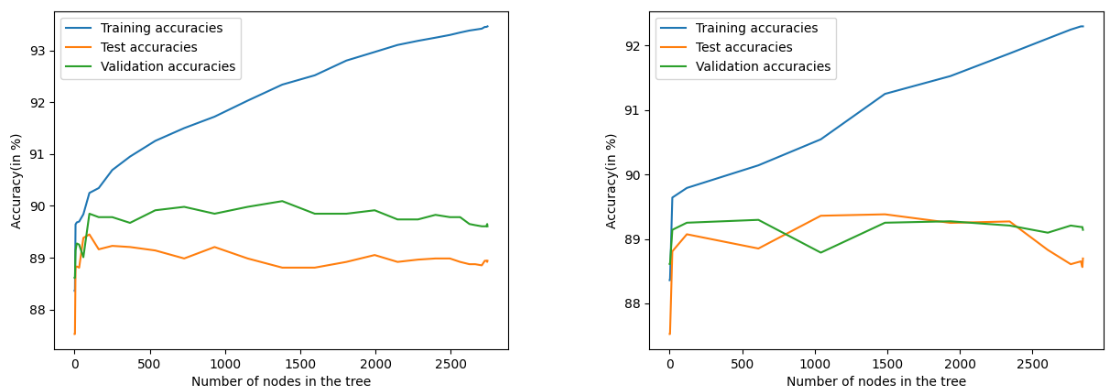
COL774 Assignment 3

Mridul Gupta(AIZ218322)

October 26, 2021

1 Q1

1.1 Q1(a)



(a) with one-hot (b) without one-hot

Figure 1: Decision Tree Accuracies vs Number of nodes in Tree

It seems that the model trained on one-hot encoded data performs slightly better than the one trained on the not one-hot encoded data. But what's more important to note here is that the validation and test accuracy saturate after about a 100 nodes are created and pretty much oscillate slightly. We can remove these extra nodes and improve out inference time. I don't know what this will be called, as the performance on train data improves, but performance on test/validation doesn't deteriorate. This does not seem to be overfitting, so no comments can be made about overfitting. Accuracy was 88.70%

1.2 Q1(b)

Decision tree pruning is performed in bottom-up fashion, the increase in accuracy is only $0.2 - 0.3\%$. Accuracy was 88.80%.

1.3 Q1(c)

The accuracies obtained are as follows:

$$\begin{aligned} \text{Train: } & 98.11\% \\ \text{Validation: } & 90.69\% \\ \text{Test: } & 90\% \end{aligned}$$

The best parameters obtained were `n_estimators: 100`, `max_features:0.9`, `min_samples_split: 10`. The random forest performs better than pruned decision tree, especially on the training set. The performance on test set is not very different.

1.4 Q1(d)

It is seen that random forests are most sensitive to n_estimators as one would expect, since more models in the ensemble, the better the estimate as the law of large number suggests (because this is just a case of averaging out the predictions). Even then, the sensitivity is not a lot, the algorithm is pretty much stable.

2 Q2

2.1 Q2(a)

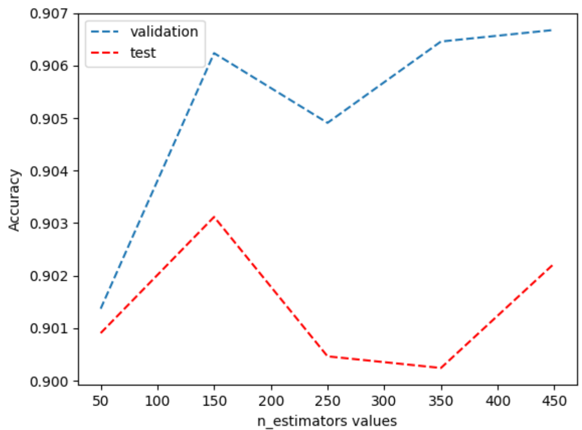
I performed one-hot encoding using pandas' `get_dummies` function.

2.2 Q2(b)

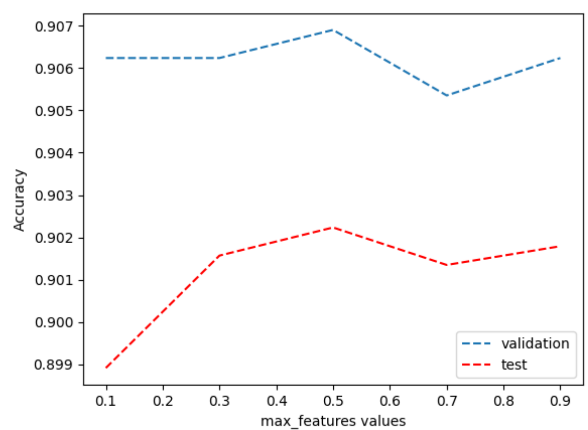
For the implementation of the backpropagation algorithm for neural networks, I follow Goodfellow et al.'s Deep Learning[1] (Section 6.5.6 General back-propagation p. 213). I use the tensorflow api style with reference from codingame.com[2].

2.3 Q2(c)

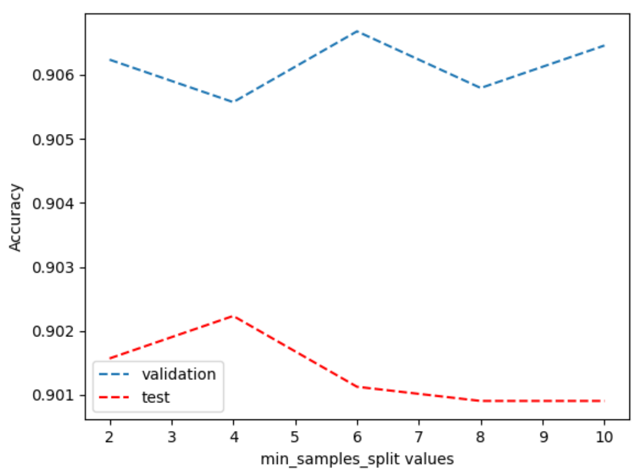
It is seen that the accuracy and training time both increase as we increase the number of nodes in the hidden layer of the neural network. Note, I also train a network with 100 nodes along with the required $\{5, 10, 15, 20, 25\}$ for this exercise. For stopping criteria I followed the following intuition: since the target/label is a one-hot vector for a 10 class classification problem, it will be a sparse vector with exactly one 1 and the rest entries being 0. When



(a) n-estimators

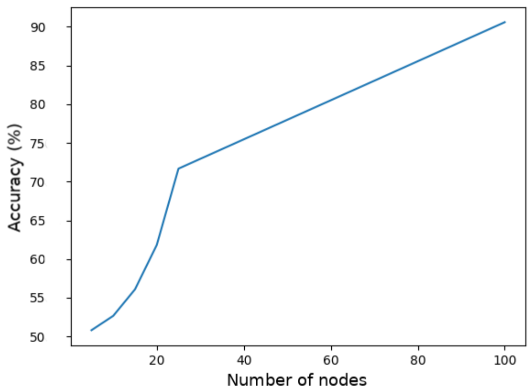


(b) max-feature

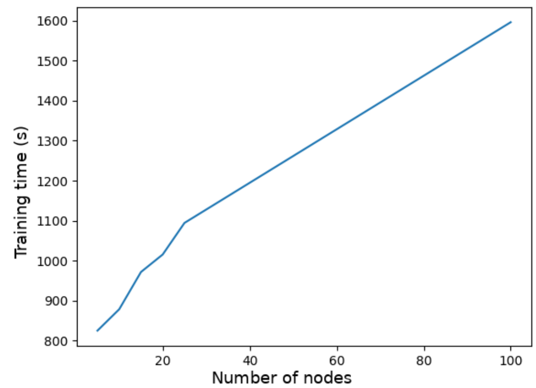


(c) min-samples-split

Figure 2: Sensitivity for parameters



(a) Accuracy



(b) Training Time

Figure 3: Metric vs number of hidden layer nodes Q2(c)

the network learns that it must output a one hot vector too, then each incorrect classification will incur an MSE of 2. Thus, per 100 examples, if the classifier makes one mistake then the loss function value will be under 0.1. This is what I use for stopping criteria, that the loss is less than some threshold. Of course, the sigmoid output is not perfect 0 or 1, thus I try for various values, and choose the stopping criteria as loss getting lower than 0.01.

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	436701	64508	0	0	0	0	0	0	0	0
Actual 1	351161	71337	0	0	0	0	0	0	0	0
Actual 2	37720	9902	0	0	0	0	0	0	0	0
Actual 3	15959	5162	0	0	0	0	0	0	0	0
Actual 4	3206	679	0	0	0	0	0	0	0	0
Actual 5	1864	132	0	0	0	0	0	0	0	0
Actual 6	1020	404	0	0	0	0	0	0	0	0
Actual 7	155	75	0	0	0	0	0	0	0	0
Actual 8	9	3	0	0	0	0	0	0	0	0
Actual 9	3	0	0	0	0	0	0	0	0	0

(a) 5

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	410040	91169	0	0	0	0	0	0	0	0
Actual 1	305964	116534	0	0	0	0	0	0	0	0
Actual 2	30992	16630	0	0	0	0	0	0	0	0
Actual 3	12682	8439	0	0	0	0	0	0	0	0
Actual 4	3101	784	0	0	0	0	0	0	0	0
Actual 5	1689	307	0	0	0	0	0	0	0	0
Actual 6	772	652	0	0	0	0	0	0	0	0
Actual 7	110	120	0	0	0	0	0	0	0	0
Actual 8	11	1	0	0	0	0	0	0	0	0
Actual 9	3	0	0	0	0	0	0	0	0	0

(b) 10

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	366279	134930	0	0	0	0	0	0	0	0
Actual 1	228000	194498	0	0	0	0	0	0	0	0
Actual 2	18978	28644	0	0	0	0	0	0	0	0
Actual 3	5906	15215	0	0	0	0	0	0	0	0
Actual 4	1386	2499	0	0	0	0	0	0	0	0
Actual 5	1591	405	0	0	0	0	0	0	0	0
Actual 6	310	1114	0	0	0	0	0	0	0	0
Actual 7	18	212	0	0	0	0	0	0	0	0
Actual 8	6	6	0	0	0	0	0	0	0	0
Actual 9	1	2	0	0	0	0	0	0	0	0

(c) 15

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	401529	99680	0	0	0	0	0	0	0	0
Actual 1	206004	216494	0	0	0	0	0	0	0	0
Actual 2	12245	35377	0	0	0	0	0	0	0	0
Actual 3	5693	15428	0	0	0	0	0	0	0	0
Actual 4	2450	1435	0	0	0	0	0	0	0	0
Actual 5	1734	262	0	0	0	0	0	0	0	0
Actual 6	145	1279	0	0	0	0	0	0	0	0
Actual 7	17	213	0	0	0	0	0	0	0	0
Actual 8	10	2	0	0	0	0	0	0	0	0
Actual 9	3	0	0	0	0	0	0	0	0	0

(d) 20

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	423642	77567	0	0	0	0	0	0	0	0
Actual 1	129407	293091	0	0	0	0	0	0	0	0
Actual 2	4287	43335	0	0	0	0	0	0	0	0
Actual 3	1627	19494	0	0	0	0	0	0	0	0
Actual 4	2397	1488	0	0	0	0	0	0	0	0
Actual 5	1827	169	0	0	0	0	0	0	0	0
Actual 6	35	1389	0	0	0	0	0	0	0	0
Actual 7	0	230	0	0	0	0	0	0	0	0
Actual 8	8	4	0	0	0	0	0	0	0	0
Actual 9	1	2	0	0	0	0	0	0	0	0

(e) 25

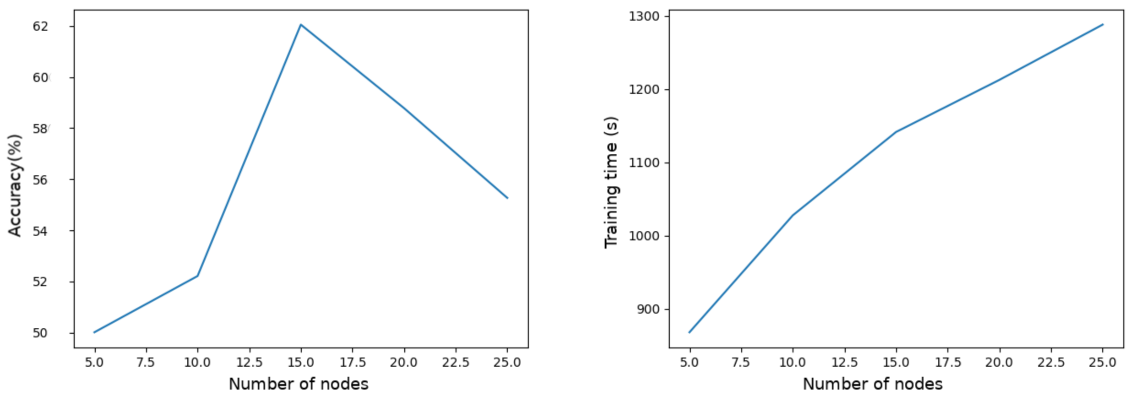
	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	496574	4628	7	0	0	0	0	0	0	0
Actual 1	13153	409337	8	0	0	0	0	0	0	0
Actual 2	149	47473	0	0	0	0	0	0	0	0
Actual 3	0	21121	0	0	0	0	0	0	0	0
Actual 4	3775	110	0	0	0	0	0	0	0	0
Actual 5	1987	9	0	0	0	0	0	0	0	0
Actual 6	1	1423	0	0	0	0	0	0	0	0
Actual 7	0	230	0	0	0	0	0	0	0	0
Actual 8	12	0	0	0	0	0	0	0	0	0
Actual 9	3	0	0	0	0	0	0	0	0	0

(f) 100

Figure 4: Confusion matrix for Q2(c)

2.4 Q2(d)

The adaptive learning makes training stable, not necessarily faster. It is seen that just like the previous case, the training time and accuracy both go up as the number of nodes is increased. It is also seen that as the number of nodes go up, the accuracy goes down because deeper networks do not train well with decreasing learning rate.



(a) Accuracy (b) Training Time

Figure 5: Metrics vs hidden layer nodes with adaptive learning Q2(d)

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	487715	13494	0	0	0	0	0	0	0	0
Actual 1	410046	12452	0	0	0	0	0	0	0	0
Actual 2	46050	1572	0	0	0	0	0	0	0	0
Actual 3	20326	795	0	0	0	0	0	0	0	0
Actual 4	3758	127	0	0	0	0	0	0	0	0
Actual 5	1976	20	0	0	0	0	0	0	0	0
Actual 6	1379	45	0	0	0	0	0	0	0	0
Actual 7	220	10	0	0	0	0	0	0	0	0
Actual 8	12	0	0	0	0	0	0	0	0	0
Actual 9	3	0	0	0	0	0	0	0	0	0

(a) 5

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	427248	73961	0	0	0	0	0	0	0	0
Actual 1	327582	94916	0	0	0	0	0	0	0	0
Actual 2	34155	13467	0	0	0	0	0	0	0	0
Actual 3	15736	5385	0	0	0	0	0	0	0	0
Actual 4	3323	562	0	0	0	0	0	0	0	0
Actual 5	1831	165	0	0	0	0	0	0	0	0
Actual 6	1011	413	0	0	0	0	0	0	0	0
Actual 7	166	64	0	0	0	0	0	0	0	0
Actual 8	12	0	0	0	0	0	0	0	0	0
Actual 9	3	0	0	0	0	0	0	0	0	0

(b) 10

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	378194	123015	0	0	0	0	0	0	0	0
Actual 1	180246	242252	0	0	0	0	0	0	0	0
Actual 2	11430	36192	0	0	0	0	0	0	0	0
Actual 3	2571	18550	0	0	0	0	0	0	0	0
Actual 4	2209	1676	0	0	0	0	0	0	0	0
Actual 5	1530	466	0	0	0	0	0	0	0	0
Actual 6	114	1310	0	0	0	0	0	0	0	0
Actual 7	3	227	0	0	0	0	0	0	0	0
Actual 8	7	5	0	0	0	0	0	0	0	0
Actual 9	3	0	0	0	0	0	0	0	0	0

(c) 15

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	366250	134959	0	0	0	0	0	0	0	0
Actual 1	200950	221548	0	0	0	0	0	0	0	0
Actual 2	13673	33949	0	0	0	0	0	0	0	0
Actual 3	5029	16092	0	0	0	0	0	0	0	0
Actual 4	2626	1259	0	0	0	0	0	0	0	0
Actual 5	1663	333	0	0	0	0	0	0	0	0
Actual 6	156	1268	0	0	0	0	0	0	0	0
Actual 7	16	214	0	0	0	0	0	0	0	0
Actual 8	3	9	0	0	0	0	0	0	0	0
Actual 9	3	0	0	0	0	0	0	0	0	0

(d) 20

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	349306	151903	0	0	0	0	0	0	0	0
Actual 1	219095	203403	0	0	0	0	0	0	0	0
Actual 2	18071	29551	0	0	0	0	0	0	0	0
Actual 3	5062	16059	0	0	0	0	0	0	0	0
Actual 4	2873	1012	0	0	0	0	0	0	0	0
Actual 5	1636	360	0	0	0	0	0	0	0	0
Actual 6	259	1165	0	0	0	0	0	0	0	0
Actual 7	12	218	0	0	0	0	0	0	0	0
Actual 8	10	2	0	0	0	0	0	0	0	0
Actual 9	3	0	0	0	0	0	0	0	0	0

(e) 25

Figure 6: Confusion matrices with adaptive learning for different number of hidden nodes for Q2(d)

2.5 Q2(e)

In this part, the multilayered architecture with **ReLU** activation trains to 92.297% accuracy within 100 epochs, under 2 minutes. The **sigmoid** hidden unit architecture however doesn't converge even after 1400 epochs, and only achieves 77.26% accuracy on test set by then. ReLU performs better than sigmoid in terms of training speed.

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	501056	153	0	0	0	0	0	0	0	0
Actual 1	131	422367	0	0	0	0	0	0	0	0
Actual 2	0	47622	0	0	0	0	0	0	0	0
Actual 3	0	21121	0	0	0	0	0	0	0	0
Actual 4	3451	434	0	0	0	0	0	0	0	0
Actual 5	1996	0	0	0	0	0	0	0	0	0
Actual 6	0	1424	0	0	0	0	0	0	0	0
Actual 7	0	230	0	0	0	0	0	0	0	0
Actual 8	11	1	0	0	0	0	0	0	0	0
Actual 9	2	1	0	0	0	0	0	0	0	0

Figure 7: Confusion matrix for Q2(e) with ReLU

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	439320	61889	0	0	0	0	0	0	0	0
Actual 1	67935	354563	0	0	0	0	0	0	0	0
Actual 2	679	46943	0	0	0	0	0	0	0	0
Actual 3	17	21104	0	0	0	0	0	0	0	0
Actual 4	3052	833	0	0	0	0	0	0	0	0
Actual 5	1764	232	0	0	0	0	0	0	0	0
Actual 6	2	1422	0	0	0	0	0	0	0	0
Actual 7	0	230	0	0	0	0	0	0	0	0
Actual 8	10	2	0	0	0	0	0	0	0	0
Actual 9	3	0	0	0	0	0	0	0	0	0

Figure 8: Confusion matrix for Q2(e) with sigmoid

2.6 Q2(f)

The MLPClassifier from sklearn does not work as one would expect. I initialized the classifier to replicate the training settings and model architecture exactly as done in Q2(e), yet it doesn't converge even after 800 epochs, and the maximum accuracy achieved over multiple trials was under 70%.

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	501160	48	0	0	0	0	0	0	0	0
Actual 1	422442	56	0	0	0	0	0	0	0	0
Actual 2	47610	12	0	0	0	0	0	0	0	0
Actual 3	21108	13	0	0	0	0	0	0	0	0
Actual 4	3885	0	0	0	0	0	0	0	0	0
Actual 5	1996	0	0	0	0	0	0	0	0	0
Actual 6	1422	2	0	0	0	0	0	0	0	0
Actual 7	230	0	0	0	0	0	0	0	0	0
Actual 8	12	0	0	0	0	0	0	0	0	0
Actual 9	3	0	0	0	0	0	0	0	0	0

Figure 9: Confusion matrix for Q2(f) with MLPClassifier

2.7 Q2(g)

I created more samples for classes 6, 7, 8, and 9 by permuting and undersampled classes 1 and 2 to balance out the training data. The training completes in under 3 minutes with around 94% accuracy. But what is more interesting is that the classifier now predicts more than just two classes as apparent from the confusion matrix. Note, I used the two hidden layers with 100 nodes each with ReLU activation as in Q2(e) for this.

	P 0	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
Actual 0	497928	3043	0	0	0	0	0	0	161	77
Actual 1	4043	413918	4423	0	0	0	0	0	67	47
Actual 2	0	12774	34774	0	0	0	74	0	0	0
Actual 3	0	154	18047	0	0	0	2763	157	0	0
Actual 4	3328	420	0	0	0	0	0	0	24	113
Actual 5	1655	99	0	0	0	0	0	0	178	64
Actual 6	0	0	723	0	0	0	634	67	0	0
Actual 7	0	0	19	0	0	0	163	48	0	0
Actual 8	4	3	0	0	0	0	0	0	3	2
Actual 9	0	0	0	0	0	0	0	0	0	3

Figure 10: Confusion matrix for Q2(g)

References

[1] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016.

[2] codingame, “Deep Learning From Scratch”, https://www.codingame.com/playgrounds/9487/deep-learning-from-scratch–theory-and-implementation/computational-graphs, accessed: 2021-10-14