AMAN BHARDWAJ

(2019SIY7580)

Assignment 3 – Part B: Neural Networks

Q2. Neural Networks

a) Implement a generic neural network architecture.

Key Features:

- One Hot Encoding for labels
- Mini Batch Stochastic Gradient Descent as Solver
- Mean Squared Error as loss function.
- **Generic Architecture:**
  - **Mini-Batch Size (M)**
  - **Number of features/attributes (n)**
  - **Hidden layer architecture:** List of number of perceptron in each hidden layer. **Eg. [100,50]**
  - **Number of target classes (r)**
  - **Activation Function for each layer: Eg. Sigmoid / ReLU**

b) A neural network having a single hidden layer: {1, 5, 10, 50, 100}

Learning Rate = 0.1
Mini Batch Size = 100
Activation Function = Sigmoid (Both Hidden and Output layers)

Stopping Criteria: I have kept it a combination of 3 things.

i. Change in Error/Cost taken **averaged over last 10 epochs** gives change less than threshold value (**Epsilon) = 1e-06.** i.e. the change is error is not significant over last 10 epochs.
ii. Error/Cost **reaches below** a certain threshold = **0.02** Error reaches a certain acceptable value which gives good accuracy. And further reduction in error is resulting in overfitting of model which reduces the test accuracy. Best accuracy is achieved at 0.02 in my implementation.
iii. Number of epochs exceeds **Max_Epochs** = **10,000**

The algorithm will stop if any of the above-mentioned criteria has been met. By above stopping criteria I can achieve early stopping and prevent overfitting.

Accuracy Table (Fixed Learning Rate):

No of Perceptron in Hidden Layers	Model Training Time	Min. Cost Reached	Total Epochs	Train Accuracy	Test Accuracy
1	0.41 Mins	0.48077	84	03.84%	03.84%
5	30.95 Mins	0.41156	6032	21.14%	20.45%
10	53.88 Mins	0.36250	10000	33.59%	31.23%
50	30.83 Mins	0.02968	3902	96.13%	84.63%
100	7.88 Mins	0.01999	840	97.60%	88.06%

Plots:

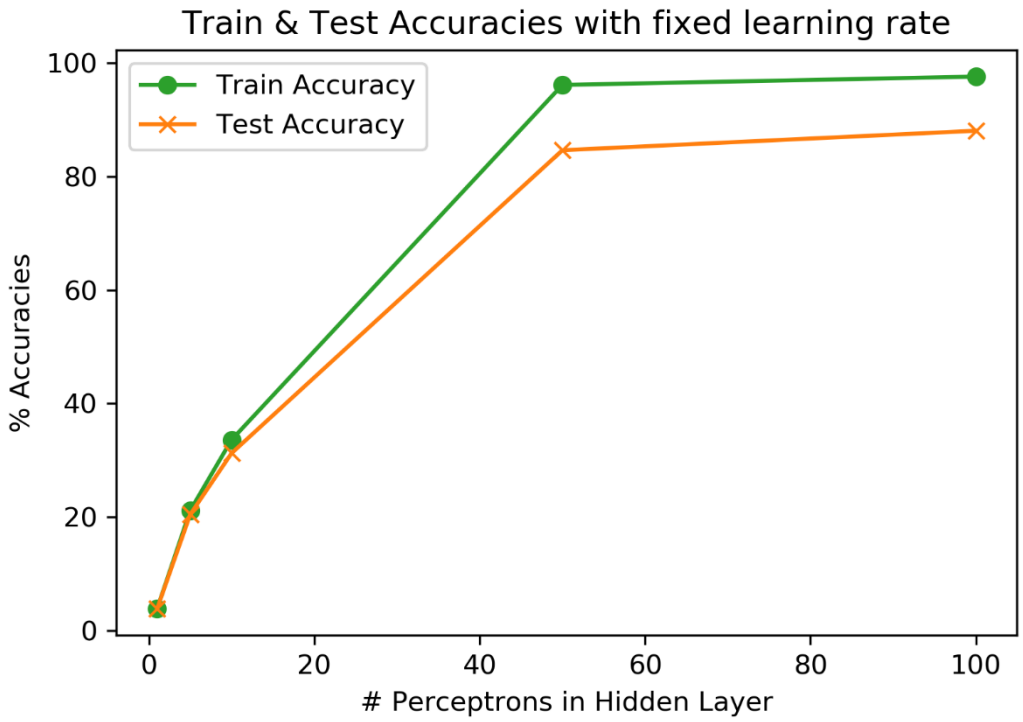

Figure: Train and Test Accuracy with fixed Learning Rate for {1,5,10,50,100} Perceptron

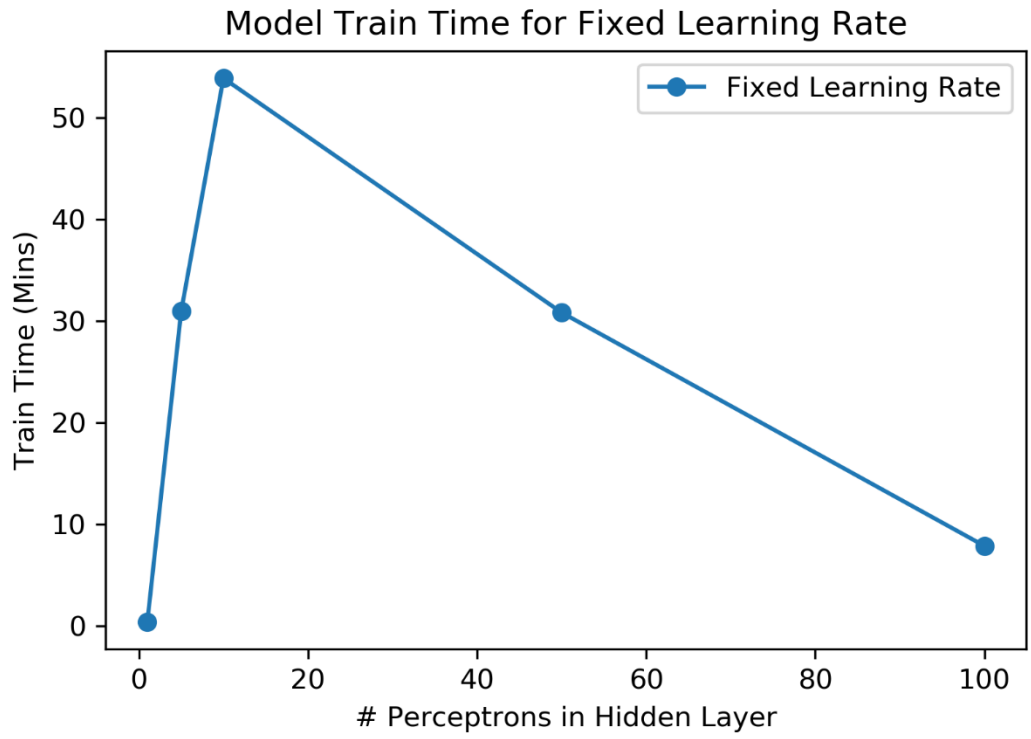

Figure: Train Time (Mins) for fixed Learning Rate for {1,5,10,50,100} Perceptron

Observations:

- For Model with **1 perceptron** in the hidden layer, it doesn't learn much about the data. The rate of change of error is very slow for the given learning rate for this case.
- The gradient descent early stopped for the case of 1 perceptron at under 100 epochs because the change in error was of order less than 1e-06. Therefore, these models did not learn anything from the data.
- For Model with **5 and 10 perceptron** the accuracy improves a bit. But they took approx. **6K and 10K epochs** to reduce the cost by very little amount. Even after running for so long they were able to get very less accuracy.
- With increase in number of perceptron in the hidden layer from 1 to 100**, both Train and Test accuracy increases**.
- But the **rate of increase** in accuracy is **more** from 1 to 50 as compared to the rate of increase of accuracy from 50 to 100 perceptron in the hidden layer.
- This shows that after a certain point the **model will saturate with increase in the number of perceptron** in each hidden layer.
- Number of epochs to learn the model first increase from 1 to 10 perceptron and then reduce from 10 to 100 perceptron in the hidden layer.
- Same is the case with time to learn the model.
- Time taken per epoch increase with number of perceptron in the hidden layers.

c) Part b with Adaptive Learning Rate: 1 hidden layer {1, 5, 10, 50, 100}

Learning Rate = 0.5/sqrt(epoch)
Mini Batch Size = 100
Activation Function = Sigmoid (Both Hidden and Output layers)

Stopping Criteria: Again, I have kept same stopping criteria as before.

i. Change in Error/Cost taken **averaged over last 10 epochs** gives change less than threshold value (**Epsilon**) **= 1e-06.** i.e. the change is error is not significant over last 10 epochs.
ii. Error/Cost **reaches below** a certain threshold = **0.02** Error reaches a certain acceptable value which gives good accuracy. And further reduction in error is resulting in overfitting of model which reduces the test accuracy. Best accuracy is achieved at 0.02 in my implementation.
iii. Number of epochs exceeds **Max_Epochs** = **10,000**

The algorithm will stop if any of the above-mentioned criteria has been met. By above stopping criteria I can achieve early stopping and prevent overfitting. **Note**: I tried to increase/decrease it, but the best stopping was achieved at the same value = **1e-06**

Accuracy Table (Adaptive Learning Rate):

No of Perceptron in Hidden Layers	Model Training Time	Min. Cost Reached	Total Epochs	Train Accuracy	Test Accuracy
1	0.33 Min	0.48078	68	03.84%	03.84%
5	7.85 Mins	0.46550	1599	10.00%	09.81%
10	50.56 Mins	0.46778	10000	07.85%	07.75%
50	71.12 Mins	0.06176	10000	92.30%	82.18%
100	32.0 Mins	0.03839	3687	95.51%	86.48%

Plots:

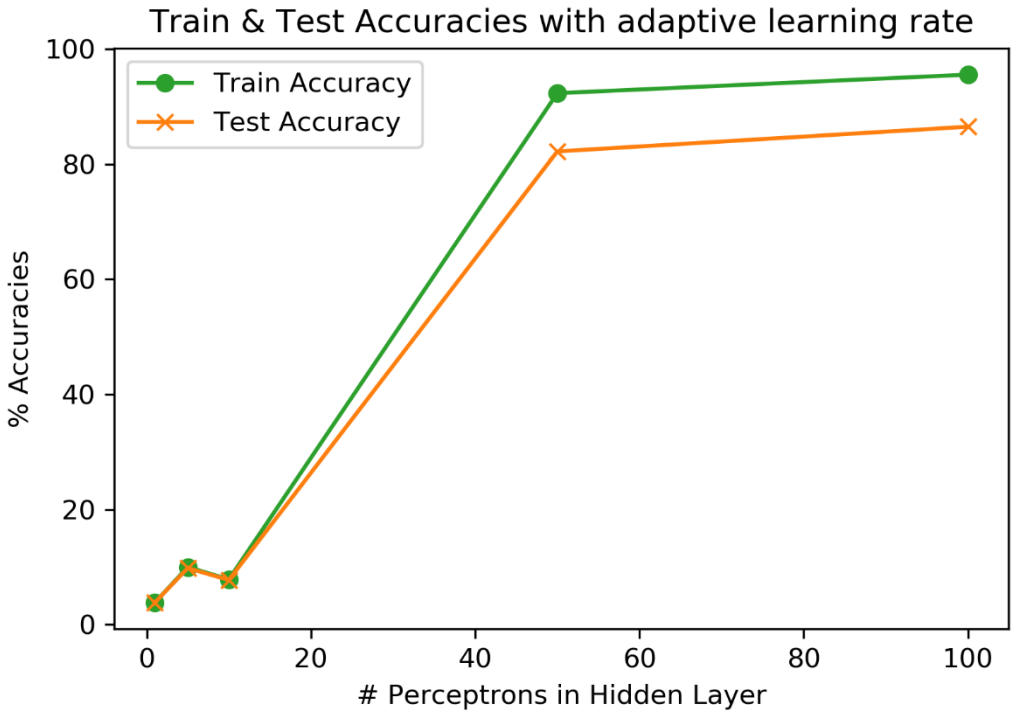

Figure: Train & Test Accuracy with Adaptive Learning Rate for {1,5,10,50,100} Perceptron

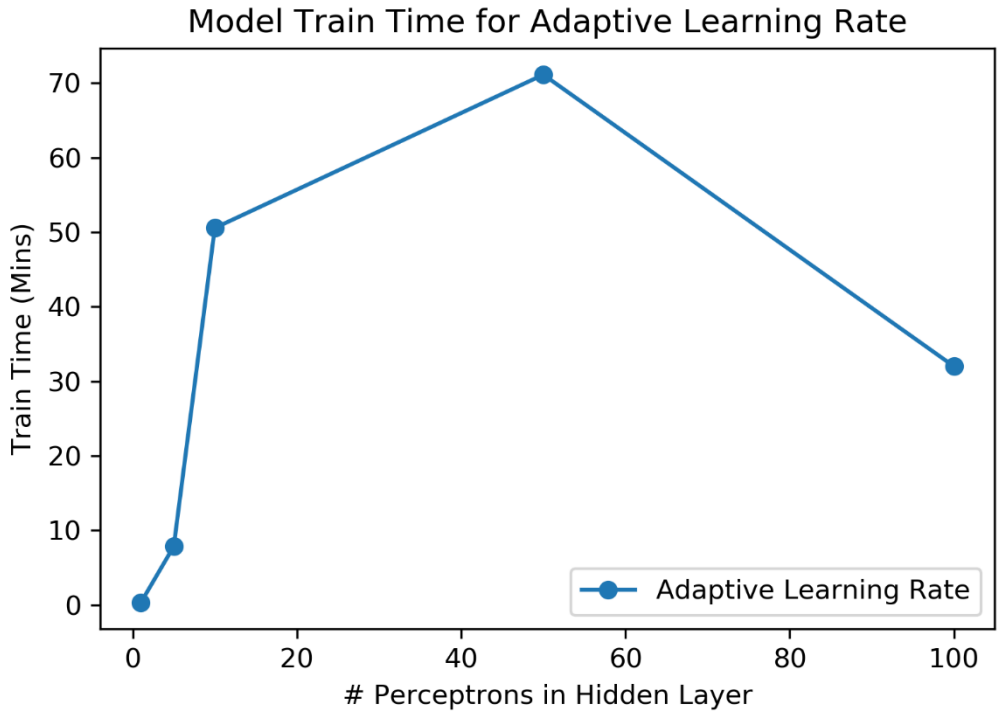

Figure: Train Time (Mins) for Adaptive Learning Rate for {1,5,10,50,100} Perceptron

Difference between Fixed (F) and Adaptive (A) Learning Rate matrices.

No of Perceptron in Hidden Layers	Training Time (A-F)	Train Accuracy (A-F)	Test Accuracy (A-F)
1	- 00.08 Min	00.00%	00.00%
5	- 23.10 Mins	-11.14%	-10.63%
10	- 03.32 Mins	- 25.74%	- 23.47%
50	+ 40.29 Mins	- 03.84%	- 02.44%
100	+ 24.12 Mins	- 02.09%	- 01.58%

Graphical comparison between Fixed (F) and Adaptive (A) Learning Rate matrices.

- **Learning Time:**

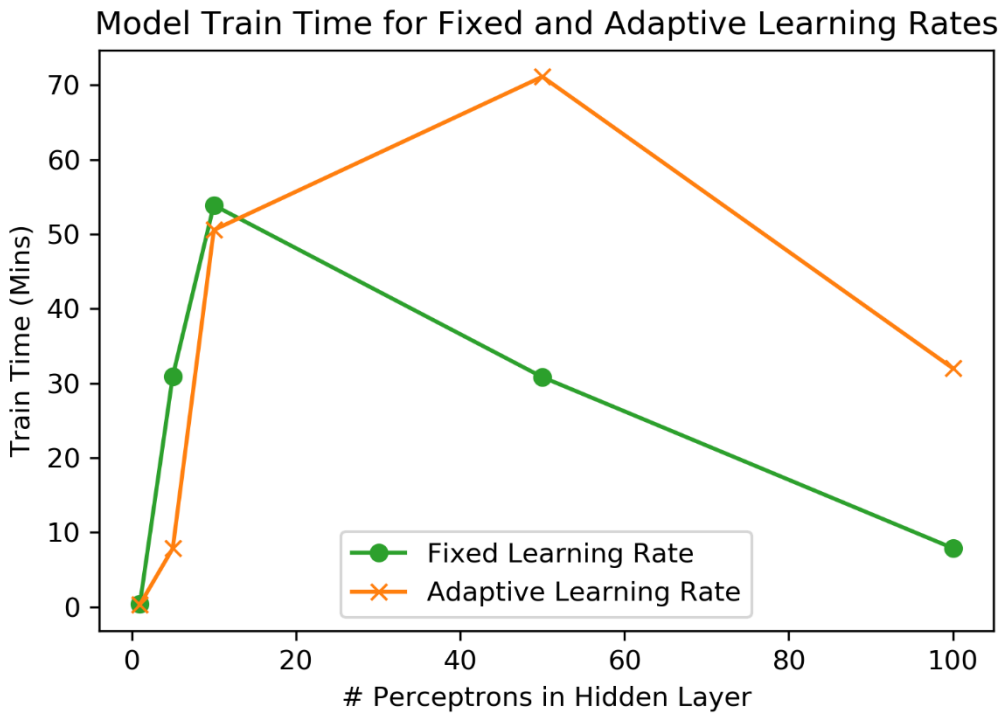

Figure: Comparison between Learning time for Fixed and Adaptive Learning rates v/s number of perceptron units in the hidden layer.

- **Train Accuracy**

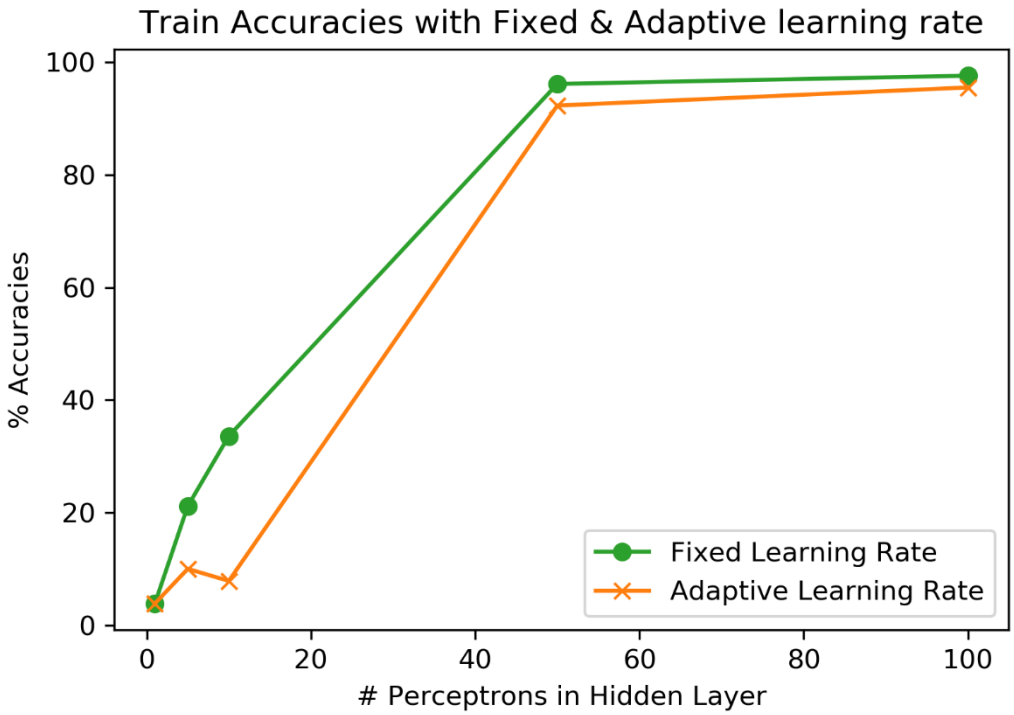

Figure: Comparison between Train Accuracy for Fixed and Adaptive Learning rates v/s number of perceptron units in the hidden layer.

- **Test Accuracy**

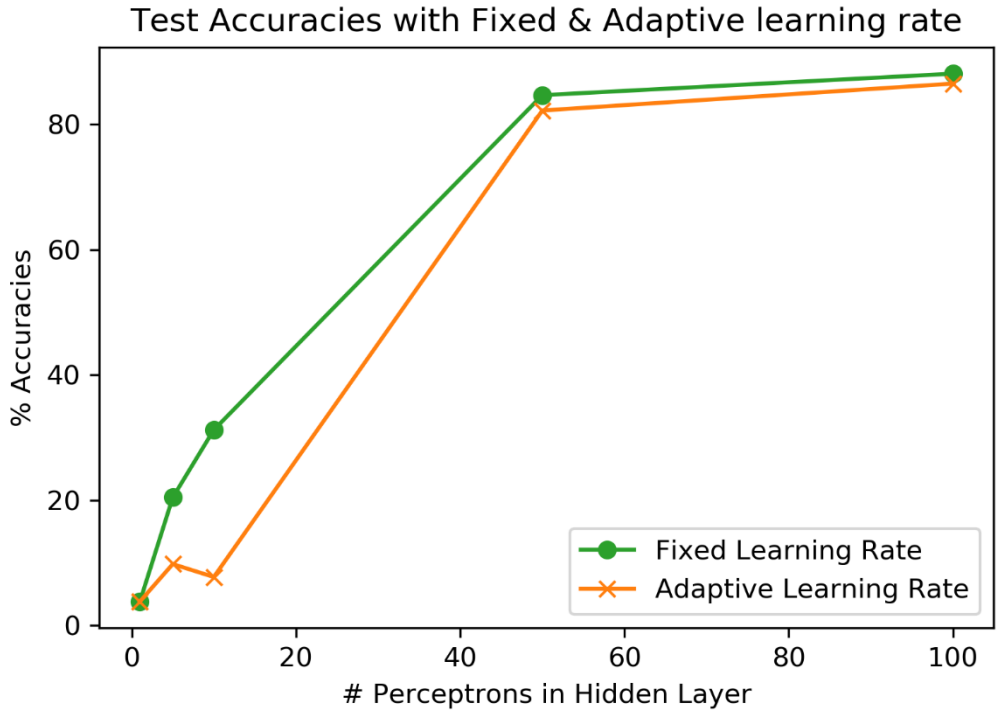

Figure: Comparison between Test Accuracy for Fixed and Adaptive Learning rates v/s number of perceptron units in the hidden layer.

Observations:

- **Learning Time:** In our case adaptive learning rates performed better for 1,5,10 perceptron units in hidden layer but took a lot more time than fixed learning rate for the case of 50, and 100 perceptron units. Ideally the purpose of adaptive learning rate is to reduce the learning time and help the model to converge faster as compared to fixed learning rate and still give the similar accuracies. But in our case, this seems to be wrong as even after running for longer time **we got lesser accuracies for all cases**. This proves that we need to do hyperparameter tuning for the adaptive learning rate too.
- **Train and Test accuracy:** Despite of taking a lot more time and epochs to fall below the error change rate of 1e-06, it did not converge well i.e. it didn't reach to the minimum error point (0.02) at which it gave the best accuracy. And thus**, performed a little bit bad in case of adaptive rate**. So, our purpose of adaptive learning rate was not solved.
- **Good Adaptive Rate:** Instead of **0.5/sqrt(epoch),** if we use **(epoch)^1/4** then that would have performed a lot better and would have resulted in faster convergence.
- **Strange behavior:** The accuracy for **10 perceptron is slightly less than 5 perceptron,** in case of adaptive learning rate. This is because for 10 perceptron case we stopped at 10k. If we had gone further (which is required for this LR), then it would have given better accuracy than 5 perceptron case for sure but with very low rate of change in error/cost. Because for fixed LR Too it learned for 10K epochs, therefore for this adaptive LR it needed to go further but was stopped by the stopping criteria. This **behavior is not strange** and can be explained.

d) ReLU as Activation Function for hidden layers [100,100]:

- **[100,100] ReLU Hidden Layers & Sigmoid Output Layer:**
  - Learning Time **= 0.94 Min**
  - Number of Epochs **= 80**
  - Train Accuracy = **98.29 %**
  - Test Accuracy **= 91.83 %**

- **[100,100] Sigmoid Hidden Layers & Sigmoid Output Layer:**
  - Learning Time **= 9.29 Min**
  - Number of Epochs **= 825**
  - Train Accuracy **= 97.30 %**
  - Test Accuracy **= 87.75 %**

Stopping criteria used:

Again, I have kept same stopping criteria as before.

i. Change in Error/Cost taken **averaged over last 10 epochs** gives change less than threshold value (**Epsilon**) **= 1e-06.** i.e. the change is error is not significant over last 10 epochs.

ii. Error/Cost **reaches below** a certain threshold = **0.02** Error reaches a certain acceptable value which gives good accuracy. And further reduction in error is resulting in overfitting of model which reduces the test accuracy. Best accuracy is achieved at 0.02 in my implementation.

iii. Number of epochs exceeds **Max_Epochs** = **10,000**

The algorithm will stop if any of the above-mentioned criteria has been met. By above stopping criteria I can achieve early stopping and prevent overfitting.

Note: I tried to with adaptive learning rate here but since it was not tuned well. Even Sklearn proves the same thing in case of 'invscaling'. So, I have done this with fixed rate of 0.1

Observations:

- **ReLU** Converged in just 80 epochs and 0.94 Mins. While the other took 825 epochs and 9.3 Mins.
- **Accuracy in case of ReLU** is quite good as compared to all the previous models.
- **Sigmoid's** performance was similar to previous models with best accuracies.

Which one of above performed better?

- **ReLU Model performed better than Sigmoid model** in terms of both Learning time and Accuracies.

Comparison with part (b) with one hidden layer:

- In Case of ReLU as activation for two hidden layers, it **performed a lot better** than the part b with 1 hidden layer. Learning time is very less and accuracy is quite good for this model.
- But the model with 2 hidden layers of sigmoid performed similar to the best model of part b. There results were comparable in terms of both Accuracy and Learning Time.
- This says that in this scenario ReLU is better activation function than Sigmoid for hidden layers. It performs quite well as compared to the other.

e) MLPClassifier from Sklearn Library:

- **[100,100] ReLU Hidden Layers: (Constant LR = 0.1)**
  - Learning Time **= 0.86 Min**
  - Number of Epochs **= 66**
  - Train Accuracy **= 100 %**
  - Test Accuracy **= 92.062 %**

- **[100,100] ReLU Hidden Layers: (Adaptive LR)**
  - Learning Time **= 1.13 Min**
  - Number of Epochs **= 147**
  - Train Accuracy **= 100 %**
  - Test Accuracy **= 92.338 %**

Comparison with d part:

- If we compare with custom implementation of ReLU (100,100) in part d with sklearn results. Custom implementation performed quite similar to Sklearn both in terms of Learn time and accuracy. Custom took **0.08 mins and just 14 epochs more** than sklearn.
- And in terms of accuracy,
    - Train Accuracy (Custom - Sklearn) = **- 1.71 %**
    - Test Accuracy (Custom - Sklearn) = **- 0.23 %**

  The results are quite **comparable** of sklearn and custom implementation. There is not much difference.

****