McMaster University
Dept. of Electrical and Computer Engineering
COE 4SL4- Term I (Fall) 2023

Assignment 4. Neural Network Classifier with

Two Hidden Layers

Tianze zhang

40028135

zhant22@mcmaster.ca

Instruction:

For the neural network model, I chose to split the test set, validation set and training set with a ratio of 60% training, 20% validation, and 20% test data. The random seed is the last four digits of my student number 8135.

The activation function I am using for the hidden units is Relu, as output logistic sigmoid is used. The gradient descent logic I am using is stochastic GD, which uses one training point each time. For the learning rate, I am starting with 0.005.

In order to initialize the weights I define functions for weight matrix initialization. standard_normal generates random values from a standard normal distribution, random_integers creates binary weights of 1 or -1, and pattern generates a checkerboard pattern of 1s and -1s.

For the early stopping strategy, I first define the epochs inside the defined function for calculating the Classifier, which is a straightforward number limitation.

The numbering limitation of epoch is set to be 50 and I iterate the whole script three times to avoid randomness.

For the n1 and n2 pairs, I iterate n1 from range of 1 to 7 and n2 is 8 -n1, with this set up, I can test all different combination with the constraint of n1 + n2 <= 8 below is the result table with learning rate of 0.005.

n1	n2	Misclassification Rate	entropy_loss
1	1	0.4788	16.3117
1	2	0.4727	16.3117
1	3	0.4848	16.3117
1	4	0.4606	8.2216
1	5	0.4970	10.3432
1	6	0.4667	16.3117
1	7	0.4788	16.3117
2	1	0.4970	8.1997
2	2	0.4242	10.4945
2	3	0.4242	16.3117
2	4	0.4606	16.3117
2	5	0.4364	16.0194
2	6	0.4667	14.8647
3	1	0.4788	16.3117

| 3 | 2 | 0.4788 | 9.2959
| 3 | 3 | 0.4364 | 12.1680
| 3 | 4 | 0.4909 | 9.3690
| 3 | 5 | 0.4485 | 12.0803
| 4 | 1 | 0.4848 | 16.3117
| 4 | 2 | 0.4242 | 0.9208
| 4 | 3 | 0.4242 | 9.7052
| 4 | 4 | 0.4364 | 10.7429
| 5 | 1 | 0.4667 | 14.1851
| 5 | 2 | 0.4485 | 12.6430
| 5 | 3 | 0.4545 | 3.9025
| 6 | 1 | 0.4545 | 6.1388
| 6 | 2 | 0.4485 | 12.5407
| 7 | 1 | 0.4424 | 11.4884

Best n1: 4
Best n2: 2

Best Misclassification Rate: 0.42424242424242425
Best Entropy Loss: 0.9208232617657668

Best Weight 1: [[1.28448885 -0.91402788 1.07002895 -1.07853617 0.9406446]
 [-0.80763153 0.95039308 -1.22432776 1.23159807 1.11768223]]

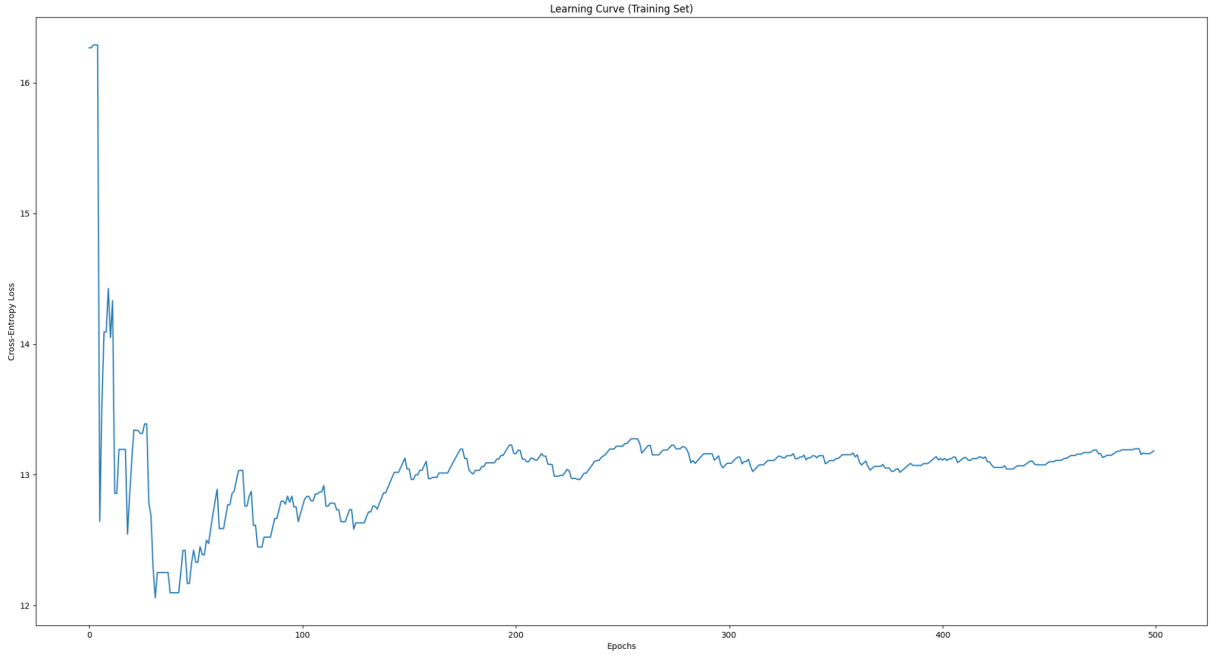
Best Weight 2: [[-0.77430561 1.15987117 1.35142006]
 [1.06744419 1.15291018 -0.99609265]]

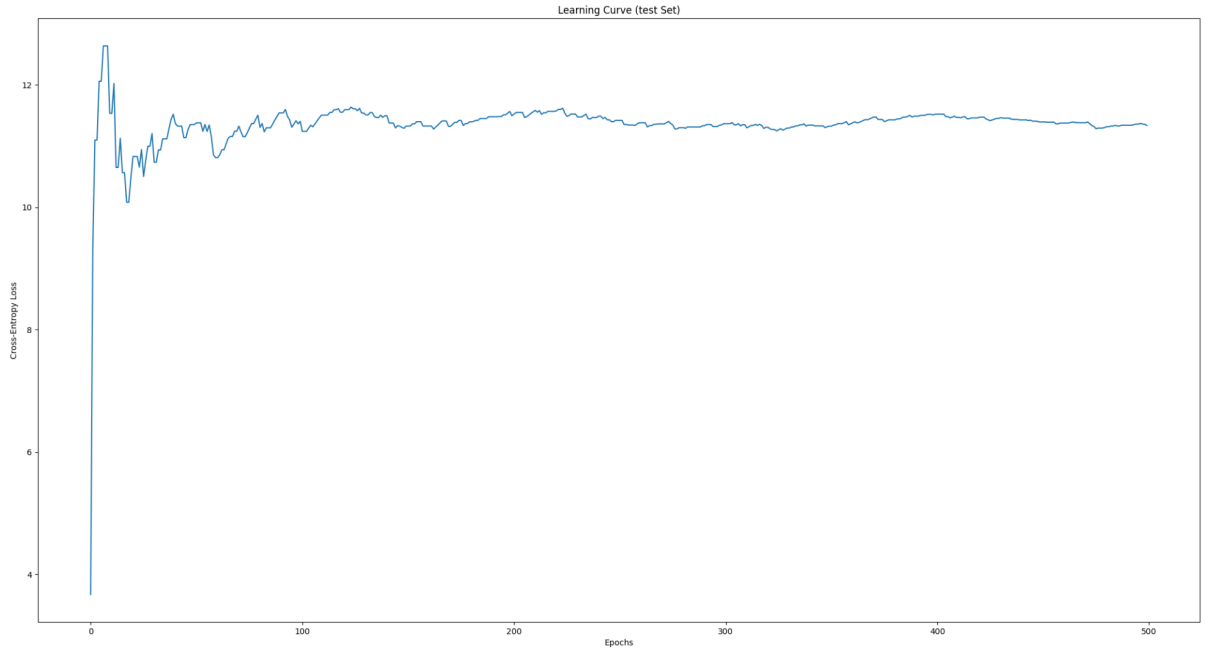
Best Weight 3: [[-0.77430561 1.15987117 1.35142006]
 [1.06744419 1.15291018 -0.99609265]]

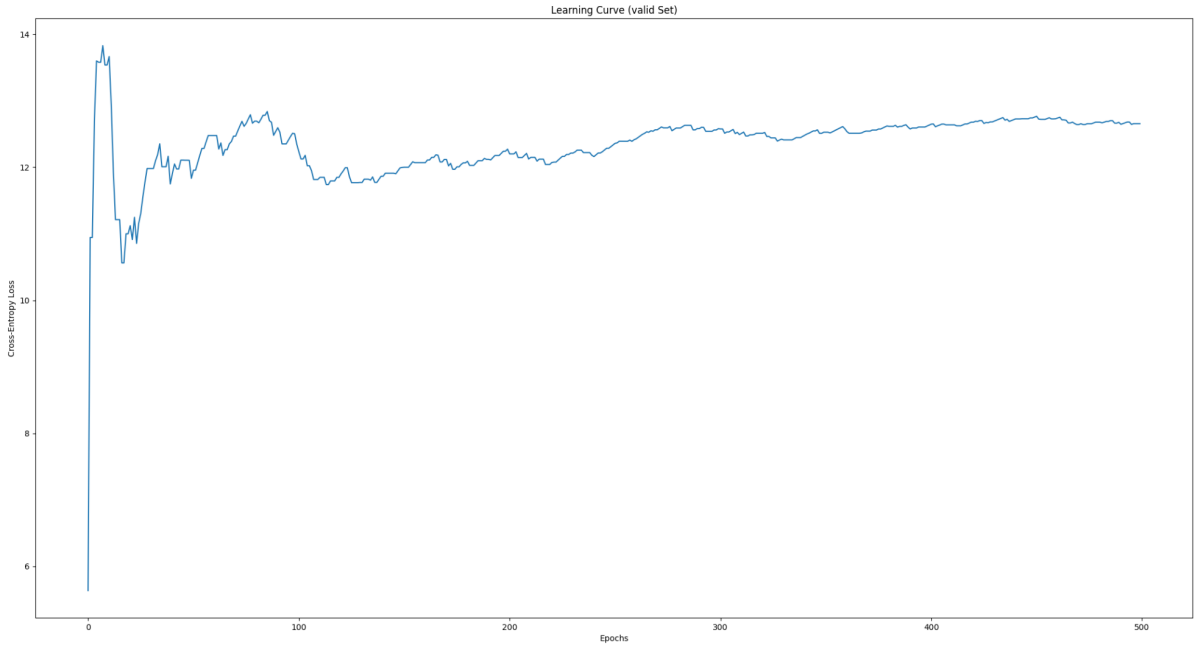
As we can see from the result, the best paris is when n1 = 4 and n2 =2. With this set up, we can have the lowest Misclassification Rate at 0.42 and the Entropy Loss is 0.920, so I think the learning rate at this stage is good enough.

Training the model with the best n1 and n2 pair we tested out:

As we can see from above data, the best n1 and n2 we can get according to MR is n1 = 4 and n2 =2, so let us train the model with this pair of number of node in each layer. Let us first find out the learning curve in training set, test set and validation set.



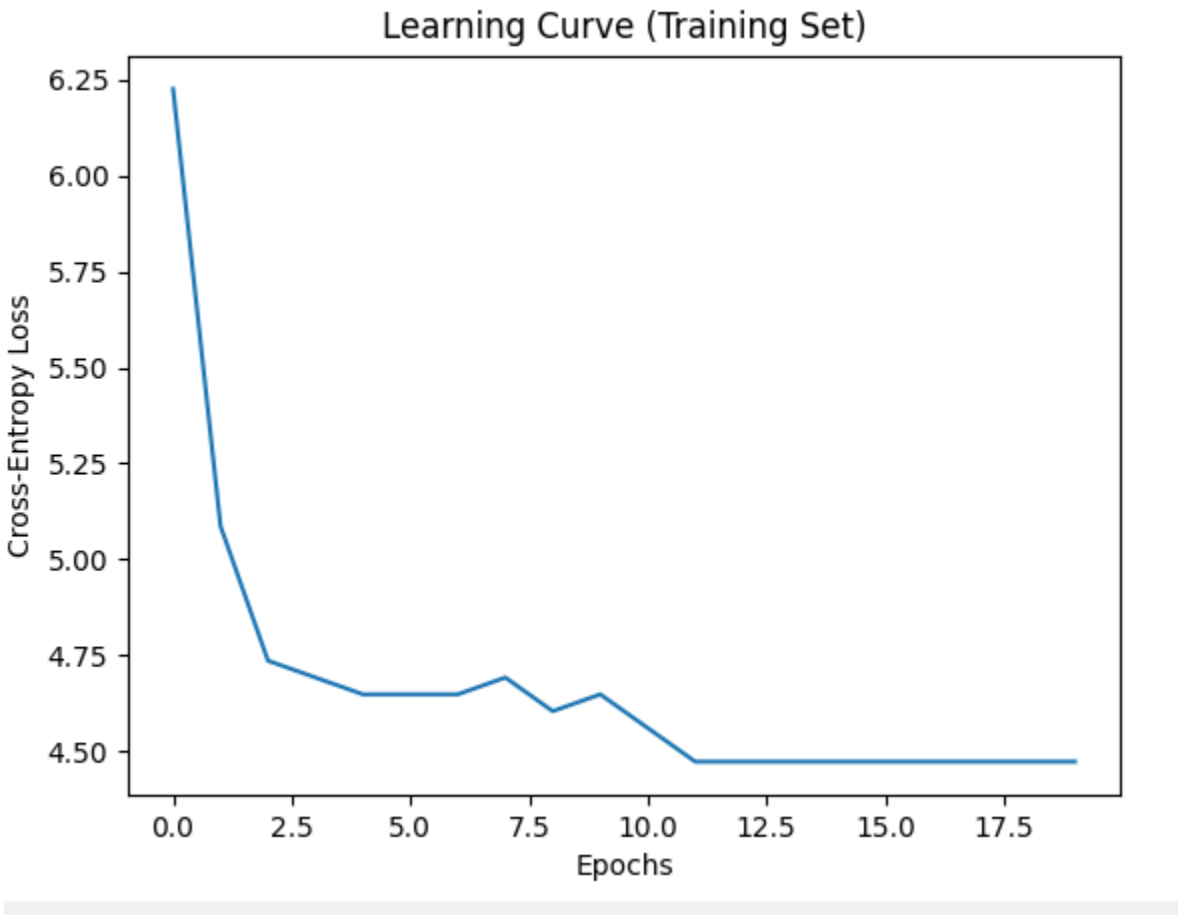


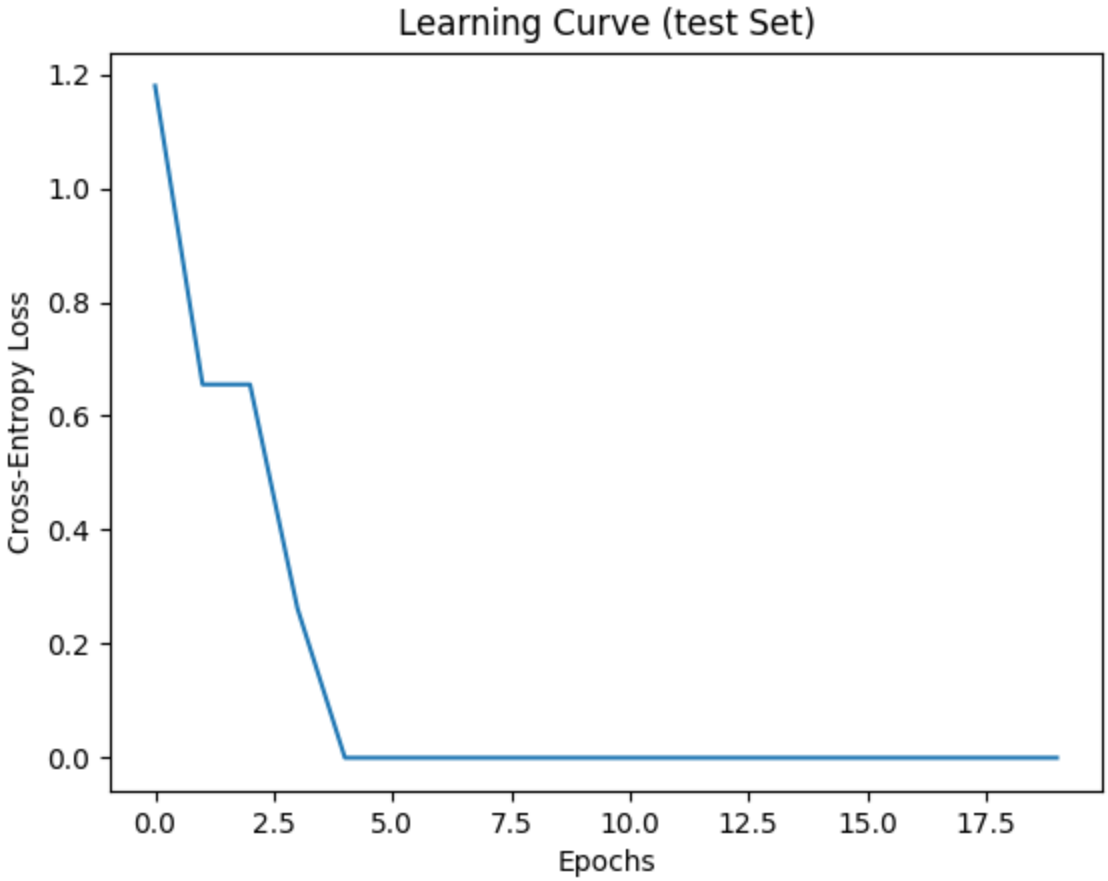


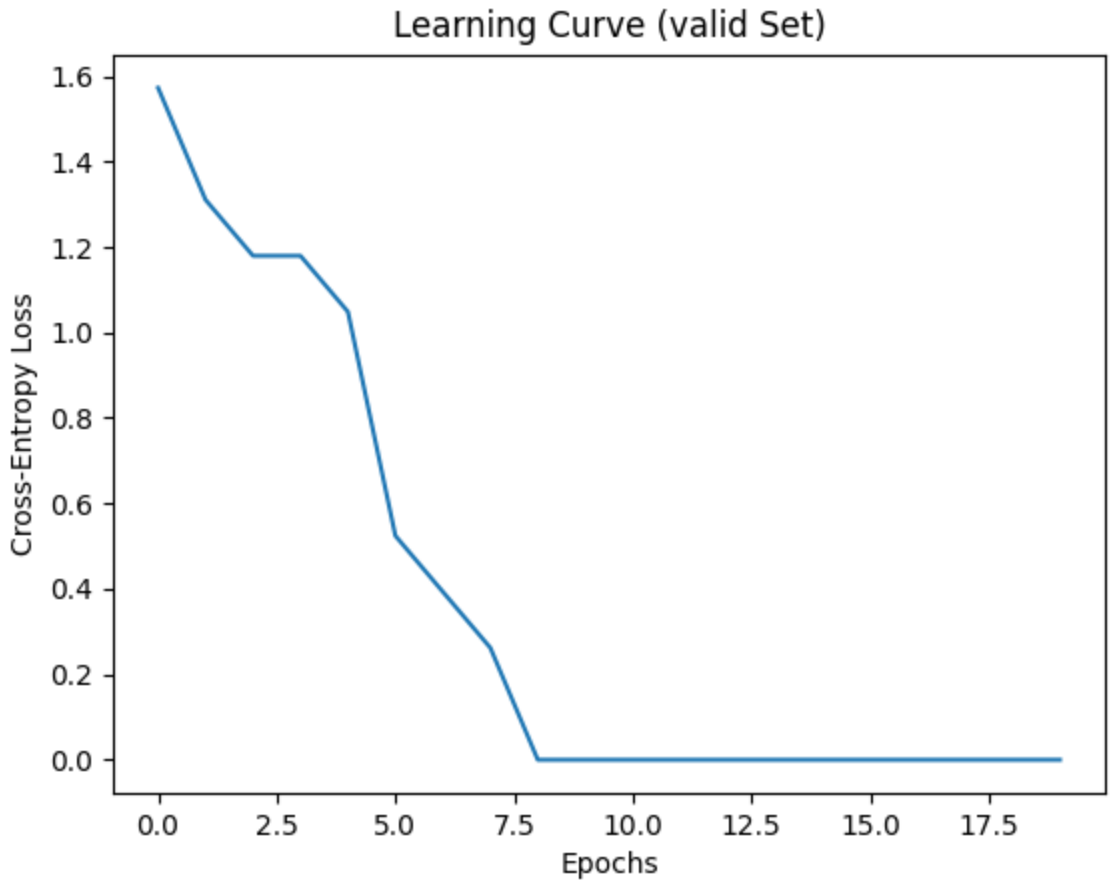
As we can see, when we set the epochs to 500, the cross-entropy loss of training set, test set and validation set have the tendency of overfitting after around 50 epochs. With this in mind, let's decrease the epochs to 100 so that we can zoom in the plot and find more details.

Misclassification Error on Training Set: 0.5474452554744526
Misclassification Error on Validation Set: 0.4909090909090909
Misclassification Error on Test Set: 0.41454545454545455

Learning rate at 0.005




Learning Curve (test Set)
Cross-Entropy Loss
Epochs
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5



The method for choosing couples involved trying every possible set of numbers between 1 and 7. Larger ranges were experimented with for a while, but performance is severely degraded by severe overfitting caused by an excessive number of neurons. These pairs were adequate since the best solution for each set of features remains constant as the number of nodes increases.

Additionally, we observe that the error lines for learning rate 0.005 is converging within 10 epochs, which suggests that the learning rate is acceptable; in fact, as long as the learning rate is not too big, it is all acceptable. The selection of the learning rate was based on the presumption that the ideal model's current performance is preferable to the worst-case outcome of training a model with an excessively high learning rate.

In conclusion, the number of n1 and n2 has a sweet spot around 2-6, if the number of nod is bigger than 6, overfitting tends to happen, below; the model is too simple, underfitting might happen.

Notice: due to random in seed and initial weight. The Plot of the learning curve will show as a straight line sometimes. But it can be fixed by running the code again.