CSE-574 Introduction to Machine Learning

Project 3

Apoorva Biseria
abiseria@buffalo.edu
UB person number -50291145

1. Introduction

This project deals with using Machine Learning algorithm that is Logistic Regression, Random Forest, Support Vector Machine and Neural Network to train on the MNIST dataset and tune the hyperparameters to improve accuracy. MNIST dataset consists of a 28*28 image that is it has 784 features which help us to recognize digits from 0 to 9. So, here we are dealing with a classification problem in which we have 10 classes. In Logistic regression, we are doing softmax regression, since it is a multiclass logistic regression. The test data is converted into a one hot vector which has 10 columns in each row denoting every class. One at any position denotes that it belongs to that particular class. In support vector machine, I am using particularly Linear SVM and SVM with kernel 'rbf'. A SVM model is a probabilistic model where each point is marked in a n-dimensional space where a hyperplane divides the two categories, where n is the number of categories. Also, deep Neural Network is implemented on the data using Keras on tensorflow. Here Random forest is implemented on the mnist dataset, which is a popoluar ensemble method for classification. After implementing all four methods on the mnist dataset, I made an ensemble model that works on majority voting as voting classifier, it takes votes from all the four models and chooses the class which has major number of votes in each class from the four models. Also, testing is performed on the USPS dataset which is resized to a 28*28 image.

2. Performance Metric

Since here we are performimg a classification problem, so the main metric for performance evaluation is accuracy, where accuracy is found out by comparing the result of the test data with test target every time and using the formula

$$accuracy = \frac{right}{right + wrong} * 100$$

3. Hyperparameter setting and result for different models

4.1 Logistic Regression

Hyperparameters

Learning rate = 0.03

Testing Accuracy on mnist dataset = 90.79

Confusion Matrix:

```
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]])
```

Testing Accuracy on USPS dataset = 34.94

Confusion Matrix:

```
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]])
```

4.2 Deep Neural Network on mnist dataset

Hyperparameters:
Dropout = 0.1
First dense layer = 512
Activation after first dense layer = relu
Validation data split = 0.2
Early_patience = 100
Epochs = 1500

Accuracy Testing obtained con mnist dataset= 98.21

Confusion Matrix:

```
array([[ 971,    0,    1,    0,    0,    0,    4,    1,    3,    0],
       [   0, 1126,    2,    1,    0,    0,    2,    1,    3,    0],
       [   3,    1, 1014,    1,    3,    0,    2,    5,    3,    0],
       [   0,    0,    0,  996,    0,    2,    0,    4,    4,    4],
       [   1,    0,    1,    1,  964,    0,    5,    2,    2,    6],
       [   2,    0,    0,    7,    2,  871,    3,    2,    3,    2],
       [   4,    3,    1,    1,    5,    3,  939,    0,    2,    0],
       [   1,    4,    7,    2,    1,    0,    0, 1008,    3,    2],
       [   3,    0,    4,    3,    7,    2,    2,    4,  945,    4],
       [   2,    3,    0,    2,    6,    3,    1,    4,    1,  987]])
```

Accuracy Testing obtained on USPS dataset = 42.32

Confusion Matrix:

```
array([[ 533,    0,  236,   80,  106,  215,  268,  267,   34,  261],
       [  28,  334,  550,  110,  339,  132,   23,  286,  147,   51],
       [  35,    4, 1623,   38,   26,  106,   86,   34,   45,    2],
       [  26,   11,  419, 1014,   13,  411,   25,   26,   36,   19],
       [  15,   31,  132,   10, 1072,  129,   57,  335,  153,   66],
       [  18,    2,  418,   69,    8, 1344,   77,   28,   31,    5],
       [  51,   17,  546,   16,   34,  207,  963,  112,   13,   41],
       [  16,   69,  180,  438,   52,   50,   46,  929,  213,    7],
       [ 157,    4,  242,  309,  146,  317,  220,  182,  396,   27],
       [   6,   34,  163,  241,  162,   42,   24,  799,  272,  257]])
```

4.3 Random Forest

Hyperparameters

n_estimators = 10

Accuracy obtained on mnist dataset= 94.48

Confusion Matrix:

```
array([[ 533,    0,  236,   80,  106,  215,  268,  267,   34,  261],
       [  28,  334,  550,  110,  339,  132,   23,  286,  147,   51],
       [  35,    4, 1623,   38,   26,  106,   86,   34,   45,    2],
       [  26,   11,  419, 1014,   13,  411,   25,   26,   36,   19],
       [  15,   31,  132,   10, 1072,  129,   57,  335,  153,   66],
       [  18,    2,  418,   69,    8, 1344,   77,   28,   31,    5],
       [  51,   17,  546,   16,   34,  207,  963,  112,   13,   41],
       [  16,   69,  180,  438,   52,   50,   46,  929,  213,    7],
       [ 157,    4,  242,  309,  146,  317,  220,  182,  396,   27],
       [   6,   34,  163,  241,  162,   42,   24,  799,  272,  257]])
```

Accuracy obtained on USPS dataset = 30.54

Confusion Matrix:

```
array([[ 602,   89,  363,   80,  312,  155,  103,  124,   16,  156],
       [  60,  526,  172,  184,  106,   89,   59,  755,   29,   20],
       [ 155,  122, 1033,  144,   64,  171,   64,  198,   18,   30],
       [  80,   78,  272, 1007,   49,  295,   15,  136,   19,   49],
       [  59,  219,  145,  132,  781,  189,   23,  336,   50,   66],
       [ 181,  102,  241,  246,   70,  923,   49,  121,   23,   44],
       [ 323,   99,  353,  100,  142,  355,  457,  104,   25,   42],
       [  61,  427,  392,  323,   39,  154,   19,  534,   41,   10],
       [ 123,  164,  281,  286,  157,  646,   84,   89,  128,   42],
       [  60,  312,  316,  370,  175,  154,   16,  397,   84,  116]])
```

4.4 Support Vector Machine

a) Linear Support Vector Machine

Accuracy Testing obtained on mnist data= 91.53

Confusion Matrix:

```
array([[ 958,    0,    2,    1,    0,    7,    8,    2,    2,    0],
       [   0, 1113,    4,    1,    0,    1,    4,    1,   11,    0],
       [  10,    9,  910,   22,   11,    4,   12,   10,   40,    4],
       [   5,    2,   20,  914,    3,   21,    5,   12,   19,    9],
       [   1,    4,    5,    3,  914,    0,   10,    3,    5,   37],
       [   9,    3,    1,   38,   11,  762,   20,    8,   31,    9],
       [   9,    4,    7,    2,    6,   20,  907,    1,    2,    0],
       [   2,    9,   20,    6,    6,    1,    1,  945,    5,   33],
       [   8,   13,    8,   23,   13,   38,    8,   15,  835,   13],
       [   6,    8,    2,   15,   35,   10,    0,   26,   12,  895]])
```

Accuracy Testing on USPS dataset = 32.15

Confusion Matrix:

```
array([[ 310,    1,  379,  313,   59,  175,  115,  486,   59,  103],
       [  44,  278,  664,  160,  353,   98,   27,  280,   74,   22],
       [  72,   46, 1283,  101,   47,  170,  149,   84,   21,   26],
       [  48,   38,  471,  750,   17,  484,   33,   80,   45,   34],
       [  53,   53,  190,  118,  594,  176,   73,  551,  134,   58],
       [  42,   23,  840,  217,   23,  645,   80,   95,   27,    8],
       [  99,   12,  729,  115,   55,  334,  510,   78,   15,   53],
       [ 126,   87,  231,  535,  115,  150,   25,  607,   84,   40],
       [ 175,   31,  146,  661,  123,  398,  113,  198,  115,   40],
       [  40,   51,  155,  553,  113,   73,   15,  709,  178,  113]])
```

b) Support Vector Machine with gamma = 1

Accuracy Testing obtained on mnist data = 17.59

Confusion Matrix:

```
[[   0    0    0    0    0    0    0  980    0    0]
 [   0  731    0    0    0    0    0  404    0    0]
 [   0    0    0    0    0    0    0 1032    0    0]
 [   0    0    0    0    0    0    0 1010    0    0]
 [   0    0    0    0    0    0    0  982    0    0]
 [   0    0    0    0    0    0    0  892    0    0]
 [   0    0    0    0    0    0    0  958    0    0]
 [   0    0    0    0    0    0    0 1028    0    0]
 [   0    0    0    0    0    0    0  974    0    0]
 [   0    0    0    0    0    0    0 1009    0    0]]
```

Accuracy Testing obtained on USPS data = 10.125

Confusion Matrix:

```
[[   0    0    0    0    0    0    0 2000    0    0]
 [   0    0    0    0    0    0    0 2000    0    0]
 [   0    0    0    0    0    0    0 1999    0    0]
 [   0    0    0    0    0    0    0 2000    0    0]
 [   0    0    0    0    0    0    0 2000    0    0]
 [   0    0    0    0    0    0    0 2000    0    0]
 [   0    0    0    0    0    0    0 2000    0    0]
 [   0    0    0    0    0    0    0 2000    0    0]
 [   0    0    0    0    0    0    0 2000    0    0]
 [   0    0    0    0    0    0    0 2000    0    0]]
```

c) Support Vector Machine on gamma = 'auto' (default)

Accuracy testing obtained on mnist dataset = 94.32

Confusion matrix:

```
[[ 967    0    1    0    0    5    4    1    2    0]
 [   0 1120    2    3    0    1    3    1    5    0]
 [   9    1  962    7   10    1   13   11   16    2]
 [   1    1   14  950    1   17    1   10   11    4]
 [   1    1    7    0  937    0    7    2    2   25]
 [   7    4    5   33    7  808   11    2   10    5]
 [  10    3    4    1    5   10  924    0    1    0]
 [   2   13   22    5    7    1    0  954    4   20]
 [   4    6    6   14    8   24   10    8  891    3]
 [  10    6    0   12   33    5    1   14    6  922]]
```

Accuracy Testing obtained on USPS data = 38.54

Confusion Matrix:

```
array([[ 573,    2,  428,   19,  285,  248,   73,   44,    6,  322],
       [ 110,  429,  285,  137,  273,  180,   46,  501,   22,   17],
       [ 128,   18, 1402,   59,   39,  198,   61,   57,   23,   14],
       [  76,    3,  186, 1123,   11,  483,    5,   70,   27,   16],
       [  18,   67,   91,   14, 1167,  267,   22,  194,   69,   91],
       [ 108,   17,  257,  102,   25, 1367,   60,   43,   15,    6],
       [ 197,    7,  489,   24,   98,  394,  748,   13,    7,   23],
       [  50,  225,  457,  265,   57,  416,   15,  452,   41,   22],
       [  73,   25,  209,  193,   87, 1006,   95,   41,  244,   27],
       [  26,  166,  228,  278,  213,  165,    8,  499,  214,  203]])
```

d) Support Vector Machine on Customizable parameters C= 5 and gamma = 0.05

Accuracy testing obtained mnist dataset = 98.28

Confusion matrix:

```
array([[ 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,  996,    0,    4,    0,    5,    4,    0],
       [   0,    1,    3,    0,  965,    0,    4,    0,    2,    7],
       [   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,    6,    2,    0,    6,    6,  971]])
```

4.5 Combination of all datasets

Accuracy Testing obtained on mnist dataset= 93.829

Confusion Matrix

```
array([[ 965,    0,    0,    1,    0,    2,    8,    1,    3,    0],
       [   0, 1117,    2,    3,    0,    0,    4,    1,    8,    0],
       [   8,    5,  940,   16,    6,    1,    9,   14,   30,    3],
       [   2,    0,   14,  950,    0,   17,    1,    9,   11,    6],
       [   1,    2,    3,    2,  921,    0,    8,    1,    7,   37],
       [   6,    1,    0,   31,    8,  805,   13,    4,   19,    5],
       [  10,    3,    2,    2,    8,   14,  917,    0,    2,    0],
       [   2,    9,   19,    5,    4,    0,    0,  962,    3,   24],
       [   5,    6,    6,   21,    7,   22,   10,   13,  872,   12],
       [   6,    6,    1,   11,   17,   13,    0,   18,    3,  934]])
```

Accuracy Testing obtained on USPS dataset= 37.631

Confusion Matrix:

```
array([[ 555,    4,  341,   98,  158,  173,  116,  208,   79,  268],
       [  99,  379,  364,  238,  204,   94,   23,  462,  117,   20],
       [ 103,   18, 1427,  107,   30,  111,   73,   77,   37,   16],
       [  50,    3,  266, 1222,    3,  337,   15,   51,   31,   22],
       [  35,   74,   96,   55,  960,  152,   41,  300,  209,   78],
       [  82,   12,  353,  168,   16, 1208,   60,   62,   25,   14],
       [ 187,   14,  510,   77,   54,  327,  727,   55,   19,   30],
       [ 103,  188,  230,  508,   43,  103,   29,  582,  184,   30],
       [ 192,   26,  161,  346,   89,  617,  113,  109,  302,   45],
       [  23,  142,  156,  462,  106,   76,   14,  604,  253,  164]])
```

4. Questions to be answered

1. **We test the MNIST trained models on two different test sets: the test set from MNIST and a test set from the USPS data set. Do your results support the "No Free Lunch" theorem?**

   Ans – No Free Lunch theorem states no model is perfect for all types of problem. Here the assumptions that are behind the model trained is that it is trained on the mnist dataset, and is giving much higher testing accuracy on the mnist dataset whereas gives low accuracy on the USPS dataset, thus the model is not fit for USPS. Thus, supporting No Free Lunch Theorem.

2. **Observe the confusion matrix of each classifier and describe the relative strengths/weaknesses of each classifier. Which classifier has the overall best performance?**

   Ans - The model gives much higher accuracy if confusion matrix of the classifier has higher number as the diagonals elements.
   For Logistic Regression, the model is giving high accuracy and more number of correct predictions, it is performing fast and is giving less accuracy as compared to other models.
   For Support Vector Machine : The model gives inaccurate results for rbf kernel gamma = 1 and classifies the data in only one class, whereas gives high accuracy for default gamma and customizable gamma = 0.03. Also, the model is very slow in performing the task, but gives high accuracy when appropriate hyperparameters are adjusted as compared to other models.
   For Neural Network : The model gets well trained in less number of epochs and is faster than SVM and gives very accurate results in lesser time.
   For Random Forrest : The model is less accurate than SVM and Neural Network but is performing the task pretty accurately in lesser time.

   Overall the best performance is of Neural Network, it gives >98% accuracy in lesser time as compared to SVM with customizable parameters.

3. **Combine the results of the individual classifiers using a classifier combination method such as majority voting. Is the overall combined performance better than that of any individual classifier?**

   Ans – The model is giving intermediate accuracy for combined classifer as compared to individual classifiers, it is giving better accuracy than 2 individual classifiers. NN and Svm are giving accuracy >98% and Logistic Regreesion is giving approx. 91% and Random

forest is giving accuracy approx. 94%,whereas combined model is giving accuracy of approx 94%

5. References

[1]"Matplotlib Bar chart – Python Tutorial", Pythonspot.com, 2018. [Online]. Available: https://pythonspot.com/matplotlib-bar-chart/. [Accessed: 17- Oct- 2018].

[2] "Understanding Learning Rates and How It Improves Performance in Deep Learning", Towards Data Science, 2018. [Online]. Available: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10. [Accessed: 31- Oct- 2018].