PROJECT REPORT

Introduction:

Handwritten digit recognition using MNIST dataset is a major project made with the help of Neural Network. It basically detects the scanned images of handwritten digits.

We have taken this a step further where our handwritten digit recognition system not only detects scanned images of handwritten digits but also allows writing digits on the screen with the help of an integrated GUI for recognition.

Approach:

We will approach this project by using a three-layered Neural Network.

The input layer: It distributes the features of our examples to the next layer for calculation of activations of the next layer.

The hidden layer: They are made of hidden units called activations providing nonlinear ties for the network. A number of hidden layers can vary according to our requirements.

The output layer: The nodes here are called output units. It provides us with the final prediction of the Neural Network on the basis of which final predictions can be made.

A neural network is a model inspired by how the brain works. It consists of multiple layers having many activations, this activation resembles neurons of our brain. A neural network tries to learn a set of parameters in a set of data which could help to recognize the underlying relationships. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria.

**Main.py**

Importing all the required libraries, extract the data from *mnist-original.mat* file. Then features and labels will be separated from extracted data. After that data will be split into training (60,000) and testing (10,000) examples. Randomly initialize Thetas in the range of [-0.15, +0.15] to break symmetry and get better results. Further, the optimizer is called for the training of weights, to minimize the cost function for appropriate predictions. We have used the “*minimize*” optimizer from “*scipy.optimize*” library with “*L-BFGS-B*” method. We have calculated the test, the “training set accuracy and precision using “predict” function.

### ****RandInitialise.py****

It randomly initializes theta between a range of [-epsilon, +epsilon].

**Model.py**

The function performs feed-forward and backpropagation.

* Forward propagation: Input data is fed in the forward direction through the network. Each hidden layer accepts the input data, processes it as per the activation function and passes it to the successive layer. We will use the sigmoid function as our “activation function”.
* Backward propagation: It is the practice of fine-tuning the weights of a neural net based on the error rate obtained in the previous iteration.

It also calculates cross-entropy costs for checking the errors between the prediction and original values. In the end, the gradient is calculated for the optimization objective.

### ****Prediction.py****

It performs forward propagation to predict the digit.

### ****GUI.py****

It launches a GUI for writing digits. The image of the digit is stored in the same directory after converting it to grayscale and reducing the size to (28 X 28) pixels.