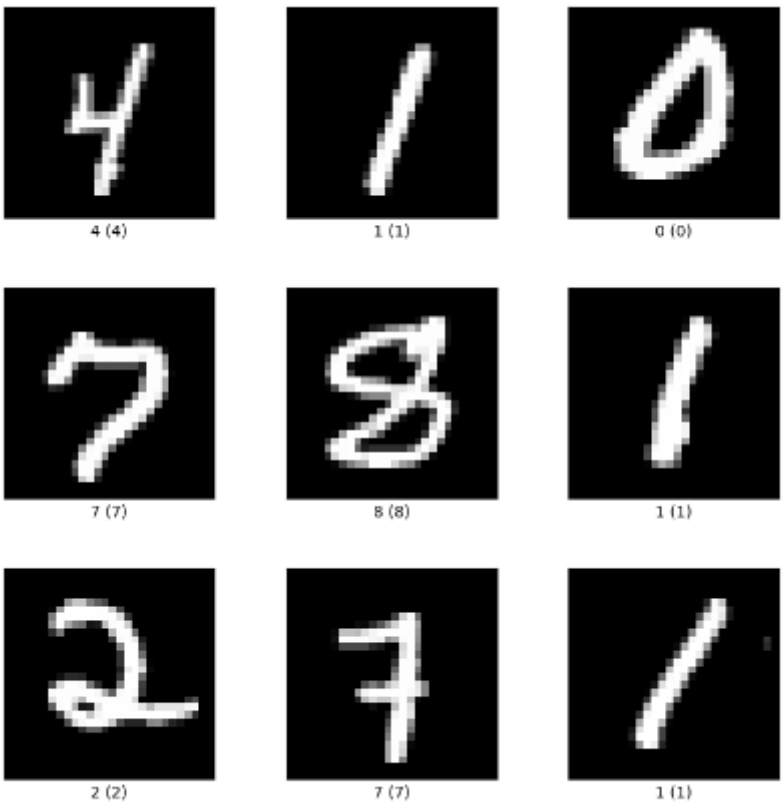
Report of first hw

1.Classfication Problem

Mnist dataset

Introduction of dataset

Mnist dataset is a dataset for classification. It contains 6,000 pictures of hand written number from 1 to 10 for training and 1,000 for test.



The reason I choose this dataset is that it is a classic dataset used before and after "deep-learning" era. For CIFAR10, CIFAR100, and other datasets, they are hard to train and evaluate.

Another reason is that it is a clear task and could be understood easily.

2.Training and Testing result

a.Decision Tree

Settings

Model: Decision tree

Dataset: MNIST

iteration:1

Pruning method: Find a subset of `ccp_alpha` and test the new decision tree on train set.

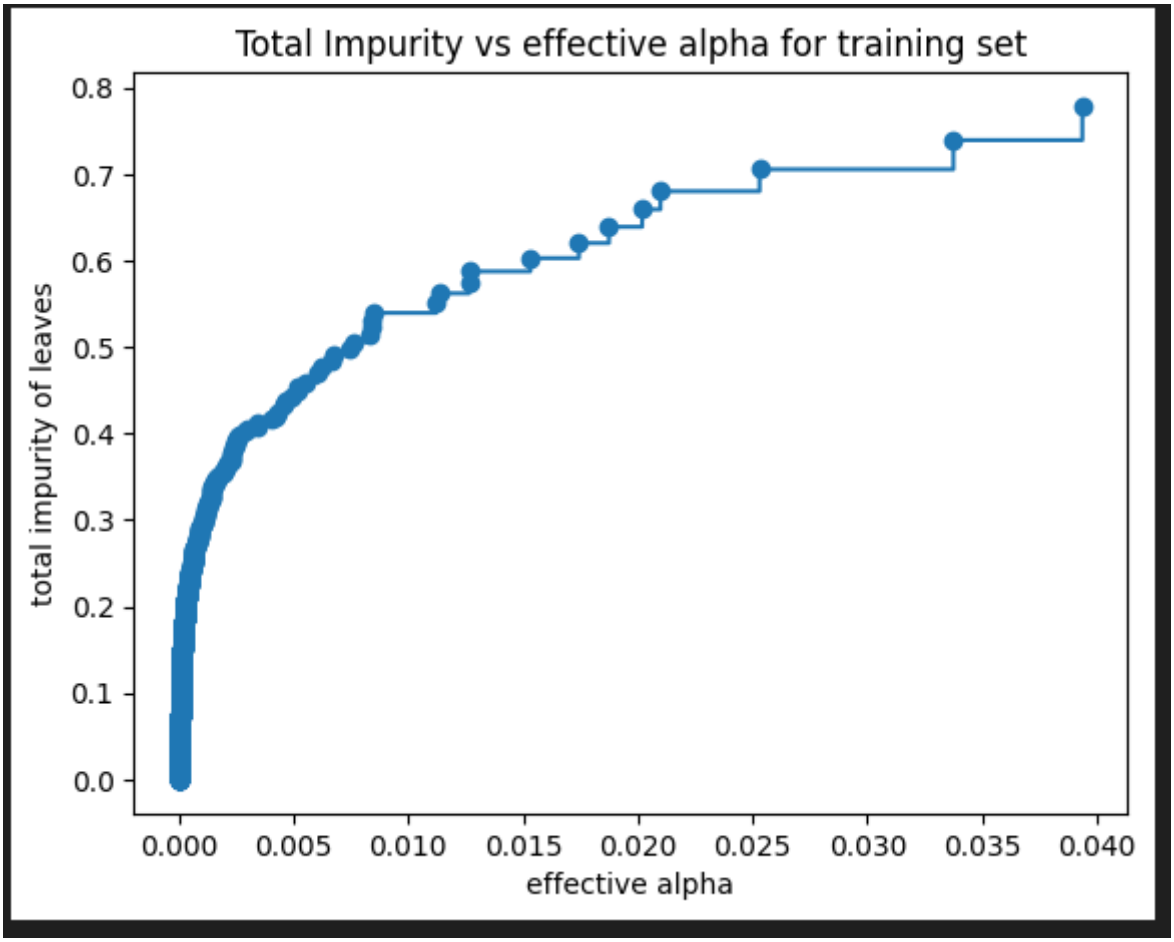
Results:

Result of non-pruning decision tree

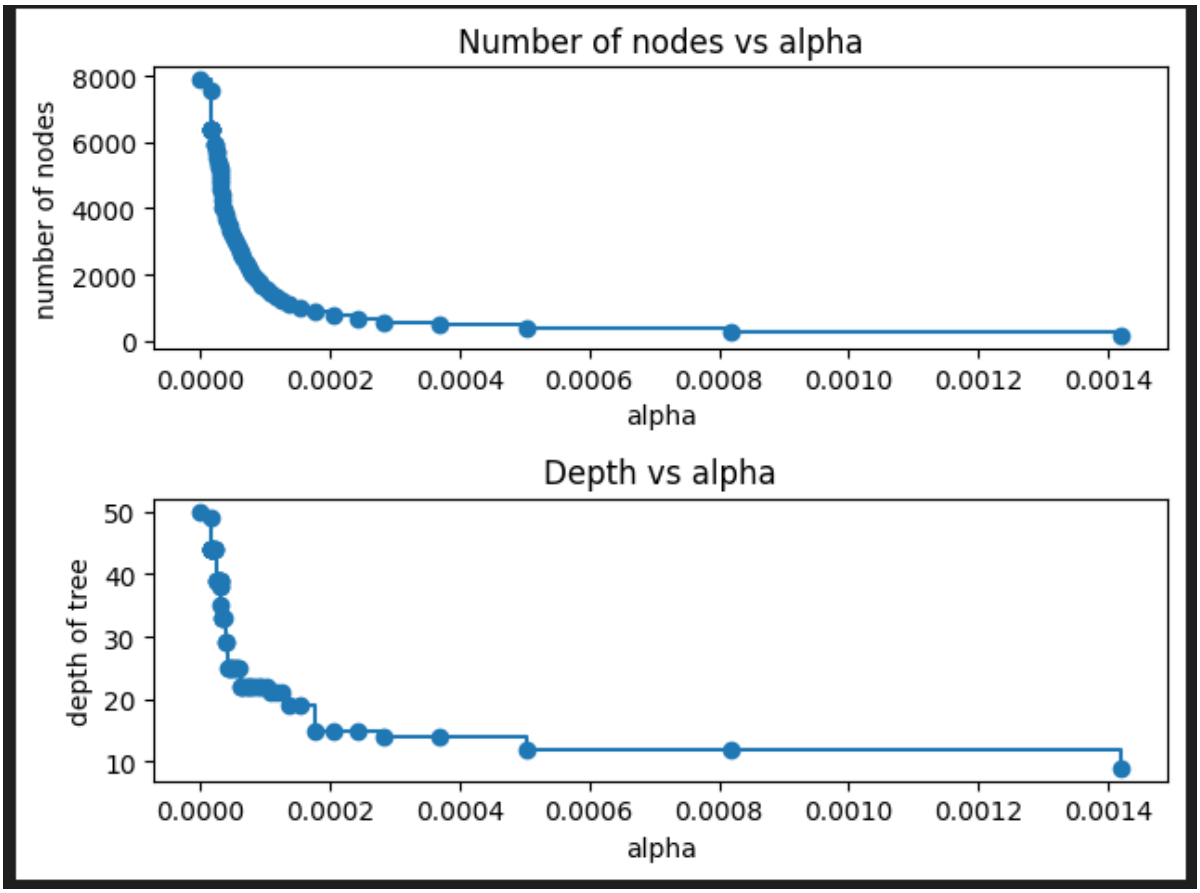
Train acc(%)	Test acc(%)
100.00	87.67~0.3

It is **overfit**.

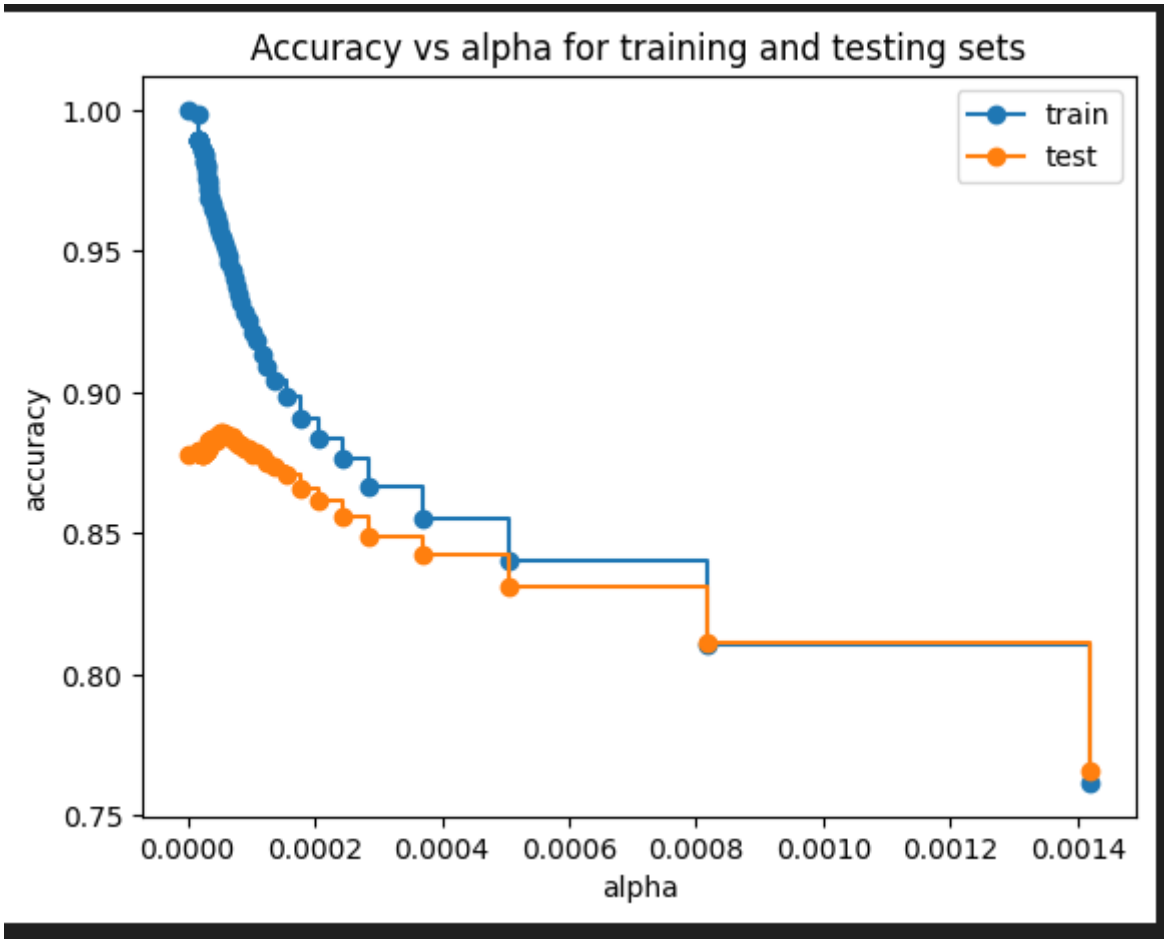
Result of pruning decision tree



First I get all the effective alpha and see the relationship with impurity. To many alphas, need to delete some.



After deleting to about 64 alphas, I draw the new graph. Different alpha vs nodes and depth. **alpha increases, the other two decreases.**



The I generate the train and test acc of diffferent alpha.The right alpha is about 0~1e-4. It shows the overfit of decision tree well. **With pruning, the training acc decreases but test acc increases then decreases.** By selecting the right alpha, we could get the best result on test case.

Train acc(%)	Test acc(%) best
98.54	88.78

b.Neural Network

The most boring part of the report. I used server for training. So please make sure you have at least one gpu and device 0 is free for fine-tuning.

Also, I use wsl as the os. Something wrong when training on ipynb. I provided python scrpit called `MobileNetV2.py` . You could use that.

Settings

Model: MobileNetV2(Pretrained from Imagenet)

Dataset: MNIST

Optimizer: Adam

Learning rate: 1e-3

Epoch: 5

batch size: 128

Results

Train acc(%)	Test acc(%)
98.85	98.67

I just train for 5 epochs. I believe training more epochs could be better

c. Boosting

The boost tree achieves almost the accuracy of previous decision tree with depth =3.

I use AdaBoost to speed up my decision tree.

Settings

Model: Decision tree

Dataset: MNIST

iteration:1

Boost: Adaboost

Depth:3

Result

Let us compare adaboost decision tree and original one. Let us set decision tree depth to 3.

Train acc Boost(%)	Test acc Boost(%)	Train acc(%)	Test acc(%)
82.58	**82.00**	49.15	49.53

Amazing! With same depth, adaboost increase the accuracy of the tree. The best accuracy of original tree is about **87%** accuracy on test dataset. But the depth is over **300**!

d. Support Vector Machines

It is really time consuming when running Support Vector Macchine for Mnist dataset.

Settings

Model: SVM

Dataset: MNIST

Result

Train acc(%)	Test acc(%)	Method
100%	99.32%	"rbf"
99.81%	99.76%	"linear"

Both achieves almost the best result of all the method.

f. K-means

No settings here, only iteration time and k changes

Results

Train acc(%)	Test acc(%)	Iteration	k
57.71	58.46	100	10
73.55	74.19	100	20
79.58	80.90	100	40

With the k increases, the accuracy increases.

Interesting to find that Test accuracy is better than train accuracy when using k-means.

This is called underfit. I remembered only happened when using large dataset and big models. It seems that k-means are the easiest and simple that can not capture the information of Mnist dataset enough. I wonder if it is correct.

3. Al together analyse

I have analyse them separately. But now let us bring them together. I have answered some questions like

"Why did you get the results you did? Compare and contrast the different algorithms. What sort of changes might you make to each of those algorithms to improve performance?" and some other questions in the separate column.

Method	**Train acc%**	**Test acc%**	**Time**
Decision tree	100.00	87.67~0.3	Fast:5s
MobileNetV2	98.85	98.67	Normal(GPU): 50s
Boost decision tree(depth = 3)	82.58	82.00	Fast: 2.7s
SVM	100	**99.32**	Slow: 2h
K-means	79.58	80.90	Slow: 1.5min

Actually, Mnist is a quite simple dataset. For difficult and large dataset, I think Neural Network would achieve the best accuracy on test dataset.

What surprises me most is that SVM achieves so good accuracy. However, the training and evaluation time is so time consuming.

Also, I found that **decision tree** is always overfit(unless we prune). It seems that it is the worst solution to Mnist dataset among 5. And for the best, I vote for **Neural Network**. It is easy, efficient on GPU and get good result.