MACHINE LEARNING - CS60050

Assignment - 3: Multi Layer Perceptron Classifier

Group Details:

Group : 17

Jatoth Charan Sai - 19CS10035

Rahul Eaga - 19CS10029

Data Set Description:

Letter recognition Data:

The data describes the character images of 26 capital letters in the English in 20 fonts

Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15.

Number of Instances: 20000

Number of Attributes: 17 (Letter category and 16 numeric features

Attribute Information:

S.No	Input Variables	Meaning	Value Range
1	x-box	Horizontal politician of box	0-15
2	y-box	Vertical position of box	0-15
3	width	Width of box	0-15
4	high	Height of the box	0-15
5	onpix	Number of on pixels	0-15
6	x-bar	Mean number of on pixels in x-direction	0-15
7	y-bar	Mean number of on pixels in y-direction	0-15
8	x2-bar	Mean x variance	0-15
9	y2-bar	Mean y variance	0-15
10	xy-bar	Mean x-y correlation	0-15
11	x2ybar	Mean of x*y*	0-15
12	xy2bar	Mean of x*y*y	0-15
13	x-ege	Mean edge count left to right	0-15
14	xegvy	Correlation of x-ege and y	0-15
15	y-ege	Mean edge count top to bottom	0-15
16	yegvx	Correlation of y-ege and x	0-15

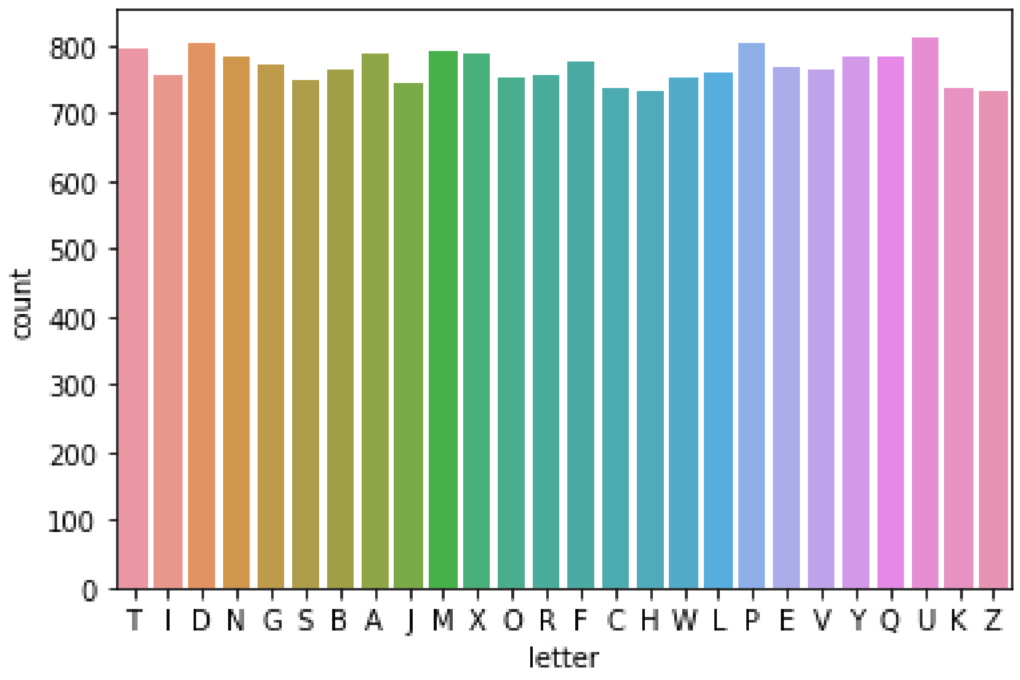
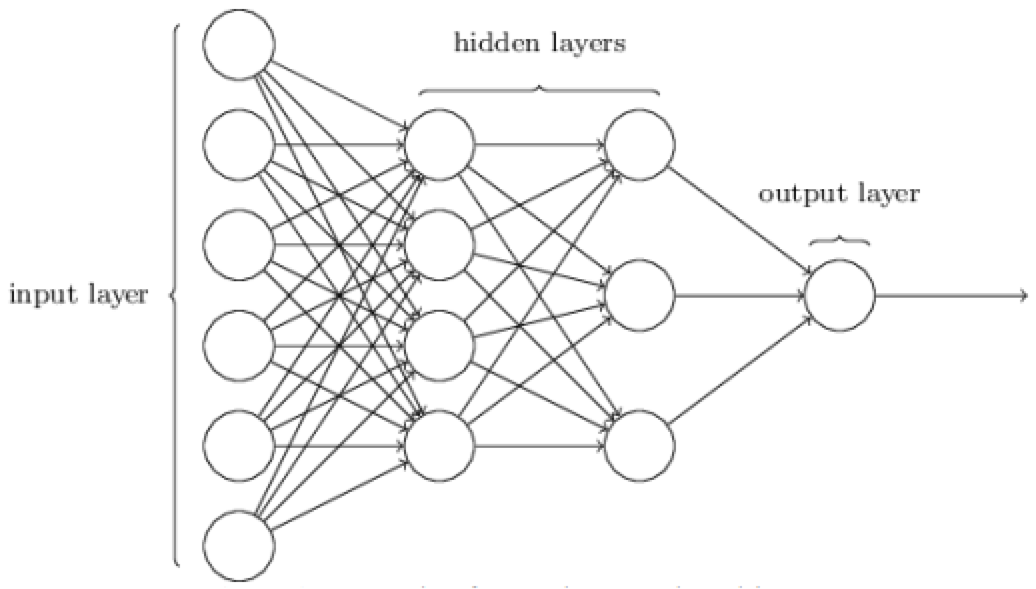


Fig: alphabet count in given data set

The flow of events:

⇒ Neural network is made up of 3 major layers named

1. Input layer
2. Middle layer(hidden layers)
3. Output layers



1. input data consists of 16 types of input classes so neural network's input layer consists of 16 nodes
2. output is an alphabet which can be any [A-Z], So, we made output into 26 classes with alphabets as classes
3. neural network will consisting of 26 output nodes
4. Total of 5 architectures are used
    a. 0 hidden layer → mlb_a in code
    b. 1 hidden layer with 2 nodes → mlb_b in code

c. 1 hidden layer with 6 nodes → mlb_c in code
d. 2 hidden layers with 2 and 3 nodes respectively → mlb_d in code
e. 2 hidden layers with 3 and 2 nodes respectively → mlb_e in code

5. With learning rates ⇒ 0.1, 0.01, 0.001, 0.0001, 0.00001.
6. Input data is normalized and output data is one-hot encoded which is then converted to torch.tensor
7. Normalized data is split into train, test and validation set with ration 70:15:15.

Class MLP(nn.module):

1. MLP class creates architectures
2. Input values are given as `(input_size,output_classes,hidden1_size=0,hidden2_size=0,no_of_layers=0)`
3. Total of 5 different architectures(neural networks) are created, with 5 different inputs.

Function `TRAIN_TEST():`

1. Input's are given as `(model, X_train,Y_train,X_val,Y_val,X_test,Y_test,epochs = 100, lr= 0.1)`
2. This function trains the model with a train and a validation set of data and returns the accuracy of test data.

All 5 models are trained with 5 different learning rates with 100 epochs:
Results(Accuracies):
Accuracies ⇒ (train accuracy, validation accuracy, test accuracy)

Models	Learning_rate = 0.1	Learning_rate = 0.01	Learning_rate = 0.001	Learning_rate = 0.0001	Learning_rate = 0.00001
mlp_a	(68.4,67.8,67.9)	(15.4,14.1,14.5)	(69.3,69.9,70.7)	(32.5,32.3,33.5)	(37.9,37.4,38.9)
mlp_b	(52.9,53,53.2)	(4.06,4.06,4.06)	(51.4,49.8,51.03)	(14.6,16,15.8)	(11.6,11.7,11.8)
mlp_c	(16.1,16.2,16.2)	(3.14,3.56,3.26)	(4.87,4.9,4.86)	(4.12,4.26,4.06)	(4.67,4.16,4.63)
mlp_d	(4.04,4.06,3.8)	(3.92,3.93,3.93)	(3.75,3.76,3.76)	(3.69,3.7,3.7)	(3.82,3.83,3.83)
mlp_e	(4.90,4.3,4.5)	(3.92,3.93,3.93)	(3.80,3.80,3.80)	(3.91,3.93,3.9)	(3.73,3.73,3.73)

plot:

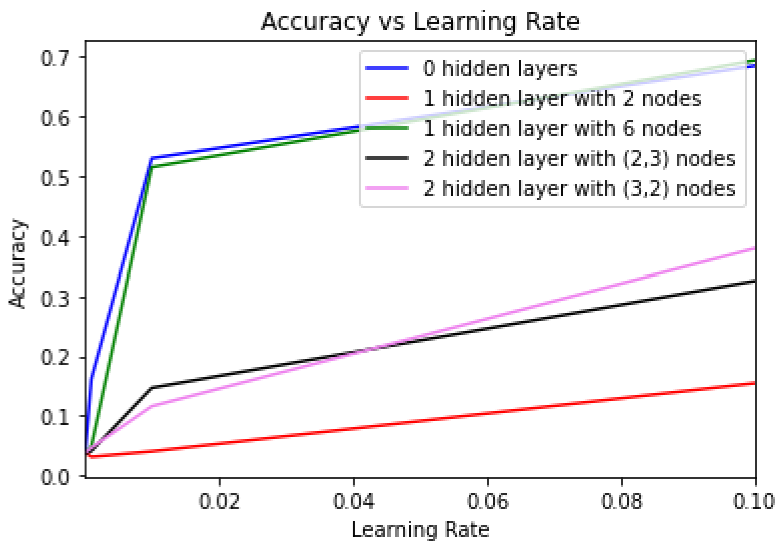


Fig: Accuracy vs learning rate.

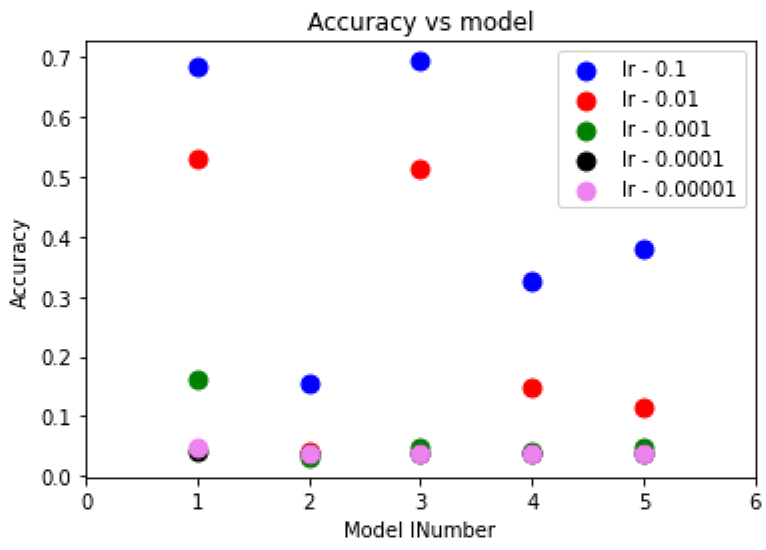
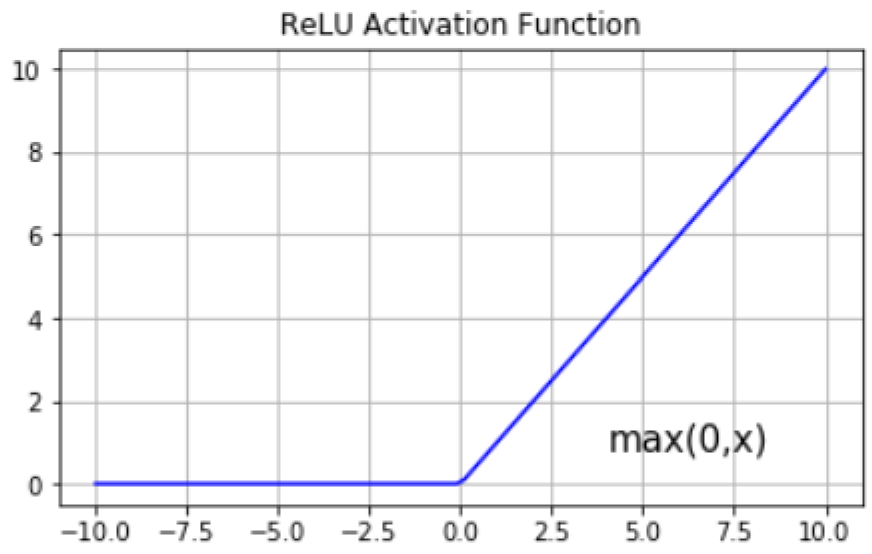


Fig: Accuracy vs model(note:mlp_a = 1,mlp_b=2,mlp_c=3,mlp_d=4,mlp_e=5)

Best Found Model:

1. The model which is best of all is which has 1 hidden layer with 6 nodes (over all)
2. With a learning rate = 0.1.
   Accuracy of that model with 0.1 learning rate is 69.3.
3. Its architecture has 16 input nodes in input layer, 6 hidden nodes in hidden layer and finally 26 output nodes in output layer. With a **Activation function Relu**

   i.



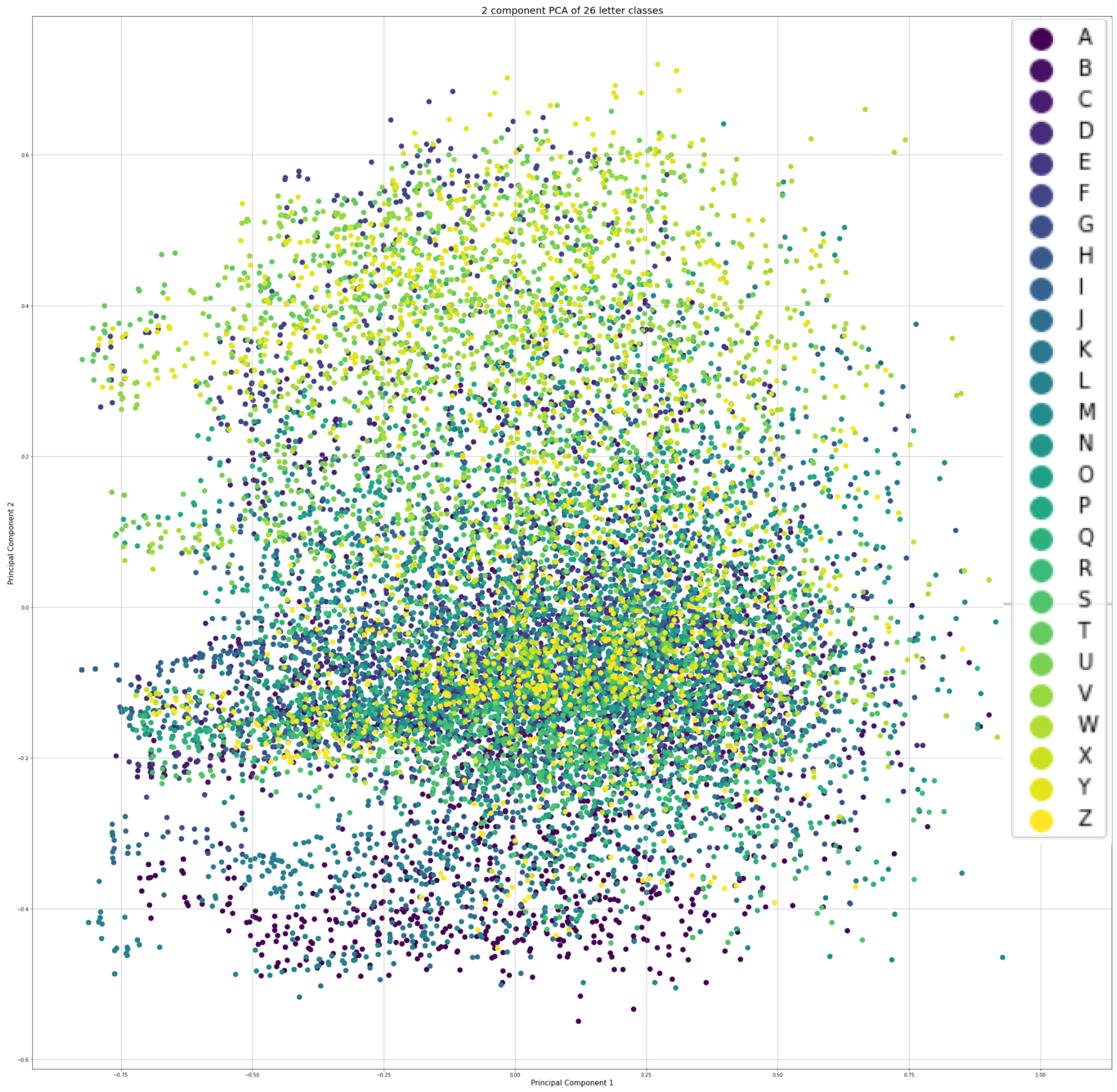
4. Used Stochastic Gradient descent for every model with batch size of 64.

Principal Component Analysis(PCA):

1. Whole 16 input data is converted to 2 input data by sklearn.decomposition package
2. PCA is initialised and fitted with training set of data and which is then transformed(reduced dimensions) to 2d data..

3. Which is then used to transform the validation set and test set with the same PCA.

Plot: of reduced data



X -axis as → principal component 1

Y-axis as → principal component 2

MLP classifier is then applied on this data which gave the following results

1. Taken learning rate = 0.1 which gave maximum accuracy for all previous 5 models.
2. New 5 models with only change in 2 input nodes are trained and tested and validated on MLP classifiers with 0.1 learning rate.
3. 5 models named
    a. mlp_a_p ⇒ 0 hidden layer
    b. mlp_b_p ⇒ 1 hidden layer with 2 nodes

c. mlp_c_p ⇒ 1 hidden layer with 6 nodes
d. mlp_d_p ⇒ 2 hidden layer with 2 nodes and 3 nodes
e. mlp_e_p ⇒ 3 hidden layer with 3 nodes and 2 nodes

model	**Learning rate = 0.1**
mlp_a_p	14.607, 13.433, 13.467
mlp_b_p	14.00, 12.933, 12.933
mlp_c_p	15.529, 14.933, 15.733
mlp_d_p	14.577, 12.900. 12.667
mlp_e_p	14.457, 14.367, 15.133

1. The accuracy by models before dimensionality reduction is far better than after dimensionality reduction.
2. May be due to the fact that reducing the whole data of 16 dimensions into 2 which basically has 26 dimensional output may lead to less accurate data for this classifier, and PCA is agnostic to Y.
3. PCA does not take information of classes into account, it just looks at the variance of each feature because it reasonably assumes that features that present high variance are more likely to have a good split between classes.