PROJECT 3 Recognizing Handwritten Digital Images using Different Classification Models

Krishna Sehgal
50291124

1. OBJECTIVE

The goal of this project is to use classification to recognize handwritten digital images by using Logistic Regression, Neural Networks, Support Vector Machine and Random Forest.

There are four tasks

1. Train Logistic Model on MNIST Data-Set and test on MNIST and USPS Data-Set
2. Train Multilayer Perceptron Neural Network on MNIST Data-set and test on MNIST and USPS Data-Set
3. Train Support Vector Machine on MNIST Data-set and test on MNIST and USPS Data-Set
4. Train Random Forest on MNIST Data-set and test on MNIST and USPS Data-Set

2. SOFTWARE USED

- Anaconda- Navigator
- Jupyter Notebook
- Python 3.6

3. LIBRARIES

- Sklearn
- Keras
- Random
- TensorFlow
- NumPy
- Matplot

4. DATA SET

- **MNIST -** The MNIST database is a large database of handwritten digits that is commonly used for training various image processing systems. The database contains 60,000 training images and 10,000 testing images.

- **USPS -** The USPS handwritten digit is another testing data for this project to test whether models could be generalized to a new population of data. Each digit has 2000 samples available for testing.

5. HYPER PARAMETERS

- **Logistic Regression**

  - Learning Rate
  - Epochs

- **Neural Networks**

  - Batch Size
  - Number of Epochs
  - Number of Hidden Layers
  - Dense Layer Nodes
  - Activation Function (Sigmoid,Relu,tanh)

- **Random Forest**

  - Number of Trees
  - Criteria: “Entropy” or “Gini”
  - Maximum Depth
  - Minimum Sample Split

- **Support Vector Machine**

  - Gamma
  - Regularizer (C)
  - Kernel (Radio Basis Function, Poly, Sigmoid, Linear)

6. Classifiers

6.1 Multi-class Logistic Regression

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic softmax function to return a probability value which can then be mapped to discrete number of classes.
Given ten classes, we could solve multiclass classification problem we can force the output layer to be a discrete probability distribution over the ten classes. To be a valid probability distribution, we want target values to be non-negative and sum up to one. We accomplish this by using the *softmax function.* Given an input vector, softmax exponentiates forcing all values positive after that it normalizes so that all values sum to 1. Since we have 10 outputs so we will need weights connecting each of our inputs to each of our outputs. We can represent these weights one for each input node, output node pair in a matrix W. We generate the linear mapping from inputs to outputs via a matrix-vector product x*W+b.* The whole model can be written as

$$target=softmax\ (xW+b)$$

6.2 Neural Networks

A neuron takes a group of weighted inputs, applies an activation function, and returns an output.
Inputs to a neuron can either be features from a training set or outputs from a previous layer's neurons. Weights are applied to the inputs as they travel along synapses to reach the neuron. The neuron then applies an activation function to the "sum of weighted inputs" from each incoming synapse and passes the result on to all the neurons in the next layer. **Bias** terms are additional constants attached to neurons and added to the weighted input before the activation function is applied. Bias terms help models represent patterns that do not necessarily pass through the origin. Bias terms typically accompany weights and must also be learned by your model.
Input Layer holds the data your model will train on. Each neuron in the input layer represents a unique attribute in your dataset.
Hidden Layer sits between the input and output layers and applies an activation function before passing on the results. There are often multiple hidden layers in a network. In traditional networks, hidden layers are typically fully-connected layers — each neuron receives input from all the previous layer's neurons and sends its output to every neuron in the next layer. This contrasts with how convolutional layers work where the neurons send their output to only some of the neurons in the next layer.
The final layer in a network. It receives input from the previous hidden layer, optionally applies an activation function, and returns an output representing your model's prediction.
A neuron's input equals the sum of weighted outputs from all neurons in the previous layer. Each input is multiplied by the weight associated with the synapse connecting the input to the current neuron.
Activation functions live inside neural network layers and modify the data they receive before passing it to the next layer. **Activation functions** give neural networks their power — allowing them to model complex non-linear relationships. By modifying inputs with non-linear functions neural networks can model highly complex relationships between features. Popular activation functions include relu and sigmoid.
A l**oss function, or cost function**, is a wrapper around our model's predict function that tells us "how good" the model is at making predictions for a given set of parameters. The loss function has its own curve and its own derivatives. The slope of this curve tells us how to change our parameters to make the model more accurate! We use the model to make predictions. We use the cost function to update our parameters. The basic modeling idea is that we're going to linearly map our input x onto 10 different real value outputs. before outputting these values, we'll want to normalize them so that they are non-negative and sum to 1. This normalization allows us to interpret the output as a valid probability distribution.

6.3 Support Vector Machine

Multi-class SVM aims to assign labels to instances by using support vector machines, where the labels are drawn from a finite set of several elements. SVM seeks to find the optimal separating hyperplane between binary classes by following the maximized margin criterion. A support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

6.4 Random Forest

A random forest consist of combination of uncorrelated decision trees and each works on various sub samples of the dataset. As each tree have different error instances averaging them all can provide better predictive accuracy and control over-fiting. Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model. Therefore, in Random Forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random, by additionally using random thresholds for each feature rather than searching for the best possible thresholds. In Random Forest there is a majority voting based on the decision made by a tree when a number of trees make decision. If the number of trees are very large, the training will be slow but it will have a better performance. A criteria such as Gini or entropy could be used. Gini calculates the purity of Kernel. We calculate the Gini node to see whether this split gives better result and very pure as compared to the parent node.

7 OUTPUTS

7.1 LOGISTIC REGRESSION

7.1.1 ACCURACY AND CONFUSION MATRIX ON MNST DATA-SET

```
***************LOGISTIC REGRESSION***************
NUMBER OF EPOCHS 700
Learning Rate  0.07
-----------MNIST--------------
accuracy of MNIST is
0.8656
confusion matrix
[[ 934    0    3    3    1   12   15    0   11    1]
 [   0 1089   10    5    1    1    8    0   21    0]
 [  14   12  853   30   20    1   20   15   50   17]
 [   7    0   30  842    1   62   11   17   25   15]
 [   2    6   11    0  856    5   16    3   20   63]
 [  19    4    5   53   19  680   26    7   64   15]
 [  20    3   10    1   24   20  868    1   10    1]
 [   3   16   27    4   14    1    2  900    5   56]
 [   8    9   16   40   12   40   20   19  792   18]
 [   9    5   11   14   59   10    1   40   18  842]]
```

7.1.2 ACCURACY AND CONFUSION MATRIX ON USPS DATA-SET

```
----------USPS Data-Set--------------
accuracy for USPS is
0.264013200660033
confusion matrix
[[517   2 224 107 222 188 200 101 107 332]
 [267 237 138 237 334  51  39 265 362  70]
 [267  29 911 148  95  69 127 101 190  62]
 [184   9 237 750  49 289  32  92 258 100]
 [ 90  39  50 162 708 198 110 147 221 275]
 [142  20 190 235  44 905 113  76 201  74]
 [264  37 315 110  84 285 506 100 240  59]
 [157 118 309 665  63 122  60 207 155 144]
 [143  27 150 393 121 521 137  60 248 200]
 [ 54  74 143 616 165 180  58 250 169 291]]
```

7.2 NEURAL NETWORK

7.2.1 ACCURACY AND CONFUSION MATRIX ON MNST DATA-SET

```
---------TESTING STATISTICS ON MNIST DATASET-------------

 accuarcy 0.9699
loss percentage 0.106668368672207
confusion matrix :
[[ 963    0    2    1    0    3    6    1    2    2]
 [   1 1122    3    0    0    1    4    1    3    0]
 [   3    0 1001    7    2    0    4    5    9    1]
 [   1    1    3  981    0    8    0    5    7    4]
 [   0    0    4    1  950    0    5    2    3   17]
 [   4    0    2   14    2  854    6    1    7    2]
 [   7    2    2    1    4    3  935    0    4    0]
 [   2    2   13    9    2    0    0  987    2   11]
 [   3    0    2    8    4    5    6    4  939    3]
 [   3    4    2    9    9    6    0    7    2  967]]
```

7.2.2 ACCURACY AND CONFUSION MATRIX ON USPS DATA-SET

```
 accuracy for usps dataset:
0.3476173808675533

loss percentage for testing on usps dataset: 4.884816352758022
confusion matrix :
[[ 388    1  233   66   82  130  468  285  182  165]
 [  22  274  579  129  153  100   53  479  183   28]
 [  61   15 1355  130    6  113  184   30   98    7]
 [  31   15  328 1103    4  327   67   34   82    9]
 [  16    9  150  107  795   64   94  464  255   46]
 [  22   12  593  238   19  702  195   38  168   13]
 [ 152   18  641   76   25  108  826   77   33   44]
 [  19   15  230  568   35   57   41  875  154    6]
 [ 104    2  168  462  137  183  216  225  490   13]
 [   8    9  198  328  106   24   41  822  320  144]]
```

7.3 RANDOM FOREST

7.3.1 ACCURACY AND CONFUSION MATRIX ON MNST DATA-SET

```
||||||||||||||||||||||**RANDOM FOREST**|||||||||||||||||||||||||
----------------Accuracy for random forest on MNIST test set---------------

 accuracy : 0.9699

 confusion matrix  [[ 969    0    1    0    0    2    3    1    4    0]
 [   0 1122    3    3    0    2    2    0    2    1]
 [   6    0 1001    4    3    0    4    8    6    0]
 [   1    0   10  974    0    6    0    9    7    3]
 [   1    0    1    0  954    0    5    0    2   19]
 [   2    0    1   11    3  861    5    2    5    2]
 [   7    3    0    0    3    3  940    0    2    0]
 [   1    4   21    1    1    0    0  989    1   10]
 [   3    0    6    5    3    6    4    5  931   11]
 [   6    5    3   12   12    2    1    5    5  958]]
```

7.3.2 ACCURACY AND CONFUSION MATRIX ON USPS DATA-SET

```
----------------Accuracy for random forest on USPS test set---------------

 accuracy : 0.40772038601930094
confusion matrix
[[ 640   12  282   56  441  154   66   97    1  251]
 [  39  573  110  104   46   97   22  992   16    1]
 [  81   26 1297   67   53  185   18  263    8    1]
 [  32    7  104 1313   48  300    2  177    5   12]
 [  10  206   54   22 1071  183   15  400   24   15]
 [ 137   35  133   71   24 1458   21  113    6    2]
 [ 294   49  237   23   85  338  825  139    2    8]
 [  33  333  380  213   33  272   31  691    3   11]
 [  42   41  155  206   99 1100   64  102  177   14]
 [  15  262  224  295  246  134   12  622   81  109]]
--------------------------------------------------------------------------
```

7.4 SUPPORT VECTOR MACHINE

7.4.1 ACCURACY AND CONFUSION MATRIX ON MNST DATA-SET

```
||||||||||||||||||**SUPPORT VECTOR MACHINE**||||||||||||||||||
----------------Accuracy for SVM on mnist test set---------------
accuracy :
0.9827
confusion matrix
[[ 974    0    1    0    0    1    1    1    2    0]
 [   0 1128    3    1    0    1    0    1    1    0]
 [   4    0 1015    1    1    0    0    6    5    0]
 [   0    0    1  997    0    3    0    5    4    0]
 [   0    1    3    0  964    0    4    0    2    8]
 [   2    0    1    7    1  872    3    1    4    1]
 [   5    2    0    0    2    3  945    0    1    0]
 [   0    3    9    1    1    0    0 1004    2    8]
 [   2    0    1    6    1    2    0    2  958    2]
 [   4    4    2    8    7    2    0    6    6  970]]
```

7.4.2 ACCURACY AND CONFUSION MATRIX on USPS DATA-SET

```
----------------Accuracy for SVM on USPS test set---------------

 accuracy :
0.2614130706535327
confusion matrix
[[ 226    0 1564    2   26   35    2    0   79   66]
 [  78  257  713  172  262   77   12  337   88    4]
 [   8    0 1944    6    2   20    1    6   11    1]
 [   4    0 1193  725    0   41    0    0   37    0]
 [   6    0 1045   18  522   96    0   56  252    5]
 [  15    0 1305   16    1  626    0    0   37    0]
 [  78    0 1534    2   10   61  290    0   22    3]
 [  17    6 1435  129    6  134    0  220   52    1]
 [   7    0 1387   14    4  221    0    0  367    0]
 [   1    0 1508   79   26   29    0   39  267   51]]
```

7.5 Ensemble Classifier

7.5.1 ACCURACY AND CONFUSION MATRIX ON MNST DATA-SET

```
ensembled mnist accuracy: 0.971
[[ 967    0    1    1    1    0    4    1    4    1]
 [   0 1123    2    2    0    1    3    0    4    0]
 [   4    1  985    3    7    0    3    6   23    0]
 [   1    0    6  981    1    7    0    4    8    2]
 [   0    0    2    0  964    0    6    2    2    6]
 [   5    1    0    9    5  849    5    0   16    2]
 [   5    3    2    1    5    2  936    0    4    0]
 [   0    6    6    5    0    0    0 1001    4    6]
 [   5    0    2    5    5    2    3    3  945    4]
 [   2    1    0    4   27    5    0    5    6  959]]
```

7.5.2 ACCURACY AND CONFUSION MATRIX ON USPS DATA-SET

```
ensembled USPS accuracy: 0.488524426221
[[ 707    4  108   17  144  156   48  165   69  582]
 [  50  563  159   36  177   52   47  750  131   35]
 [  74   75 1255  120   24  169   73  109   95    5]
 [  26   16  127 1279    4  333    5   96  100   14]
 [  12   21   70   11 1033   89   35  482  181   66]
 [  45   15   29   58   15 1621   13   92   99   13]
 [  71   31  363   29   67  274 1071   11   73   10]
 [  35  101  222  163   29   77    7 1125  213   28]
 [  59   34  169  210   76  481   57  136  750   28]
 [  12   67   65  124  183   50    5  789  339  366]]
```

8 TABULAR REPRESENTATION

8.1 Logistic Regression by varying Learning Rates

8.1.1 MNST

Learning Rate	Accuracy
0.07	0.8656
0.1	0.8799
0.15	0.8913
0.20	0.898
0.25	0.90

8.1.2 USPS

Learning Rate	Accuracy
0.07	0.264
0.1	0.2801
0.15	0.284
0.20	0.286
0.25	0.3

8.2 Neural Network by varying Batch Size

8.2.1 MNST

Batch Size	Accuracy
128	0.96
150	0.97
200	0.969
250	0.97

300	0.97

8.2.2 USPS

Batch Size	Accuracy
128	0.34
150	0.33
200	0.383
250	0.35
300	0.36

8.3 Random Forest by varying the Number of Trees

8.3.1 MNST

Number of Trees	Accuracy
250	0.9696
300	0.968
350	0.97
400	0.9704
450	0.969

8.3.2 USPS

Number of Trees	Accuracy
250	0.39
300	0.402
350	0.408
400	0.4048
450	0.407

8.4 Support Vector Machine using different Kernels

8.4.1 MNST

Kernel	Accuracy
Radio Basis Function	0.98
Sigmoid	0.33

8.4.2 USPS

Kernel	Accuracy
Radio Basis Function	0.2614
Sigmoid	0.14

9 Results

We have trained our classification models on MNIST Data-Set and have tested our trained models on MNIST Data set and USPS Data Set. Based on our observations the accuracy of trained models on MNIST testing set comes out to be in between 80-100% whereas the accuracy on USPS Data set comes out to be in between 20-40%. Hence we can conclude that training our model on one dataset might not provide good results when a new data-set is fed into our model this helps us to understand that a model cannot be trained to give good results on any Data-set. Therefore our results support the “No Free Lunch” theorem.

Based on the confusion matrix of different classifiers we observe that the accuracy on a Neural Network model is better as compared to other classifiers followed by Random Forest, Support Vector Machine and Logistic Regression.

Moreover the overall combined performance of classifiers is better as compared to the individual classifier.