Classification

Gagan Suneja
School of Management
University at Buffalo
Buffalo, NY 14260
gagansun@buffalo.edu

Abstract

The report presents implementation of machine learning models for a given classification task of recognizing handwritten images and classifying it into 10 number classes among 0,1,2….9. The classifiers required in this project are – Multinomial Logistic Regression, A publically available multilayer perceptron neural network, publically available Random Forest package and a publically available SVM package. The machine learning models are trained on MNIST data and are test on test sets from MNIST data and USPS data set.

1 Introduction

In supervised learning, a task of classification involves training a machine learning model to identify to which set of classes a data point belongs. For example, training a model to identify which email is Spam and which is Not Spam. Another example involves identifying from a given set of images, which of them is a car, truck, a bike or a bus. For this, first model is trained on existing data with the target result already known. When there are only two classes to choose upon, it is called as binary classification (for eg- Spam, Not Spam). However, when we have more than one classes to choose upon, it is called as multiclass classification. For the given task of recognizing handwritten images and classification, we use multinomial classification as we need to classify it among 10 possible values- 0,1,2,3,4,5,6,7,8,9.

Multinomial Logistic Regression

Multinomial Logistic Regression finds its use in cases when there are more than 2 classes to identify. It basically generalises logistic regression to the case of more than two classes. The basic difference between logistic regression and multinomial logistic regression is that the latter involves use of softmax and one-hot encoding. For each data row, softmax function computes the probability for each of the classes summing up to 1. For each class, it is given as =

$$p(C_k|x) = y_k(x) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where k represents the class. The denominator term is the summation of exp(a) for each of the classes for a particular data point. Further the algorithm involves computing one-hot encoding which is nothing but converting the actual set of target values to k – column representation of 0s and 1 with 1 at the class which the target value belongs to and the rest 0s. For example, for a target value 4, its equivalent one-hot encoded output will be [0 0 0 0 1 0 0 0 0 0]. Similarly, [1 0 0 0 0 0 0 0 0 0] for 0, [0 0 0 0 0 0 0 0 1 0] for 8 and etc.

For each of the epochs, weights are updated as follows-

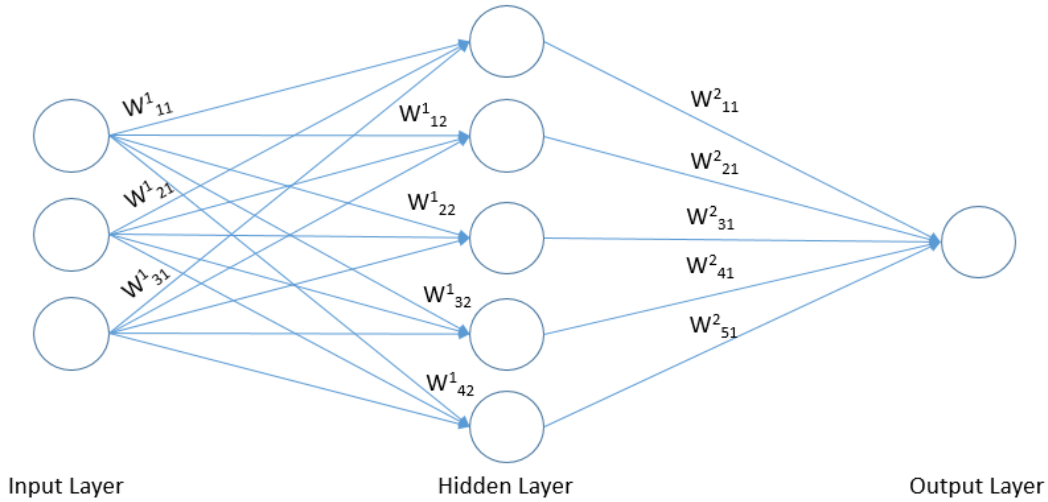
$$w_j^{t+1} = w_j^t - \eta \nabla_{\mathrm{w_j}} E(x)$$

where E(x) is the Error function and $\nabla_{\mathrm{w_j}} E(x)$ represents its gradient.

Neural Networks

Artificial Neural Networks(ANNs) are an important class of machine learning algorithms that exhibit behavior analogous to a human brain. A simple neural network consists of 3 components –

Input Layer, *Hidden Layer* and *Output Layer*. There can be multiple hidden layers in the network and each hidden layer comprises of nodes which are called neurons. A neuron takes input from previous layer, processes it with a weight W and an activation function A and passes the output to next layer.



Support Vector Machine

Support Vector Machine is a machine learning model used in supervised learning for the task of classification and regression analysis. It is a discriminative model that categorizes data among classes using hyperplane. It is analogous to a line dividing a plane in a 2-D space, where each class data lies. SVMs can be optimized by fine-tuning parameters like Kernel, Regularization, Gamma and Margin.

Kernel – The Kernel provides some linear algebra which is further used in learning the hyperplane. It can be of types like linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid.

Regularization (C) – The Regularization parameter helps the model avoid misclassification of each of the training example. Higher the value of C, lower is the margin of the hyperplane and vice versa.

Gamma – The Gamma value decides how far a single training data example can influence. High value means 'close' meaning only close data points to the hyperplane are considered for calculation whereas a low value denotes that far data points are also taken into consideration in calculation.

Margin – It specifies the distance of the hyperplane from the closest data points. A good margin will have high margin for either of the categorized data.

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. Random decision forests correct for decision trees' habit of overfitting to their training set.

Dataset

The dataset used in the project are the images in the MNIST and USPS dataset. The machine learning models need to be trained on MNIST dataset and test on both the MNIST and USPS dataset.

2 Implementation

The project has been implemented using 5 methods, namely multinomial logistic regression, neural networks, SVM and Random Forest and an ensemble of all 4.

Data Pre-Processing

This step involves preparing the data for use as training input and test data in the machine learning models

a. MNIST data - Segregate the training, validation and test data set from mnist.pkl.gz file
b. USPS data – Resize the png images to 28 x 28 size and further to single row vector of 784 values

Multinomial Logistic Regression Implementation

Multinomial Logistic Regression has been implemented as per below pseudocode

1. Train Dataset and Compute Weights
   a. Add Bias to the training Data Set
   b. Compute T = One-Hot-Encoder
   c. Compute Weights
      For each epoch (till 500),
      i. Compute A($WX + b$)
      ii. Compute Y = Softmax(A)
      iii. Compute YminusT = Y – T
      iv. Compute Gradient of Error Function
      $$\nabla_{\mathrm{w_j}} E(x) = X^T(Y - T)$$
      v. Update Weights
      $$w_j^{t+1} = w_j^t - \eta \nabla_{\mathrm{w_j}} E(x)$$
2. Compute Multiplication of input dataset and Weights(computed from Step 1) with Regularizer if any
   $$A = WX + b$$
   A = Activation, W = Weights, X = Input Data, b = regularization
3. Compute Softmax of the result from Step 2:
   $$Y = Softmax(A)$$
   Y is the probability vector computed for each of the classes(k=10) for each of the data point.
4. For each of the probability factor for each data point, choose the class which has the maximum probability.

Neural Network Implementation

For implementing Neural Networks, keras library has been used. The hyper parameters have been fine tuned to obtain best performance. The first layer is the input layer. The second layer has 32 nodes with sigmoid function applied. The third and the last layer has softmax operation applied. The optimizer used in the model is sgd with loss as categorical cross entropy.

Support Vector Machines

This machine learning Model has been implemented using the SciKit learn python library. The Classifier object parameters have been set as kernel = ‘linear’, C(Regularizer) = 2, gamma = 0.05

Random Forest

For Random Forest, SciKit learn python library has been used with varying number of estimators starting from 10.

Confusion Matrix

Confusion Matrix, which is also called as the error matrix, facilitates visualization of the performance of the machine learning model. Each of the rows represent the number of predicted values corresponding to the actual column values.

3 Analysis, Inference and Conclusion

The Analysis has been done for all the machine learning models and with the ensemble.

Multinomial Logistic Regression

The model achieved maximum performance with Weights trained at epoch=500 with Accuracy of 92.06 on MNIST Test data and 34.28 on USPS Data (Table 3).

Below is the relation shown between the number of Epochs and Accuracy(Fig1). We can see that as the number of Epoch increase, the Accuracy increases. Further, as we reach an accuracy of 90%, it starts increasing slowly.

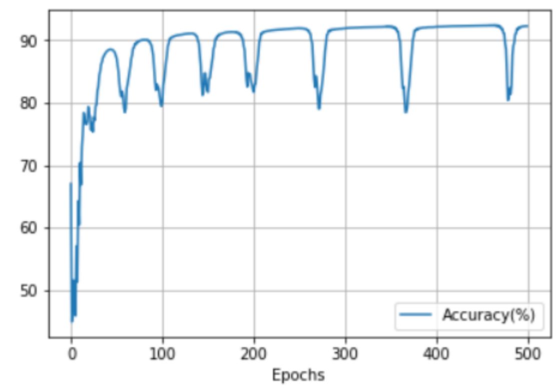


Fig 1: Accuracy vs # Epochs

```
CONFUSION MATRIX for MNIST Test Data
        Logistic Regression
   Rows    -----> Predicted Values
   Columns -----> Actual Values
[[ 957    0    6    2    1    9   11    3    6   11]
 [   0 1110    8    0    3    3    3    7    6    6]
 [   3    3  924   23    5    5    6   22    7    2]
 [   1    2   14  914    1   34    1    9   27   11]
 [   0    0   10    0  910    9   10    6   10   33]
 [   5    2    5   31    0  771   12    1   27    9]
 [  10    4   11    2   11   18  911    0   11    0]
 [   1    2   11   11    3    8    2  949   12   29]
 [   3   12   34   17    7   28    2    1  859    7]
 [   0    0    9   10   41    7    0   30    9  901]]
```

```
CONFUSION MATRIX for USPS Data
      Logistic Regression
 Rows    -----> Predicted Values
 Columns -----> Actual Values
[[ 490   71   93   49   41   73  130  108  159   19]
 [   1  325   12    3   43   11    6  130   18   72]
 [ 221  259 1296  215   53  226  567  192  158  120]
 [  93  190  127 1056   38  158   74  556  314  521]
 [ 143  188   29    7  858   23   56   51   84  107]
 [ 365  208  265  533  226 1329  488  257  793  127]
 [  58   14   51    6   29   54  597    8  105   14]
 [ 169  597   59   72  332   71   19  472   79  538]
 [ 103  116   37   31  205   39   15  159  226  275]
 [ 357   32   30   28  175   16   48   67   64  207]]
```

Fig 2: Confusion Matrix for Logistic Regression

From the Fig2 and Table 3, it can be inferred that it is not necessary that the model trained on a known dataset (MNIST in this case) will perform good on an unknown dataset(USPS in this case as accuracy in USPS is 34.28). Hence, No free Lunch Theorem is supported here..

Neural Networks

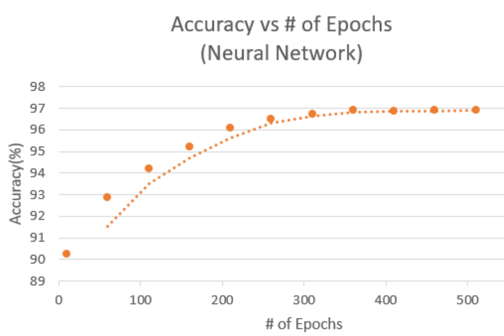
Neural Networks has been implemented using Keras library in python with one hidden layer. The hidden layer comprises of 32 nodes and have sigmoid function applied on it. The last layer has 10 nodes which is the number of classes and has softmax function applied on it. The optimizer used for the model is 'sgd' (Stochastic Gradient Descent). The maximum accuracy observed is 95.53% for MNIST dataset and 40.10% for USPS dataset(Table 3).

Table 1: Observations for Loss and Accuracy

with increasing the number of Epochs

Epoch	Loss	Accuracy
10	0.3806	90.22
60	0.2585	92.84
110	0.1992	94.18
160	0.1613	95.21
210	0.1379	96.08
260	0.1236	96.50
310	0.1152	96.74
360	0.1106	96.89
410	0.1087	96.86
460	0.1086	96.90
510	0.1101	96.93

From the table and the below Figure, we can see that as the as the number of Epoch increase, the Accuracy increases and the Loss decreases.



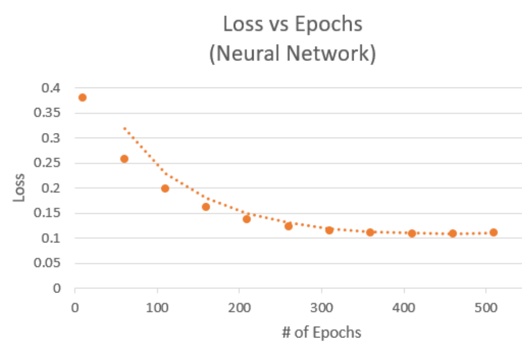


Fig 3: Relation representing Accuracy v/s Epochs, and

Loss v/s Epochs

```
          CONFUSION MATRIX for MNIST Test Data
                    Neural Network
               Rows    -----> Predicted Values
               Columns -----> Actual Values
[[ 962    0    7    2    1    7    9    1    9    7]
 [   0 1120    2    1    0    2    2    7    2    6]
 [   1    3  989   11    5    2    2   14    1    0]
 [   1    1    5  955    0   22    2    7   16   11]
 [   0    1    6    0  939    3    6    4    3   18]
 [   3    1    3   15    2  824    7    0    9    7]
 [   6    3    6    1    6   13  922    0    6    2]
 [   5    2    6   12    4    2    2  979    6   11]
 [   2    4    6   11    3   16    6    2  919    3]
 [   0    0    2    2   22    1    0   14    3  944]]
```

```
          CONFUSION MATRIX for USPS Data
                    Neural Network
               Rows    -----> Predicted Values
               Columns -----> Actual Values
[[ 477   61  116   51   17   82  189   37  124   11]
 [   0  280    8    0   10    4    6   83    5   24]
 [ 154  301 1374  124   43  198  489  163  122   85]
 [ 114  107  159 1380   53  181   77  637  371  429]
 [ 174  188   38    1  953   16   55   24   75   87]
 [ 204  154  137  315  164 1296  263  201  595   74]
 [  53   32   48    3   25   43  799    5   78    8]
 [ 180  691   46   48  385   76   30  684   74  666]
 [ 166  102   58   62  191   90   44  131  499  337]
 [ 478   84   15   16  159   14   48   35   57  279]]
```

Fig 4: Confusion Matrix for Neural Network

From table 3 and Fig 4, it can be seen that the model does not necessarily perform good on an unknown dataset (USPS in this case). Hence, No free lunch theorem is supported.

Support Vector Machines

Support Vector Machines has been implemented using SciKit Learn library. With linear kernel, Regularization(C) = 2 and Gamma = 0.05, maximum accuracy of 93.81% for MNIST dataset and 28.45% for USPS dataset has been observed.

```
      CONFUSION MATRIX for MNIST Test Data
                 SVM
           Rows    -----> Predicted Values
           Columns -----> Actual Values
[[ 956    0    5    4    1   11    6    2    8    7]
 [   0 1117    7    0    1    5    3    8    5    7]
 [   4    5  969   15    9    5   14   21    6    3]
 [   1    3   11  950    0   37    2   12   28   13]
 [   1    0    2    1  943    6    6    7    7   32]
 [   8    1    4   19    2  794   18    2   26    5]
 [   7    2    7    1    5   12  906    0    8    1]
 [   1    1    9    7    1    2    1  956    6   18]
 [   0    6   16   11    2   15    2    2  872    5]
 [   2    0    2    2   18    5    0   18    8  918]]
```

```
      CONFUSION MATRIX for USPS Data
                 SVM
           Rows    -----> Predicted Values
           Columns -----> Actual Values
[[ 360   46  118   50   23   37  144   27  115   11]
 [   1  294   70   60   23   17   20   75   14   31]
 [ 528  560 1269  390  249  699  853  218  360  212]
 [ 168  205  104  795   98  218   63  639  466  608]
 [ 156  253   35    9  765   29   68   49   90  122]
 [ 321  204  261  588  216  892  338  274  655  100]
 [  77   20   67    8   19   30  443   11   65    4]
 [ 194  358   40   53  456   32   32  575   65  606]
 [   8   44   22   34   87   33    2  102  142  151]
 [ 187   16   13   13   64   13   37   30   28  155]]
```

Fig 5: Confusion Matrix for Support Vector Machines

Fig 5 and Table 5 clearly demonstrate that No Free Lunch Theorem is supported here as the model performs very good in the data set in which it is trained on (MNIST here) but fails to perform good on an completely unknown dataset (USPS)

Random Forest

Random Forest has been implemented using SciKitlearn python library. With the No. of Estimators at 100, maximum accuracy of 97.07% was observed. From Table 2 and Fig6, it can be observed that as the No. of Esimators are increased, the accuracy also increases.

Table 2: Observations of Accuracy with increasing number of Epochs

# Estimators	Accuracy(%)
10	94.89
20	95.9
30	96.44
40	96.55
50	96.71
60	96.9
70	96.93
80	96.84
90	96.98
100	97.07

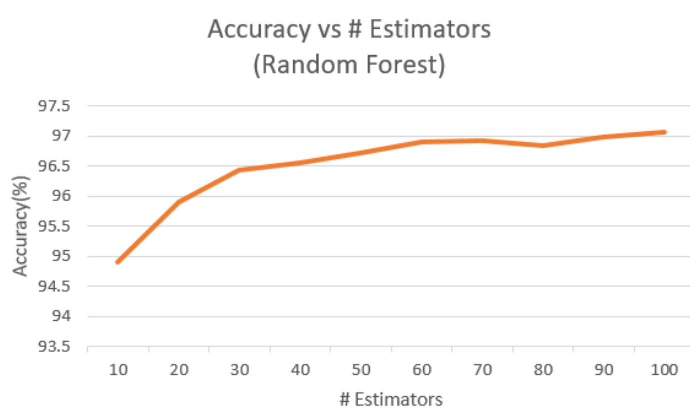


Fig 6: Relationship between Accuracy and No. of Estimators

```
CONFUSION MATRIX for MNIST Test Data
           Random Forest
      Rows    -----> Predicted Values
      Columns -----> Actual Values
[[ 971    0    6    0    1    2    7    2    3    5]
 [   0 1123    0    0    0    2    3    2    0    5]
 [   1    3  999   11    1    1    1   21    6    2]
 [   0    3    5  974    0   12    0    1   13   10]
 [   0    0    3    0  957    3    5    1    3   10]
 [   1    2    0    7    0  855    3    0    7    5]
 [   4    2    4    0    4    5  935    0    3    1]
 [   1    0    9    8    1    2    0  991    3    7]
 [   2    1    6    7    3    7    4    2  926    6]
 [   0    1    0    3   15    3    0    8   10  958]]
```

```
CONFUSION MATRIX for USPS Data
           Random Forest
      Rows    -----> Predicted Values
      Columns -----> Actual Values
[[ 631   49   83   39   12  132  293   42   54   20]
 [  13  579   24    4  202   30   48  342   55  273]
 [ 282  145 1296  113   67  149  251  418  173  251]
 [  51   94   86 1234   32   88   35  233  218  282]
 [ 457   60   50   52 1091   33   86   40  100  252]
 [ 134   87  184  374  147 1404  346  191 1037  132]
 [  84   24   22    3   21   37  807   35   67   16]
 [ 115  946  241  158  381  110  113  685  106  592]
 [   1   13    8    6   32    8    9    3  160   86]
 [ 232    3    5   17   15    9   12   11   30   96]]
```

Fig 7: Confusion Matrix for Random Forest

From Fig 7 and Table 3, it is evident that No Free Lunch Theorem is supported as Random Forest fails to perform good on a dataset on which it is not trained (USPS)

Ensemble

From Fig 8 and Table 3, it can be seen that a maximum accuracy of 95.45% (for MNSIT data) and 37.59%(for USPS data) is achieved. It has been implemented based on maximum voting. If there is a tie, it picks up first maximum observed value.

```
CONFUSION MATRIX for MNIST Test Data
           Ensemble
      Rows    -----> Predicted Values
      Columns -----> Actual Values
[[ 970    0    8    2    1    8    8    2    6    9]
 [   0 1124    3    0    0    3    3    6    2    6]
 [   1    2  990   17    5    1    7   22    4    2]
 [   1    2    5  963    0   29    1    7   22   14]
 [   0    0    4    0  955    6    7    3    5   25]
 [   2    1    1   11    0  822   11    1   17    5]
 [   4    2    4    0    3   10  917    0    8    0]
 [   1    1    9    9    0    1    2  975    7   16]
 [   1    3    7    7    3   10    2    1  899    2]
 [   0    0    1    1   15    2    0   11    4  930]]
```

```
CONFUSION MATRIX for USPS Data
           Ensemble
      Rows    -----> Predicted Values
      Columns -----> Actual Values
[[ 567   71  117   51   23   86  203   59  143   17]
 [   4  389   20    8   60   11   13  161   21   92]
 [ 323  312 1437  233  102  292  618  242  208  169]
 [ 100  163  106 1244   46  171   68  584  380  515]
 [ 223  181   28    2 1004   17   58   34   76  109]
 [ 233  177  169  382  197 1302  344  233  788   99]
 [  45   18   34    2   14   28  629    6   59    8]
 [ 142  627   57   44  334   58   27  564   72  594]
 [  49   48   18   26  131   25    7   94  220  235]
 [ 314   14   13    8   89   10   33   23   33  162]]
```

Fig 8: Confusion Matrix for Ensemble

Table 3 : Maximum Accuracy Observed for all Machine Learning Models

For both the datasets

Machine Learning Model	MNIST Test Data	USPS Data
Logistic Regression	92.06	34.28
Neural Network	95.53	40.1
SVM	93.81	28.45
Random Forest	96.89	39.91
Ensemble	95.45	37.59