CS5644 Assignment 4

Q1. Data Analysis on Congressional Voting Records dataset

a. Introduction

This report documented the machine learning analysis of the MNIST dataset. The goal is to create an MLP classifier that can correctly determine the digits in the dataset—the implementation of MLP architectures including 4 different kinds of layers and 2 different activation functions. The result of precision, recall, and f score for 5-fold cross-validation was recorded for comparison. In addition, the time spent fitting each model as the size of layers and the number of layers changed were also recorded, and graphs were used for observing the difference.

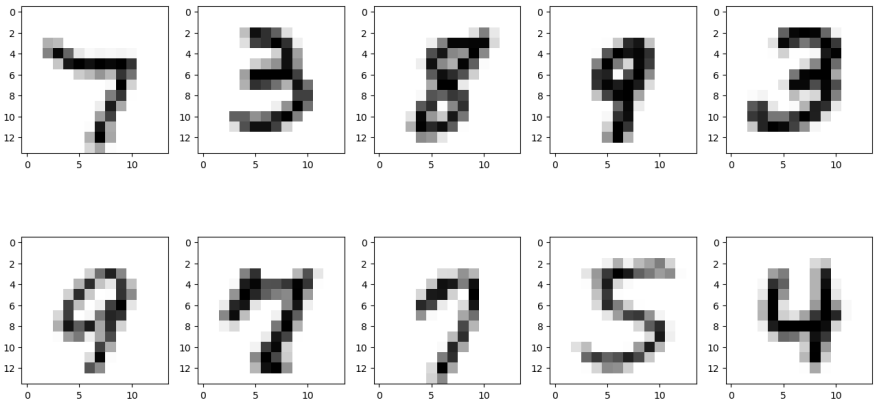
b. Methods

1) Data preprocessing

The data was organized fairly well in the CSV file, which is convenient for transferring it into the data frame and applying it to the machine learning models..

	label	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	...	13x4	13x5	13x6	13x7	13x8	13x9	13x10	13x11	13x12	13x13
0	7	0	0	0	0	0	0	0	0	0	...	0	0	41	88	3	0	0	0	0	0
1	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	8	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

The label indicates the actual digit the image represents, the other columns (e.g., 0x0, 0x1) represent pixel values for each position. A portion of the data can be viewed as:



2) Machine learning models

For a fair comparison, a 5-fold cross validation with random state 2 object was initialized (single initialization) beforehand and been used consistently across all architectures in the evaluation process

There are 4 different layers setting and 2 different activation functions were applied, totally 8 different scenarios were tested. The data was scaled by using `MinMaxScaler` before applying to the model for optimum results. The MLP classifier is using the the arguments of max_iter=1000 and solver=lbfgs. The result of precision, recall, f score, and time spent were recorded.

c. Results and Discussions

1) ReLU (Rectified Linear Unit) activation function

I) 2 hidden layers with 20 nodes in each layer

Precision Score	Recall Score	Test F1 Score	Time
0.9107 ±0.0088	0.9100 ±0.0087	0.9100 ±0.0087	10.43

II) 2 hidden layers with 100 nodes in each layer

Precision Score	Recall Score	Test F1 Score	Time
0.9403 ±0.0079	0.9395 ±0.0082	0.9396 ±0.0082	20.44

III) 5 hidden layers with 20 nodes in each layer

Precision Score	Recall Score	Test F1 Score	Time
0.9072 ±0.0077	0.9063 ±0.0080	0.9064 ±0.0080	41.94

IV) 5 hidden layers with 100 nodes in each layer

Precision Score	Recall Score	Test F1 Score	Time
0.9377 ±0.0068	0.9370 ±0.0070	0.9371 ±0.0070	52.08

Discussion:

Generally, more nodes can improve the capacity to learn, but have increased computation time. Also, more layers can learn high-level features better, but requires more computation time. The result is reflecting this well, with increasing the number of nodes per layer (from 20 to 100) has a slight increase in performance but doubles the computation time. In addition, increasing the number of hidden layers from 2 to 5, while keeping the number of nodes constant, leads to a marginal improvement in performance metrics, but increase the computation time.

From the finding, we can observe that the most "budget" solution is the one with 2 hidden layers with 100 nodes in each layer, since it has almost identical performance with the one with 5 hidden layers with 100 nodes in each layer but requires much less time to compute.

2) tanh (hyperbolic tangent) activation function

I) 2 hidden layers with 20 nodes in each layer

Precision Score	Recall Score	Test F1 Score	Time
0.9141 ±0.0104	0.9130 ±0.0102	0.9131 ±0.0102	6.49

II) 2 hidden layers with 100 nodes in each layer

Precision Score	Recall Score	Test F1 Score	Time
0.9357 ±0.0084	0.9352 ±0.0085	0.9351 ±0.0085	9.95

III) 5 hidden layers with 20 nodes in each layer

Precision Score	Recall Score	Test F1 Score	Time
0.9085 ±0.0101	0.9072 ±0.0105	0.9072 ±0.0104	15.90

IV) 5 hidden layers with 100 nodes in each layer

Precision Score	Recall Score	Test F1 Score	Time
0.9404 ±0.0070	0.9397 ±0.0070	0.9397 ±0.0070	28.37

Discussion:

Besides a much less computation time when we apply the model with the tanh function, the performance result is similar to the model with the ReLU function. The difference between each scenario also makes sense. While increasing the model's complexity improves performance metrics, the computation time will also be increased. The reason why the tanh

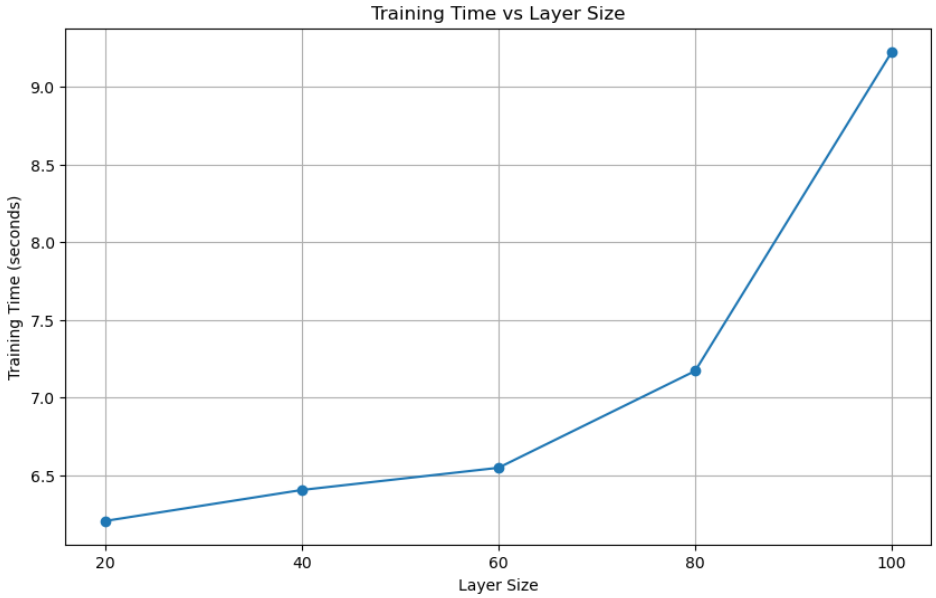
function requires less computation time than ReLU may be because the ReLU function can have a dead neuron problem. The dataset has many 0s; even with the scaling, it doesn't help transfer 0s into other numbers. When a neuron's output is consistently zero, the gradients for that neuron during backpropagation will also be zero. This means that the weights for that neuron won't get updated and become useless for the learning, which further affects the computation time. On the other hand, tanh doesn't have the dead neuron issue.

From the findings, we can observe that the most "budget" solution is also the one with 2 hidden layers with 100 nodes in each layer for the same reason.

3) The time it takes to fit each model as size of layers increase for various layer sizes.

Function used: tanh (hyperbolic tangent) activation function.
Number of layers: 2

Time: [6.20, 6.40, 6.54, 7.17, 9.22]

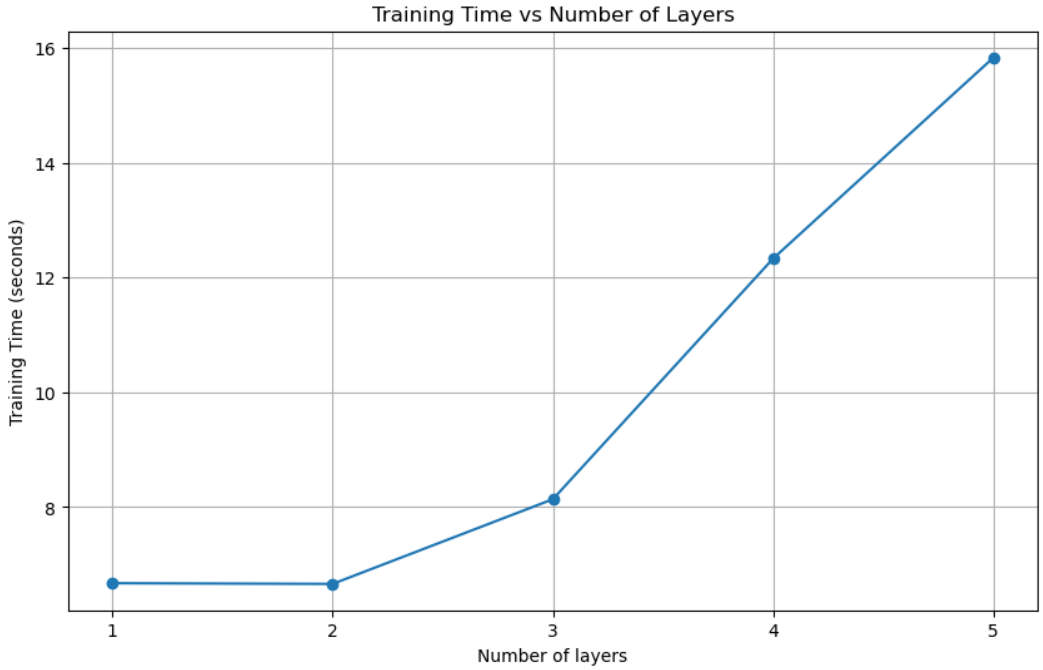


Effect of layers on training time discussion:

From this graph, we can find that when the layer size is increasing from 20 to 100, the computation time is also increasing. More specifically, when the size is from 20 to 60, the grow of the computation time is increasing relatively slow. However, when it is beyond 60, we can see that the computation time is rapidly increased. Generally, more nodes in each layer needs more computation time since the complexity increased. This chart reflects that in the initial stages of training, increasing the small amount of size of each layer may not cause a significant difference in time. However, after it surpasses the certain threshold, the difference will become more and more obvious.

4) The time it takes to fit each model as number of layers increase for various numbers of layers.

Function used: tanh (hyperbolic tangent) activation function.
Time: [6.67, 6.66, 8.14, 12.33, 15.84]
Layer size: 20



Effect of layers on training time discussion:

In general, the computation time grows when the number of layers increases due to as we add more layers, the model becomes more complex. This might be due to the ability of deeper models to learn features more efficiently. In the initial stage, the difference is not obvious, with almost didn't change from 1 to 2 and a slight increase from 2 to 3. On the other hand, starting from 3 layers, the computation time had massive jumps and became exponential, which shares a similar result with the previous one.

4. What happens in simple/smallish network if you stop using `MinMaxScaler`. Why?

Layers: 2 hidden layers with 20 nodes in each layer
Function: tanh (hyperbolic tangent) activation function

Without using MinMaxScaler

Precision Score	Recall Score	Test F1 Score	Time
0.8608 ±0.0106	0.8588 ±0.0109	0.8591 ±0.0109	70.83

Using MinMaxScaler

Precision Score	Recall Score	Test F1 Score	Time
0.9141 ±0.0104	0.9130 ±0.0102	0.9131 ±0.0102	6.49

Effect of scaling discussion:

The goal of using MinMaxScaler is to convert the range of the pixel values to the values between 0 and 1, which range from 0 to 255. Scaling ensures the features have consistent scales and helps the gradient magnitudes be more consistent. Without using the scaling, the LBFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) may fail to converge, which is sensitive to the scale of the problem. The result indicates that performance worsens, and the time spent becomes more than ten times as before. The scaled data can help the optimization landscape become more uniform, which leads to letting the optimization algorithm converge much faster and provides a better scoring in this case.

Q2. In the scikit learn package, the default value for the learning rate of an MLP is 0.001. Explain what will happen to the classification result if we set this parameter to 0.5 and why.

Effect of learning rate:

The learning rate decides the frequency of the model's weight updates. Applying a much higher learning rate is not good for effectively reaching stable solutions. If we set the parameter to 0.5, which is a much higher learning rate, the model will update the weight much more often, which may lead to causing instability to the training process. The model will overreact to the optimal points in the function and may cause it to not converge to the best solution since there is some chance the results could degrade. Since the solutions update so frequently, the model may make the decision back and forth, generating lots of unnecessary computation, which makes it more difficult to reach a stable solution.