ML ASSIGNMENT 3

-Shubham Kawatgi

-S20180010079

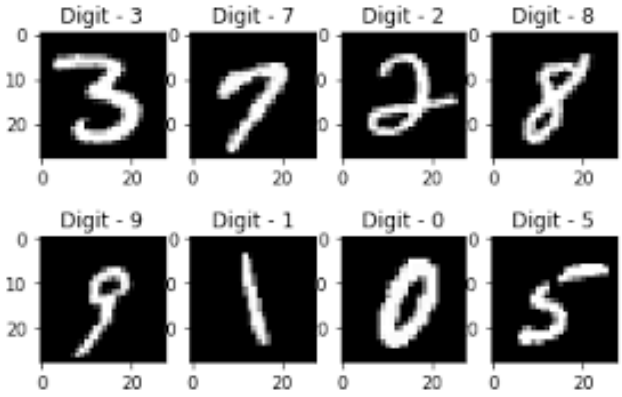
Introduction:

The aim of this project is to classify handwritten digits using softmax regression and feed forward neural network and check whether they are classified correctly according to the given labels.

Dataset:

Each training and test example (pattern) is of 192 dimensions given in a row. In the same row the last value (i.e., 193rdvalue) is the class label. Class labels are 0, 1, 2…. 9.

Ex

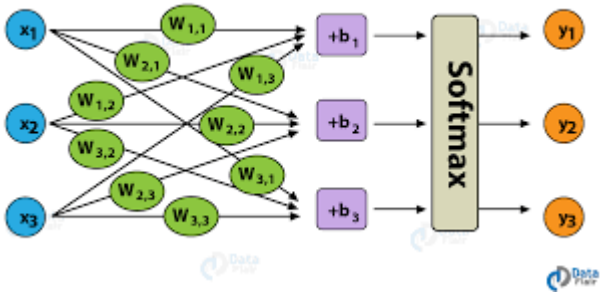


Softmax Regression:

It is used to classify data when multiple classes are present in the dataset.

The data is labelled with the class which has the highest probability.

Functions used:



i)To find the probability of a class:

$$h_\theta(x) = \begin{bmatrix} P(y=1|x;\theta) \\ P(y=2|x;\theta) \\ \vdots \\ P(y=K|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)} \begin{bmatrix} \exp(\theta^{(1)\top} x) \\ \exp(\theta^{(2)\top} x) \\ \vdots \\ \exp(\theta^{(K)\top} x) \end{bmatrix}$$

ii)Loss: Cross Entropy

$$H(p,q) = -\sum_{x \in \mathcal{X}} p(x) \log q(x) \quad \textbf{(Eq.1)}$$

iii)Cost function

$$J(\theta) = -\left[\sum_{i=1}^{m} \sum_{k=1}^{K} 1\left\{ y^{(i)} = k \right\} \log \frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x^{(i)})} \right]$$

Observations:

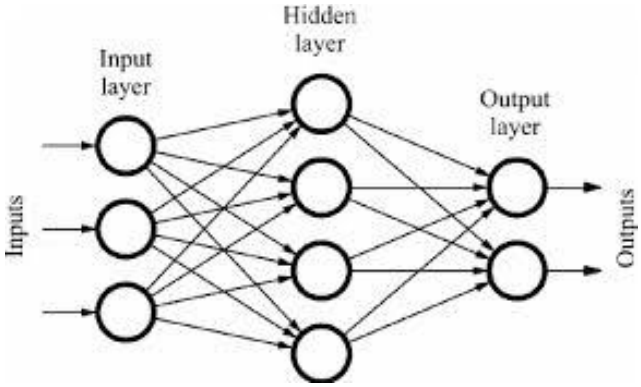
For 1000 epochs,

0.01 learning rate: Training accuracy: 87.70 %,Test accuracy: 86.46 %

0.1 learning rate: Training accuracy: 93.50 %,Test accuracy: 91.26 %

Feed Forward Neural Network:

A feed forward neural network is an artificial neural network wherein connections between the nodes do *not* form a cycle.[1] As such, it is different from its descendant: recurrent neural networks.



Functions used:

i) Backpropogation using chain rule

$$\frac{\partial J}{\partial w_{ji}} = \underbrace{\frac{\partial J}{\partial net_j}}_{-\delta_j} \underbrace{\frac{\partial net_j}{\partial w_{ji}}}_{x_i} = \left(\sum_{r=1}^{c} \underbrace{\frac{\partial J}{\partial \text{net}_r}}_{-\delta r} \underbrace{\frac{\partial \text{net}_r}{\partial y_j}}_{w_{rj}} \underbrace{\frac{\partial y_j}{\partial \text{net}_j}}_{f'(net_j)} \right) x_i$$

$$= \underbrace{\sum_{r=1}^{c} -\delta_r w_{rj} f'(net_j)}_{\frac{\partial J}{\partial net_j} = -\delta_j} x_i$$

$$\triangle w_{ji} = \eta x_i \delta_j$$

ii)Update weights

$$W_x = W_x - a\left(\frac{\partial Error}{\partial W_x}\right)$$

New weight; Old weight; Learning rate; Derivative of Error with respect to weight

Observations:

Stopping criteria: no of epochs

For 350 epochs and learning rate as 0.1,

Training score: 89.29%

Test score: 88.41%

Comparison of softmax with feed forward neural network:

FFNN is slightly better than the Softmax Regression because of its Computational Powers against a Gradient Descent Based Softmax Regression Due to the Back Propogation Algorithm.