

CSE 574 – INTRODUCTION TO MACHINE LEARNING – PROJECT 3

Submitted By: Keerthana Baskaran

UB# 50288944

Contents:

1. *Introduction*
 - *Given Dataset*
2. *Implementation*
 - *Preprocessing the data*
 - *Softmax regression model*
 - *Deep neural network*
 - *Convolutional neural network*
 - *Support vector machine*
 - *Random forest classifier*
 - *Ensemble model classifier*
3. *Questions to be answered*
4. *Experimental results*
5. *Conclusion*

1. Introduction

The goal of the project requires you to apply machine learning to solve the task of classification. We should will first implement an ensemble of four classifiers for a given task. Then the results of the individual classifiers are combined to make a final decision.

The classification task will be that of recognizing a 28_28 grayscale handwritten digit image and identify it as a digit among 0, 1, 2, ..., 9. We are required to train the following four classifiers (logistic regression, Neural network, SVM and Random forest) using MNIST dataset and test using the MNIST and the USPS dataset.

Given Dataset

The MNIST database contains 60,000 digits ranging from 0 to 9 for training the digit recognition system, and another 10,000 digits as test data. The USPS dataset is similar to MNIST and is used only to test the model and has 2000 samples for each digit.

2. Implementation

2.1 Preprocessing the dataset

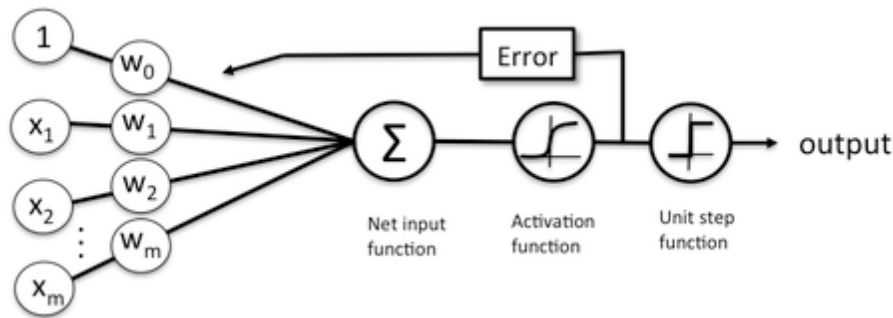
2.1.1 Preprocessing the MNIST dataset

Each digit is normalized to fit in 20X20 pixel box while preserving the aspect ratio and centered in a gray-level image with size 28X28, or with 784 pixel in total as the features. The intensity of the images varies from 0 to 255.

2.1.2 Preprocessing the USPS dataset

The images are in form of segments scanned at a resolution of 100ppi and cropped. We resize the image to 28X28 like MNIST digits and feed into the model.

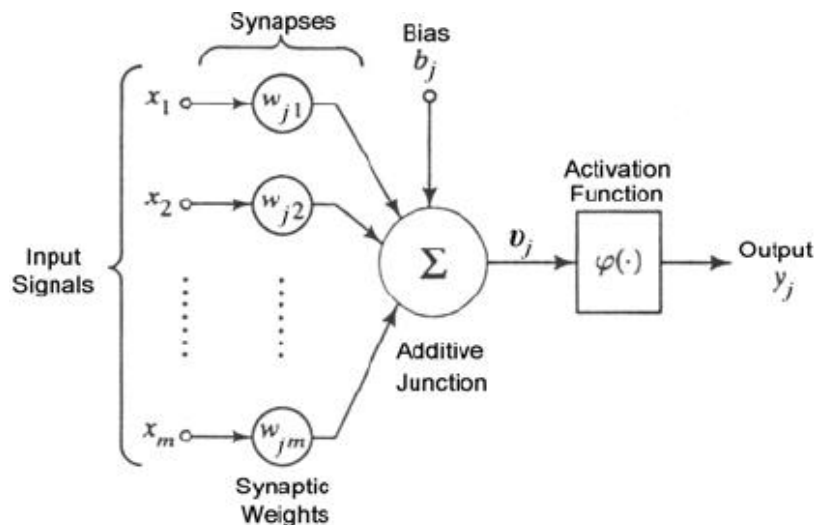
2.2 Softmax Regression Model



There is only 3 steps involved in softmax regression, get the softmax function, calculation cost and then do gradient descent.

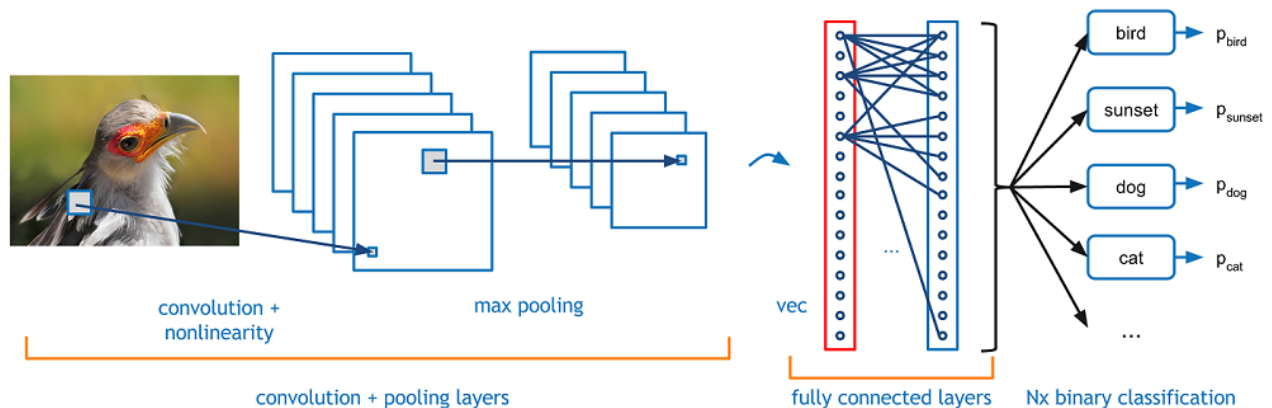
1. Logistic regression is a **classification algorithm** used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic **softmax function** to return a probability value which can then be mapped to two or more discrete classes.
2. We use binary logistic regression on the dataset and apply softmax function on it.
3. The **genesis equation** is given by target = softmax x (weights and the input). Initially we randomly **initialize** the weights and send these weights along with input to the model. We compute the softmax value and the learning rate.
4. According to the computational graphs, we **backtrack** from the learning rate to find the loss function by taking derivative with respect to the weights.
5. For each epoch we find the loss or the errors and then keep **updating** the weights using the gradient descent solution.

2.3 Deep Neural Network



1. We are processing the data using neural network. A neural network has **input layer, hidden and output layer**. Layers are made of nodes connected by neurons. Node is the place where computation happens.
2. We use **activation function** to eliminate linearity in the network and improve accuracy. **Loss function** is the sum of squared errors. We use adam,rmsprop etc.
3. To stop **overfitting** we use earlystopping and dropout rate

2.4 Convolutional neural network



1. We use **feature detector** or kernel to detect the input image
2. Apply the **Relu** activation to increase non linearity.
3. **Pooling** on the features and get the pooled feature map. (example place the matrix in 2x2 feature map and pick largest value In box)
4. Do **flattening** by transforming the entire pooled feature map to single column
5. Pass this to the **neural network** with different layers and get the error prediction. Backpropagate the same to improve predictions.

2.5 Support vector machine

SVM is a supervised machine learning algorithm which can be used for **classification or regression problems**. It uses a technique called the kernel trick to transform the data and then based on these transformations it finds an optimal boundary (called **margin**) between the possible outputs. SVM does some extremely complex data transformations, then figures out how to separate the data based on the labels or outputs we defined.

In the case of support vector machines, a data point is viewed as a **n-dimensional** vector and we want to know whether we can separate such points with a n-1 dimensional **hyperplane**. This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the **largest separation**, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the **maximum-margin hyperplane**

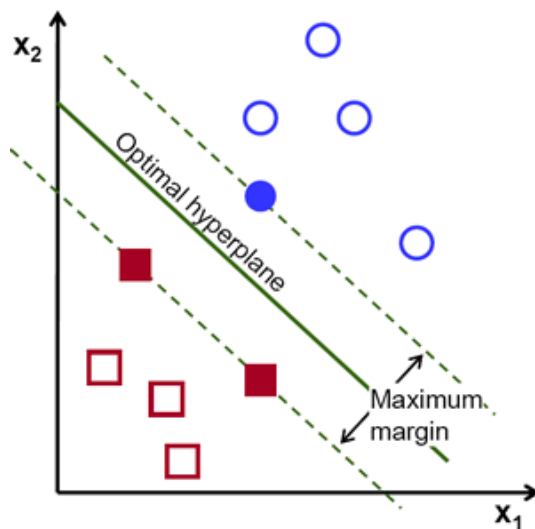
In a two dimension it is easy for the SVM to separate whereas in multidimension it just does a transformation to bring one more dimension and call it z-axis. Let's assume value of points on z plane, $w = x^2 + y^2$. In this case we can manipulate it as distance of point from z-origin. Now if we plot in z-axis, a clear separation is visible, and a line can be drawn. When we transform back this line to original plane, it maps to circular boundary. These transformations are called **kernels**.

The Regularization parameter (often termed as C) tells the SVM optimization how much you want to **avoid misclassifying** each training example.

The gamma parameter defines how **far the influence** of a single training example reaches. A margin is a **separation of line** to the closest class points. Low gamma values give us **strong accuracy**. (**Intuition:** It would mean the data points are sparse, far enough from decision boundary in graph plot).

Why we use SVM

For **classification problem**, given a set of datapoints the SVM model will first predict the **perpendicular distance** between the datapoint and the hypothesis. Then choose the hypothesis with **maximum** of perpendicular distance. The **tangential line** is drawn parallel to the hypothesis covering the closest datapoint. The area between these two tangential points is called the **marginal space**. SVM chooses and keeps this **space maximum**.



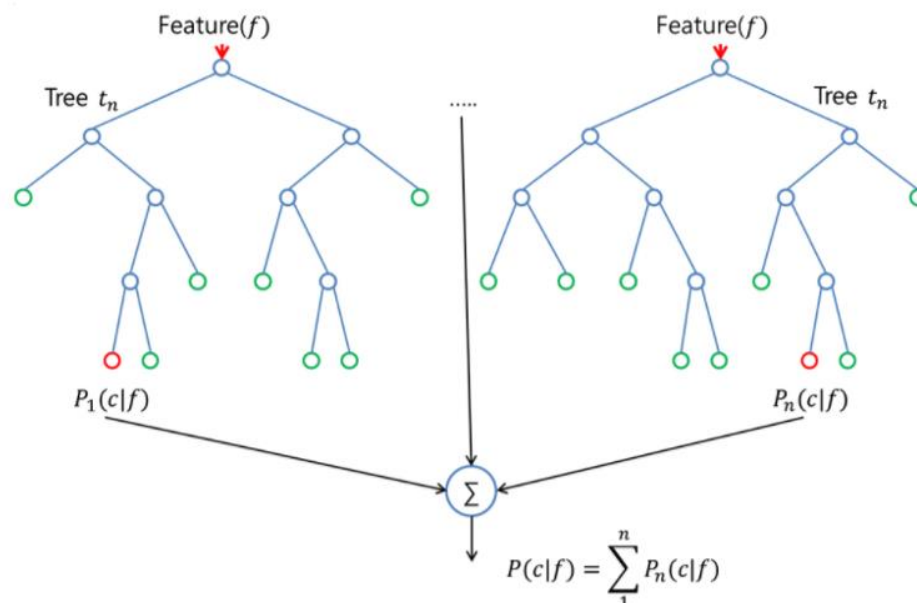
How SVM works?

1. In logistic regression, we take the output of the linear function and squash the value within the range of $[0,1]$ using the **sigmoid function**. If the squashed value is greater than a threshold value (0.5) we assign it a label 1, else we assign it a label 0.
2. In SVM, we take the output of the linear function and if that output is greater than 1, we identify it with one class and if the output is -1, we identify it with another class. Since the threshold values are changed to 1 and -1 in SVM, we obtain this range of values $([-1,1])$ which acts as margin.

3. The loss function that helps **maximize the margin** is hinge loss.
4. The cost is 0 if the predicted value and the actual value are of the same sign. If they are not, we then calculate the loss value.
5. We also add a **regularization parameter** the cost function. The objective of the regularization parameter is to balance the margin maximization and loss.
6. Now that we have the loss function, we take **partial derivatives** with respect to the weights to find the **gradients**. Using the gradients, we can update our weights.
7. When there is no misclassification, i.e our model correctly predicts the class of our data point, we only have to **update the gradient** from the regularization parameter.
8. When there is a misclassification, i.e our model makes a mistake on the prediction of the class of our data point, we **include the loss** along with the regularization parameter to perform gradient update.

2.6 Random forest

Random Forest is a supervised learning algorithm and capable of performing both **regression and classification** tasks. As the name suggests, Random Forest algorithm creates a forest with a number of decision trees as shown below:



As we see, there is multiple decision trees as base learners. Each decision tree is given a subset of random samples from the data set.

For **one dataset we will have only one tree**. But when we split the dataset into subsets we will get **many** decision trees.

We use **random forest over decision** trees because it gives view about the **world**. Decision tree might be many, but random forest will come up and **combine all**.

How random forest works

Subset the dataset and formulate the decision trees. We do this by taking the features that gives the **maximum information**.

Then find the **entropy** for each subset to get probability and use it to build the decision tree.

Gain Is the one obtained with respect to another variable and **information gain** is the information obtained with respect to particular variable.

There are many times of decision trees one is **ID3** (Iterative Dichotomiser 3). The first step is to choose the root node.

Entropy Is given by,

$$S = - \sum_{i=1}^N p_i \log_2 p_i,$$

S — Current group for which we are interested in calculating entropy.

Pi — Probability of finding that system in ith state, or this turns to the proportion of a number of elements in that split group to the number of elements in the group before splitting(parent group).

Where 0.5 **entropy is high** because when probability is very low or very high there is no information about anything.

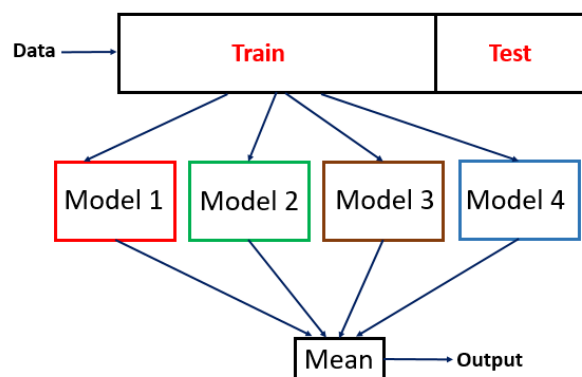
Information gain is given by,

$$IG(Q) = S_O - \sum_{i=1}^q \frac{N_i}{N} S_i,$$

We find the **gain of root** with other node and pick the highest entropy

2.7 Ensemble classifier

Ensemble methods use multiple learning algorithms to obtain better predictive performance. We train various different models, aggregating their predictions to improve stability and predictive power as shown below:



There are many ensemble methods like bagging, boosting, stacking and maximum voting etc

Here I have used **maximum voting** technique.

We can train your model using **diverse algorithms** and then ensemble them to predict the final output. If we use a Random Forest Classifier, SVM Classifier, logistic regression and neural

network models are pitted against each other and selected upon best performance by voting using the VotingClassifier Class from sklearn.ensemble.

Hard voting is where a model is selected from an ensemble to make the final prediction by a simple majority vote for accuracy.

Soft Voting can only be done when all your classifiers can calculate probabilities for the outcomes. Soft voting arrives at the best result by averaging out the probabilities calculated by individual algorithms.

Hard voting is the simplest case of **majority voting**. Here we predict the class label y via majority or plurality voting of each classifier C_j

$$Y = \text{mode} \{C_1(x), C_2(x), \dots, C_m(x)\}$$

If we combine three classifiers that classify a training sample as follows,

Classifier 1 \rightarrow class 0

Classifier 2 \rightarrow class 0

Classifier 3 \rightarrow class 1

$$Y = \text{mode} \{0, 0, 1\} = 0$$

Through majority voting we would classify the sample as class 0.

In this project I have implemented **maximum voting ensemble method in python** and have implemented using hard voting using the sklearn.ensemble library also.

3. Questions to be answered

1. Does the result support “No free lunch theorem”?

The “**No Free Lunch**” theorem states that there is no one model that works best for every problem. The assumptions of a great model for one problem may not hold for another problem, so it is common in machine learning to try multiple models and find one that works best for a particular problem.

A model that works well could also be trained by multiple algorithms – for example, **linear regression** could be trained by the **normal equations** or by **gradient descent**.

Depending on the problem, it is important to assess the trade-offs between **speed**, **accuracy**, and **complexity** of different models and algorithms and find a model that works best for that particular problem.

Usually, No Free Lunch theorem states that no **optimization technique** (algorithm/heuristic/meta-heuristic) is the best for the generic case and all special cases (specific problems/categories). We can have a technique that is better for the generic case, but it will lose to certain techniques for specific problems/categories. Likewise, a technique for a given problem/category **will not be comparable** to some techniques for generic cases.

In this project both MNIST and USPS are **closely related** similar dataset. The models that work great for MNIST is **not doing good** for the USPS dataset. According to the result obtained by this project by training the MNIST dataset we are **not** getting a **good accuracy** for the USPS dataset. Which means we **cannot** train with one dataset and test with another. Therefore our results clearly **prove** that it **supports 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?

In machine learning a **confusion matrix**, also known as an error matrix, is a specific table layout that allows **visualization** of the performance of an algorithm, typically a supervised learning one. Each row of the Matrix represents the instances in a **predicted class** while each column represents the instances in an **actual class** (or vice versa).

All correct predictions are located in the **diagonal of the table**, so it is easy to visually inspect the table for prediction errors, as they will be represented by values outside the diagonal.

We can **infer** many things directly from the confusion matrix like the **error rate and the accuracy**. The model is predicting good if all the values in the diagonal and other values are close to zero.

From our experimental results we can see that **MNIST dataset** has good accuracy in all models which is shown by the confusion matrix. The **diagonal values are higher** than the other values. Whereas the **USPS confusion** matrix has values that are similar in both diagonal and non-diagonal which shows why the model **accuracy is low**. As we can see from our results that as the accuracy goes high, for example the SVM model for MNIST dataset, there is very less variations in the non diagonal values. This shows that model is predicting good for the given dataset.

Comparison of the models

The **convolutional neural network** gives best accuracy than any other model because they are specifically designed to exploit the structure of an image. For any problem CNNs are very **good feature extractors**. This means that you can extract useful attributes from an already trained CNN with its trained weights by feeding your data on each level and tune the CNN a bit for the specific task. They have proven good in most of the recognition task and in our project also the CNN gives a accuracy of **99%**. (implemented but not able to complete the execution as it takes longer time in my machine)

The **SVM** works with even **unstructured** and semi structured data like text, Images. With an appropriate **kernel function**, we can solve any complex problem. Unlike in neural networks, SVM is **not solved for local optima**. But choosing a good kernel is not easy task. And the training time is usually **long**. SVM usually gives superior accuracy than neural network. They are also fairly **robust against overfitting**, especially in high-dimensional space. **Disadvantage** is, SVM's are **memory intensive**, trickier to tune due to the importance of picking the right kernel, and **don't scale well** to larger datasets.

Random forest is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier. It runs efficiently on **large databases**. It gives estimates of what variables are important in the classification. It has an effective method for **estimating missing data** and maintains accuracy when a large proportion of the data are missing.

Disadvantage is Random forest sometimes **overfit** the for some datasets with noisy classifications.

Among all the models, The **SVM and convolutional neural network worked well for** our dataset given their advantages and properties they are best for data recognition.

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?

All ensemble methods share the following 2 steps: Producing a **distribution** of simple models on subsets of the original data. Combining the distribution into one **aggregated** model

An ensemble itself is a learning algorithm, because it can be **trained** and then used to make **predictions**. The trained ensemble, therefore, represents a single hypothesis. Ensembles tend to yield better results when there is a significant diversity among the models.

Ensemble Methods uses **two or more diverse classifiers** in the hope that, a set of classifiers, on average, will have a **higher performance** than a single classifier. If we have 10 classifiers, each one misclassify 10% of the test pattern imagine all classifiers are 100% diverse. using all classifiers will get us 100% accuracy, while a single one will give us 90% accuracy. That's why, in an ensemble method, you want to generate classifiers that have different regions of competence (they are wrong in different areas of the space).

Implementation of ensemble model in python involved taking all the **predicted value** of all 4 models and **comparing** each predicted value with other model and **picking** the majority one. This newly computed value is used to calculate the accuracy. We got **96%** accuracy when implemented in python and **94%** accuracy when used sklearn.ensemble

The **accuracy is better** than logistic and deep neural network classifier **but not** better than SVM and convolutional neural network. Ensemble are unlikely to overfit and the model is more stable. If the ensemble is more accurate than a linear additive model, the ensemble is probably **exploiting** some non-linear and interaction effects.

3. Experimental results

3.1 Logistic Regression

Hyperparameter setup for Linear Regression

Lambda is 0.01, learning rate is 0.01, iterations is 1000

3.1.2 MNIST and USPS dataset

```
Console 1/A ✕
-----LOGISTIC REGRESSION-----
loss
0.6962275131853127
-----
Test Accuracy for MNIST dataset: 0.8712
-----
Test Accuracy for USPS dataset: 0.3317165858292915
-----
Confusion matrix for MNIST dataset
[[ 947   0   3   3   0   3  15   1   8   0]
 [   0 1094   5   3   1   2   4   0  26   0]
 [  16  19 849  26  19   0  27  22  47   7]
 [   5   3  22 883   1  28   8  19  27  14]
 [   3   8   5   0 858   1  17   2  11  77]
 [  26  15   5  83  23 644  28   9  42  17]
 [  20   5  13   2  13  19 880   0   6   0]
 [   4  39  26   1  13   0   4 887  10  44]
 [   9  14  14  39  12  22  18  14 812  20]
 [  14  13  11  12  53  10   1  26  11 858]]
-----
confusion matrix for USPS dataset
[[ 708   5 414  46 331  43  73  37  85 258]
 [ 294 290 162 254 281  34  42 303 326  14]
 [ 285  39 1120 116  79  51 104  99  89  17]
 [ 175   4 141 1116  47 205  47  72 127  66]
 [ 144  96  39  51 1078  91  30 127 240 104]
 [ 250  23 239 227  56 848 141  81  96  39]
 [ 524  15 387  92 121 120 626  21  66  28]
 [ 213 243 370 359  70  78  47 283 300  37]
 [ 277  40 191 204 181 409 137  36 442  83]
 [ 105 224 180 403 199  64  17 364 321 123]]
-----
```

Observation

The **MNIST dataset** has a higher accuracy of **87%**. This is because we train and test using MNIST dataset. Whereas the **USPS** has a **lesser** accuracy because training is not done in USPS dataset. Another reason is that the images of USPS dataset are not preprocessed to 28X28 pixels. This is also one of reason for lower accuracy. The **confusion matrix** clearly shows the model predictions through its diagonal value.

The accuracy was around 60% initially when we use different learning rate and lambda value. With current network setting I got the highest accuracy.

3.2 Neural Network

Hyperparameter setting for Convolution neural network

Epoch is 10, Dropout rate is 0.25, activation is relu, number of layer is 7, filter size is 3x3, number of filter used is 32-64 in different layers

Observation

Apparently, the accuracy of convolution neural network should be greater than other models, intuitively it should be around **99%**. I was not able to complete the run of convolution neural network in my machine since each time it took more than 45 mins for one epoch and gradually became slow as it crossed three epochs. So, I had to kill the kernel. I have implemented the convolutional neural network model and attached the code in separate file.

Hyperparameter setting for deep neural network

batch size=20, epoch =10, early patience =100, number of layers=2, activation function=sigmoid and softmax, optimizer is adam, loss function is categorical crossentropy

3.2.1 MNIST and USPS dataset

```
Console 1/A ✕
Epoch 9/10
- 6s - loss: 0.1131 - acc: 0.9686 - val_loss: 0.1825 - val_acc: 0.9444
Epoch 10/10
- 5s - loss: 0.1054 - acc: 0.9707 - val_loss: 0.1774 - val_acc: 0.9480
-----NEURAL NETWORKS-----
Loss for MNIST testdata: 0.1403198441900313
Accuracy for MNIST testdata: 0.9583
-----
confusion matrix for MNIST dataset
[[ 964   0   1   2   0   3   7   1   2   0]
 [   0 1121   3   0   0   1   5   1   4   0]
 [   9   1  976   4   5   1   7  10  19   0]
 [   0   3   7  958   1  15   0   5  20   1]
 [   1   0   3   0  938   0  11   2   3  24]
 [   8   3   3  17   2  833   7   1  14   4]
 [   7   2   1   1   3   5  933   0   6   0]
 [   0   6  18  12   3   0   0  974   3  12]
 [   4   5   6   8   3   5   4   5  933   1]
 [   4   7   0  12  13   3   0   6  11  953]]
-----
Loss for USPS testdata: 3.9021079528211944
Accuracy for USPS testdata: 0.3397669883479273
-----
confusion matrix for USPS dataset
[[ 332   0  299   62   81  247  331  351   30  267]
 [  60  261  564  122  203  226   29  340  150   45]
 [  52  12 1382   69   13  224  153   38   48    8]
 [  24   3  309  925    6  554   56   49   58   16]
 [   9   4  150   38  875  189   65  425  143  102]
 [  30   3  531  137   16  991  180   50   44   18]
 [  99   1  655   45   40  151  773  154   23   59]
 [  32  23  145  627   36  138   33  792  152   22]
 [ 174   7  201  468  145  257  220  223  286   19]
 [  13   3  140  390  111   38   26  858  243  178]]
-----
```

Observation

For the above networking setting for MNIST and USPS the accuracy of MNIST dataset is relatively high than USPS dataset. The confusion matrix shows the model prediction visually. The accuracy was around 84% initially with different optimizer technique. With **adam** and after tuning the hyperparameters I got the highest accuracy

3.3 Support Vector machines

Hyperparameter setting for support vector machines

Kernel used is rbf, gamma value is 0.05 all others are kept as default

```
Console 1/A ✕
-----SVM-----
SVM accuracy for MNIST dataset: 0.9827
-----
confusion matrix for MNIST dataset
[[ 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]]
-----
SVM accuracy for USPS dataset: 0.2614130706535327
-----
confusion matrix for USPS dataset
[[ 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]]
-----
```

Observation

The **SVM model** gives a higher accuracy for MNIST dataset of **98%** whereas the **USPS** gives **26%**. The SVM model with **linear** kernel gives an accuracy of **91%** and with gamma as 0.05 and rbf kernel it gives 98%. The accuracy is not changing much for different gamma values (when they are very small). However, it is recommended to have the gamma value as **minimum** for better

accuracy. By setting the environment to **rbf** and **gamma value as 1** we get an accuracy of only 17% which shows that we have to use gamma value minimum and optimum

Kernel	Gamma	Accuracy for MNIST and USPS
Linear	Default	91, 23
Rbf	0.05	98, 26
Rbf	1	17, 10

3.4 Random Forest classifier

Hyperparameter setting for random forest classifier

The number of trees is 10

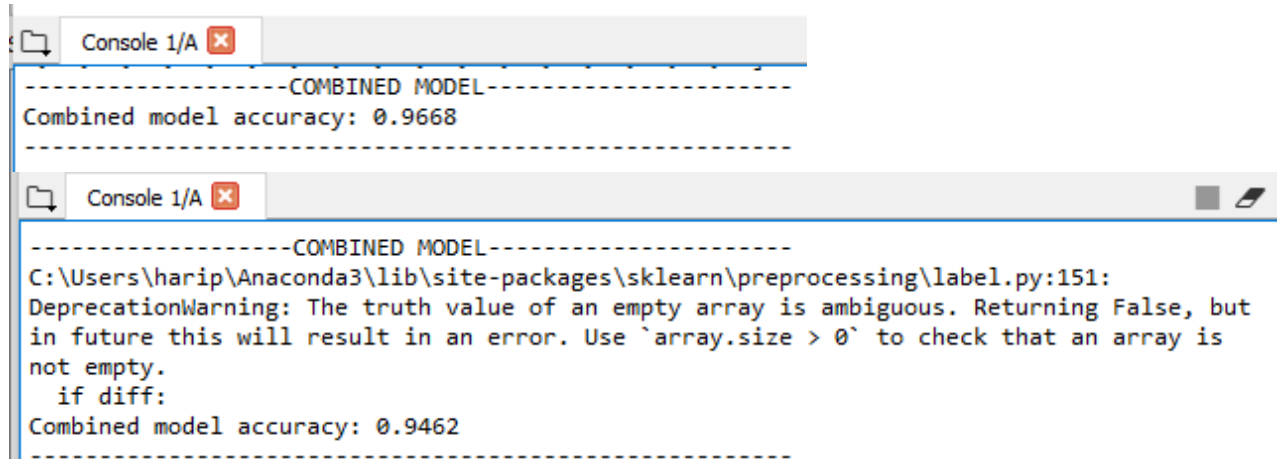
```
Console 1/A [X]
-----RANDOM FOREST-----
Random forest accuracy for MNIST dataset: 0.9463
-----
confusion matrix for MNIST dataset
[[ 969   1   1   0   0   2   2   2   3   0]
 [   0 1122   3   4   1   1   2   0   2   0]
 [   8   0 984   9   5   0   4   8  12   2]
 [   3   0  23 946   1  16   1   8   9   3]
 [   3   2   3   1 935   1   8   1   5  23]
 [  13   2   4  27   4 820   9   1   6   6]
 [  15   3   1   2   6   8 921   0   2   0]
 [   3  10  27   2   5   1   1 964   2  13]
 [  10   1  10  17  14  16  12   2 883   9]
 [   6   6   2  16  26  12   1   8  13 919]]
-----
Random forest accuracy for USPS dataset: 0.30596529826491325
-----
confusion matrix for USPS dataset
[[ 602  21 320  99 375 208  90  94  13 178]
 [ 105 455 196 139 112  78  50 839  17  9]
 [ 160  76 1055 128  80 215  65 194  13  13]
 [ 130  34  216 947  82 403  25 114  15  34]
 [  74 195 166 113 844 127  43 337  29  72]
 [ 225  59  207 259  82 955  59 112  14  28]
 [ 466  81  341  98 157 267 478  72  20  20]
 [ 109 255 468 296  68 203  40 538  14  9]
 [ 160 113 261 336 190 580  98  95 122  45]
 [  90 225 370 340 204 125  41 429  53 123]]
-----
```

Observation

For the above networking setting for MNIST and USPS the accuracy of MNIST dataset is relatively high than USPS dataset. We get an accuracy of 94% for MNIST. The confusion matrix shows the model prediction visually.

I tried **changing the number of trees** to different values from 5 to 20 but every time there is not much change in the accuracy of the model. As I increase the number of trees the model took more time to compute but the accuracy was almost same or 1% higher.

3.5 Ensemble classifier



```
-----COMBINED MODEL-----
Combined model accuracy: 0.9668
-----COMBINED MODEL-----
C:\Users\harip\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151:
DeprecationWarning: The truth value of an empty array is ambiguous. Returning False, but
in future this will result in an error. Use `array.size > 0` to check that an array is
not empty.
  if diff:
Combined model accuracy: 0.9462
-----COMBINED MODEL-----
```

Observation

I have used to maximum voting ensemble classifier implemented in python and got the accuracy of 96%. By using the inbuild ensemble classifier I got an accuracy of 94%. For USPS dataset I got 37% using python implementation

4. Conclusion

Thus, by doing different model implementation for classifying the handwritten samples of MNIST and USPS dataset, I found out that **convolutional neural network and SVM** gives higher accuracy than other models. The ensemble model also gives higher accuracy than the logistic model and random forest is mostly same as ensemble model. The USPS dataset has less accuracy for all the models that the MNIST dataset.

Models (Accuracy)	MNIST dataset	USPS dataset
Softmax regression	0.87	0.33
Deep neural network	0.94	0.33
SVM	0.98	0.26
Random forest	0.94	0.30
Ensemble model	0.96	0.37

References

1. <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>
2. <https://www.youtube.com/watch?v=X3Wbfb4M33w>

3. <https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72>
4. <https://blog.statsbot.co/support-vector-machines-tutorial-c1618e635e93>
5. <https://medium.com/@rakendd/building-decision-trees-and-its-math-711862eea1c0>
6. http://rasbt.github.io/mlxtend/user_guide/classifier/EnsembleVoteClassifier/
7. <https://chemicalstatistician.wordpress.com/2014/01/24/machine-learning-lesson-of-the-day-the-no-free-lunch-theorem/>
8. <https://www.quora.com/What-is-the-No-Free-Lunch-theorem-and-what-is-its-significance>
9. https://en.wikipedia.org/wiki/Confusion_matrix
10. <http://amateurdatascientist.blogspot.com/2012/01/random-forest-algorithm.html>
11. https://www.reddit.com/r/MachineLearning/comments/3dnolr/disadvantages_of_neural_networks_deep_learning_why/
12. <https://arxiv.org/ftp/arxiv/papers/1006/1006.5902.pdf>
13. <https://medium.com/@aravanshad/ensemble-methods-95533944783f>
14. http://brianfarris.me/static/digit_recognizer.html
15. <https://www.kaggle.com/atorin/mnist-digit-recognition-with-random-forests>
16. <https://www.kaggle.com/thunderflash/svm-for-mnist>
17. <https://stackoverflow.com/questions/48987959/classification-metrics-cant-handle-a-mix-of-continuous-multioutput-and-multi-label>
18. <https://medium.com/@awjuliani/simple-softmax-in-python-tutorial-d6b4c4ed5c16>
19. <https://www.youtube.com/watch?v=X3Wbfb4M33w>
20. <https://www.geeksforgeeks.org/applying-convolutional-neural-network-on-mnist-dataset/>
21. Images are from google.