CSE 574 – Introduction to Machine Learning

Project 3.0 – Classification

Asish Kakumanu
UB Person No :50288695
asishkak@buffalo.edu

Abstract

The Abstract of the project is to develop different classifier models and evaluate the models. The Machine Learning models used to classify digits in this exercise are Logistic Regression, Multi-Layer Perceptron, Convolutional Neural Network, Deep Neural Network, Random Forest and Support Vector Machines.

1 Problem Description

The problem of classifying digits is to be solved using different classification Models.

1. **Logistic Regression.**
2. **Multi-Layer Perceptron.**
3. **CNN & DNN.**
4. **Random Forest.**
5. **Support Vector Machines.**

Input: The input is the set of features from images.

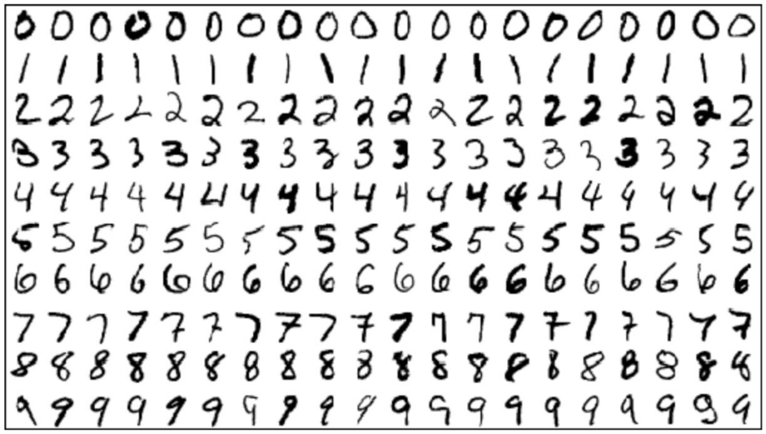
Output: A number class which is any integer between 0 to 9.

The Task is to recognize a 28 x 28 grayscale handwritten digit image and identify the class. We use MNIST dataset to train our models and also test models along with USPS Dataset which is solely used to test the models.

2 Overview of Dataset

2.1 MNIST Dataset

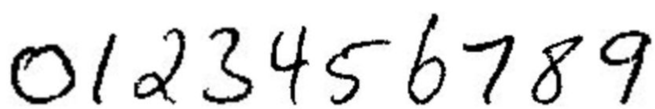
It is a collection of handwritten digits used for training and testing in various fields of machine learning and image processing. The overall size of the dataset is of 70000 images, which is divided into three parts i.e., 50000 for training, 10000 for validation, 10000 for testing. Each image has 28 x 28 viz., 784 features



The target value is of dimensions 70000 x 1 which corresponds to the labels associated with each digit.

2.2 USPS Dataset

The USPS datasets consist of images of digits in folders marked from 0 to 9. The dataset is processed and loaded using the Python imaging library (PIL). There are totally 2000 images in the collection. This dataset is solely used for testing the performance of MNIST-trained dataset on it. This test dataset is of the size 19999 x 784.



2.3 Processing of the Datasets

The Dataset are partitioned as below.

MNIST Dataset

70000 images in total

Training – 50000 (80%)

Validation – 10000 (10%)

Testing – 10000 (10%)

USPS Dataset

19999 images in total

Entire Dataset is used for testing the Classification Models we design in this project.

3 Implementation

3.1 Logistic Regression

The task of classifying 10 classes using Logistic regression can be represented by

$$p(\mathcal{C}_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{exp(a_k)}{\sum_j exp(a_j)},$$

Where a_k is the activation function $a_k = \mathbf{w}^\top\mathbf{x} + b_k$. Here w^T are the weights

$$\boldsymbol{w_k} = [w_{k,1}, .., w_{k,10}]^{\mathrm{T}}$$

Here, the cross-entropy function for the multiclass function in terms of training samples is

$$E(\mathbf{x}) = -\sum_{k=1}^{K} t_k \ln y_k,$$

Where $y_k = y_{k(x)}$. The gradient of the error function which would be,

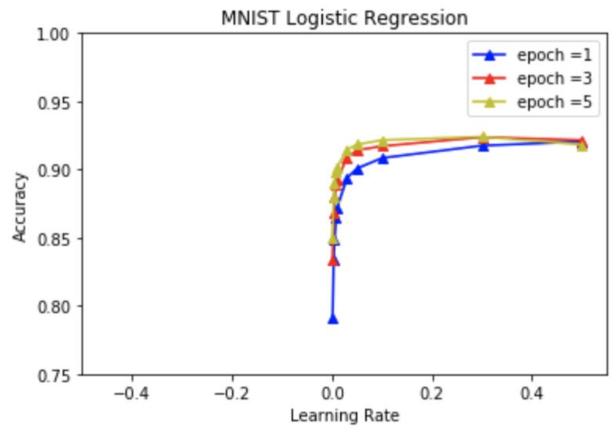
$$\nabla_{w_j} E(\mathbf{x}) = (y_j - t_j)\mathbf{x}.$$

And to find the optimum of the error function and the solution of w_j. We use the below

$$\mathbf{w}_j^{t+1} = \mathbf{w}_j^t - \eta \nabla_{w_j} E(\mathbf{x}).$$

3.1.1 Logistic Regression: Experiments

From the graph above, it is clear that accuracy increased with the increase of learning rate. During this experiment, we gradually increased the learning rate starting from 0.001 to 0.5 and also the number of epochs which increased the accuracy of the model.



<u>Testing on MNIST Dataset</u>

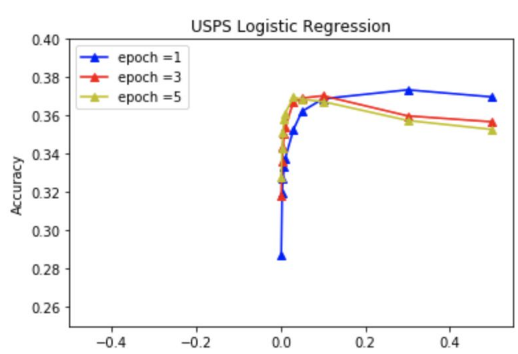
Experiment 1:
Epochs – 1
Learning Rate – 0.001
Accuracy: 0.8462

Experiment 2:
Epochs – 2
Learning Rate – 0.01
Accuracy: 0.896

Experiment 3:
Epoch – 3
Learning Rate – 0.5
Accuracy: 0.9277

<u>Testing on USPS Dataset</u>

Similarly, we tested the model using USPS Dataset and the below are the observations gathered.



Experiment 1:
Epochs – 1
Learning Rate – 0.001
Accuracy: 0.3183

Experiment 2:
Epochs – 2
Learning Rate – 0.01
Accuracy: 0.36125

Experiment 3:
Epoch – 3
Learning Rate – 0.5
Accuracy: 0.35205

We have observed a decrease of the accuracy as the learning rate and the number of epochs increased.

Accuracies on MNIST Datasets

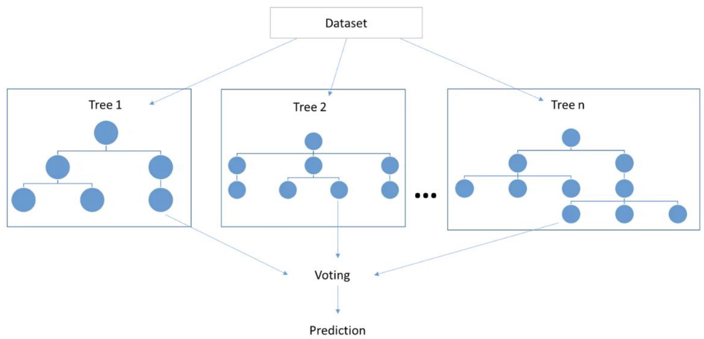
Accuracy- Training set
0.91954
Accuracy - Validation set
0.9208
Accuracy - Testing set
0.9125

3.1.2 Confusion Matrix

```
Confusion Matrix for Logistic Regression: Validation_MNIST
[[ 926    0   14    8    1   33    9   22    9   11]
 [   0 1019   22    9   16   24   16   39   47   19]
 [   7    7  828   27    3   15   16   18   19   10]
 [   5    2   15  886    0  119    1    2   58   17]
 [   1    1   16    1  842   25    7   15    3   47]
 [   2    5    1   20    1  600    7    0   13    4]
 [  24    4   30   10    9   27  902    0    8    0]
 [   4    2   22    4    2   10    0  953   14   43]
 [  19   21   32   48   16   32    9    7  807    8]
 [   3    3   10   17   93   30    0   34   31  802]]
Confusion Matrix for Logistic Regression: Testing_MNIST
[[ 938    0   18    5    3   33   21    5    9   16]
 [   0 1089   34    4   12   26   10   44   26   15]
 [   5    9  827   25    5    7   19   23   15   12]
 [   5    3   27  880    0  130    2    0   56   13]
 [   0    1   21    1  840   26   12   14   11   60]
 [   3    2    0   20    0  560   17    0   17    9]
 [  19    4   31    8   21   33  871    3   20    3]
 [   1    0   22   20    2   11    0  885   15   36]
 [   9   27   48   31   14   46    6   11  783   11]
 [   0    0    4   16   85   20    0   43   22  834]]
Confusion Matrix for Logistic Regression: Testing_USPS
[[ 766  306  327  207  191  287  626  228  312  130]
 [   7  294   47    5  102   23   16  260   53  246]
 [ 409  187 1110  147   41  262  377  381  228  185]
 [  52  216  109 1109   55  279   85  336  204  397]
 [ 368  266   76   55 1079   64  122   67  195  208]
 [  24   26   30  155   69  714   67   66  323   38]
 [  70   43  102   50   26  138  604   56  144   19]
 [  35  331  100   79  135   87   23  282   42  361]
 [  68  315   84  131  219  107   55  289  428  313]
 [ 201   16   14   62   83   39   25   35   71  103]]
```

3.2 Random Forest

Random forests are an ensemble learning method for classification, regression tasks, that operate by constructing a multitude of decision trees at training time and output the class that is the mode of the classes or mean prediction of the individual trees.



We have used a library function from Sklearn to solve this problem. The parameters used in this problem are ***n_estimators, criterion, random_state*** and ***max_depth, verbose*** etc.,

By keeping the below constant
verbose = 0,
random_state = None,
max_features = auto and
max_depth = None

We change a couple of parameters like ***n_estimator*** and ***n_jobs***.

By keeping everything constant, we changed parameters like ***n_estimator*** and ***n_jobs, criterion.***

3.2.1 Random Forest: Experiments

Experiment 1:

n_estimators = 10
n_jobs = 2
criterion: 'entropy'

Validation Results :: MNIST Dataset...

Precision : 0.9525066177792418

Recall : 0.9525
F1 Score : 0.9523923341326412
Accuracy : 0.9525
Mean Accuracy : 0.9525

Testing Results :: MNIST Dataset ...

Precision : 0.9466675310394334
Recall : 0.9466
F1 Score : 0.9464681342532542
Accuracy : 0.9466
Mean Accuracy : 0.9466

Testing Results :: USPS Dataset ...

Precision : 0.31865230768854325
Recall : 0.30591529576478826
F1 Score : 0.2874935094293345
Accuracy : 0.30591529576478826
Mean Accuracy : 0.30591529576478826

Experiment 2:

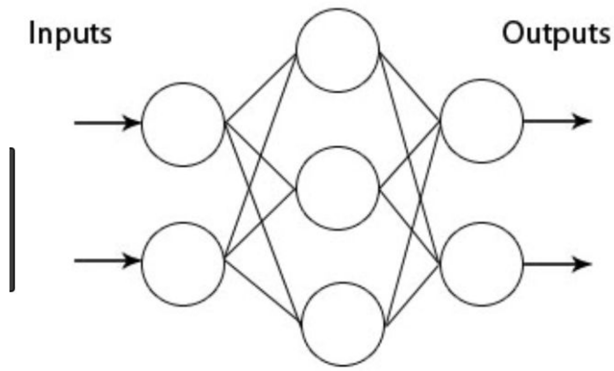
By changing the **n_estimators** to 100. We have observed the accuracy to be 0.97 (approx) for the **MNIST Dataset**.
0.34 for **USPS Dataset**.

3.2.2 Confusion Matrix

```
Confusion Matrix for Random Forest: Validation_MNIST
[[ 978    0    5    2    1    5    1    0    2    5]
 [    0 1050    1    0    5    1    0    5    4    3]
 [    4    5  962    3    0    4    0    7    5    3]
 [    0    1    0  996    0   15    0    2    4    8]
 [    0    1    2    0  954    2    2    4    2    8]
 [    0    2    1   11    1  867    3    0    6    6]
 [    2    1    2    1    2   12  958    0    3    0]
 [    0    1    8    5    1    1    0 1061    1    8]
 [    5    3    6    7    1    5    3    0  974    6]
 [    2    0    3    5   18    3    0   11    8  914]]
Confusion Matrix for Random Forest: Testing_MNIST
[[ 968    0    7    0    1    2    7    1    4    6]
 [    0 1124    0    0    0    1    3    6    0    5]
 [    0    2  997   10    1    1    1   19    5    0]
 [    0    3    4  973    0   12    0    2    7    9]
 [    0    0    4    0  952    4    3    2    5   11]
 [    3    2    0    7    0  859    2    0    6    4]
 [    4    2    4    0    5    4  937    0    5    1]
 [    1    0    9   10    0    1    0  985    5    5]
 [    3    1    7    8    2    6    5    2  929    8]
 [    1    1    0    2   21    2    0   11    8  960]]
Confusion Matrix for Random Forest: Testing_USPS
[[ 621   22   73   31    7  131  306   28   50   18]
 [  12  597   47   10  211   36   75  356   70  297]
 [ 230   85 1217   82   55  125  197  335  162  211]
 [  48   98   73 1265   24   83   25  226  193  279]
 [ 471   54   45   62 1068   34   96   30  116  253]
 [ 142   79  164  282  148 1395  327  241 1075  112]
 [  60   18   19    1   13   18  781   26   64   13]
 [ 167 1033  356  242  444  166  183  748  118  680]
 [   1   13    2    6   13    8    1    3  129   59]
 [ 248    1    3   19   17    4    9    7   23   78]]
```

3.3 Multi Layer Perceptron

A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of, at least, three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function.



The Parameters that we use in this model are as following
hidden_layer_sizes=(100,)
max_iter=50,
alpha=1e-4,
solver='sgd',
verbose=10,
tol=1e-4,
random_state=1,
learning_rate_init=.1

We have kept few as constant while changing the ***solver, hidden layer sizes, max_iterations, activation and learning rate.***

Solvers used are **sgd, adam.**
Activation functions used are **tanh, relu.**
Learning Rates used are **Adaptive** and **Constant.**
Alpha to be **0.01** and **0.05.**

3.3.1 Multi Layer Perceptron

Best Settings:

Solver : sgd
Activation Function : relu
Learning Rate : Adaptive

Alpha: 0.05
Init_learning Rate: 0.1

Validation :: MNIST

	precision	recall	f1-score	support
0	0.99	0.99	0.99	991
1	0.99	0.99	0.99	1064
2	0.99	0.98	0.98	990
3	0.97	0.98	0.98	1030
4	0.99	0.97	0.98	983
5	0.98	0.96	0.97	915
6	0.98	0.99	0.99	967
7	0.98	0.99	0.99	1090
8	0.97	0.98	0.97	1009
9	0.96	0.97	0.97	961
avg / total	0.98	0.98	0.98	10000

Accuracy : **0.9804**

Testing :: MNIST

	precision	recall	f1-score	support
0	0.99	0.99	0.99	980
1	0.99	0.99	0.99	1135
2	0.98	0.97	0.98	1032
3	0.97	0.98	0.97	1010
4	0.98	0.98	0.98	982
5	0.99	0.97	0.98	892
6	0.98	0.98	0.98	958
7	0.98	0.98	0.98	1028
8	0.97	0.98	0.97	974
9	0.98	0.97	0.97	1009
avg / total	0.98	0.98	0.98	10000

Accuracy : **0.9792**

Testing :: USPS

	precision	recall	f1-score	support
0	0.55	0.27	0.37	2000
1	0.62	0.24	0.35	2000
2	0.48	0.77	0.59	1999
3	0.52	0.73	0.61	2000
4	0.51	0.53	0.52	2000
5	0.58	0.70	0.63	2000
6	0.78	0.53	0.63	2000
7	0.32	0.45	0.37	2000
8	0.34	0.40	0.37	2000

9	0.23	0.14	0.17	2000
avg / total	0.49	0.48	0.46	19999

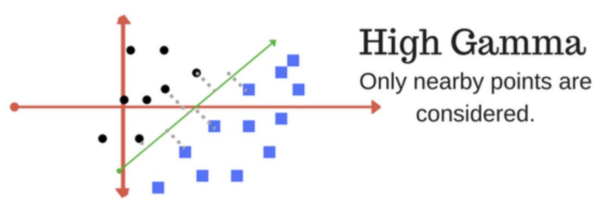
Accuracy : **0.47657382869143455**

3.4 SVM (Support Vector Machine)

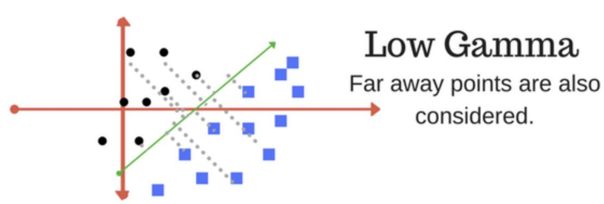
Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.
We have used a library function from Sklearn to solve this problem. The Parameters we use to tune the model are C, kernel, gamma.

Gamma

When Gamma is high, Nearby points are considered. When it is low, Farther points are also considered.



High Gamma



Low Gamma

C (Regularization Parameter)

When C is high, it accepts zero tolerance. When C is low, it accepts tolerance.

Left: low regularization value, right: high regularization value

Kernel

The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role.

Linear Kernel

For linear kernel the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

$$f(x) \;=\; B(0) \;+ Sum(a_i * (x, x_i))$$

Radial Basis Function Kernel

The RBF kernel on two samples x and x', represented as feature vectors in some input space, is defined as

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

3.4.1 Experiments

Best Parameters

C : 5
Kernel : Linear
Gamma : 0.01

Validation :: MNIST

Accuracy : 0.9828

Testing :: MNIST

Accuracy : 0.982

Testing :: USPS

Accuracy : 0.41527076353817693

3.4.2 Confusion Matrix

```
Confusion Matrix for SVM: Validation_MNIST
[[ 972    0    5    1    1    5    2    2    3    5]
 [    0 1049    3    3    7    5    0   10   25    6]
 [    3    3  929   13    4    8    5   13    9    4]
 [    2    2    5  949    0   27    0    3   16   14]
 [    1    1   10    1  944    6    3   11    4   21]
 [    0    2    3   34    0  830    4    0   20    4]
 [    8    0   12    3    3   20  950    0    4    0]
 [    1    1    6    3    2    2    0 1034    6   21]
 [    3    4   11   17    3   11    3    1  911    6]
 [    1    2    6    6   19    1    0   16   11  880]]
Confusion Matrix for SVM: Testing_MNIST
[[ 967    0    9    1    1    7   10    2    4   10]
 [    0 1120    1    1    1    4    3   13    6    6]
 [    1    2  962   14    7    5    4   22    6    0]
 [    0    3    7  950    0   33    1    5   14   12]
 [    0    0   10    1  937    7    5    7    8   33]
 [    5    1    1   17    0  808   10    1   24    5]
 [    4    3   13    1    7   11  924    0   10    1]
 [    1    1   11   10    2    2    0  954    8   14]
 [    2    5   16   11    2   10    1    4  891    6]
 [    0    0    2    4   25    5    0   20    3  922]]
Confusion Matrix for SVM: Testing_USPS
[[ 573  110  128   76   18  108  197   50   73   26]
 [    2  429   18    3   67   17    7  225   25  166]
 [ 428  285 1402  186   91  257  489  457  209  228]
 [  19  137   59 1123   14  102   24  265  193  278]
 [ 285  273   39   11 1167   25   98   57   87  213]
 [ 248  180  198  483  267 1367  394  416 1006  165]
 [  73   46   61    5   22   60  748   15   95    8]
 [  44  501   57   70  194   43   13  452   41  499]
 [    6   22   23   27   69   15    7   41  244  214]
 [ 322   17   14   16   91    6   23   22   27  203]]
```

4 Conclusion

As per the observations made on the classifiers post tuning the parameters for each model. The results observed were as below

Model	USPS Testing Accuracy	MNIST Test Accuracy
CNN	0.49	0.9885
SVM	0.415	0.982
DNN	0.47	0.9875
MLP	0.47	0.97
Random Forest	0.305	0.94
Logistic	0.307	0.91

4.1.1 Questions

4.1.1.1 No Free Lunch Theorem

No Free lunch theorem states that there is no universal model/ algorithm which fits perfectly to all the problems. By looking at the accuracy results for each model clearly explains the No Free Lunch Theorem.

4.1.2 Confusion Matrix

```
Confusion Matrix for USPS after majority voting
[[ 683  129  185  107   49  171  416  114  155   59]
 [    6  468   32    3  120   22   19  287   43  234]
 [ 372  230 1347  157   65  231  410  416  232  199]
 [   40  179   82 1347   36  168   50  344  225  387]
 [ 367  245   54   24 1178   46  111   51  130  190]
 [ 128  123  114  244  157 1226  233  231  797   96]
 [   52   35   51    6   14   50  708   22   89   10]
 [   52  520  100   74  194   63   19  444   49  482]
 [   16   66   24   27  107   17   16   76  250  215]
 [ 284    5   10   11   80    6   18   15   30  128]]
```

4.1.3 Majority Voting

Post the Majority Voting, the accuracy is 0.55. Which is very much less than the individual models.