Project-3

CIS 666 Artificial Intelligence

Multi-Layer Perceptron Implementation in python

Fathima Syeda

Student ID: 2790024

Abstract

This project implements a Multi- Layer Perceptron in recognising Handwritten Digit Images into Digits from (0-9). The Handwritten digit images are taken from the MNIST dataset . Two sets of image datasets are taken, the smaller one is reserved for testing and is used only after the model is built using the other dataset. The multi-layer perceptron model is built by taking 10000 images for training the model and 3000 images from the testing dataset. Then the Error is calculated by comparing the output with the corresponding labels data. We make use of the Logistic sigmoid for activation function in this project to improve the learning of the model . The model is built multiple times with 3 different values of learning rates with different value so hidden nodes=10,35,100. The percentage error in recognising each digit is found for all three models constructed with three 3 different values of learning rates and it is plotted on bar charts.

Contents

1. Introduction:

A multi-layer perceptron consists of an input layer, one or more hidden layers and a final output layer. The layers close to the input layer are usually called the lower layers, and the ones close to the outputs are usually called the upper layers. Every layer except the output layer includes a bias neuron and is fully connected to the next layer. A neural network with a deep stack of hidden layers is called a deep neural network.

Since the multi-layer perceptron is more complex than a simple-layer perceptron network, the step function is replaced with the logistic sigmoid function. MLPs are used in classifications tasks and especially in binary classification problem takes a single neuron and with the logistic activation function gives out an output between 0,1 which you can interpret as the estimated probability of the positive class.

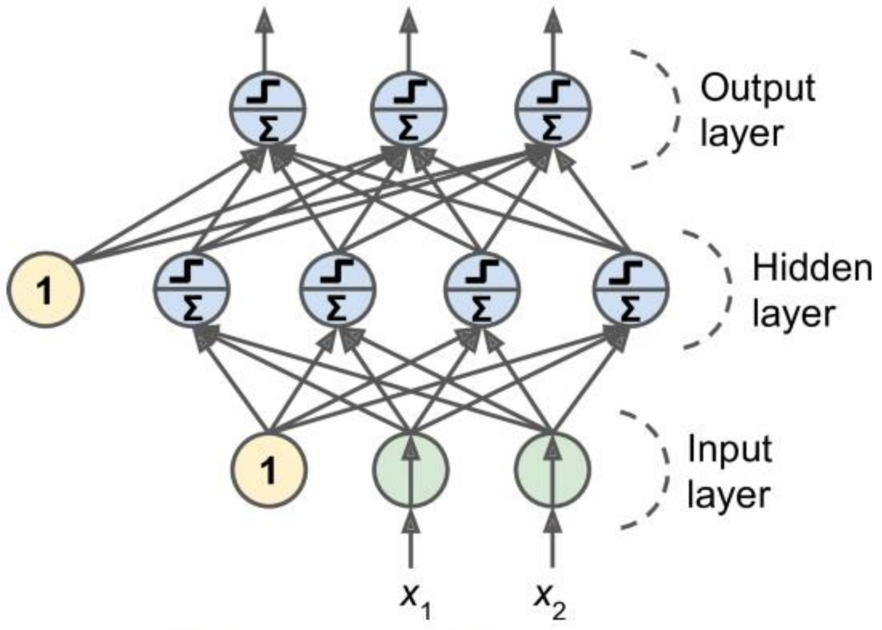


Figure 10-7. Architecture of a Multilayer Perceptron with two inputs, one hidden layer of four neurons, and three output neurons (the bias neurons are shown here, but usually they are implicit)

Géron, Aurélien. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (p. 289). O'Reilly Media. Kindle Edition.

2. Steps involved in Multi-Layer Perceptron implementation

2.1 Training Steps

2.1.1 Forward Path

The following steps are required in the implementation of a multi-layer perceptron .

1. Normalize the datasets. The input data is a 28x28 pixel image that is it contains 784 pixels. Each point is divided by 255 to normalize the entire image. This process is done for all 10000 images.
2. Initialize the weights for both the hidden layer $\mathbf{W_{ji}}$ and the input layer $\mathbf{U_{ki}}$ as values between [0, 1]. at first the weights are initialized as any random values between 0,1 , The weight for the input layer is are represented as $\mathbf{U_{ki}}$ and that for the hidden layer as $\mathbf{W_{ji}}$, where k is the number of hidden nodes in the hidden layer, j is the output nodes and i represents the input nodes.
3. Apply the input x^s from the training dataset. The input data set has 784 nodes and s patterns where s is the number of input images.
4. Compute the $\mathbf{Net_k = \sum_{i=1}^{N} U_{ki} * X_i + U_k(0)}$ This is the summation of the product of the weights for the input layer and the input nodes.
5. Compute the $Z_k = f(\mathrm{Netk})$. This is the sigmoid function of calculated as $\mathbf{Z_k = f(Netk) = \frac{1}{1+e^{-Netk}}}$
6. Compute the $\mathbf{V_j = \sum_{k=1}^{L} W_{jk} * Z_k + W_j(0)}$ This is the summation of the product of the weights $\mathbf{W_{jk}}$ for the hidden layer and the input nodes.
7. Find the system output Y_j using the Sigmoid function. The sigmoid function is $Y_j = f(Vj) = \frac{1}{1+e^{-Vj}}$

2.1.2 Back Path (Backpropagation for error correction)

8. Calculate δ j=(D_j- Y_j) and δ k=$\sum_{j=1}^{M}$ $W_{jk} * \delta$ j. These values will be used in the calculation of errors for the hidden and the output layer.
9. Calculate the error $e_j = (D_j - Y_j) * Y_j * (1_j - Y_j)$, , where D_j is the desired output or the classification label and Y_j, is the output from the training model and j=1,2,..M
10. Calculate the error $e_k = (\sum_{j=1}^{M} W_{jk} * \delta j) * Z_k * (1_j - Z_k)$, , where δ k is gotten from step-8 , Z_k is the hidden layer nodes and W_{jk} are the weights for the hidden layer and j=1,2,..M.
11. Once we calculate e_j we find the total error as the means square error as $E = \frac{1}{M*P} \sum_{s=1}^{P} = \sum_{j=1}^{M} [D_j - Y_j]^2$. If the error found is greater than the Epsilon value then we must update the weights.
12. Inorder to update the weights w emust first fond the DeltaW= Beta*e_j* Z_k and DeltaU = Beta*e_k* X_i .The weights are updated as W_{jk}(t+1)=W(t) + Beta*DeltaW_{jk} and U_{ki} t+1)= U_{ki} (t) + Beta*Delta U_{ki}

13. Repeat steps 4 to 12 until the model has error that is almost close to zero or less than the Epsilon value.

2.2 Testing Steps:

1. Get the testing Data X_test and its corresponding labels Test_Labels from the testing dataset.
2. Using the weights obtained at the end of training , Test the data by doing steps 4-6 from the training data set.
3. Calculate the error in classifying and plot the corresponding chart for it.

3 Program Outline

The project consists of three main parts, viz reading the input values and their labels along with modelling the Single layer perceptron, finding the mean Square error and plotting it as a curve and finally plotting a bar chart for different values of learning curve .

1. Design a fully connected network structure of 784 input nodes and 10 output nodes.
   - This is done by reading the training and testing files from the idx3-ubyte files and then calling *getData(Data, Labels, number*) to get 10000 training and 3000 testing data.
   - The labels values are replaced with an array[1x10] containing zeros and one 1 in the position of the value which is done by this method- *createLabelArray(Labels)*
   - The input X values must be normalized by dividing by 255.

2. Plot a learning curve that illustrates the mean square error versus iterations. (One iteration: apply all the training inputs once to the network and compute the mean square error).
   - The data read and prepared in the above section is then sent for training to build a model, by calling the M*LP_Training(X ,D, Beta=0.00025,k=10)* function. This returns the Weights W and an array of MSE values calculated at the end of each iteration which are then plotted .

3. Plot the percentage error in testing your handwritten digit recognition system as a bar chart. (Mean error occurred while testing each digit with the test data).

   To complete this section of the project, the testing(X, D, W, Beta) and the draw_barCharts(x_train, D, x_test, TestD) functions are called extensively.

   - <u>Task #1:</u> Repeat this experiment for different learning rate parameters (at least 3 experiments. Start with a large value and gradually decrease to a small value).
     This task is done by giving the inputs read from section1 and the weights obtained from train the model in section 2 along with the testing data to the *draw_barCharts(x_train, D, x_test, TestD)* function which then calls the training and testing functions 3 times for different values of Beta=0.5,0.05,0.005. It then calculates the percentage error in identifying each digit and plots this on bar charts.
   - <u>Task #2:</u> Repeat Task #1 with different number of hidden nodes (k=35,100,300)

4 Program Implementation

The program consists of 6 functions to implement the single layer perceptron algorithm to correctly identify handwritten images into 10 digits from 0-9. The following are the functions:

- getData(Data, Labels, number)

- createLabelArray(Labels)
- MLP_Training(X ,D, Beta=0.00025)
- testing(X, D, W, Beta)
- draw_barCharts(x_train, D, x_test, TestD, sig=False)

4.1 getData(Data, Labels, number)

This function takes in the set of input values and their corresponding labels along with the number of elements to be read and returns a subset if the data and its corresponding labels according to the given number.

4.2 createLabelArray(Labels)

This function is called to convert the value sin the labels dataset from decimals to array of size 10 filled with zeros and 1 in the position of the value of the label. It takes in the set labels returns an array of zeros and 1 .

4.3 MLP_Training(X ,D, Beta=0.0005,k=10)

This function is called to train the single layer perceptron model. It takes in the set of training data X and the corresponding labels D along with optional arguments Beta(the learning rate) and sigmoid (represents which activation function to use.) At first the value of NetK is calculated from which Z_K is calculated which is then used to find Vj and then Yj .The multiplication is done using numpy library's **numpy.matmul()** function. Then the sigmoid function is used for activation after which the Mean Square Error is calculated and all the steps ,mentioned in the training steps are done. It returns the weights and the MSE in each iteration which is then plotted against a curve using **matplotlib.pyplot.plot()**

4.4 testing(X, D, W)

This function takes the testing data and the corresponding labels, the weights from the trained model and following the steps in the testing steps mentioned earlier classifies the data. It also calculates the error and returns it .

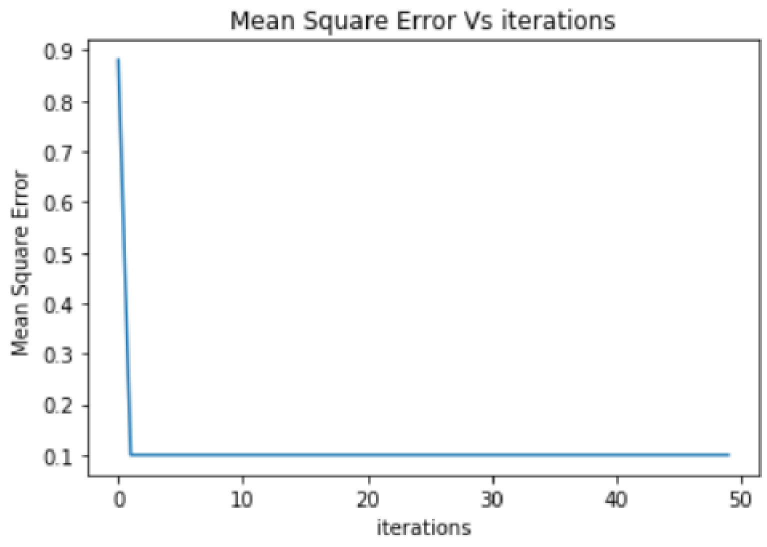
4.5 draw_barCharts(x_train, D, x_test, TestD,k)

This function is given the training data and the corresponding labels, the testing data and the corresponding labels and a flag telling if you should use the sigmoid activation function. It takes the training data and trains three different models with 3 different learning curves **Beta=0.005, 0.0005, 0.00025** by calling **the MLP_Training(X ,D, Beta=0.00025)** function. Then it takes the weights obtained from those training models and tests them by calling the **testing(X, D, W, Beta**) function. Then it calculates the percentage error in identifying each digit (0-9) and plots three bar charts for each learning curve using the **matplotlib.pyplot.bar()** function**.**

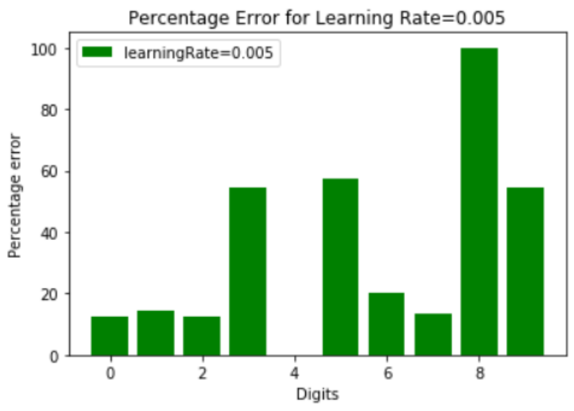
5 Obtained Results

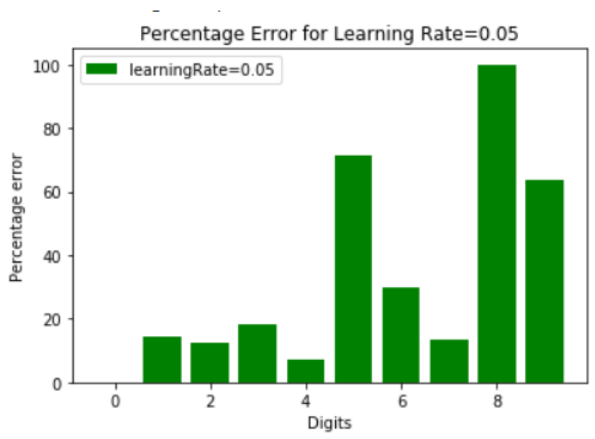
During the construction of the model by training it on a training set of 10000 images we calculate the Mean Square Error(MSE) at the end of each iteration. These MSE computed are plotted against the number of iterations at the end of the training process. The below curve was obtained with an Epsilon value=0.001 and the learning curve Beta=0.00025

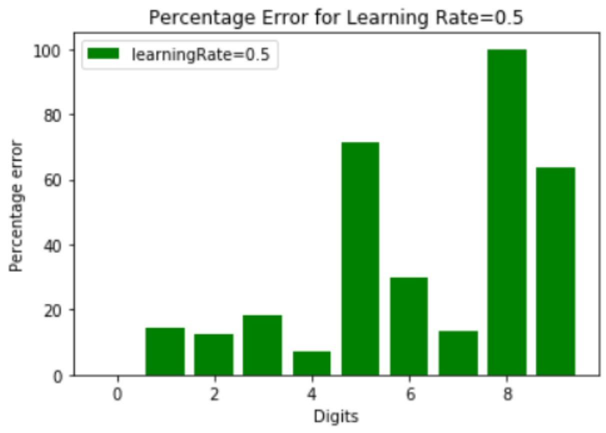
The curve was also plotted for various values of learning rate and it was observed that as the learning rate decreased the training process was slower and the result was more accurate. As the number of iteration increased the mean square error reduced along the curve.



Task 1:
Below are three bar charts that were plotted for various values of the learning rate Beta=0.005,0.0005,0.00025. The training data used was of 10000 handwritten images and the testing data=3000 images. Epsilon=0.01, is used to check the error. Each bar chart below shows the Percentage Error in recognising each handwritten digit.
For k=10 hidden nodes:





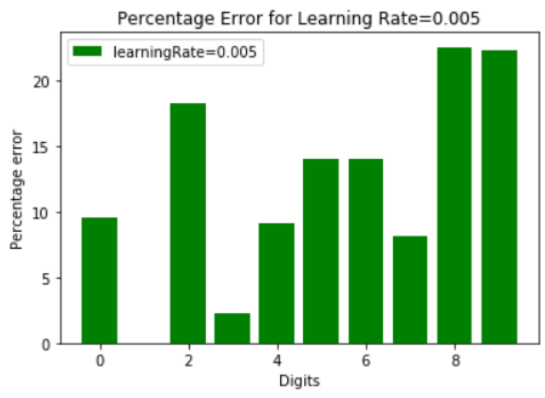


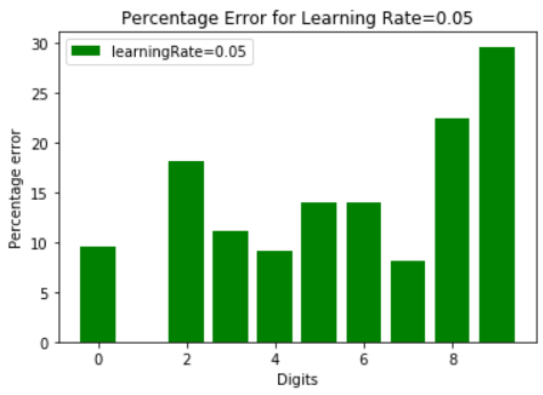
After observing the above charts we get that the digit with the maximum error in recognition is digit 8 with 100% error and the digits with minimum error are 0,4, with zero and 10% error in recognition.
We also notice that the error is high for Beta=0.5 with zero error for recognising digit 0 and less error for digit 4 and for Beta=0.05 it doesn't make much of a difference and then it decreases for Beta=0.005 but we also notice that the error for zero increased to 12% and the error in recognising digit 4 became zero.

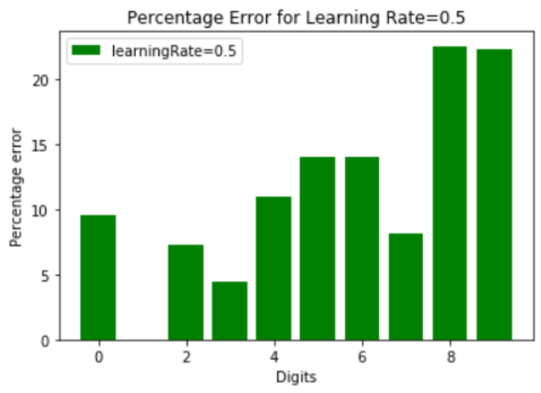
Max Error	Min Error	Beta value	Digit with Max	Digit with Min
25	0	0.5	8	0
22	0	0.05	8	0
25	2	0.005	8	4

Task 2:
Training data set-5000, testing datset-500,Epsilon=0.001. This task is the same as the previous task, however we check for different values of the hidden layers viz 10,35,100.

For hidden layers=35.

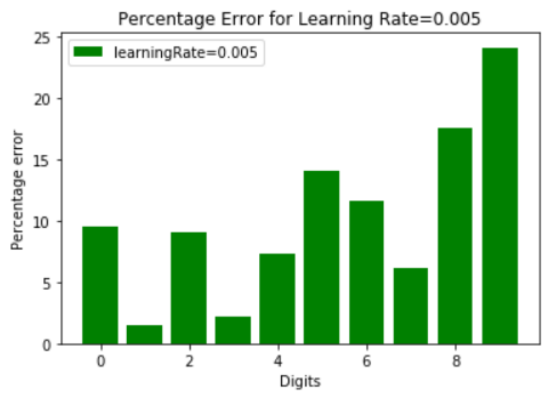


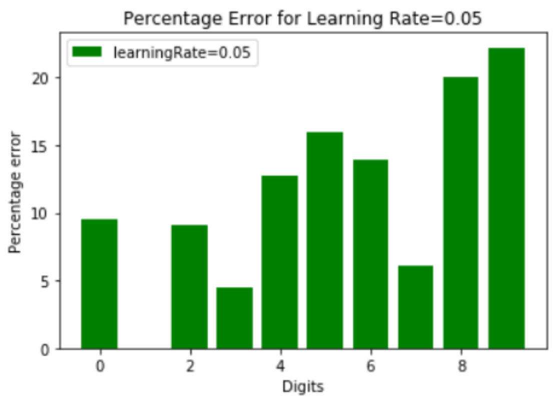


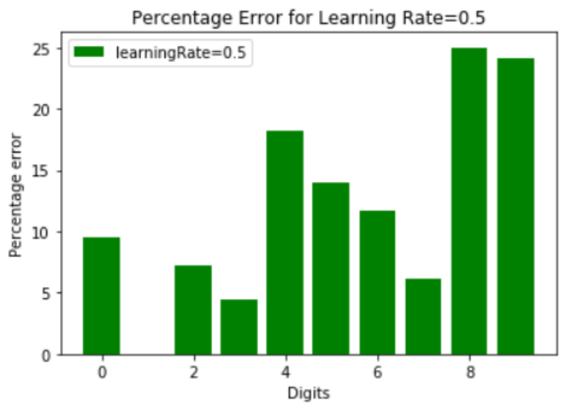


The digit with the maximum error in recognition is digit 8 followed by digit 9 with around 30% error the minimum error digits are 1, 3 with zero and 4% error in recognition.
We also notice that the error is high for Beta=0.5 then it increases for Beta=0.05and then decreases for Beta=0.005.

For k=100

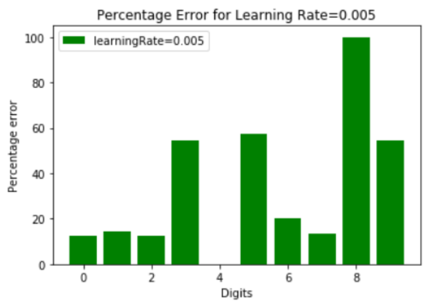


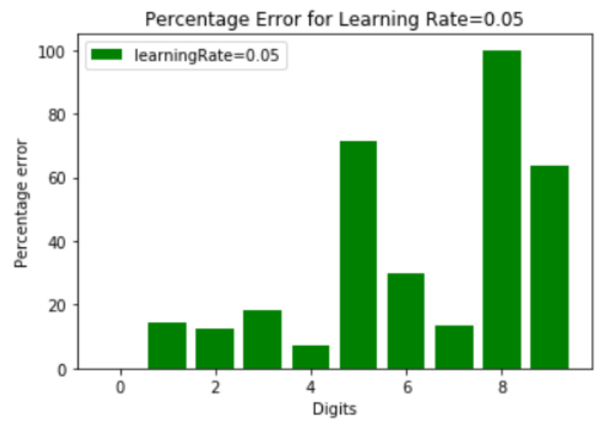


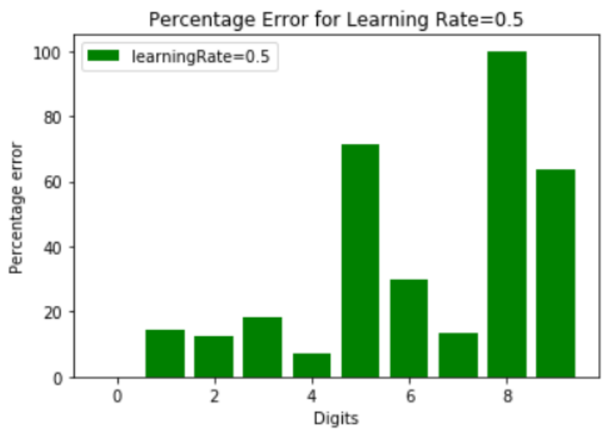


The digit with the maximum error in recognition is digit 9 with 25 % error the minimum error digit is 1 with zero error in recognition.
It is noticed as in task-2 that the error is high for Beta=0.5 then it reduces for Beta=0.05and then slightly increases again for Beta=0.005.

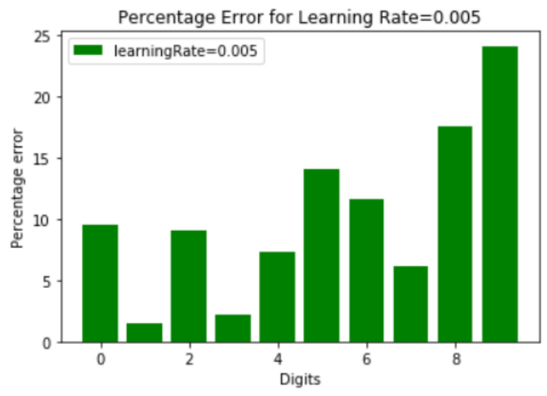
For k=100

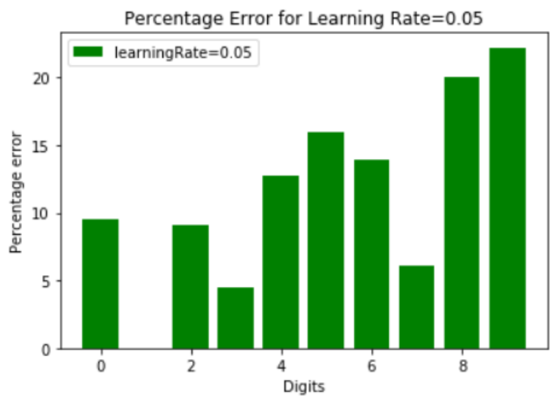


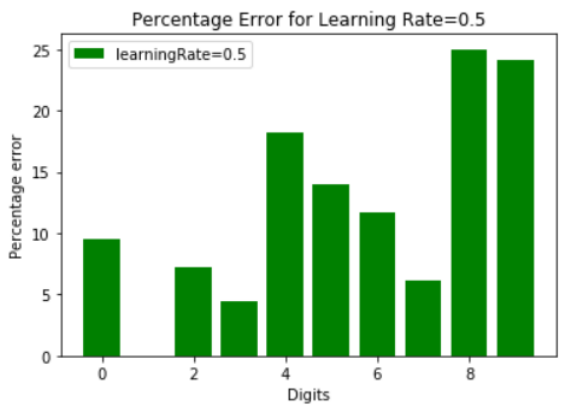

Percentage Error for Learning Rate=0.05
learningRate=0.05
Percentage error
Digits


Percentage Error for Learning Rate=0.5
learningRate=0.5
Percentage error
Digits

For k=300


Percentage Error for Learning Rate=0.005
learningRate=0.005
Percentage error
Digits


Percentage Error for Learning Rate=0.05
learningRate=0.05
Percentage error
Digits



6 Conclusion

- We notice that the error in correctly identifying the handwritten images was lower in the multi-layer perceptron than the single-layer perceptron.
- The lower the learning rate Beta, the higher the accuracy in recognising the digit.
- The higher the number of hidden nodes in the neural network the higher accuracy, however the training time significantly increases as the number of iterations go up.
- The lower the value of epsilon the higher the accuracy rate, but the training model does a lot of iterations, which can take a lot of time.
- The larger the training dataset the more accurate model is built.
- The handwritten digit with the least amount error in recognition were Digits 1,7,0 as they are very simple in shape.
- The handwritten digit with the greatest amount error in recognition was Digit 8 followed by Digit

8. References

- Géron, Aurélien. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (p. 289-294). O'Reilly Media. Kindle Edition.
- Class Notes