March 3, 2016
Shravya Thandra [UB # 50169587]
Shiva Prasad Reddy Munagala [UB # 50168800]
GROUP - 9

CSE 474/574 - Introduction to Machine Learning

Handwritten Digits Classification Using Neural Networks

Description: -

The main aim of this project is build and train a Neutral Network, i.e., learn the weight parameters and evaluate its performance in classifying handwritten digits from 0 to 9.

The steps followed are

1. Preprocess the data.
2. Train the Neural Network
3. Predict and calculate the accuracy.

The Neural Network we have built has one hidden layer and has regularization to control the bias-variance tradeoff, i.e., we have add a regularization term (λ) into our error function to control the magnitude of parameters in Neural Network.

Preprocess: -

The given data is preprocessed into appropriate matrices i.e., the input values and the true labels are separated.

Then we do "**Feature Selection**" on the input data.

In feature selection: -

- We have check for the columns in the input matrix i.e., train_data in which all the row entries are the same.
- To check that we have first converted the matrix values from numerical to binary by using a numpy function "**ndarray.astype(bool)"**
- After this conversion we have used another numpy function **np.any(ndarray, axis=0)**. This function when executed as axis=0, columns which have at least one true (non-zero value) will be returned as true and we can select those columns eliminating the others i.e., in this process we are just eliminating the columns (features of images) which initially had all zero entries.

The **Feed Forward** and **Back Propagation** are implemented in the "**nnObjFunction**". This function returns two values –**obj_val** (the error function value) and **obj_grad** (the derivate of the error function) to the **scipy.minimize function** to calculate the Gradient Decent and the weight matrices are updated.

The scipy.minimize function iterates for maxiter – value no of times over the entire data and in each iteration to converge on the error i.e., it runs until the neural network trains itself and gives a minimal amount of error; the weight matrices are then updated.

As mention above the regularization factor is included while calculating the obj_val and obj_grad as mentioned below:

Calculation of obj_val:

$$\widetilde{J}(W^{(1)}, W^{(2)}) = J(W^{(1)}, W^{(2)}) + \frac{\lambda}{2n}\left(\sum_{j=1}^{m}\sum_{i=1}^{d+1}(w_{ji}^{(1)})^2 + \sum_{l=1}^{k}\sum_{j=1}^{m+1}(w_{lj}^{(2)})^2\right)$$

Where,

$$J(W^{(1)}, W^{(2)}) = \frac{1}{n}\sum_{p=1}^{n} J_p(W^{(1)}, W^{(2)}) = \frac{1}{n}\sum_{p=1}^{n}\frac{1}{2}\sum_{l=1}^{k}(y_{pl} - o_{pl})^2$$

Calculation of obj_grad:

This value of obj_grad is the derivate of J with respect to each entry in the weight matrix of the hidden layer and the weight matrix of the output layer.

For the output layer weight matrix W2:

$$\frac{\partial \widetilde{J}}{\partial w_{lj}^{(2)}} = \frac{1}{n}\left(\sum_{p=1}^{n}\frac{\partial J_p}{\partial w_{lj}^{(2)}} + \lambda w_{lj}^{(2)}\right)$$

Where,

$$\begin{aligned}\frac{\partial J_p}{\partial w_{lj}^{(2)}} &= \frac{\partial J_p}{\partial o_l}\frac{\partial o_l}{\partial b_l}\frac{\partial b_l}{\partial w_{lj}^{(2)}} \\ &= -\delta_l z_j\end{aligned}$$

$$\delta_l = \frac{\partial J_p}{\partial o_l}\frac{\partial o_l}{\partial b_l} = (y_l - o_l)(1 - o_l)o_l$$

For the hidden layer weight matrix W2:

$$\frac{\partial \widetilde{J}}{\partial w_{ji}^{(1)}} = \frac{1}{n}\left(\sum_{p=1}^{n}\frac{\partial J_p}{\partial w_{ji}^{(1)}} + \lambda w_{ji}^{(1)}\right)$$

Where,

$$\begin{aligned}\frac{\partial J_p}{\partial w_{ji}^{(1)}} &= \sum_{l=1}^{k}\frac{\partial J_p}{\partial o_l}\frac{\partial o_l}{\partial b_l}\frac{\partial b_l}{\partial z_j}\frac{\partial z_j}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}^{(1)}} \\ &= -\sum_{l=1}^{k}\delta_l w_{lj}^{(2)}(1 - z_j)z_j x_i \\ &= -(1 - z_j)z_j(\sum_{l=1}^{k}\delta_l w_{lj}^{(2)})x_i\end{aligned}$$

The **Regularization Factor (λ)** is used to avoid the over-fitting problem i.e., to have an optimal learnt Neural Network which does pretty good on both the train and test data.
The model that is learnt is said to be

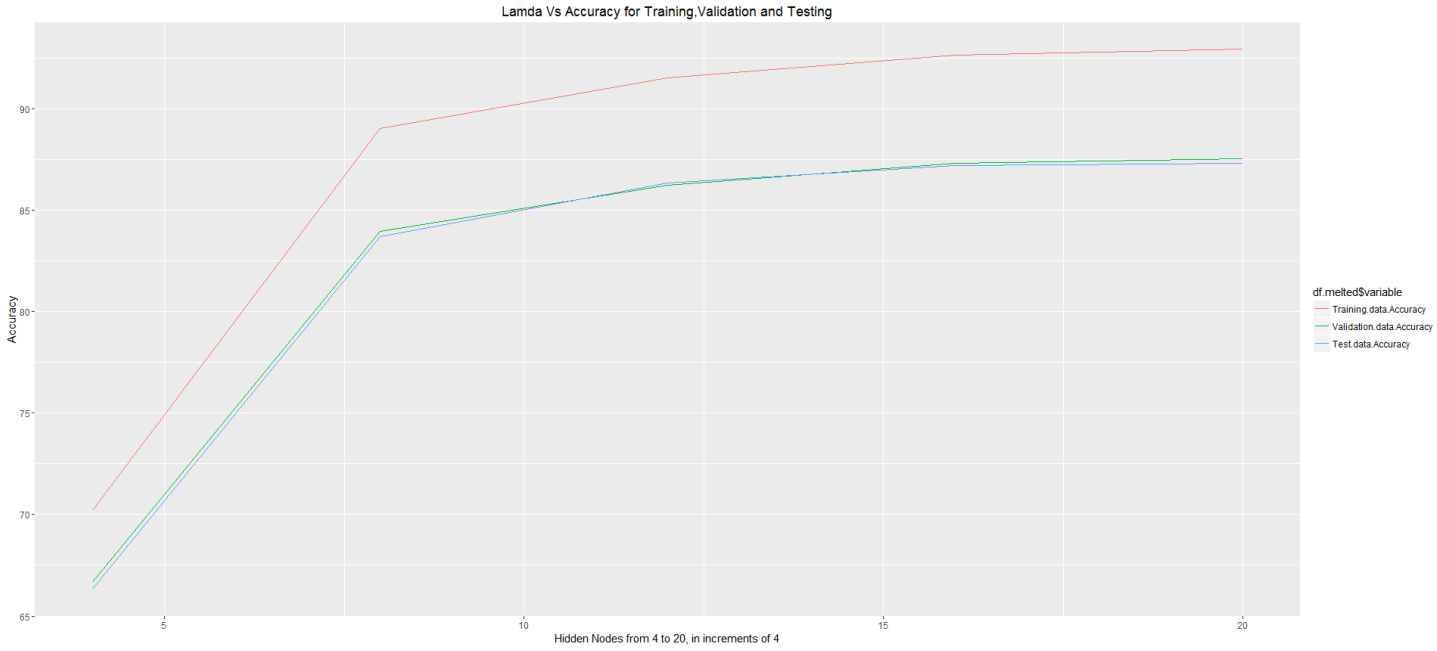
- Under fitting if it performs badly (achieving low prediction accuracy) on the training data set itself.
- Over fitting if it performs very well on the training data set but performs badly on the test data set.

The efficiency or accuracy of the trained neural network also depends on the number of layer present in its design and also depends on number of nodes present in each layer.

The neural network which we have designed has only one hidden layer. Initially we have trained the neural network with a regularization factor of 0.1 i.e., λ = 0.1 and we have tried for different numbers of hidden nodes (n_hidden ranging as 4, 8, 12, 16, 20).

N_hidden	Training data Accuracy	Validation data Accuracy	Test data Accuracy
m = 4	70.21	66.67	66.32
m = 8	89.034	83.96	83.68
m = 12	91.496	86.21	86.34
m = 16	92.638	87.32	87.18
m = 20	92.92	87.52	87.32

With this we clearly see that as the number of hidden nodes increases the accuracy over all the three data sets has increased. The more complex the neural network is built; the more accurate will it's prediction be.



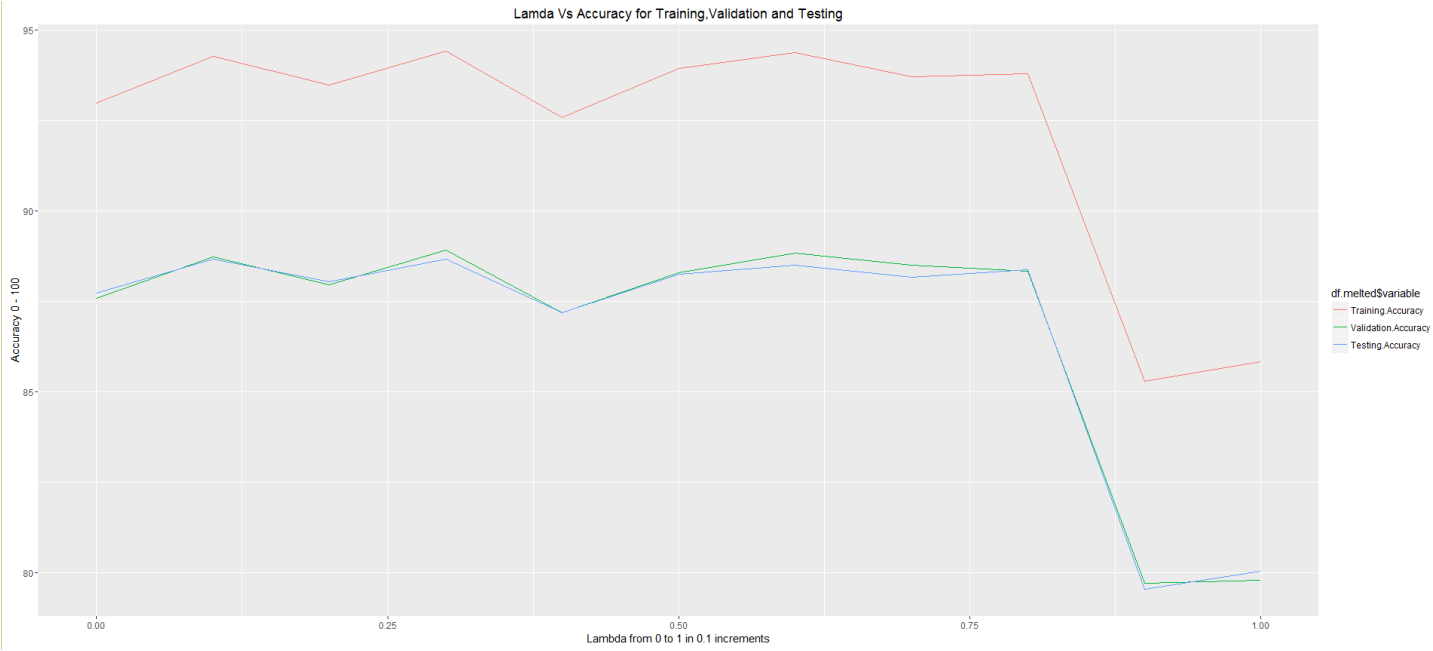
Next we have tried for different values of λ having the hidden nodes number as 50(n_hidden).

Below is the table showing values of accuracy on the three data sets for different values of λ.

λ	Training data Accuracy	Validation data Accuracy	Test data Accuracy
0	92.98	87.58	87.74
0.1	94.27	88.73	88.68
0.2	93.48	87.96	88.04
0.3	94.42	88.92	88.68
0.4	92.59	87.2	87.2
0.5	93.94	88.3	88.25
0.6	94.38	88.84	88.51
0.7	93.726	88.51	88.17
0.8	93.808	88.34	88.38
0.9	85.304	79.71	79.54
1	85.828	79.8	80.05

In this varied values of λ; as it increases it belief on the training data decreases gradually i.e., its accuracy of prediction on the training data set decreases and with fluctuating values of validation data and test data accuracies. We can clearly see the over fitting problem when λ is high and the under fitting problem when λ is low.

We have plotted this on a graph for clear and visual view: -



After seeing the above graphs and the varying values of accuracy for λ and n_hidden, we have come to a conclusion that the neural network behaves good on the three data sets for values of λ = 0.1 as and n_hidden as 50.

Final values: Training set Accuracy: 94.246%; Validation set Accuracy: 88.68%; Test set Accuracy: 88.77%