Assignment 3B

Priyadarshi Hitesh

28 April 2020

1 Neural Networks

Before Implementing we have normalized the given data by dividing every pixel value by 255. This is done for every part of the assignment.

1.1 Implementing Neural Network

Neural Network algorithm has been implemented from scratch for Multi-class Classification. In this outputs are converted into one hot encoded form. We have used Mini-Batch Stochastic Gradient Descent to train the network. A fully connected architecture is assumed while implemented the algorithm.

The implementation is generic for following parameters :

(a) Mini-Batch Size

(b) Number of features/attributes

(c) Hidden Layer Architecture : List of numbers denoting the number of perceptrons in the corresponding hidden layer.

(d) Number of target classes

1.2 Experimenting with the implemented network

1.2.1 Fixed Parameters :

To train the network following parameter are fixed.

1. Number of Hidden Layer = 1
2. $\eta = 0.1$
3. Hidden Layer Units = {1,5,50,100}
4. Mini Batch Sizer = 100

1.2.2 Convergence Criteria :

For every epoch we train the network using stochastic gradient descent, where summation of cost of every batch is calculated and then average is taken. We are using difference between consecutive average cost of epochs. If it is less than 1e-5 then it would terminate.

1.2.3 Accuracy

Hidden Layer Units	1	5	50	100
Train Accuracy	3.7692%	3.9153%	89.8%	97.1923%

Hidden Layer Units	1	5	50	100
Test Accuracy	3.8461 %	4.1538 %	85.8461%	90.9846%

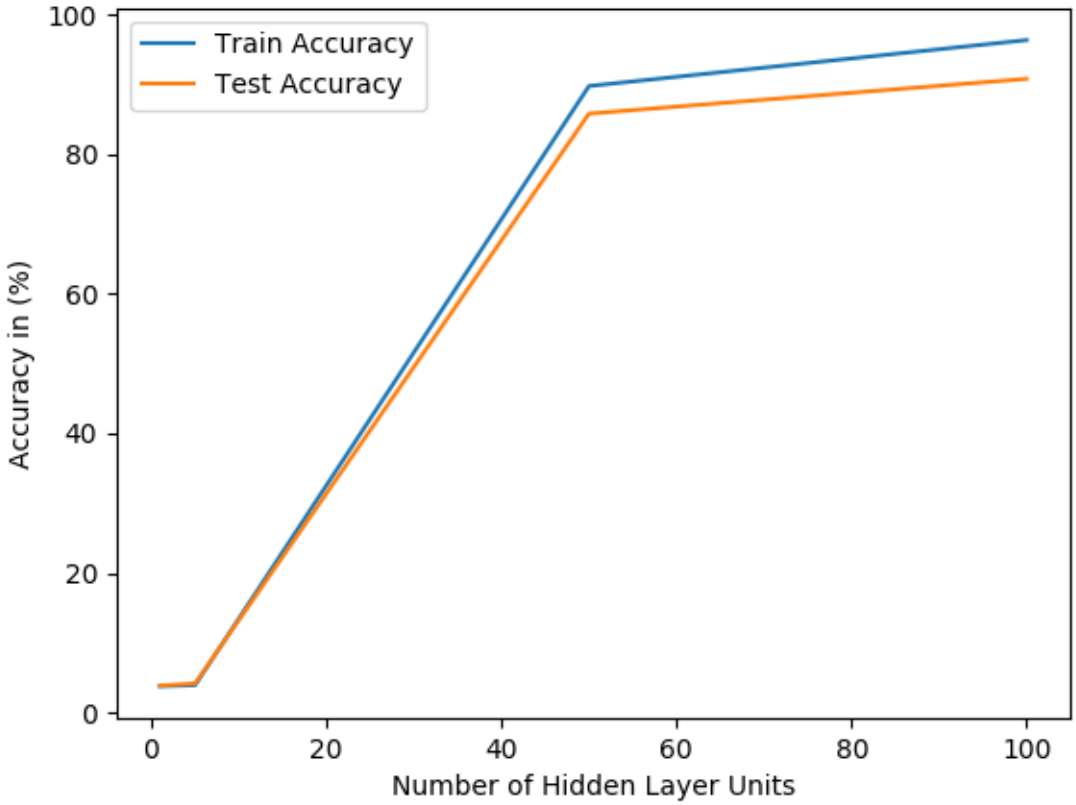

Figure 1: Accuracy for different hidden layer units

1.2.4 Total epochs while training the network

Hidden Layer Units	1	5	50	100
Total epochs	51	63	2246	2441

1.2.5 Time taken to train the network

Hidden Layer Units	1	5	50	100
Time taken	3.9637 secs	4.1251 secs	617.4890 secs	825.7314 secs

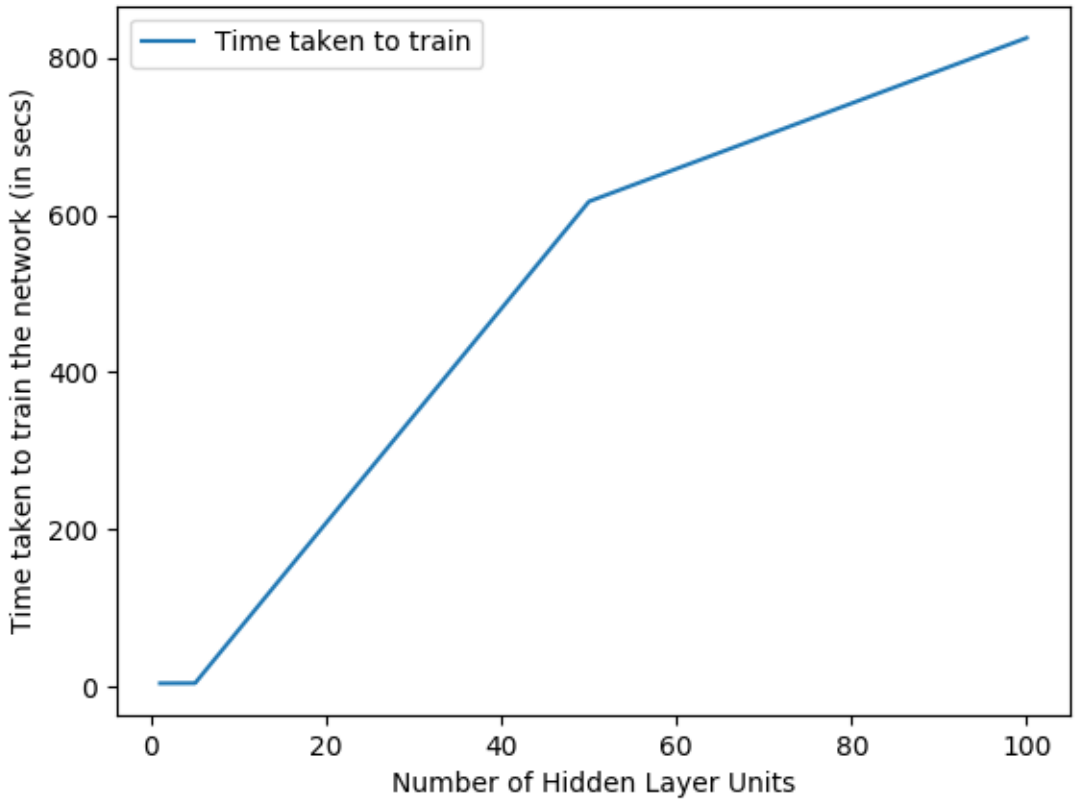

Figure 2: Time taken to train the network for different hidden layers

1.2.6 Observations

1. For small hidden layer units, it is taking almost few seconds to train the model.
2. For large hidden layer units, time taken to train the model is increasing substantially as we can see it from the above tables.
3. For small hidden layer units, network is not learning well as we can see from table of accuracy. Accuracy is like we have some random prediction.
4. For large hidden layer units like 50,100 we can see that network has learned well as we see it from train and test accuracy.
5. So, we can clearly observe that network need some considerable hidden layer units, else it won't learn properly.
6. If we decrease our convergence criteria from 1e-6 to further smaller value, then it could also lead to overfitting.

1.3 Using Adaptive Learning Rate

In this part, we are keeping everything same just like in above part. The only change is learning type for the network. Here the learning type is "Adaptive" which basically says that learning rate(η) would get updated after every epoch i.e. $\eta_e = \eta_0/\sqrt{e}$ where $\eta_0 = 0.5$ and e is the current epoch number.

1.3.1 Convergence Criteria :

Convergence criteria for adaptive learning is same as without adaptive learning. For every epoch we train the network using stochastic gradient descent, where summation of cost of every batch is calculated and then average is taken. We are using difference between consecutive average cost of epochs. If it is less than 1e-5 then it would terminate. When we tried to make convergence criteria 1e-6 it wasn't converging even after being trained for more than half an hour. So, it has been kept same as earlier.

1.3.2 Accuracy

Hidden Layer Units	**1**	**5**	**50**	**100**
Train Accuracy	3.5692 %	3.8615 %	88.4923%	90.1769%

Hidden Layer Units	**1**	**5**	**50**	**100**
Test Accuracy	3.7230 %	3.9076 %	84.7693%	86.5076%

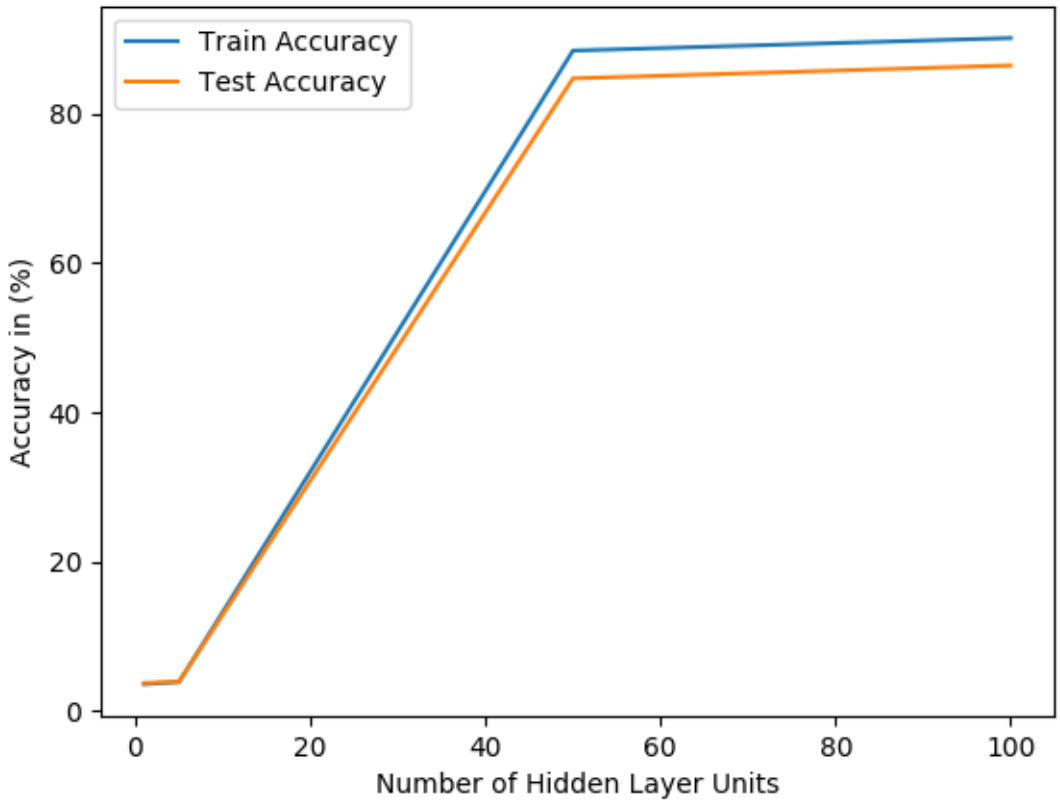

Figure 3: Accuracy for different hidden layer units for apaptive learning

1.3.3 Total epochs while training the network

Hidden Layer Units	**1**	**5**	**50**	**100**
Total epochs	35	55	2475	2480

1.3.4 Time taken to train the network for adaptive learning

Hidden Layer Units	1	5	50	100
Time taken	4.6191 secs	6.8160 secs	466.5153 secs	723.1876 secs

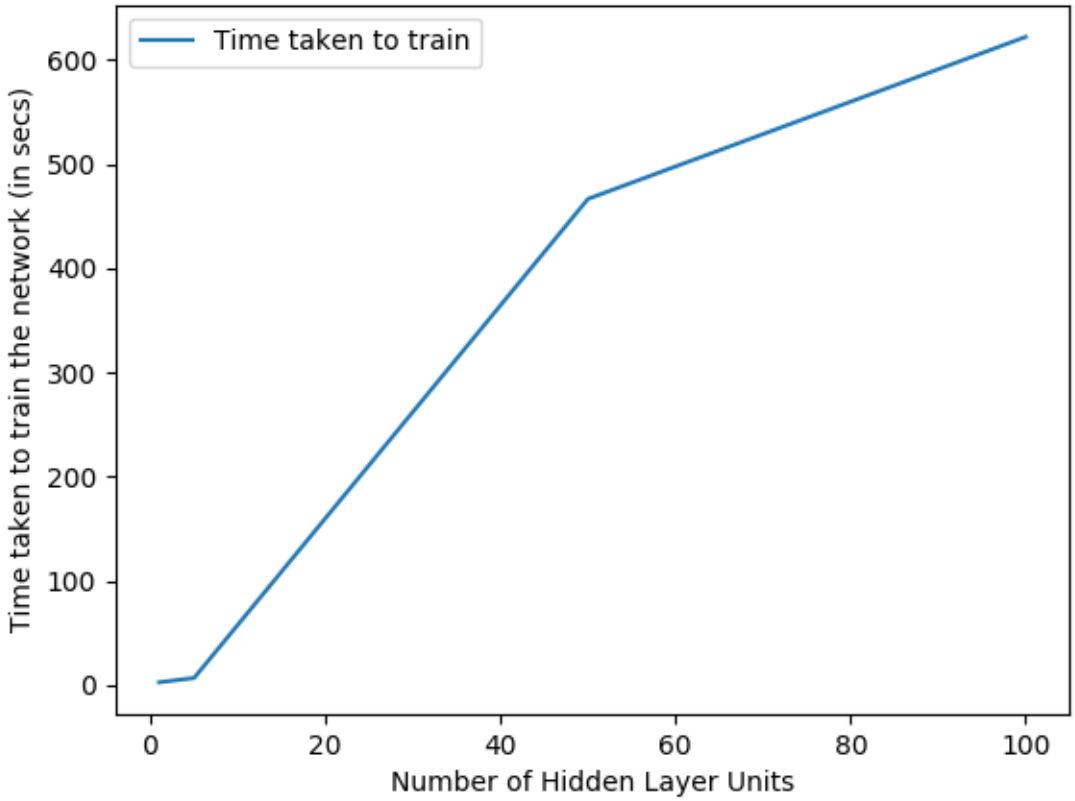

Figure 4: Time taken to train the network for different hidden layers for adaptive learning

1.3.5 Observations :

1. Network trained using adaptive learning rate takes more time than what it had taken in non-adaptive learning.
2. We can see accuracy has also been decreased than what we had earlier for each hidden layer units.
3. Result we have got using adaptive learning rate is not better than what we had earlier in any aspect.
4. Adaptive learning rate has taken more time than non-adaptive learning rate as learning rate keeps on decreasing with each epoch and that is leading to its slow convergence.

1.4 Experimenting with activation units

A network is implemented with 2 hidden layers with 100 units each. We would experiment with activation type of hidden layer for this network. Learning type has been kept adaptive for this network.

1.4.1 Using "ReLU" as activation unit for hidden layers

Train Accuracy	Test Accuracy	Time taken to train	Total Epoch
94.9538 %	89.5538 %	413.3364 secs	879

1.4.2 Using "sigmoid" as activation unit for hidden layers

Train Accuracy	Test Accuracy	Time taken to train	Total Epoch
90.1230 %	86.124 %	1636.025 secs	4697

1.4.3 Relative Comparison between above two networks

1. The network with relu activation unit performed better than sigmoid in every aspect.
2. The relu network is taking very less time as compared to sigmoid one.
3. Better test accuracy is obtained through relu network.
4. Relu network converges with less number of epoch than sigmoid one.

1.4.4 Comparison with network having single hidden layer with sigmoid in earlier part

1. Single hidden layer network takes less time to converge than networks of this part.
2. If comparing with relu network test accuracy is almost same.
3. If comparing with sigmoid network single hidden layer network performed slightly better.

1.5 Using MLP Classifier

Here the network has been trained using MLP classifier from Scikit Learn library with architecture same as the above part. Here, binary cross entropy loss is used for multi-class classification.

1.5.1 Using "ReLU" as activation unit for hidden layers

Train Accuracy	Test Accuracy	Time taken to train	Total Iteration
87.6692 %	84.8307 %	583.1231 secs	1056

1.5.2 Comparison between sklearn MLP results and our results

1. We have got better test accuaracy with our own implementation in above part.
2. Our implementation is taking slightly lesser time than scikit learn implementation.
3. Our implementation is overfitting the data upto some extent while scikit learn is avoiding that which is clearly visible from accuacy obtained.
4. Result is varying as convergence criteria of both implementation is not same.