May 8, 2020 ENTS669D Project II Report Pranav H. Deo

Abstract:

The MNIST dataset has training dataset size of 60,000 images and testing dataset of 10,000 images (all gray scale). The number of classes we have is 10. In this project I have implemented the **SVM** using 3 approaches (**LDA**, **PCA** and **simple linear SVM**).
Another implementation of the MNIST dataset being the Neural Networking (CNN). I have then calculated the results on various epochs, learning rate and batch-size.

Classification Techniques:

1. Support Vector Machines (SVM)

a. LDA-SVM

- The training images and training labels are fed to the ***"fitcdiscr"*** MATLAB in-built function for discriminant analysis.
- The testing images along with the testing labels are also fed to the ***"fitcdiscr"*** MATLAB in-built function for discriminant analysis.
- I have used a pseudo-quadratic discrimination method/type for both training and testing.
- The new training dataset is obtained by multiplying the original training dataset to a vector "X" and the training labels by multiplying the original training labels to a vector "Y" which are generated by ***"fitcdiscr"***.
- Similar with the testing dataset and testing labels.
- An SVM model, "lda-svm-model", is thus produced by the ***"fitcecoc"***, which takes the transformed training data and training labels as the parameters.
- This model, "lda-svm-model", is now used to test the images in the testing dataset.
- We use an in-built MATLAB function ***"predict"***, for finding the accuracy.

b. PCA-SVM

- The training images and training labels are fed to the ***"pca"*** MATLAB in-built function for generating the principal component analysis.
- The testing images along with the testing labels are also fed to the ***"pca"*** MATLAB in-built function for principal component analysis.
- The coefficient vector is the Eigenvector and we multiply this vector to the training dataset and testing dataset.

- An SVM model, “pca-svm-model”, is thus produced by the ***“fitcecoc”***, which takes the transformed training data and training labels as the parameters.
- This model, “lda-svm-model”, is now used to test the images in the testing dataset.
- We use an in-built MATLAB function ***“predict”***, for finding the accuracy.

c. Simple Linear SVM

- The training images and labels are fed to the ***“fitcecoc”*** an in-built MATLAB function for SVM.
- The model generated as a result of the ***“fitcecoc”*** is then used as a parameter along with the testing images to generate the resulting labels.

2. **CNN:**

a. The first step we perform is to read the dataset and the labels using the “MNIST_Data_Read” and “MNIST_Labels_Read” function.

b. Then we reshape these images as per the dimensions (28x28) stated in the paper by **Y. LeCun**.

c. To initialize, we setup the CNN layers using structure and state the layers to be ‘input’, ‘convolutional’, ‘subsampling’.

d. I have setup a **5-layer** NN in this project (including the input layer).

e. The first convolutional layer has 6 kernels and the next has 6x12.

f. A wider CNN would be able to take 12 and 24 in the 1st and 2nd layers resp.

g. Using a pre-defined function: *cnn_setup* (which initializes the layers), I passed the cnn layers, training data and its pertaining labels as the parameters.

h. I initially used the learning rate as 1, batch_size as 50 and the no. of epochs as 1.

i. Sending these as the parameters to the *cnntrain,* a model was generated which contains the converged weights **(Feed-Forward Weights)**.

j. Using a pre-defined function: *cnntest*, we then implement the testing (i.e. compare the testing labels and the labels generated by the model).

k. The activation function used is sigmoid.

$$L = \frac{1}{2} * \sum_{k=1}^{10} (Z(k) - t(k))^2$$

l. Thus, increasing the number of epochs will decrease the ***“L”*** (error rate/loss function).

Analysis:

Dataset (MNIST)	**PCA-SVM**	**LDA-SVM**	**Linear SVM**	**Average (%)**
Training: 60,000 Testing: 10,000	94.46	94.38	94.38	**94.40**

**Table 1. Shows the overall accuracy for each SVM classifier**

Epochs	**Batch Size**	**Learning Rate**	**Accuracy (%)**
1	50	0.5	**99.34**
2	50	0.5	**99.50**
10	100	0.5	**99.99**
50	100	0.5	**99.99**
100	100	0.5	**99.99**

**Table 2. Shows the CNN classifier**

Results:

- In SVM, we notice that, the best result is obtained in PCA-SVM with a 94.46% as compared to LDA and linear SVM.
- In CNN, increased number of epochs causes increased accuracy. The MNIST dataset with 5-6 layers NN achieves an accuracy of ~ 99.7% on average with a test error rate of 0.35%.

****