Home Assignment 1 report

Dataset used :

UCI repository : https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State

Back-Propagation Algorithm

Backpropagation algorithm is an enhanced version of the supervised multi-layered perceptron. This program consists of a four-layer network with backpropagation logic, where the algorithm is used to effectively train a neural network through a method called chain rule. In simple terms, after each forward pass through a network, backpropagation performs a backward pass while adjusting the model's parameters (weights and biases). In other words, backpropagation aims to minimize the cost function by adjusting network's weights and biases. The level of adjustment is determined by the gradients of the cost function with respect to those parameters.

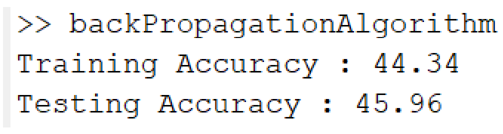
Implementation:

Initial values of weights and biases have been initialized randomly. Whereas, previous weights of all 4 layers in first epoch is considered as a zero matrix. **Learning rate** has been varied across the epochs in between **0.1 to 1E-5**. **Number of epochs are 50**. In each loop, before passing the data to a network, I have shuffled it and then passed it through 4 layers.

For the **forward computation**, middle layer's outputs have been passed to next layer as an input. Each layer's output has been calculated with simple formula i.e. activationFunction(weight * input to the layer). Hyperbolic function has been used as an activation function. Once we get the output in the final layer. We calculate the error in the network by comparing with the target value.

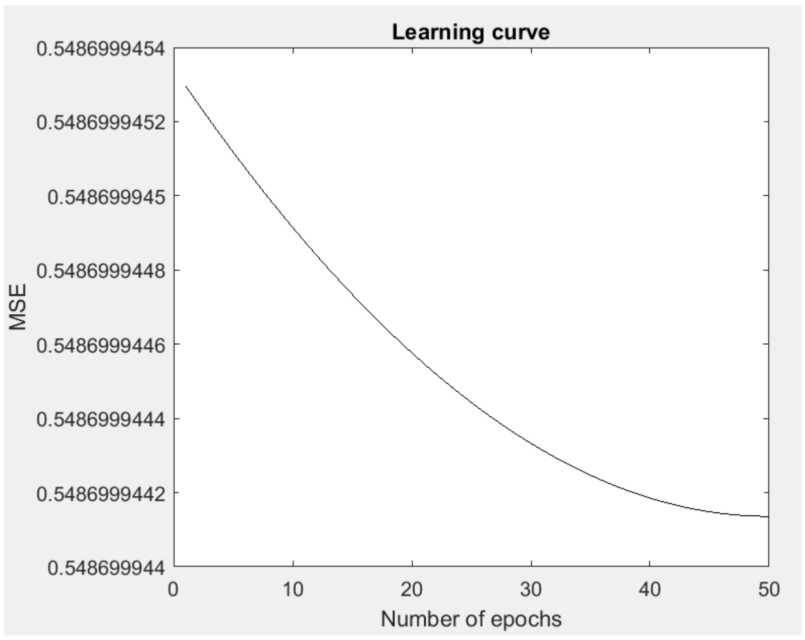
This error will be used to calculate the loss function in the **backward computation**. We calculated the delta loss in each layer, and which resulted in the weight adjustments with the delta values of it. These new weights will be used in the next epoch for training the next instance. And likely at the end, we will get the backpropagation model which will be used on testing dataset to calculate the accuracy and performance of it.

The results achieved in backpropagation:



```
>> backPropagationAlgorithm
Training Accuracy : 44.34
Testing Accuracy : 45.96
```

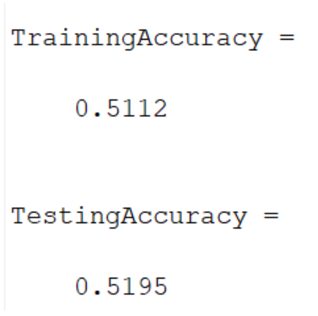
Learning curve built by backpropagation:



ELM algorithm:

Given a single layer feed forward network (SLFN) with a type of nonconstant piecewise continuous hidden nodes, if any continuous target function f(x) can be approximated by such SLFNs with appropriate hidden node parameters, then there is no need to find an algorithm to tune the hidden node parameters. Hidden node parameters are independent of target function f(x) but also of training samples.

Results of ELM:



```
TrainingAccuracy =

    0.5112


TestingAccuracy =

    0.5195
```

Analysis and discussion:

From these results, you can see that, the accuracies achieved by **ELM is higher than (5%) the backpropagation algorithm**. For this specific EEG eye state dataset, ELM has performed better than the backpropagation algorithm with 4-layer network. We **have not normalized the data** at any point because it was not required for this dataset. We have **changed the learning rate** over the epochs **from 0.1 to 1E-5** and for backpropagation we have used **hyperbolic activation function** whereas, for ELM we have used **sine activation function** and if we specifically use **sigmoid function**, we might achieve **higher accuracy** of around 55.8%. When I used sine function for forward computation and its derivative for backward computation in backpropagation algorithm, all the values were approximated to not-a-number (NAN). Both algorithms have ran over **50 epochs**. **Number of hidden neurons in ELM is 20** whereas, number of neurons in all the **4 layers of backpropagation have been adjusted to 30**. **Learning momentum used is 0.1**. Even if we vary learning momentum, the accuracy is not varying much.

Steps to run backpropation algorithm:

1. Open backpropagation algorithm file named 'backPropagationAlgorithm.m' in MATLAB. And just run it without any parameters. To load the dataset, dataset file name has been already included in matlab file.

Steps to run ELM algorithm:

1. Open ELM.m file and run it by using following commands:
   - Sigmoid activation function command -> ELM("EEGTrainingDataset.csv", 'EEGTestingDataset.csv', 1, 20, 'sig')
   - Sine activation function command -> ELM("EEGTrainingDataset.csv", 'EEGTestingDataset.csv', 1, 20, 'sin')

   Whereas, first parameter is training dataset, second testing dataset, third is for classification dataset, fourth is number of hidden neurons, and fifth is activation function. This code has been picked up from d2l.