Assignment 1

Group: Sarthak Sheoran (2018A7PS0206H), Siddharth Jagota (2018A7PS0166H),
Ananay Gupta (2018A7PS0203)

Pre-processing:

The training data was first read from "mnist_train.csv" which contained the relevant data information and this was stored in the form of a data frame.
This dataset was separated into features and target attribute. The first column corresponds to target attribute and the rest 784 columns correspond to features.

Then normalization was applied using the following formula:
X=X/255
Target attribute was encoded as follows:
dummy_y = np_utils.to_categorical(y) where np_utils is a library from keras package.

Testing data was first read from "mnist_test.csv" and similar transformations were applied to it as well.

Architecture of 12 Different Neural network models:

Loss function used is *categorical cross entropy* as it is well suited for multi-class classification problems.
Batch gradient descent is used with batch size 100 and no. of epochs 10.

Model 1:

Hidden layers used: 1
Input layer of 784 nodes
One hidden layer with 100 nodes and activation function = softplus
Output layer with 10 nodes and activation function = softplus

Accuracy achieved:

Training Accuracy : 0.9843500256538391

Testing Accuracy : 0.9725000262260437

Model 2:

Hidden layers used: 2
Input layer of 784 nodes
First hidden layer with 100 nodes and activation function = softplus
Second hidden layer with 100 nodes and activation function = softplus
Output layer with 10 nodes and activation function = softplus

Accuracy achieved:

Training Accuracy : 0.9904333353042603

Testing Accuracy : 0.9771000146865845

Model 3:

Hidden layers used: 3
Input layer of 784 nodes
First hidden layer with 100 nodes and activation function = softplus
Second hidden layer with 100 nodes and activation function = softplus
Third hidden layer with 100 nodes and activation function = softplus

Output layer with 10 nodes and activation function = softplus

Accuracy achieved:
Training Accuracy : 0.9912166595458984
Testing Accuracy : 0.9778000116348267

Model 4:

Hidden layers used: 2
Input layer of 784 nodes
First hidden layer with 400 nodes and activation function = relu
Second hidden layer with 100 nodes and activation function = relu
Output layer with 10 nodes and activation function = relu

Accuracy achieved:
Training Accuracy : 0.09871666878461838
Testing Accuracy : 0.09799999743700027

Model 5:

Hidden layers used: 2
Input layer of 784 nodes
First hidden layer with 400 nodes and activation function = sigmoid
Second hidden layer with 100 nodes and activation function = sigmoid
Output layer with 10 nodes and activation function = sigmoid

Accuracy achieved:
Training Accuracy : 0.9952666759490967
Testing Accuracy : 0.9804000025844574

Model 6:

Hidden layers used: 2
Input layer of 784 nodes
First hidden layer with 400 nodes and activation function = softplus
Second hidden layer with 100 nodes and activation function = softplus
Output layer with 10 nodes and activation function = softplus

Accuracy achieved:

Training Accuracy : 0.9931499958033833

Testing Accuracy : 0.9768999814987183

Model 7:

Hidden layers used: 2
Input layer of 784 nodes
First hidden layer with 1 nodes and activation function = tanh
Second hidden layer with 1 nodes and activation function = relu
Output layer with 10 nodes and activation function = softplus

Accuracy achieved:

Training Accuracy : 0.26756668090820312

Testing Accuracy : 0.26620000600814824

Model 8:

Hidden layers used: 2
Input layer of 784 nodes
First hidden layer with 10 nodes and activation function = tanh
Second hidden layer with 10 nodes and activation function = relu
Output layer with 10 nodes and activation function = softplus

Accuracy achieved:

Training Accuracy : 0.9402999877929688

Testing Accuracy : 0.9337000250816345

Model 9:

Hidden layers used: 2
Input layer of 784 nodes
First hidden layer with 100 nodes and activation function = tanh
Second hidden layer with 100 nodes and activation function = relu
Output layer with 10 nodes and activation function = softplus

Accuracy achieved:

Training Accuracy : 0.9946833252906799

Testing Accuracy : 0.9758999943733215

Model 10:

Hidden layers used: 2
Input layer of 784 nodes
First hidden layer with 1 nodes and activation function = sigmoid
Second hidden layer with 1 nodes and activation function = relu
Output layer with 10 nodes and activation function = softmax

Accuracy achieved:

Training Accuracy : 0.11236666887998581

Testing Accuracy : 0.11349999904632568

Model 11:

Hidden layers used: 2
Input layer of 784 nodes
First hidden layer with 10 nodes and activation function = sigmoid
Second hidden layer with 10 nodes and activation function = relu
Output layer with 10 nodes and activation function = softmax

Accuracy achieved:

Training Accuracy : 0.9385833144187927

Testing Accuracy : 0.9294999837875366

Model 12:

Hidden layers used: 2
Input layer of 784 nodes
First hidden layer with 100 nodes and activation function = sigmoid
Second hidden layer with 100 nodes and activation function = relu
Output layer with 10 nodes and activation function = softmax

Accuracy achieved:

Training Accuracy : 0.9902166724205017

Testing Accuracy : 0.9767000079154968

Traditional machine learning classification techniques used:

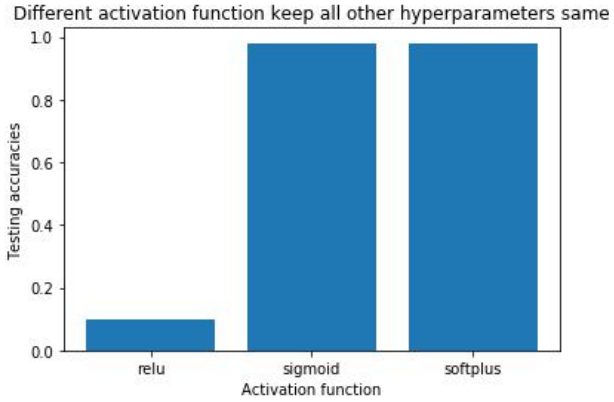
- Logistic Regression
- Naive Bayes Classification
- Decision Tree Classification
- Support Vector Classification

Accuracy achieved:

	Classification	Training Accuracy	Testing Accuracy
1	LogisticRegression	0.9279	0.9201
2	Naive Bayes	0.5649	0.5558
3	Decision Tree	1.0000	0.8864
4	SVC	0.9430	0.9446

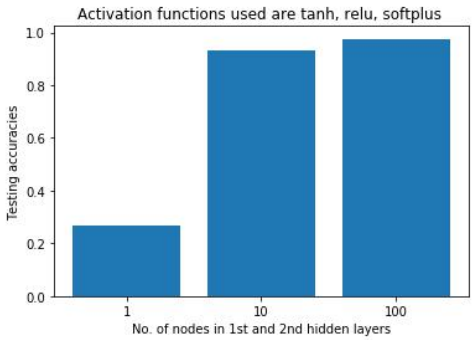
Comparisons

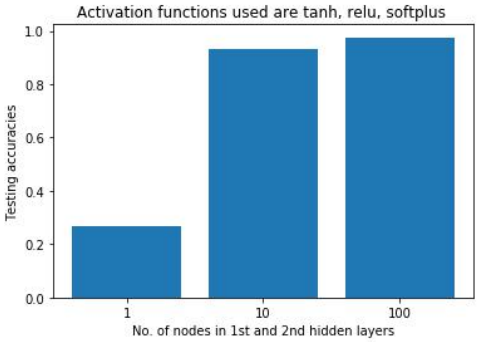
1. Effect of activation function keep all other hyper parameters same:



As we can see from the above graph, sigmoid and softplus turned out to give a much better testing accuracy with our given dataset upon implementing an architecture of 2 hidden layers with 100 neurons in each layer.

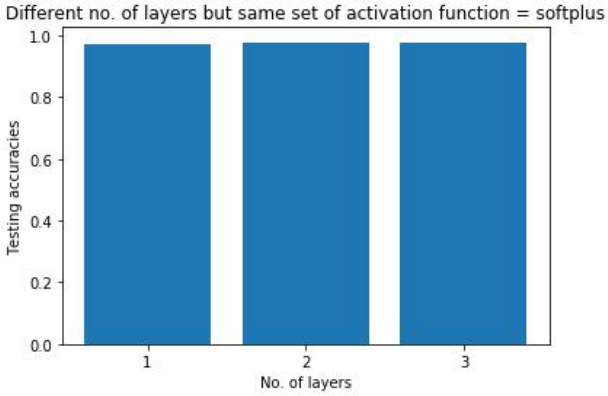
2. Effect of no. of nodes in different layers





It is evident from the above 2 graphs that upon increase in the no. neurons per hidden layer, accuracy increases. We experimented this on two different architectures as shown above with varying activation functions. Both gave similar inferences suggesting an evident trend upon increasing the no. of neurons per hidden layer.

3. Accuracy of different models having same set of activation function but different no. of layers:



We are not able to see any evident increasing or decreasing trend by increasing the no. of layers by one keeping the activation functions, no. of neurons per hidden la yer same on our data-set

.

Traditional Machine Learning models vs Deep Learning models:

- **Time consumption:-** While testing different basic deep learning models and traditional ML models, we observed that there is a significant difference in time taken. Deep Learning models (with small and simple architecture, like 3 hidden layers used and 400 neurons per hidden layer) when compared to traditional ML models like SVC have been observed to be less time consuming.

- **Experimentation Flexibility And Testing Accuracy:-** We observed that traditional models gave good accuracy (like SVC, logistic regression) while deep learning models allowed more flexibility in choosing hyper-parameters (activation function etc.) and implementing architecture (number of neurons per hidden layer and number of hidden layers) and upon implementing a simple architecture of 2 hidden layers with 100 number of neuron per layer we managed to get a superior model (in terms of testing accuracy) than any of the traditional models.

Conclusion

- Increasing the no. of nodes resulted in a superior model.
- We observed that change in activation functions resulted in inferior/moderate/superior models (in terms of testing accuracy). Eg.- Relu gave us an inferior model with the given dataset while softplus and sigmoid gave us superior models for the given dataset.
- Increase in the no. layers didn’t give us an evident trend to report.
- Deep learning models offer greater experimentation flexibility and better testing accuracy upon appropriate choice of hyper-parameters and architecture when compared with traditional ML models.
- Basic Deep learning models (with small and simple architecture) proved to be less time consuming as compared to traditional machine learning models like SVC for the given dataset.