLPClassifier`