Machine Learning-Assignment 3 (Part B)

Kartik Jain-2019MCS2563

Neural Networks

Part (a)

Implemented the required architecture for neural network.

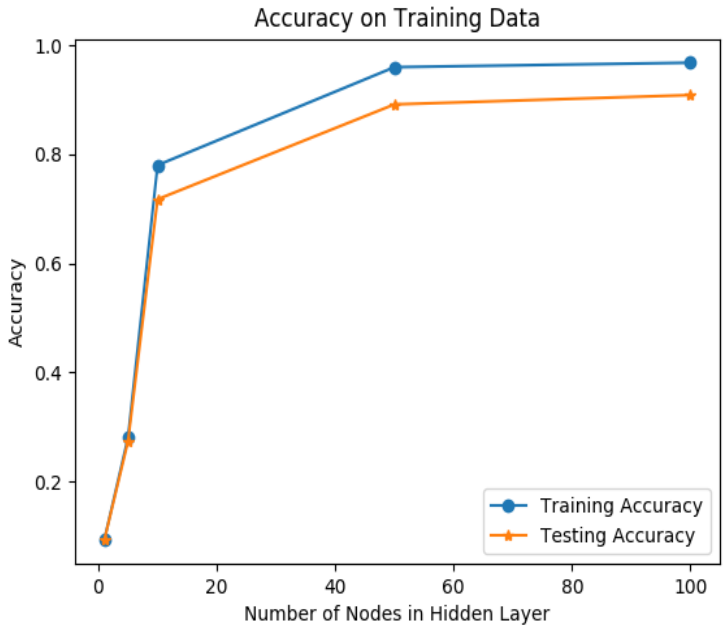
Part (b)

Used the implemented code for training the given architectures and for the stopping criteria I am finding out the average cost after 30 batches and then if the difference between the previous average cost and the current average cost is less than some tolerance limit (taken as 1e-8) then stop otherwise keep iterating until a maximum number of epochs are reached (here taken to be 2000 epochs as the max limit by default).
We observed that the accuracy increased as we increased the number of units in the hidden layer the accuracy and the time taken to train the model both increased as well.
The accuracies and time taken for the various architectures are as follows:

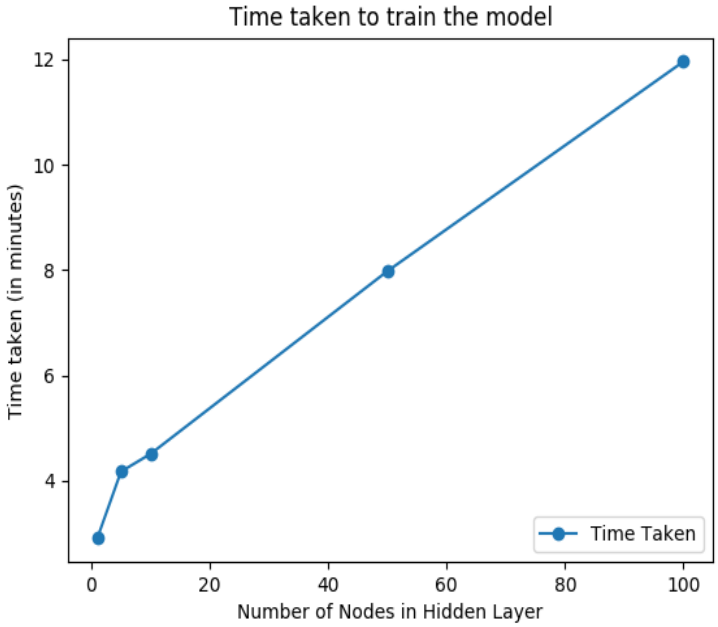
Size	**Train Set Accuracy**	**Test Set Accuracy**	**Time Taken**
1	9.31	9.29	174.71 secs
5	28.04	27.35	250.62 secs
10	78.0	71.75	270.76 secs
50	95.96	89.13	478.82 secs
100	96.77	90.84	717.42 secs

The corresponding plots are as follows:

1. Plot showing Number of Nodes vs Accuracy on Training and Testing Data.



2. Plot showing Number of Nodes vs Time taken.



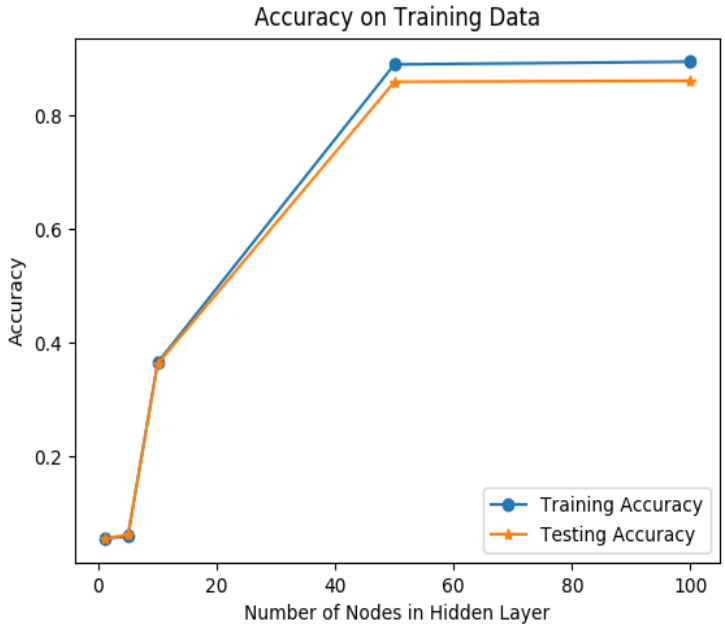
Part (c)

In this part also, the stopping criteria used is the same as used above but the learning rate is decreased according to the given criteria and the accuracies obtained after training the model are as follows:

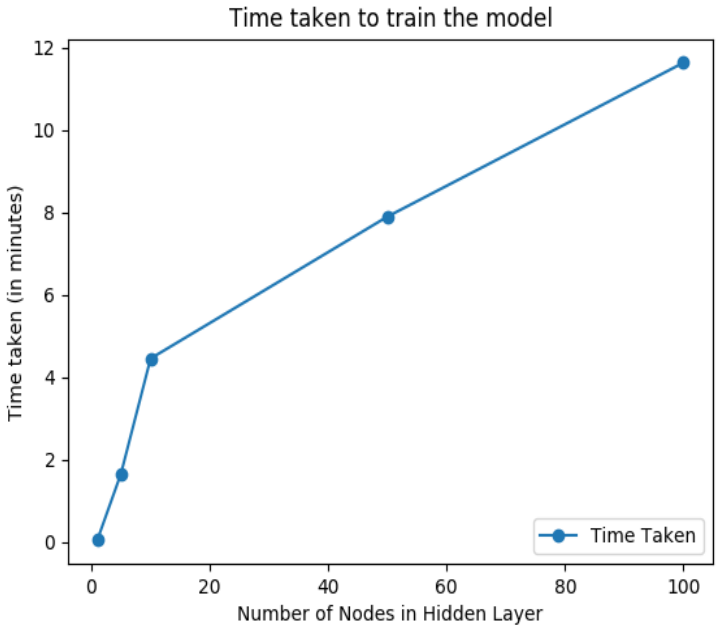
Size	Train Set Accuracy	Test Set Accuracy	Time Taken
1	5.61	5.50	3.18 secs
5	5.98	6.32	99.89 secs
10	36.53	36.32	267.67 secs
50	89.038	85.96	473.62 secs
100	89.51	86.16	697.48 secs

The corresponding plots are as follows:

1. Plot showing Number of Nodes vs Accuracy on Training and Testing Data.



2. Plot showing Number of Nodes vs Time taken.



Thus as it is clear from the above values and plots the accuracies on the testing and training sets have reduced a little after using the adaptive learning rate. It might be because the learning rate is still decreasing too fast.
But using the adaptive learning rate increased the convergence rate of the algorithm and now the convergence was much faster as compared to when non adaptive learning rate was used which is clear from the values of time taken in the above table. And the effect of increasing the number of nodes in the hidden layer remained the same which is it lead to an increase in accuracy and the time taken to train also increased.

Part (d)

The training and test set accuracies obtained after using 2 hidden layers with 100 neurons in each layer with adaptive learning rate starting from 0.5 are as follows:

Activation unit	**Train Set Accuracy**	**Test Set Accuracy**
Sigmoid	83.86	81.73
Relu	97.40	91.73

As it is clear from the accuracies obtained the relu activation unit provides much better accuracy than the sigmoid activation unit.It might be because using relu function has the advantage of removing vanishing gradient problem.
The accuracy obtained by using the relu activation unit is the maximum of

what we obtained in the previous parts on the testing as well as the training data while the accuracy obtained by the sigmoid activation unit when using two hidden layers is not as good as obtained in the previous parts using only a single hidden layer. This behaviour can still be understood as vanishing gradient problem will be higher when using two layers as compared to when we were using a single layer.

Part (e)

To implement the same architecture as used in the previous part, since MLP uses softmax function by default in the output layer and we needed the sigmoid function in the output layer we needed to pass one hot encoded Y while training the data so that loss is calculated as binary cross entropy over each class and then added to calculate the total loss.
Also, MLP classifier by default considers it as multilabel classification problem so we needed to predict the class with maximum probability.
After training the MLP classifier with 0.5 starting learning rate with parameter invscaling which we were doing previously we get the following results:

Activation unit	**Train Set Accuracy**	**Test Set Accuracy**
Sigmoid	25.67	26.21
Relu	86.73	84.04

There was another parameter, adaptive, for decaying the learning rate and after using that we get the following results:

Activation unit	**Train Set Accuracy**	**Test Set Accuracy**
Sigmoid	100	92.3
Relu	100	91.4

When using the normal softmax function for output we get the following results:

Activation unit	**Train Set Accuracy**	**Test Set Accuracy**
Sigmoid	32.56	33.92
Relu	90.88	87.53

Training using existing library when learning rate starts from 0.5 and decay is invscaling which we have used in the previous part results in accuracy being decreased. The model we implemented above performs better in out case. But, here also relu activation unit performs better when compared with sigmoid activation unit.