Assignment 1 Report

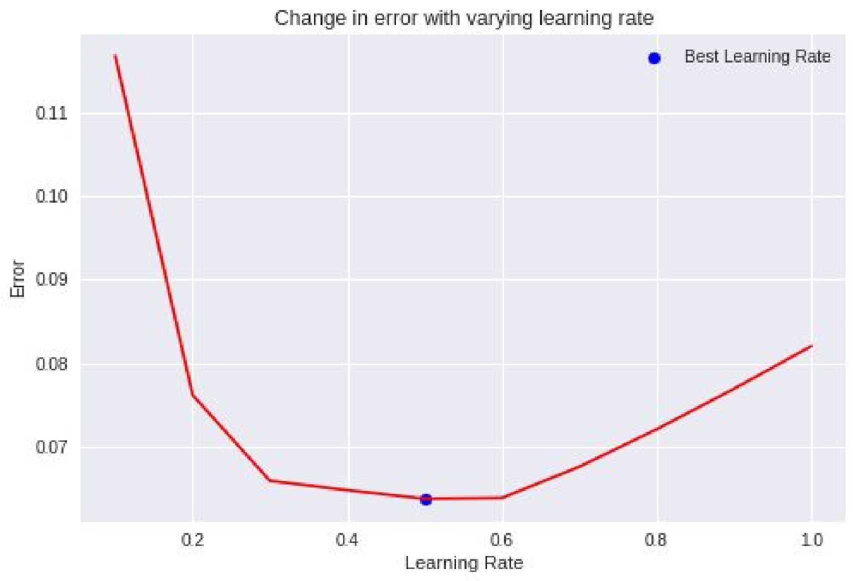
Q1. Implement two perceptron model to perform classification on the iris database from scratch. You should not be using any inbuilt function for this implementation (except reading the data).
(a) vary the learning rate and show the best learning rate value when you run it for 50 epochs.
(b) vary the number of epochs from 10 to 100 in a step of 10 and show the loss value curve (using the best learning rate obtained from (a))

Iris Data Set is used for training and testing. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

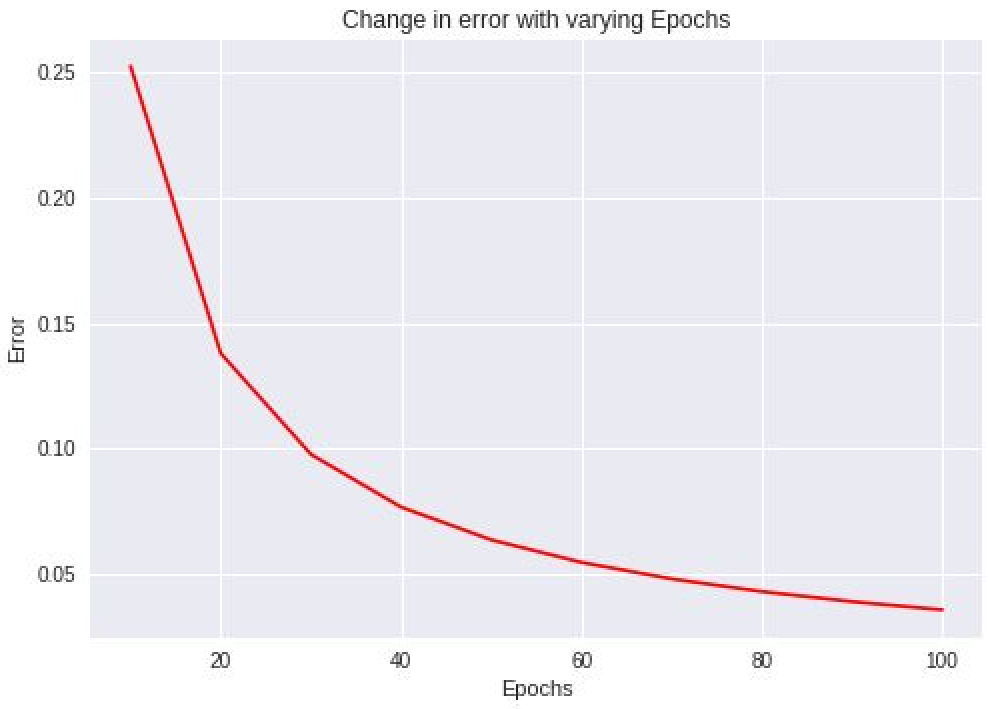
Approach Used:
We have to classify 3 class data using two perceptron model. We use one-vs-rest approach. In this approach, the first perceptron is used to classify Setosa from rest of all and second perceptron is used to classify Virginia from rest of all.

Model information:
Binary cross-entropy loss also called as log loss is used for calculating loss. For Optimization gradient descent approach is used. Total loss is calculated as the average of loss of two perceptrons.

Observation:
Following curve is obtained by fixing epoch to 50 and varying learning rate. Obtained best learning rate is marked in the below curve.



Following curve is obtained by fixing the best learning rate (for which loss is minimum) obtained in the previous step and varying epochs from 10 to 100 vs error.



As the number of epochs increases, the error rate also reduces.

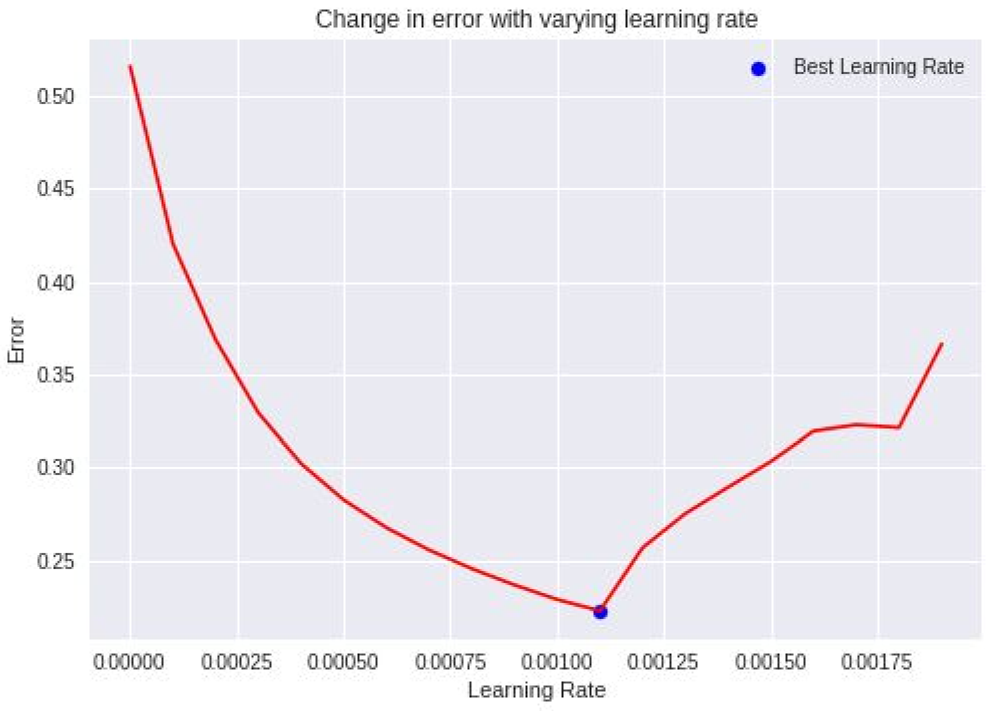
Q2. Implement a 3-class backpropagation NNet on your own to classify iris data, i.e. from scratch. You should not be using any inbuilt function for this implementation (except reading the data).
(a) vary the learning rate and show the best learning rate value when you run it for 50 epochs.
(b) vary the number of epochs from 10 to 100 in a step of 10 and show the loss value curve (using the best learning rate obtained from (a))

Iris Data Set is used for training and testing. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

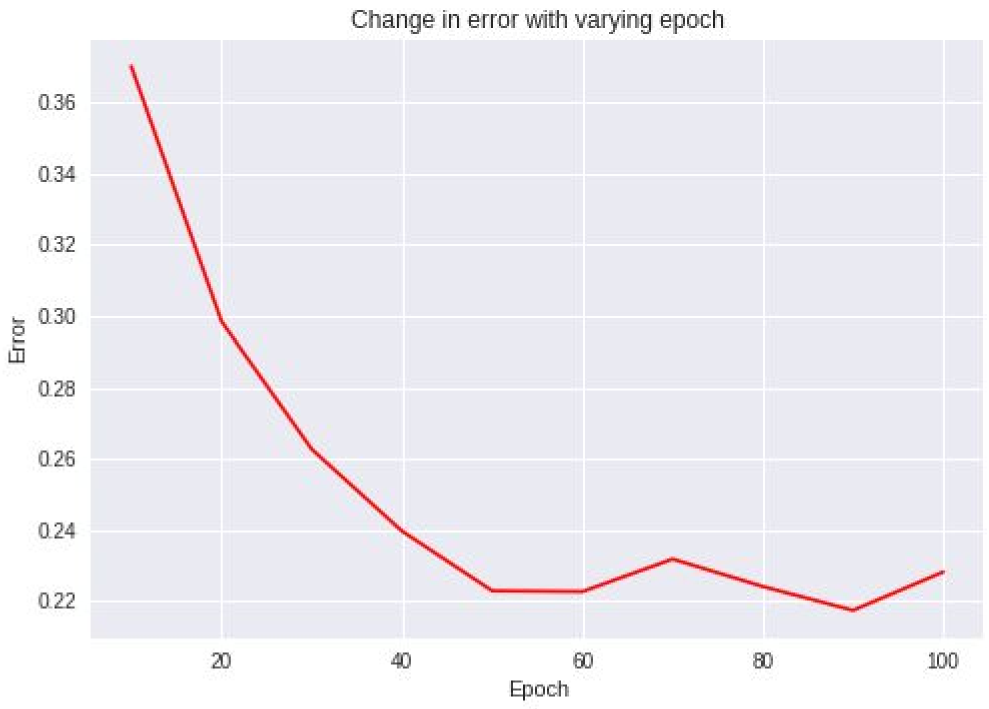
Approach Used:
Iris data is fed to the neural network which in turns classify these data to three class.

Model information:
Model is composed of input layer, output layer, and two hidden layers.
Binary cross-entropy loss also called as log loss is used for calculating loss. For Optimization gradient descent approach is used. Sigmoid is used as the activation function. Each hidden layer is composed of 6 neurons, output layer contains 3 neurons, and the input layer contains 5 neurons.

Observation:
Following curve is obtained by fixing epoch to 50 and varying learning rate. Obtained best learning rate is marked in the below curve.



Following curve is obtained by fixing the best learning rate (for which loss is minimum)



obtained in the previous step and varying epochs from 10 to 100 vs error.

As the number of epochs increases, the error rate approximately reduces.

Q3. Use any toolbox in python and implement RBF NNet to solve one of the problems/databases (of your choice from the UCI ML database Repo). Analyze your results with respect to the varying learning rate and epochs.

The MNIST dataset is used for implementing RDF NNet. It is a dataset of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9

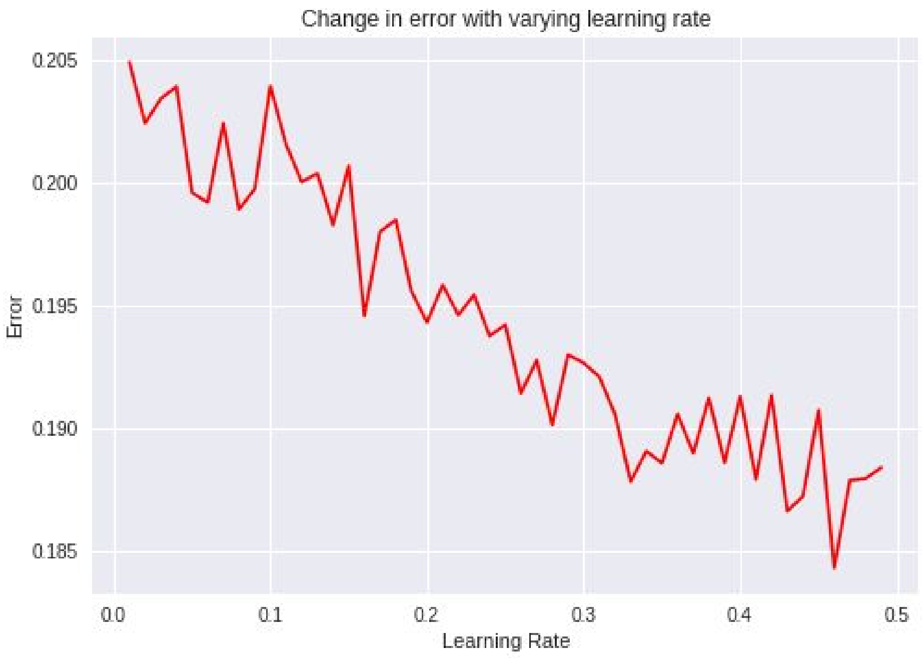
Model information:
A Radial Basis Function (RBF) neural network has an input layer, a hidden layer and an output layer. The neurons in the hidden layer contain Gaussian transfer functions whose outputs are inversely proportional to the distance from the centre of the neuron. The RBF layers are an alternative to the activation functions used in regular artificial neural networks. Implementation of RBF NNet is done using module PyTorch.
In an RBF layer, the distances between the input and a number of positions called centres are calculated and scaled. Then an RBF is applied to each scaled distance. I.e,

$$\begin{bmatrix} - & \boldsymbol{x}_1 & - \\ & \vdots & \\ - & \boldsymbol{x}_n & - \end{bmatrix} \rightarrow \begin{bmatrix} \phi(\sigma_1 \left\| \boldsymbol{x}_1 - \boldsymbol{c}_1 \right\|_2) & \cdots & \phi(\sigma_m \left\| \boldsymbol{x}_1 - \boldsymbol{c}_m \right\|_2) \\ \vdots & & \vdots \\ \phi(\sigma_1 \left\| \boldsymbol{x}_n - \boldsymbol{c}_1 \right\|_2) & \cdots & \phi(\sigma_m \left\| \boldsymbol{x}_n - \boldsymbol{c}_m \right\|_2) \end{bmatrix}$$

where the x's are the inputs, phi is a radial basis function, the sigmas are the scaling factors and the c's are the centres. The centre positions are found by clustering the data. Adam optimizer is used for optimization. Cross-Entropy loss or log loss is used as loss function. Batch size is kept at 100.

Observation:

Accuracy of the network is **89.5200** on MNIST Data set.
Following curve is obtained by fixing epoch to 50 and varying learning rate.



Q4. Using the MNIST database, code Autoencoder model with three encoding and three decoding layers. Show the visualization of the feature maps. On the features, add a classifier to perform 10-class classification and show the training loss curve and test accuracy.

The MNIST dataset is used for implementing RDF NNet. It is a dataset of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.

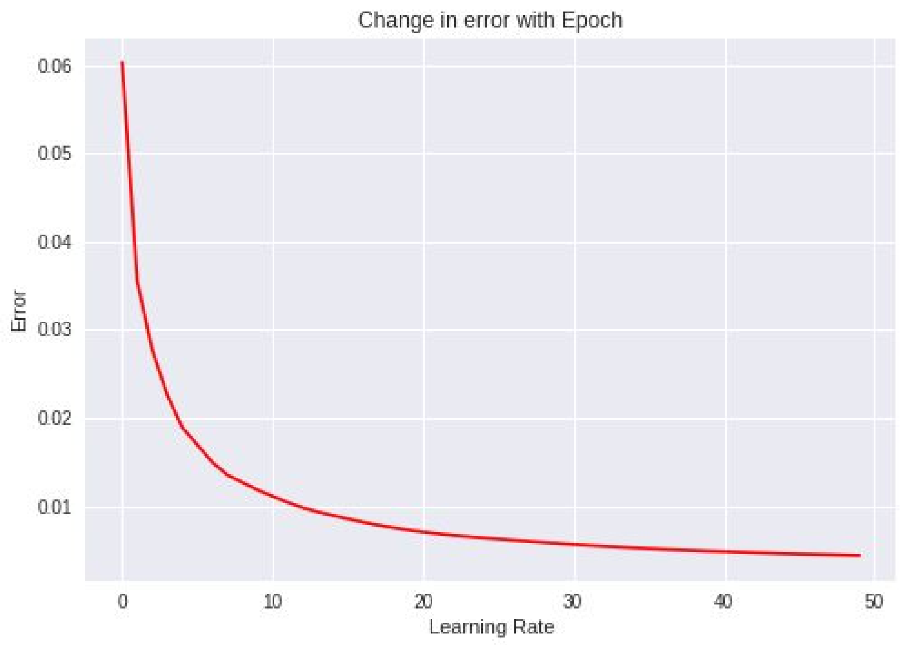
Model information:
An autoencoder is a type of neural network that finds the function mapping the *features x* to itself. This objective is known as *reconstruction*, and an autoencoder accomplishes this through the following process:

1. An encoder learns the data representation in lower-dimension space, i.e. extracting the most salient features of the data, and
2. A decoder learns to reconstruct the original data based on the learned representation by the encoder.

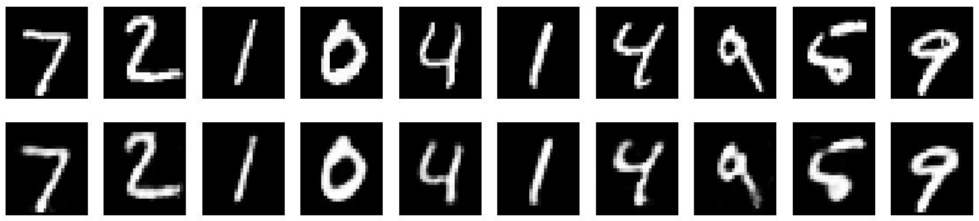
Here, Adams is used as an optimizer, Mean squared loss is used as loss function which is also called as reconstruction loss. Relu activation function is used in the hidden layer and sigmoid is used in the final layer.

Observation:

Following curve is obtained for loss vs epoch



Following is the original sample v/s reconstructed sample



First-line contains original image second line contain their corresponding reconstructed image. Reconstructed image outline is smooth as compared to the original image.