CS-7641 Assignment 1: Supervised Learning

Shangwen Huang
shuang443@gatech.edu

***Abstract*—This assignment is to implement five types of machine learning algorithms: decision tree, neural network, boosting, support vector machine and k-nearest neighbors. The algorithms will be tested on two self-chosen classification problems. This report is to document algorithm performance analysis.**

1 CLASSIFICATION PROBLEM

1.1 Problem 1 - MNIST

The first classification problem is taking the images of handwritten digits to predict the true digit. The dataset is downloaded through Pytorch. The MNIST database contains 60,000 training images and 10,000 testing images. The original creators of the database keep a list of some of the methods tested on it. In their original paper, they use a support-vector machine to get an error rate of 0.8%.

The reason I picked this dataset is because it is a unstructured dataset, specifically images and the volume are big. I am interested in comparing the performance by different algorithms on the image data. Some algorithm like CNN by nature is created for solving images classification problem and some other algorithm like Decision Tree is not.

***Figure 1*—Sample images from MNIST test dataset.**

1.2 Problem 2 – Loan Prediction

The second classification problem is loan status prediction – whether the loan application is approved or not. The dataset is downloaded from Kaggle[1]. The dataset contains 614 entries, 13 columns. The reason I chose this prediction problem is this dataset is structured data. Each of the variables has semantic meanings. The Decision Tree by nature can mimic human's logic when approving the loan application. I am interest to know if the simpler algorithm can have a good performance and what's the incremental prediction accuracy other sophisticated algorithms can achieve.

```
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

Figure 2 — Data info of Loan Prediction.

2 LEARNING CURVES

Figure 3 and figure 4 contains the learning curves by different algorithm for MNIST digit classification and loan prediction. For algorithm Decision Tree, Boosting, Support Vector Machine (SVM) and K-Nearest Neighbor (KNN), the learning curve is generated by scikit-learn Learning Curve function with randomly shuffle training and testing data in different data size. For Neural Network algorithm, the learning curve is generated by plotting the accuracy and loss in iteratively training epochs.

[1] Data download url: https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problem-dataset?resource=download

2.1 Learning Curves for MNIST

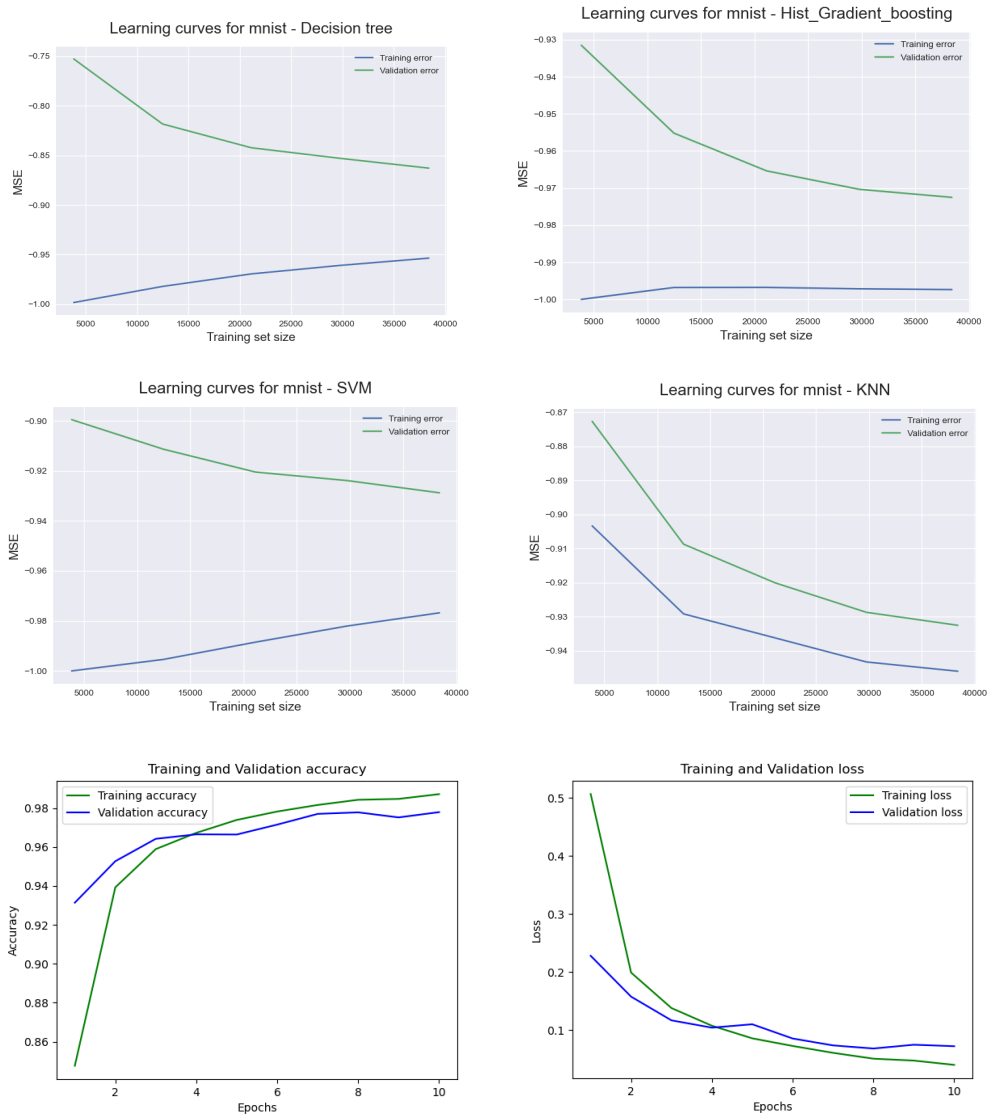

Figure 3 — Learning curves for MNIST digit classification

2.2 Learning Curves for Loan Prediction

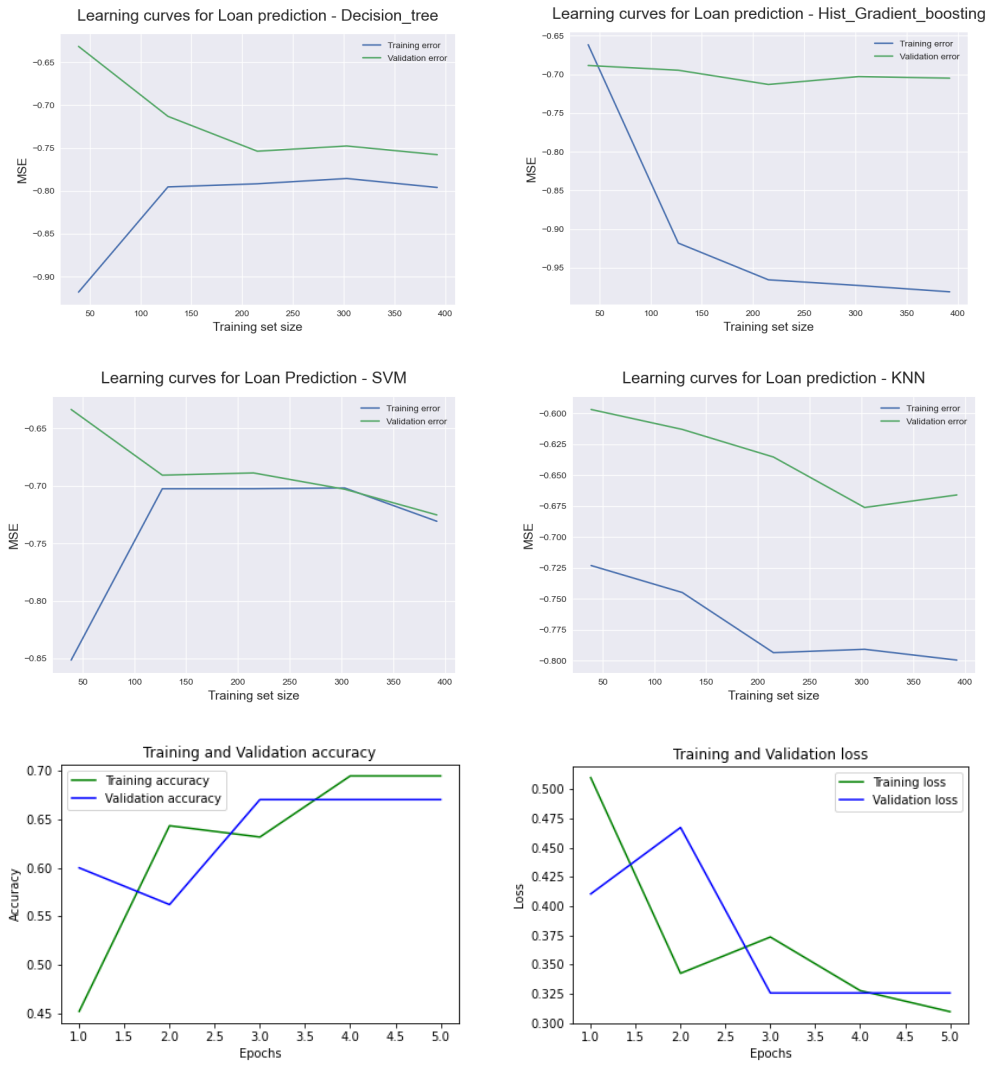

Figure 4 — Learning curves for loan prediction

3 PERFORMANCE

Algorithm	Test Accuracy of MNIST	Running Time for MNIST	Test Accuracy of Loan Prediction	Running Time for Loan Prediction
Decision Tree	0.8863	11 seconds (441 seconds for 40 iterations)	0.7886	9 ms (0.3775 seconds for 40 iterations)
Histogram Gradient Boosting	0.9763	113 seconds	0.7723	943 ms
SVM	0.9399	148 seconds	0.7298	66 seconds
KNN	0.9417	8 seconds (320 seconds for 40 iterations)	0.7479	376 ms
Neural Network	0.9746	447 seconds for 10 epochs	0.3171	166 ms

Table 1 — Accuracy and running time comparison

3.1 Performance Analysis for Problem 1 – MNIST

3.1.1 *Best Algorithm*

For problem 1: mnist digit image classification, Histogram Gradient Boosting is the best algorithm because it has the highest test accuracy and it is faster than Neural Network and SVM. For problem 2: loan prediction, Decision Tree is the best algorithm with the highest accuracy and lowest training time.

3.1.2 *Learning Curves Comparison*

By comparing the learning curves, Neural Network is more efficient in terms of lowering training loss and testing loss at the same time. At epoch 10, the difference of training loss and testing loss is around 0.05 compared at epoch 1, the difference is 0.3. The reason is for Neural Network, it starts with random or zero weights and the learned weights from gradient descent at each epoch will be the initial value for next iteration, so it can be iteratively closing to the true value.

Interestingly, for Decision Tree, Histogram Gradient Boosting and, SVM, as training size increases, the training loss increases, since the algorithm can perfectly fit one data instance but it cannot perfectly fit all the data instances, so the training errors increases. While for KNN, the training errors decrease as training size increases. That is because KNN calculate the distances between the data point and all other points, and find the nearest neighbors by ranking points by increasing distance. Then vote on the predicted class labels based on the k nearest neighbors. Because of the voting process, actually more training samples there are, the more accurate prediction will be.

3.2 How to improve the performance?

3.2.1 *Decision Tree*

For Decision tree and KNN, we can search for the maximum depth of tree and K to find the better performance. Limiting the maximum depth of the tree is the pruning process for Decision Tree. For Loan Prediction problem, if the tree is too big, it will overfit to the training data and doesn't generalize well to the unseen samples, so it has lower testing accuracy. For image classification problem, the bigger tree doesn't hurt the testing accuracy but it will plateau once maximum depth is at 11.

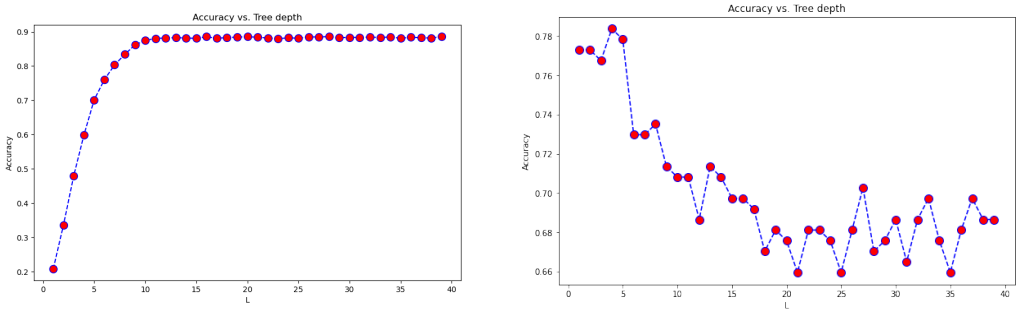

Figure 5 — Decision Tree Testing Accuracy by varying max tree depth for MNIST (left) and Loan Prediction (right)

3.2.2 *KNN*

For KNN, by searching the optimal K, which is the cross-validation process, we can have better testing accuracy. If the number of neighbors to look at is too few, it is not accurate to have the right ranking of classes. But if the number of neighbors to look at is too large, the radius can reach

to multiple classes, which defeats the principle of KNN – neighbors that are nearer have similar densities or classes, the testing accuracy decreases.

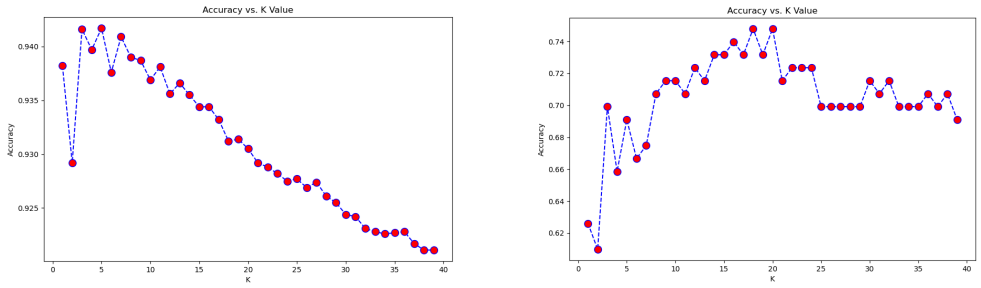

Figure 6 — KNN Accuracy by varying K nearest neighbors for MNIST (left) and Loan Prediction (right)

3.2.3 *Boosting*

Boosting is a method for creating an ensemble. It starts by fitting an initial model to the data. Then a second model is built to explain the prediction error from the initial model. So, the successive model attempts to correct for the shortcomings of the previous model. The process of boosting can repeat many times and minimize the overall prediction error.

Although, the gradient boosting performs well, the model can be slow to train. Originally, I used the normal gradient boosting, the training time is 2,163 seconds for MNIST problem. With that training speed, I even couldn't finish generating the learning curve.

Switching to Histogram Gradient Boosting method, the training time reduced to 100 seconds, which is faster than SVM. For Gradient Boosting, the most expensive part is the search for the best split in the tree which is a brute-force approach: all possible splits are evaluated and the best one is picked. But Histogram Gradient Boosting approximates the threshold values using some form of quantization. The algorithm transforms the feature distribution into a uniform distribution for split finding. The complexity is reduced from n log(n) to n of bins.

3.2.4 *SVM*

SVM ranked number four for both classification problems. The reason of SVM is not very good compare to other algorithms is probably SVM is not suitable for large data sets. With 10,000 images and 728 dimensions, it may be hard to find a hyperplane that can separate the classes clearly.

One thing could have been done to improve the performance is to normalize the input data. When calculate the distance of hyperplane to the classes, using normalized input data may be more accurate. Another thing can improve SVM performance is to introduce class weights, so the magnitude of the positive support will be proportionately higher than that of the negative support vector, since the classes are imbalanced.

3.2.5 *Neural Network*

Neural Network is the second best algorithm, and it is very close to the best for image classification problem in terms of both accuracy and training time. But it has the worst accuracy for the loan predication problem.

The Neural Network algorithm I used is the basic MLP net which has three linear layers and two Relu activation function in between the linear layers. The batch size is set as 64, learning rate is 0.01. To improve the performance for MNIST image classification, I could have tried Convolutional Neural Network like LeNet, which may work better for image classification problems. Other techniques to improve performance could be changing learning rate by iterations, learning rate could be bigger at first few iterations and dial it low at later iterations. Also, could search for the best batch size.

Since the loan predication problem has much fewer data points, Neural Network may not work well with small data sample. I reduced the original three linear layers to two linear layers and reduced the hidden size. The accuracy increases from 0.3 to 0.67. For problems with smaller sample and structured data, simpler algorithm can work better.