Project 3: Classification

Sagnik Ghosh
Person Number : 50289151
University at Buffalo - Department of Computer Science
sagnikgh@buffalo.edu

Abstract

This project is about implementation of machine learning methods for the task of classification. First, an ensemble of four classifiers for a given task was implemented. Then the results of the individual classifiers are combined to make a final decision.

The classification task was to recognizing a 2828 grayscale handwritten digit image and identify it as a digit among 0, 1, 2, ... , 9.

The following four classifiers were used to train on MNIST digit images :

1. Logistic regression, which was implemented yourself using back propagation.

2. Multi-layer perceptron neural network, which was trained on the MNIST digit images.

3. Random Forest package, which was trained on the MNIST digit images.

4. SVM package, which was trained on the MNIST digit images.

1 Types of Datasets:

Two data-sets have been used in this project to train and test different models:

1.1 MNIST Data-set:

For both training and testing of the classifiers, MNIST dataset was used. The MNIST database is a large database of handwritten digits that is commonly used for training various image processing systems.The database is also widely used for training and testing in the field of machine learning.

Figure 1:

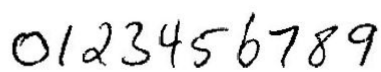
Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

The original black and white (bilevel) images from MNIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. the images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.

Intensity value of each pixel (between 1 and 255) will acct as one feature of the dataset. So we can say that the dimension of the MNIST data set will be (50000*784).

1.2 USPS Data

We use USPS handwritten digit as another testing data for this project to test whether your models could be generalize to a new population of data. Examples of each of the digits are given below.



Examples of each of the digits

Figure 2:

Each digit has 2000 samples available for testing. These are segmented images scanned at a resolution of 100ppi, cropped and Resized or filled to 28x28 like MNIST digits, so that it can be fed into into trained model and compare the result on USPS data and MNIST test data.

2 Steps followed:

1. Data Partition: The MNIST data set was originally partitioned into a training set and a testing set. This partition was used to train the models on the training set.

2. Train model parameter: For a given group of hyper-parameters such as the number of layers and the number of nodes in each layer, trained the models parameters on the training set.

3. Tune hyper-parameters: Validated the classified performance of the following models on the validation set. Changed the hyper-parameters and repeated step 3.

4. Evaluate on testing sets: Trained the following models **only on MNIST test set** and tested the models on **both MNIST and USPS data-set**. https://www.overleaf.com/project/5bf5e0f68664ca5aeafef7ad

3 Implementation of Multi-class Logistic Regression with Softmax (Softmax Regression)

Softmax regression (or multi-class logistic regression) is a generalization of logistic regression to the case where we want to handle multiple classes. In logistic regression we assumed that the labels were binary: y(i)={0,1}. We used such a classifier to distinguish between two kinds of hand-written digits. Softmax regression allows us to handle y(i)={1,...,K} where K is the number of classes.

In the softmax regression setting, we are interested in multi-class classification (as opposed to only binary classification), and so the label y can take on K different values, rather than only two. Thus, in our training set {(x(1),y(1)),...,(x(m),y(m))}, we now have that y(i)={1,2,...,K}. For example, in the MNIST digit recognition task, we would have K=10 different classes.

"Softmax" functions returns a probability value between 0 and 1 for each class (Eg: when k=10, softmax will return 10 probability values) and all 10 probability values add up to 1. All the probability value represents the probability of corresponding classes of being the correct prediction. The class with the highest probability will be considered as the predicted class.

The mathematical representation of the softmax function is as follows:

$$\boxed{\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}}$$

Figure 3:

3.1 The code implemented is as follows:

```
learningRate = 0.005
nb_classes   = 10
La           = 0.000001

W_Now = np.ones((c,10))

#<----------------------------------------------------Test Data---------------------------------------------------------->#

for i in range(0,50000):

    t_data        = trainingdata[i].reshape(784,1)
    y             = softmax(t_data.T,W_Now)
    tr = trainingtarget[i].reshape(1,10)
    Target_new    = np.subtract(y,tr)
    Delta_E       = np.dot(t_data,Target_new)
    Delta_E_La    = Delta_E + La
    Delta_W       = np.dot(learningRate,Delta_E_La)
    W_T_Next      = np.subtract(W_Now,Delta_W)
    W_Now         = W_T_Next
```

In this code Stochastic gradient descent approach was followed, where the regression was performed on each row of the data (1*784) and the wights were updated based on that.

Another Key term in this model is "One-hot vector". One hot encoding converts a categorical vector into a classified matrix (One-hot vector) which suitable for multi-class regression problem.

Here, the target vector was converted into one-hot vector so that it can be subtracted from output of the "Softmax" function (which is a multi-class matrix) in order to calculate the error.

3.2 Hyper-parameters:

In this code The following hyper parameters were tuned:

1. Regularizer (Lambda)

2. Learning rate

3. Number of iterations

The higher learning rate and higher value of the lambda both was observed to have negative impact on the performance of the model.

Because of this reason the value of lambda is set to be very low and the value of Learning rate was observed to give the best performance when set in between 0.005 an 0.01.

Since, the SGD approach was followed in this code, the number of epochs was set to 1. Increasing the number of epochs was observed to have no visible effect on the performance of the model.

3.3 Results :

3.3.1 MNIST dataset:

Testing Accuracy : 90.84%

```
array([[ 955,    0,    1,    4,    0,    5,    9,    1,    5,    0],
       [   0, 1113,    0,    5,    0,    3,    4,    2,    8,    0],
       [   9,    9,  881,   31,   11,    3,   13,   24,   43,    8],
       [   2,    0,   11,  927,    0,   37,    1,   12,   12,    8],
       [   2,    4,    4,    1,  887,    1,   12,    4,   10,   57],
       [   8,    5,    1,   44,    6,  775,   10,    9,   26,    8],
       [  12,    3,    3,    3,    8,   29,  892,    3,    5,    0],
       [   2,   14,   19,    8,    5,    0,    0,  951,    3,   26],
       [   3,   10,    4,   46,    7,   55,   12,   16,  808,   13],
       [   8,    8,    2,   13,   20,   19,    0,   40,    4,  895]],
      dtype=int64)
```

Figure 4: Confusion Matrix

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       980
           1       0.95      0.98      0.97      1135
           2       0.95      0.85      0.90      1032
           3       0.86      0.92      0.89      1010
           4       0.94      0.90      0.92       982
           5       0.84      0.87      0.85       892
           6       0.94      0.93      0.93       958
           7       0.90      0.93      0.91      1028
           8       0.87      0.83      0.85       974
           9       0.88      0.89      0.88      1009

   micro avg       0.91      0.91      0.91     10000
   macro avg       0.91      0.91      0.91     10000
weighted avg       0.91      0.91      0.91     10000
```

Figure 5: classification Report

3.3.2 USPS dataset:

Testing Accuracy : 34.9467%

```
array([[ 563,    4,  255,   84,  167,  238,   95,   88,  146,  360],
       [ 159,  356,  161,  341,  153,  108,   29,  464,  202,   27],
       [ 173,   21, 1101,  233,   37,  149,   87,   85,   81,   32],
       [  59,    2,  108, 1337,    3,  331,    3,   67,   56,   34],
       [  64,   81,   32,   75,  833,  169,   46,  210,  321,  169],
       [ 134,   19,  162,  210,   24, 1232,   82,   72,   48,   17],
       [ 262,   10,  334,  147,   65,  396,  677,   26,   41,   42],
       [ 161,  200,  216,  523,   58,  116,   22,  368,  276,   60],
       [ 222,   32,  116,  248,   76,  737,  102,   56,  343,   68],
       [  30,  144,  111,  526,   88,  112,   14,  460,  336,  179]],
      dtype=int64)
```

Figure 6: Confusion Matrix

	precision	recall	f1-score	support
0	0.31	0.28	0.29	2000
1	0.41	0.18	0.25	2000
2	0.42	0.55	0.48	1999
3	0.36	0.67	0.47	2000
4	0.55	0.42	0.48	2000
5	0.34	0.62	0.44	2000
6	0.59	0.34	0.43	2000
7	0.19	0.18	0.19	2000
8	0.19	0.17	0.18	2000
9	0.18	0.09	0.12	2000
micro avg	0.35	0.35	0.35	19999
macro avg	0.35	0.35	0.33	19999
weighted avg	0.35	0.35	0.33	19999

Figure 7: classification Report

4 Neural Network :

In this project a Deep Neural Network (DNN) has been implemented. The structure of the neural network is as follows:

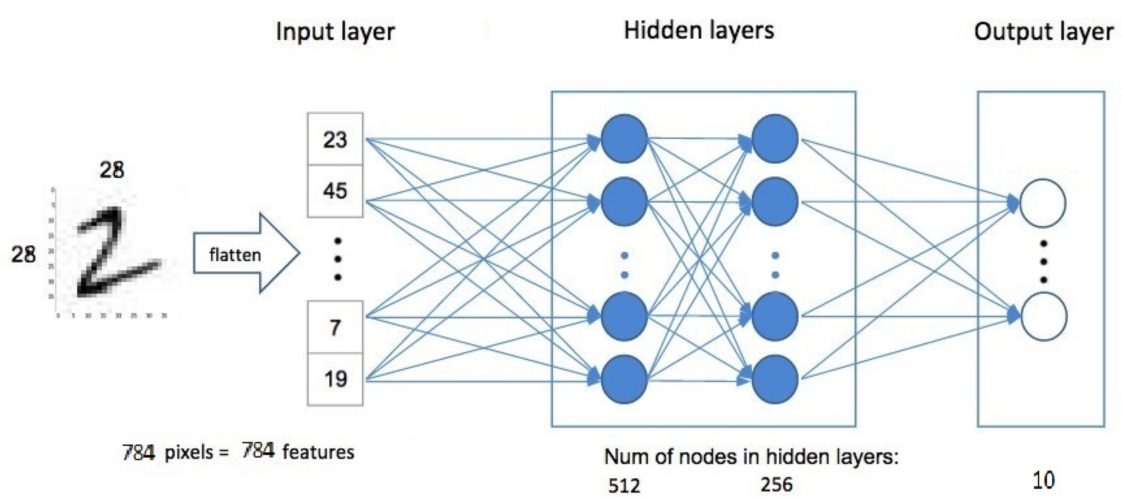


Figure 8: classification Report

Two Dense-Layer has been implemented in this model. Since the accuracy was very high already in this setting, it appeared unnecessary to increase any more Dense-Layer to the model.

The number of nodes in the input layer was set to 784, since there are 784 features in the input matrix.

The number of nodes in the first dense layer was 512 and the number of nodes in the second Dense-layer was 256 and in the final and third dense layer which is the output layer as well, has 10 nodes (since we are supposed to have 10 classes in the output vector).

In the first dense layer, "Relu" Activation was applied and in the second and third dense layer, "Softmax" activation was applied.

4.0.1 Hyper-parameters:

In this code The following hyper parameters were tuned:

1. number of epochs

2. Model Batch Size

3. Activation Functions

All the above parameters were tuned and it was observed to give the maximum accuracy with the following setting:

number of epochs = 10000 Model Batch Size = 128

4.1 Results :

The following image represents the graph generated by plotting loss and accuracy for training and validation data.

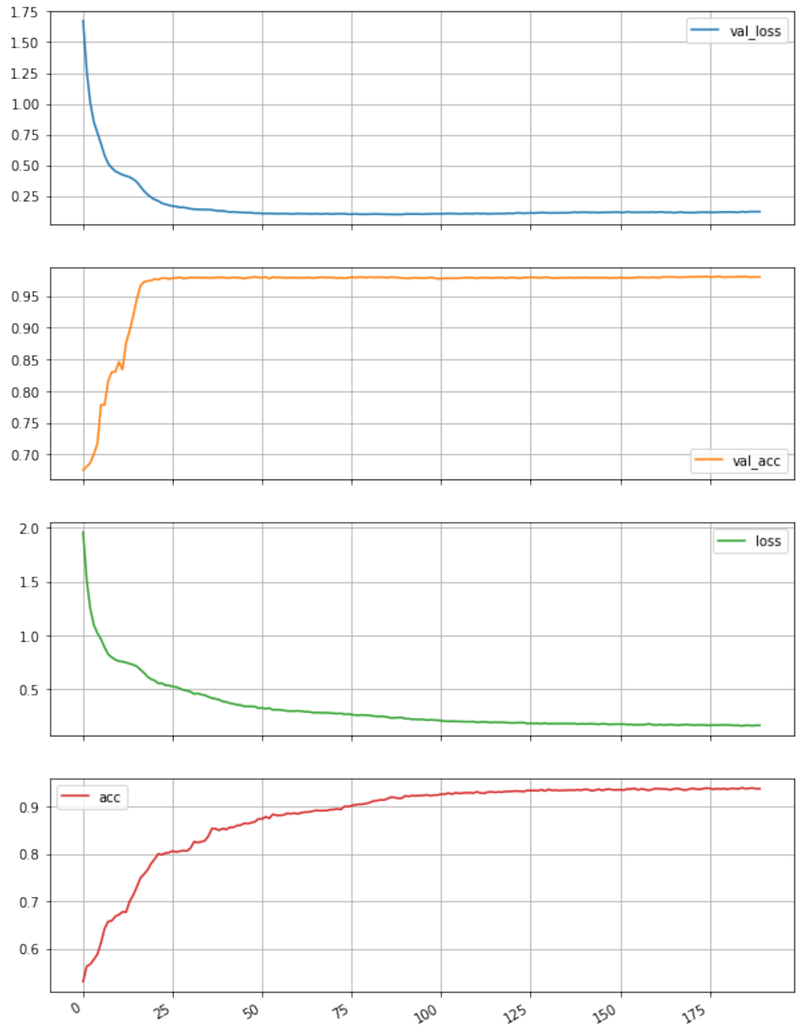


Figure 9: Confusion Matrix

4.1.1 MNIST dataset:

Testing Accuracy : 97.92%

```
array([[ 968,    1,    0,    0,    0,    0,    7,    0,    3,    1],
       [   0, 1124,    2,    1,    0,    0,    1,    0,    7,    0],
       [   1,    0, 1007,    2,    3,    0,    2,    9,    8,    0],
       [   0,    0,    3,  988,    0,    8,    0,    2,    7,    2],
       [   0,    0,    0,    0,  960,    0,    7,    1,    1,   13],
       [   3,    0,    0,   10,    0,  865,    5,    0,    7,    2],
       [   4,    2,    0,    0,    6,    2,  942,    0,    2,    0],
       [   0,    1,   10,    1,    0,    0,    0, 1008,    4,    4],
       [   0,    1,    2,    5,    6,    3,    1,    1,  952,    3],
       [   2,    3,    0,    6,   11,    3,    1,    3,    2,  978]],
      dtype=int64)
```

Figure 10: Confusion Matrix

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       0.99      0.99      0.99      1135
           2       0.98      0.98      0.98      1032
           3       0.98      0.98      0.98      1010
           4       0.97      0.98      0.98       982
           5       0.98      0.97      0.98       892
           6       0.98      0.98      0.98       958
           7       0.98      0.98      0.98      1028
           8       0.96      0.98      0.97       974
           9       0.98      0.97      0.97      1009

   micro avg       0.98      0.98      0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

Figure 11: classification Report

4.1.2 USPS dataset:

Testing Accuracy : 47.73%

```
array([[ 810,   34,   49,  181,  136,  271,   82,   62,   73,  302],
       [  37,  488,  248,  105,  437,   80,    8,  207,  285,  105],
       [  69,   31, 1434,   79,   11,   87,   52,   41,  186,    9],
       [  17,   11,   48, 1588,    5,  175,    1,   16,  129,   10],
       [  34,  190,   23,   44, 1157,   65,   31,  115,  273,   68],
       [ 162,   24,   54,  394,    4, 1139,   38,   28,  141,   16],
       [ 336,   15,  325,   43,   36,  147,  943,   50,   81,   24],
       [  34,  128,  170,  641,   61,   27,    6,  589,  272,   72],
       [ 135,   32,   49,  536,   57,  344,   72,   44,  672,   59],
       [  14,  130,   72,  388,  146,   29,    5,  500,  390,  326]],
      dtype=int64)
```

Figure 12: Confusion Matrix

	precision	recall	f1-score	support
0	0.49	0.41	0.44	2000
1	0.45	0.24	0.32	2000
2	0.58	0.72	0.64	1999
3	0.40	0.79	0.53	2000
4	0.56	0.58	0.57	2000
5	0.48	0.57	0.52	2000
6	0.76	0.47	0.58	2000
7	0.36	0.29	0.32	2000
8	0.27	0.34	0.30	2000
9	0.33	0.16	0.22	2000
micro avg	0.46	0.46	0.46	19999
macro avg	0.47	0.46	0.44	19999
weighted avg	0.47	0.46	0.44	19999

Figure 13: classification Report

5 Support Vector Machine :

A support vector machine (SVM) is machine learning algorithm that analyzes data for classification and regression analysis. SVM is a supervised learning method that looks at data and sorts it into one of two categories. An SVM outputs a map of the sorted data with the margins between the two as far apart as possible. SVMs are used in text categorization, image classification, handwriting recognition and in the sciences.

SVM is a high margin classifier. SVM can take different kernels such as Linear kernel and Gaussian kernel.

Both Linear and gaussian (with gamma = 1 and gamma = default) has been applied in this project.

Here the results are shown for the "Linear kernel" only.

5.1 Results :

5.1.1 MNIST dataset:

Testing Accuracy : 93.9%

	precision	recall	f1-score	support
0	0.95	0.98	0.96	980
1	0.97	0.99	0.98	1135
2	0.92	0.94	0.93	1032
3	0.90	0.93	0.92	1010
4	0.93	0.96	0.95	982
5	0.92	0.89	0.91	892
6	0.96	0.95	0.95	958
7	0.95	0.93	0.94	1028
8	0.93	0.89	0.91	974
9	0.95	0.91	0.93	1009
micro avg	0.94	0.94	0.94	10000
macro avg	0.94	0.94	0.94	10000
weighted avg	0.94	0.94	0.94	10000

Figure 14: Confusion Matrix

959	0	5	2	2	4	7	0	1	0
0	1121	3	3	0	1	2	1	4	0
6	8	968	9	3	2	11	10	13	2
5	2	17	944	4	13	1	8	13	3
2	1	10	1	943	0	4	2	2	17
13	4	2	39	5	792	9	1	22	5
10	3	11	1	5	14	911	2	1	0
1	8	20	10	6	1	0	961	3	18
8	4	9	25	11	27	6	5	871	8
7	6	2	13	32	4	0	18	7	920

Figure 15: classification Report

5.1.2 USPS dataset:

Testing Accuracy : 29.1265%

	precision	recall	f1-score	support
0	0.36	0.17	0.24	2000
1	0.49	0.15	0.23	2000
2	0.25	0.65	0.36	1999
3	0.25	0.45	0.32	2000
4	0.46	0.40	0.43	2000
5	0.24	0.44	0.31	2000
6	0.61	0.23	0.33	2000
7	0.23	0.26	0.24	2000
8	0.25	0.08	0.12	2000
9	0.28	0.08	0.13	2000
micro avg	0.29	0.29	0.29	19999
macro avg	0.34	0.29	0.27	19999
weighted avg	0.34	0.29	0.27	19999

Figure 16: Confusion Matrix

348	0	476	152	222	345	74	172	10	201
60	303	534	275	230	172	17	351	37	21
139	63	1293	115	33	221	55	45	21	14
56	58	341	898	8	520	9	45	48	17
24	24	221	82	800	215	10	464	82	78
47	25	652	240	41	876	30	35	41	13
146	19	903	55	86	264	462	38	2	25
19	74	201	706	54	294	12	522	84	34
100	16	298	449	126	692	82	58	160	19
18	38	204	588	142	104	8	580	155	163

Figure 17: classification Report

6 Random Forest :

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.[1][2] Random decision forests correct for decision trees' habit of overfitting to their training set.

There is only one hyper-parameter i.e. "number of tress" that can be tuned in case of random-forest algorithm.

The following image describes the working process of random forest alogorithm :

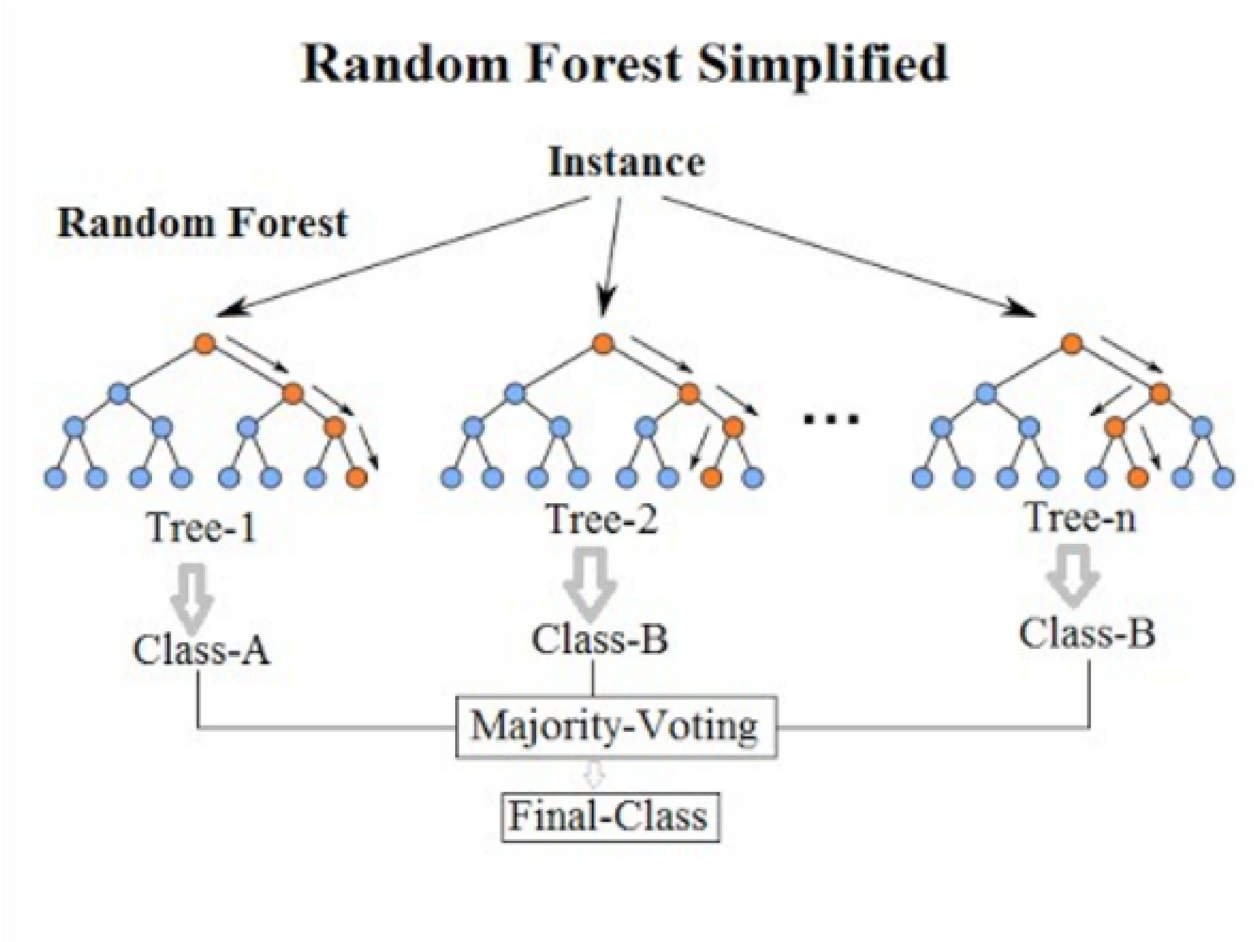


Figure 18: Confusion Matrix

6.1 Results :

6.1.1 MNIST dataset:

Testing Accuracy : 90.3%

	precision	recall	f1-score	support
0	0.51	1.00	0.68	980
1	1.00	0.98	0.99	1135
2	0.99	0.89	0.94	1032
3	1.00	0.87	0.93	1010
4	0.99	0.88	0.94	982
5	1.00	0.85	0.92	892
6	1.00	0.93	0.96	958
7	0.99	0.90	0.95	1028
8	1.00	0.83	0.91	974
9	0.99	0.88	0.93	1009
micro avg	0.90	0.90	0.90	10000
macro avg	0.95	0.90	0.91	10000
weighted avg	0.95	0.90	0.91	10000

Figure 19: Confusion Matrix

```
array([[ 979,    0,    0,    0,    0,    0,    0,    1,    0,    0],
       [  24, 1109,    0,    1,    0,    0,    1,    0,    0,    0],
       [ 102,    0,  923,    0,    0,    0,    2,    4,    1,    0],
       [ 130,    0,    1,  874,    0,    2,    0,    3,    0,    0],
       [ 107,    0,    0,    0,  869,    0,    0,    0,    1,    5],
       [ 132,    0,    0,    1,    0,  758,    1,    0,    0,    0],
       [  64,    2,    0,    0,    1,    0,  891,    0,    0,    0],
       [  90,    0,    8,    0,    0,    0,    0,  929,    0,    1],
       [ 160,    0,    0,    0,    1,    0,    0,    0,  811,    2],
       [ 118,    0,    0,    0,    3,    0,    0,    1,    0,  887]],
      dtype=int64)
```

Figure 20: classification Report

6.1.2 USPS dataset:

Testing Accuracy : 14.94%

	precision	recall	f1-score	support
0	0.11	0.99	0.19	2000
1	0.81	0.11	0.19	2000
2	0.94	0.04	0.08	1999
3	1.00	0.06	0.12	2000
4	0.97	0.10	0.19	2000
5	0.98	0.06	0.12	2000
6	0.97	0.02	0.04	2000
7	0.34	0.10	0.16	2000
8	1.00	0.00	0.00	2000
9	0.20	0.00	0.00	2000
micro avg	0.15	0.15	0.15	19999
macro avg	0.73	0.15	0.11	19999
weighted avg	0.73	0.15	0.11	19999

Figure 21: Confusion Matrix

```
array([[1987,    0,    3,    0,    6,    0,    0,    0,    0,    4],
       [1499,  220,    0,    0,    0,    0,    1,  280,    0,    0],
       [1899,    0,   85,    0,    0,    0,    0,   15,    0,    0],
       [1876,    0,    0,  124,    0,    0,    0,    0,    0,    0],
       [1727,    5,    0,    0,  205,    0,    0,   63,    0,    0],
       [1876,    0,    0,    0,    0,  123,    0,    1,    0,    0],
       [1960,    1,    0,    0,    0,    0,   39,    0,    0,    0],
       [1757,   34,    2,    0,    0,    0,    0,  207,    0,    0],
       [1995,    0,    0,    0,    0,    2,    0,    0,    3,    0],
       [1952,   11,    0,    0,    0,    0,    0,   36,    0,    1]],
      dtype=int64)
```

Figure 22: classification Report

7 Combining Models :

In this project Hard-Voting technique has been applied to combine the models. All four predicted output vectors had been stored in different variables and for each data point, that value which got the maximum vote (majority of the algorithms predicted that value to be correct) was selected to be the final value for that data point.

This technique was applied on both the data sets and the final output vector was generated.

The following is the implementation of the hard voting algorithm:

```
class_matrix = np.ones(10)

final_pred_target_mnist = np.zeros((r1,1))

for i in range(1, len(mnist1)):
    i1 = mnist1[i]
    i2 = mnist2[i]
    i3 = mnist3[i]
    i4 = mnist4[i]
    class_matrix[i1] += 1
    class_matrix[i2] += 1
    class_matrix[i3] += 1
    class_matrix[i4] += 1
    y = np.argmax(class_matrix)
    final_pred_target_mnist[i] = y
    class_matrix = np.ones(10)

counter = 0

for i in range (0,len(testtarget)):
    if(final_pred_target_mnist[i] == testtarget[i]):
        counter += 1
final_acc_mnist = (counter/len(testtarget))*100
```

Merged Accuracy on MNIST after hard-Voting ——> 90.47
Merged Accuracy on USPS after hard-Voting ——> 37.8119

8 Conclusion :

We trained our model on MNIST data set as a consequence, its performance on the MINIST test data is reasonable how ever as expected from the no free lunch theorem, the USPS test data gives poor results with less than 50% accuracy.

T he performance of the individual classifiers for MINIST data set was better than the combined performance in most cases. In case of USPS data set, the average accuracy of the four classifiers was close to the combined accuracy.

9 references

https://www.coursera.org/learn/machine-learning

Bishop - Pattern Recognition And Machine Learning - Springer 2006

https://en.wikipedia.org/wiki/Machine_learning

http://deeplearning.stanford.edu/tutorial/supervised/SoftmaxRegression/