Project 3: Classification

Sarang Agarwal
Graduate Student, Department of Computer Science
University at Buffalo
sarangag@buffalo.edu

Abstract

The basic task here is of classification. We are working on training 4 different types of classifiers name Neural Network, Random Forest, Support Vector Machine and Logistic Regression Classifier.

We have a 28 by 28 greyscale image of digits from 0 to 9. Our task is to classify each one of them to a class with values ranging from 0 to 9.
Towards the end, we are also classifying them according to Majority Voting Scheme in which we will take the majority output of all the 4 classifiers and will take that value as to be our predicted output.

1 Introduction to Different Data Set Used

We have 2 data sets in our problem.

MNIST Dataset: We have 70,000 data images which are 28 by 28 greyscale images. We have converted them into 50,000 by 784 training data, 10,000 by 784 validation data and 10,000 by 784 testing data. All these images are from range 0 to 9 representing a digit.

USPS Dataset: This is USPS hand written digit that are 20,000 by 784. These will be mainly used for testing purpose on the classifiers that we train on MNIST data. This will help us in validating the 'No Free Lunch Theorem'.

2 Code Structure

Code structure is simple with just one file Main.py which has all the code systematically arranged which will download required data set, pre-process them and run Logistic Regression followed by Neural network, SVM and Random forest. All the required code comments are included in that file itself.2

3 Support Vector Machine

Here we have used the library from SK- Learn where we have used 3 configurations:

1. Using linear kernel

So the kernel which I have used first is the linear kernel. As expected, it did not give as great accuracy as RBF kernel gave. I got an accuracy of about 90% with this seting.

Setting

Classifier = SVM
Kernel = Linear
Other setting = Default.

Accuracy on MNIST = 90%
Accuracy on USPS = 32%

2. Radial Basis Function Kernel with Gamma Default

Setting

Kernel = Rbf
Gamma= Auto/Default
Other values= Default

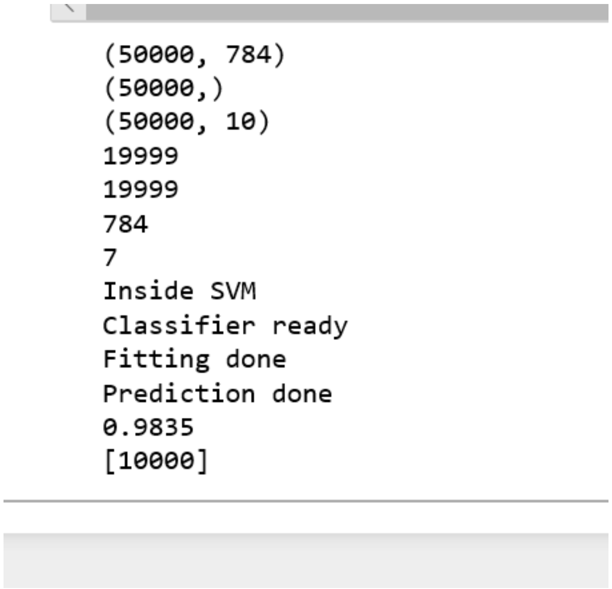
Accuracy = 94%
Accuracy on USPS = 37%

3. Radial Basis Function Kernel with Gamma One

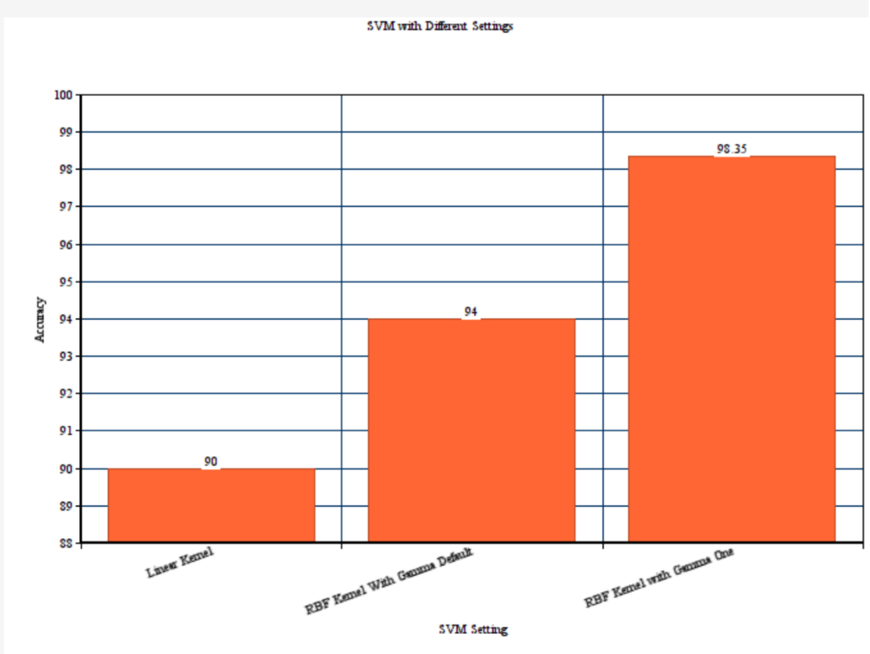
Setting

Kernel = Rbf
Gamma= 1
Other Values = Default

Accuracy = 98.35% on MNIST
Accuracy on USPS = 42%



```
(50000, 784)
(50000,)
(50000, 10)
19999
19999
784
7
Inside SVM
Classifier ready
Fitting done
Prediction done
0.9835
[10000]
```



Comparison of Different SVM Setting

4 Logistic Regression

Logistic Regression establishes the relationship between Input and Output variable as a function of Natural Logarithm and find the most optimal weights or coefficients to model the data.

We use Softmax function as classifier. Earlier we have used Sigmoidal as activation function but now we have 10 classes into which we want to classify our output to.

It is a K Class Classifier.

$$P(y=j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^{\mathsf{T}}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^{\mathsf{T}}\mathbf{w}_k}}$$

Source: Wikipedia

Therefore, we use softmax.
Softmax internally uses one versus all type of classification and will give the class prediction probabilities of a particular dataset to be from one of 10 classes.
Then we can use Hot Vector representation and get the predicted value as the class with the highest probability.

HOT Vector Representation

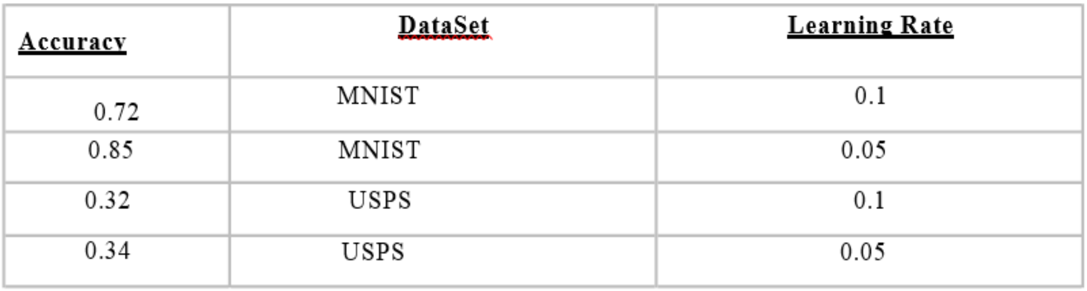
Suppose we have 10 classes, then hot vector will be 0’s at all the places other than the actual digit and 1 at the digit’s position.

Not explaining Logistic regression as such in detail since it was already done in the last project.

Setting

Learning Rate = 0.05
Iteration = 1000
Lambda = 1

Accuracy = 85% on MNIST
Accuracy = 32% on USPS



Accuracy	DataSet	Learning Rate
0.72	MNIST	0.1
0.85	MNIST	0.05
0.32	USPS	0.1
0.34	USPS	0.05

So this was how we get around Logistic regression. However ,I tried improving the accuracy to 90%, I could only get till 0.85. Maybe for very high number of iterations it could be achieved but unfortunately my Laptop was taking a lot of time to run such a high number of iterations.

5 Neural Network

I am not giving too much detail about Neural Network as we did the same thing in Project 1.2.

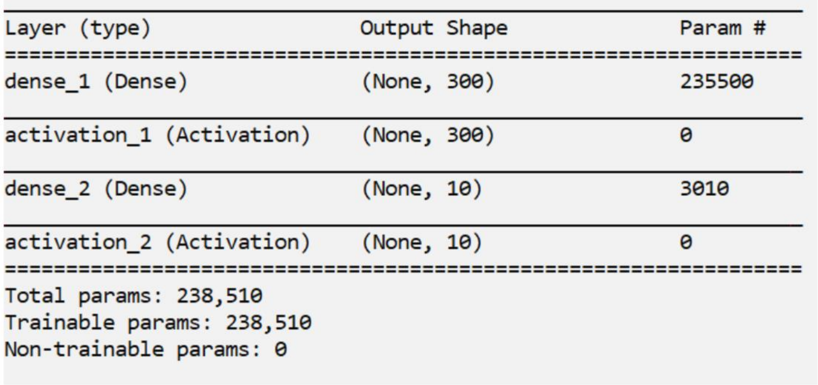
I am basically using:

-> Sequential Model
-> 3 Layered model
-> Relu as activation function in Input layers and Softmax function in Output layer to get 10 classed Output.

Loss function is categorical_crossentropy as this is the best suited.

Setting

Layers = 3
Dropout = 0
Epochs = 1200
Batch Size = 4000
Accuracy on MNIST = 97%
Accuracy on USPS = 41%



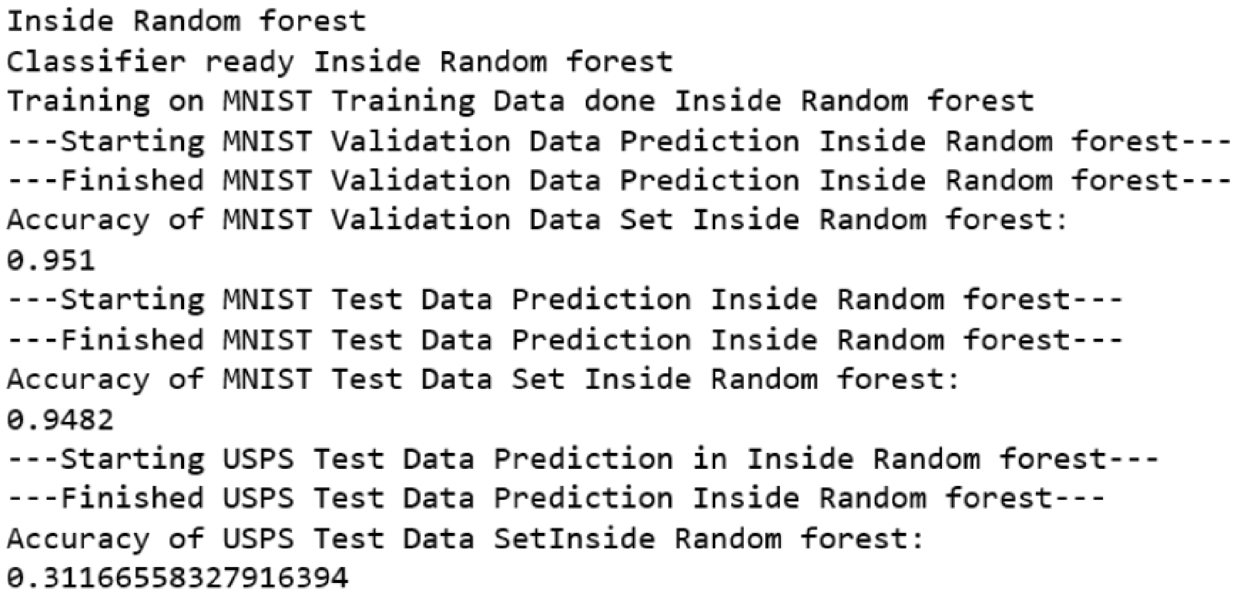
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 300)               235500
_________________________________________________________________
activation_1 (Activation)    (None, 300)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                3010
_________________________________________________________________
activation_2 (Activation)    (None, 10)                0
=================================================================
Total params: 238,510
Trainable params: 238,510
Non-trainable params: 0
```

Accuracy	Dataset	Epochs	No on nodes in intermediate
97	MNIST	1200	300
93	MNIST	500	300
41	USPS	1200	300
35	USPS	400	300

Neural Network Configuration

6. Random forest

Random forest as taught in the class are basically construction multiple decision trees and then selecting which path to follow.Basically the way data flows can give rise to many different settings of decision trees, and hence the name forest.
We are using library function of SK-Learns here to construct our random forest.
Due to its dense nature, it was expected to give a good accuracy which it gave around 95%.



```
Inside Random forest
Classifier ready Inside Random forest
Training on MNIST Training Data done Inside Random forest
---Starting MNIST Validation Data Prediction Inside Random forest---
---Finished MNIST Validation Data Prediction Inside Random forest---
Accuracy of MNIST Validation Data Set Inside Random forest:
0.951
---Starting MNIST Test Data Prediction Inside Random forest---
---Finished MNIST Test Data Prediction Inside Random forest---
Accuracy of MNIST Test Data Set Inside Random forest:
0.9482
---Starting USPS Test Data Prediction in Inside Random forest---
---Finished USPS Test Data Prediction Inside Random forest---
Accuracy of USPS Test Data SetInside Random forest:
0.31166558327916394
```

Setting

Just Classifier of Random Forest through SK-Learns library.

Accuracy MNIST	:	95%
Accuracy USPS	:	31%

7. Questions to be answered

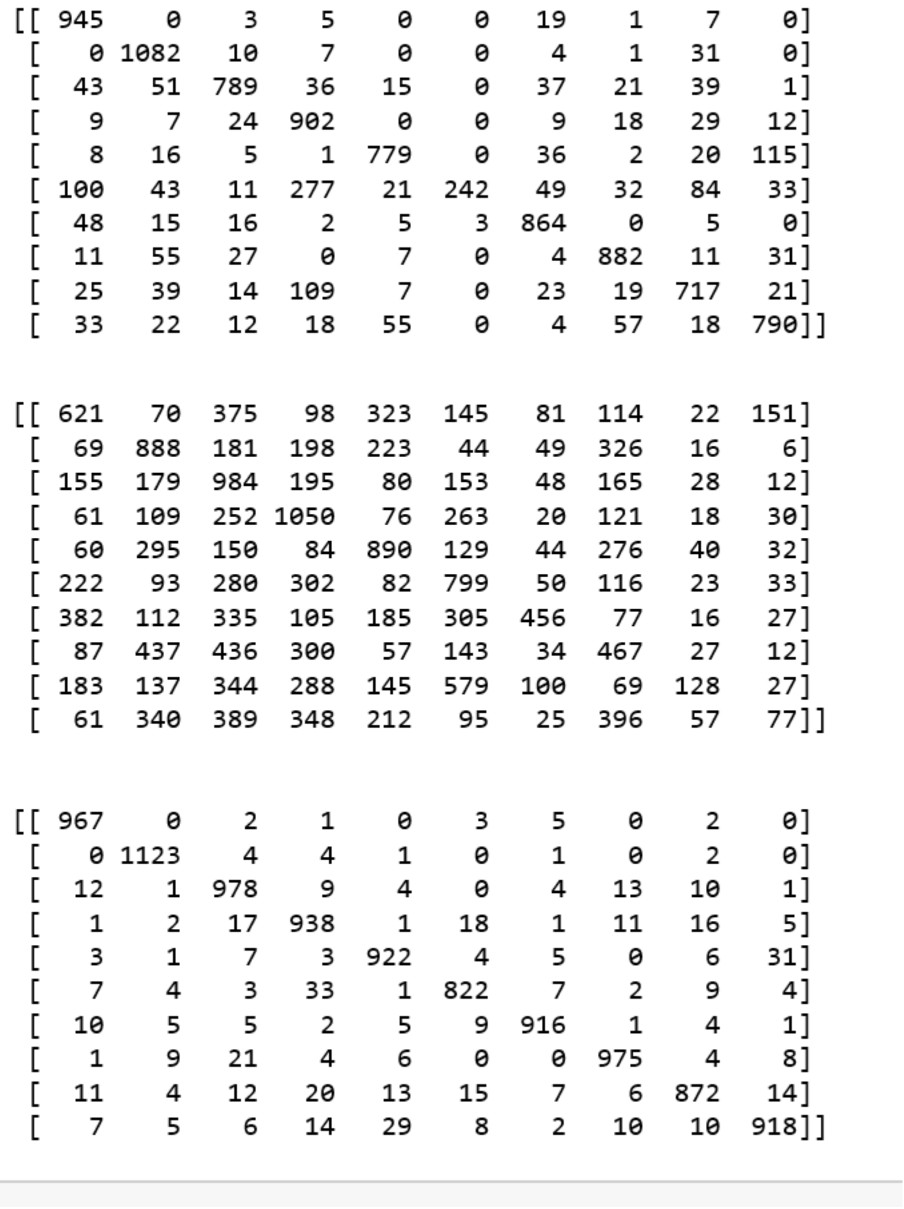
1. Does testing on different dataset support ‘Free Lunch Theorem’

Ans. Yes. It is clear from the above observations that when I trained the model on MNIST dataset and then I tested on MNIST , I am getting an accuracy of around 85-98%.

Whereas, when I test the same model on USPS data set, accuracy drops to 40%. This means that model that has been trained on one type of problem data set i.e. MNIST, it not full proof for a different type of dataset which is USPS. This is in fact the ‘No free lunch’ theorem and hence my observation supports this 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. I have printed the confusion matrix of each classifier. Some of the screen shots of the matrix are as below



```
[[ 945    0    3    5    0    0   19    1    7    0]
 [   0 1082   10    7    0    0    4    1   31    0]
 [  43   51  789   36   15    0   37   21   39    1]
 [   9    7   24  902    0    0    9   18   29   12]
 [   8   16    5    1  779    0   36    2   20  115]
 [ 100   43   11  277   21  242   49   32   84   33]
 [  48   15   16    2    5    3  864    0    5    0]
 [  11   55   27    0    7    0    4  882   11   31]
 [  25   39   14  109    7    0   23   19  717   21]
 [  33   22   12   18   55    0    4   57   18  790]]

[[ 621   70  375   98  323  145   81  114   22  151]
 [  69  888  181  198  223   44   49  326   16    6]
 [ 155  179  984  195   80  153   48  165   28   12]
 [  61  109  252 1050   76  263   20  121   18   30]
 [  60  295  150   84  890  129   44  276   40   32]
 [ 222   93  280  302   82  799   50  116   23   33]
 [ 382  112  335  105  185  305  456   77   16   27]
 [  87  437  436  300   57  143   34  467   27   12]
 [ 183  137  344  288  145  579  100   69  128   27]
 [  61  340  389  348  212   95   25  396   57   77]]

[[ 967    0    2    1    0    3    5    0    2    0]
 [   0 1123    4    4    1    0    1    0    2    0]
 [  12    1  978    9    4    0    4   13   10    1]
 [   1    2   17  938    1   18    1   11   16    5]
 [   3    1    7    3  922    4    5    0    6   31]
 [   7    4    3   33    1  822    7    2    9    4]
 [  10    5    5    2    5    9  916    1    4    1]
 [   1    9   21    4    6    0    0  975    4    8]
 [  11    4   12   20   13   15    7    6  872   14]
 [   7    5    6   14   29    8    2   10   10  918]]
```

I will compare the different classifiers as below headings:

Speed

Logistic Regression classifier takes the least amount to run.
This is followed by Neural Network.
This is followed by SVM
Random Forest took the most amount of time to run as it is complex.

Accuracy

Deep Neural Network gave the highest accuracy of about 98%
Random forest gave accuracy of about 95%
SVM gave accuracy of about 91%
Logistic gave the least accuracy of about 85%

What is confusion Matrix ?
Confusion matrix is simple 0-9 row by column representation of actual vs the predicted value.
So if the actual value is same as predicted output, then for that diagonal index, the total for all the such datasets will be its accuracy for that particular class.
So Total Accuracy will be total Diagonal Values / total samples.

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?**

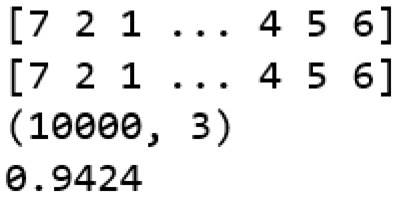
Here I have used Majority Voting method.

So what I have done is , for each of the predicted value in the dataset, for each row, I have seen what all 4 classifiers have predicted for that data point.

Let's say

Logistic =	Class 2
Deep Neural Network =	Class 3
SVM =	Class 2
Random Forest =	Class 1

Ensemble Model Output = Class 2.



```
[7 2 1 ... 4 5 6]
[7 2 1 ... 4 5 6]
(10000, 3)
0.9424
```

Here the shape is 3, as I have in this example run with Neural Network, Logistic and Random Forest Average.

Accuracy

I got an accuracy of 95% on MNIST data set.

Conclusion: I won't say this method helped in increasing the accuracy as it was already 98% with Neural Network. But yes it might be useful in certain situations where we have to use model like Logistic regression along with other models. In such case since we are having kind of low accuracy with one model, we can implement voting mechanism and this will give us the best average result in this case **95%.**

8 Conclusion

We saw the 4 classifiers that are widely used today and how their accuracy compared.

We also proved the 'No Free Lunch theorem' through our testing on MNIST data set and USPS data set. We saw that our trained model on MNIST data set couldn't perform well on USPS data set.

Last we constructed an ensemble model of our all 4 models , where we used the majority voting concept. Though it improved accuracy for 2 classifiers, it was less than the other 2.

References

1. https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
2. https://en.wikipedia.org/wiki/Random_forest
3. https://simple.wikipedia.org/wiki/Logistic_Regression
4. Previous Project 1.1 and 1.2 , 2 for code references.
5. https://www.onlinecharttool.com/graph?selected_graph=xy
6. https://scikit-learn.org/stable/modules/svm.html
7. https://github.com/schwalbe10/thinkageDeepLearning/blob/master/03_mnist.py
8. https://gluon.mxnet.io/chapter02_supervised-learning/softmax-regression-scratch.html

All above links are mostly used for Code references.