CSE 574: Introduction to Machine Learning (Fall 2018)

Instructor: Sargur N. Srihari

Project 3: Image Classification with MNIST Dataset

November 19, 2018

Report By:

Siddheswar Chandrasekhar

Objective

The goal of this project is to implement machine learning methods to classify images in the MNIST dataset. We first implement an ensemble of four classifiers. Then the results of individual classifiers are to be combined to make a final decision.

Plan of Action

We train the following four classifiers using MNIST digit images:

1. Logistic Regression
2. Multilayer perceptron Neural Network
3. Random Forest Package
4. SVM (Support Vector Machine) Package

Based on the above implementations, we wish to answer the following questions:

1. We test the MNIST trained models on two different test sets: the test set from MNIST and a test set from the USPS dataset. Do our results support the "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?

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?

Preparing the Datasets

The MNIST dataset is divided into three parts:

1. Training dataset having 50,000 samples
2. Validation dataset having 10,000 samples
3. Testing dataset having 10,000 samples

We train our models using the MNIST training dataset (i.e. our models are trained over 50,000 samples). We then test our model over the MNIST testing dataset and the USPS dataset, both individually and combined. The USPS dataset has 20,000 samples (i.e. our models are tested over 30,000 samples)

Logistic Regression (Linear Regression vs Logistic Regression [1])

It's tempting to use the linear regression output as probabilities but it's a mistake because the output can be negative, and greater than 1 whereas probability cannot. As regression might actually produce probabilities that could be less than 0, or even bigger than 1, logistic regression was introduced.

- **Outcome**

  In linear regression, the outcome (dependent variable) is continuous. It can have any one of an infinite number of possible values.

  In logistic regression, the outcome (dependent variable) has only a limited number of possible values.

- **The dependent variable**

  Logistic regression is used when the response variable is categorical in nature whereas Linear regression is used when your response variable is continuous.

- **Equation**

  Linear regression gives an equation which is of the form $Y = mX + C$, means equation with degree 1.

  However, logistic regression gives an equation which is of the form $Y = e^X + e^{-X}$

- **Coefficient Interpretation**

  In linear regression, the coefficient interpretation of independent variables are quite straightforward (i.e. holding all other variables constant, with a unit increase in this variable, the dependent variable is expected to increase/decrease by xxx).

  However, in logistic regression, depends on the family (binomial, Poisson, etc.) and link (log, logit, inverse-log, etc.) you use, the interpretation is different.

- **Error minimization technique**

  Linear regression uses ordinary least squares method to minimise the errors and arrive at a best possible fit, while logistic regression uses maximum likelihood method to arrive at the solution.

  Linear regression is usually solved by minimizing the least squares error of the model to the data, therefore large errors are penalized quadratically.

  Logistic regression is just the opposite. Using the logistic loss function causes large errors to be penalized to an asymptotically constant.

Logistic Regression Approach

1. Features → X
2. Target → y
3. Initialize weights (W) “normally” (i.e. using Normal Distribution)
4. $z = W^T\ X$
5. $$\sigma = \frac{1}{1+e^{-x}}$$
6. Gradient Descent
   $X^T\ (\hat{y} - y)$ → / number of features → * learning rate → – W → new W
7. Repeat steps 4 thru 6 for N number of epochs
8. Test accuracy on unseen data

Logistic Regression Observations

With learning rate as 0.01 and 500 epochs, we get the following results using Logistic Regression on our dataset:

Testing on	**Accuracy**
MNIST dataset	85.95 %

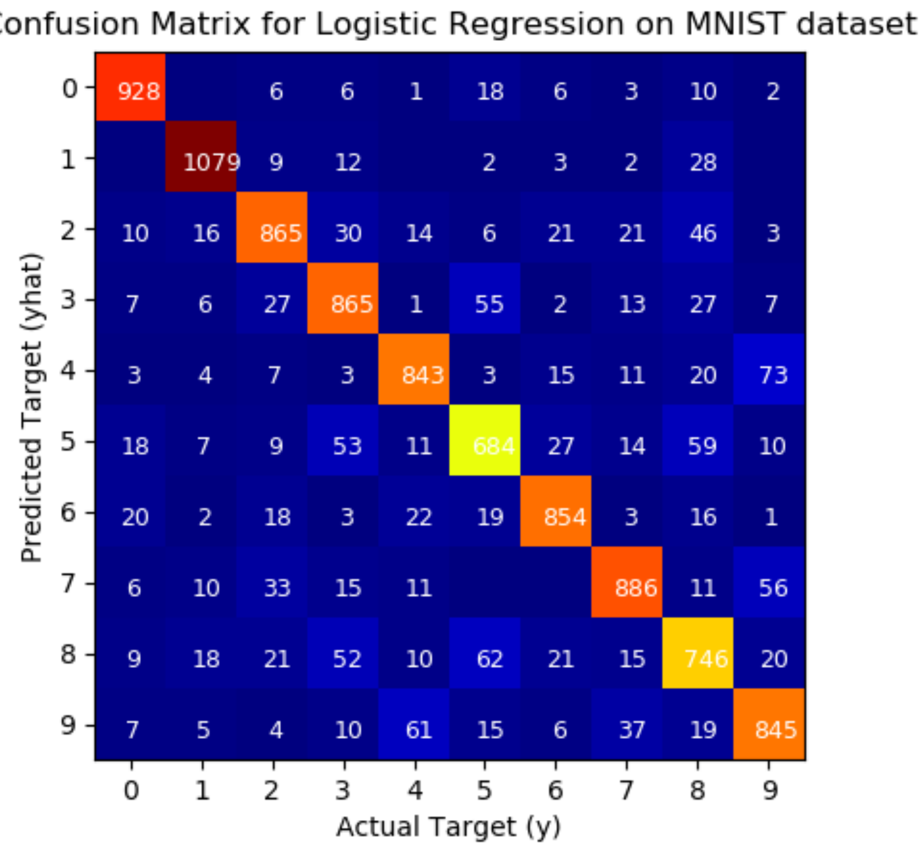


Fig 1 – Logistic Regression on MNIST dataset

We see some pretty large inconsistencies, such as nine being predicted as four over seventy times. This is somewhat expected as Logistic Regression is not a good choice of model for multi-class regression.

Testing on	**Accuracy**
USPS dataset	25.12 %

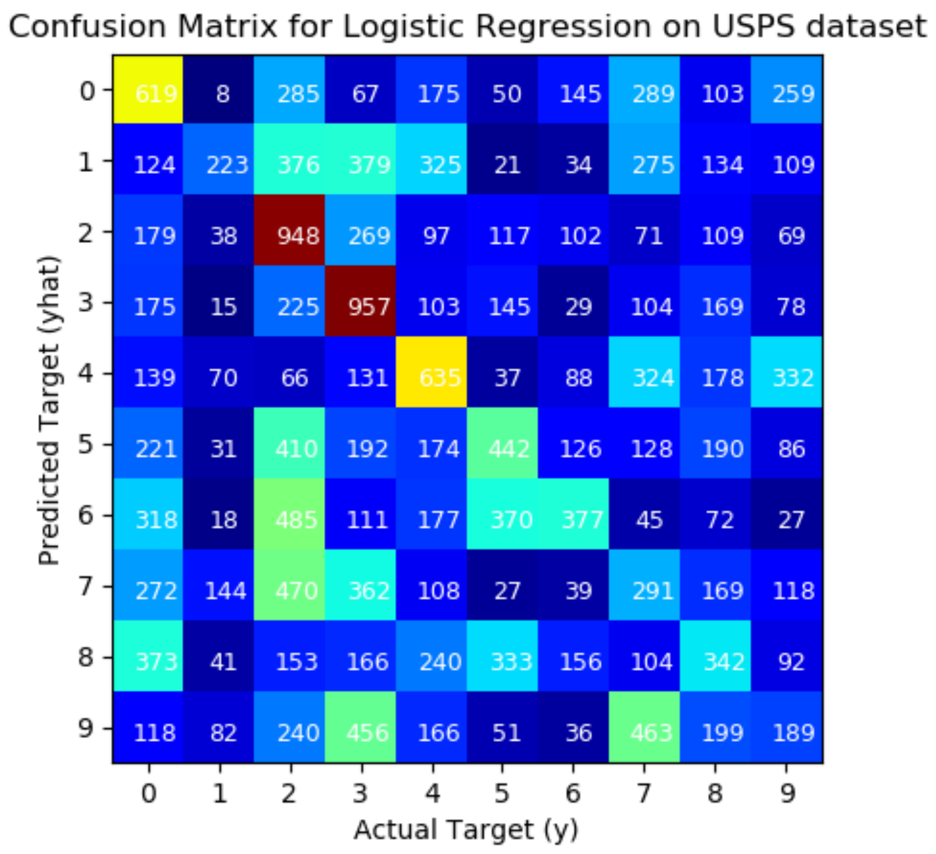


Fig 2 – Logistic Regression on USPS dataset

We clearly see poor accuracy when testing our model on the USPS dataset. By looking at the confusion matrix, we can deduce that several numbers in the USPS dataset are classified incorrectly by our model that has been trained over the MNIST dataset. An accuracy of 25% is proof of this.

Testing on	**Accuracy**
MNIST + USPS datasets	45.39 %

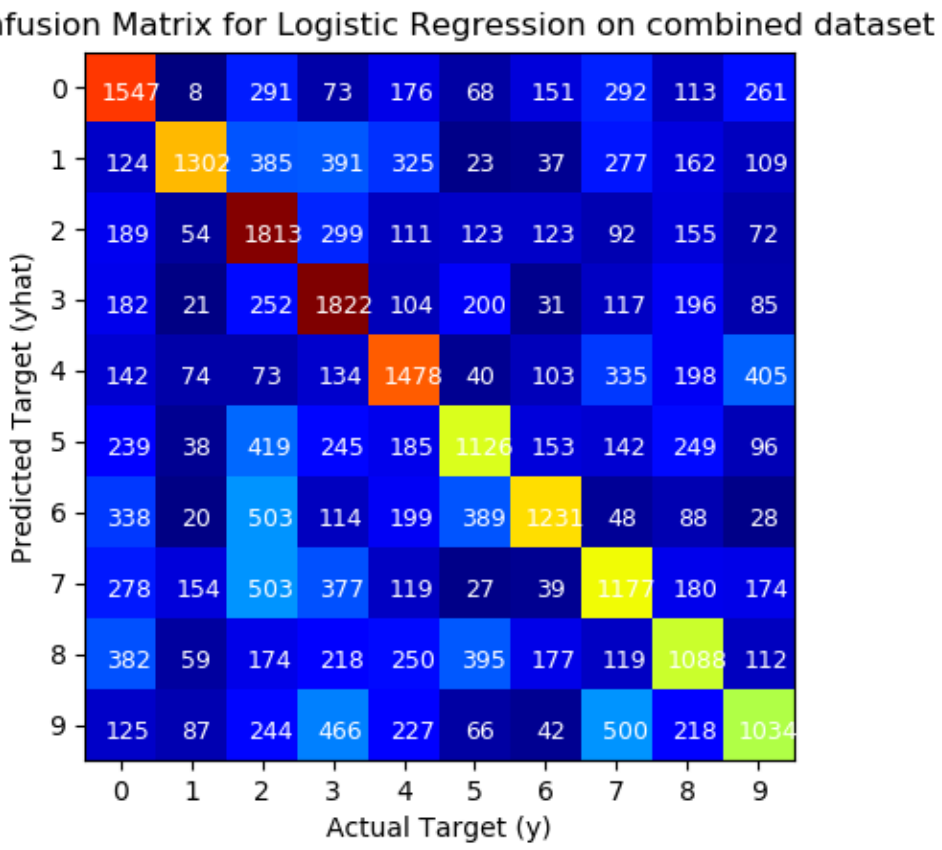


Fig 3 – Logistic Regression on Combined datasets

As expected, the accuracy results on combined dataset is better than that of testing on USPS dataset alone, but worse than testing on MNIST dataset alone. We also notice that the accuracy is more inclined towards that of the accuracy when testing on USPS dataset alone. This is simply because the USPS dataset has twice the number of samples than that of the MNIST dataset. Hence the average is weighed more towards the USPS dataset testing accuracy.

Neural Network

A neural network has a simple architecture that consists of an input layer, a black box which is a set of hidden layers, and an output layer. All the neurons in the network, connect with every neuron in its adjacent layer forming a mesh-like architecture.
The figure below is a good depiction of a simple neural network with three input neurons, one hidden layer of two neurons, and a single output neuron.

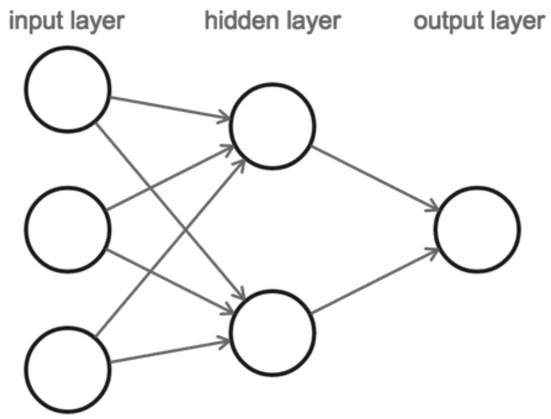


Fig 4 – Neural Network Architecture

We use ReLU (Rectified Linear Unit) as our Activation function for neural network which has the form:

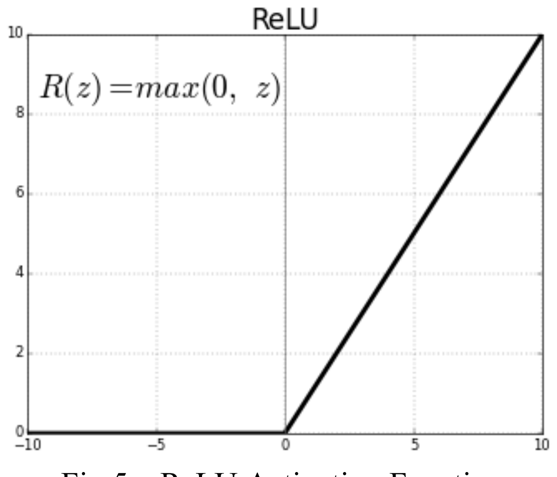
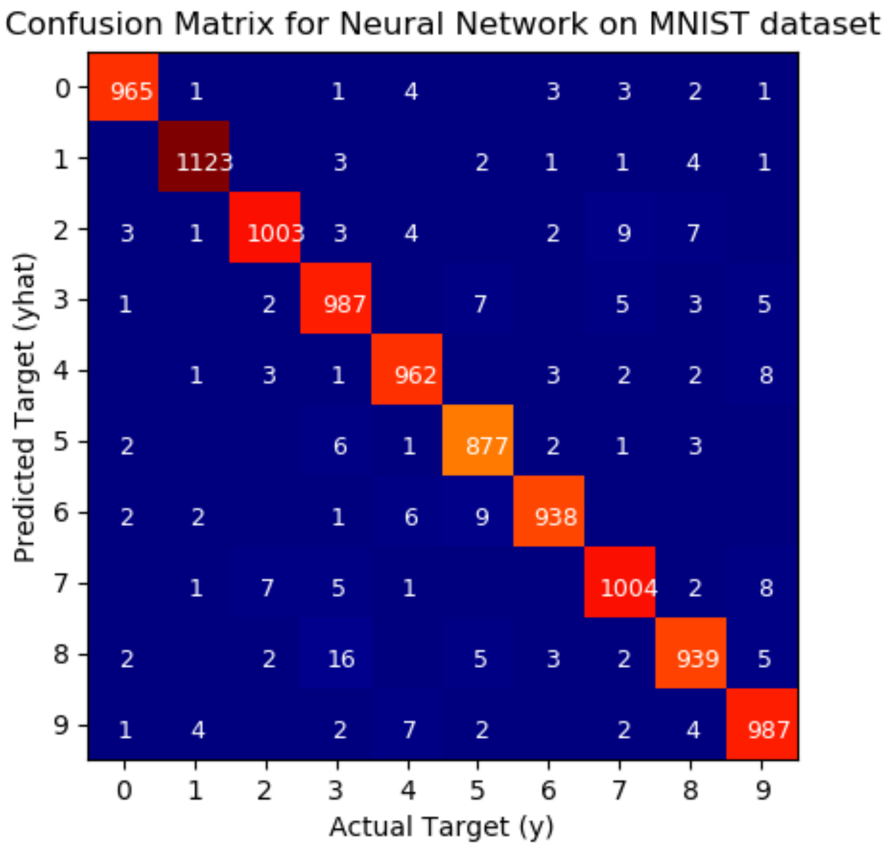


Fig 5 – ReLU Activation Function

Neural Network Observations

With 500 epochs, 500 batch size and a decaying learning rate, we get the following results performing Neural Network on the MNIST dataset:

Activation Function	Testing on	Accuracy
ReLU	MNIST dataset	97.85 %



	0	1	2	3	4	5	6	7	8	9
0	965	1		1	4		3	3	2	1
1		1123		3		2	1	1	4	1
2	3	1	1003	3	4		2	9	7	
3	1		2	987		7		5	3	5
4		1	3	1	962		3	2	2	8
5	2			6	1	877	2	1	3	
6	2	2		1	6	9	938			
7		1	7	5	1			1004	2	8
8	2		2	16		5	3	2	939	5
9	1	4		2	7	2		2	4	987

Fig 6 – Neural Network on MNIST dataset

The results are pretty impressive, with the only major inconsistencies being in four being wrongly predicted as nine, and vice versa.
We can hence deduce that the digits four and nine, look somewhat similar when handwritten.

Activation Function	**Testing on**	**Accuracy**
Sigmoid	MNIST dataset	92.08 %

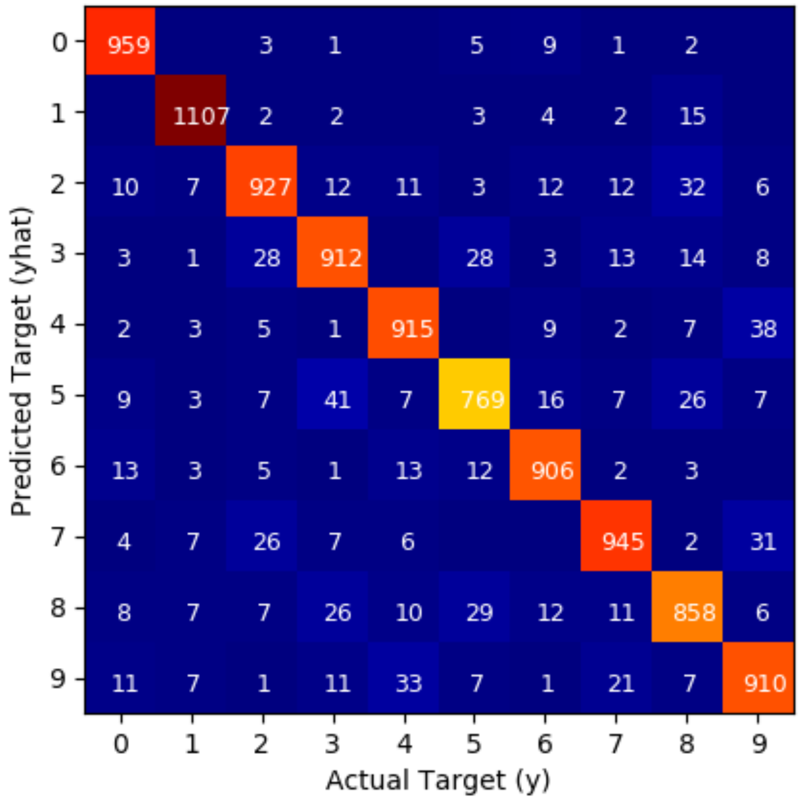
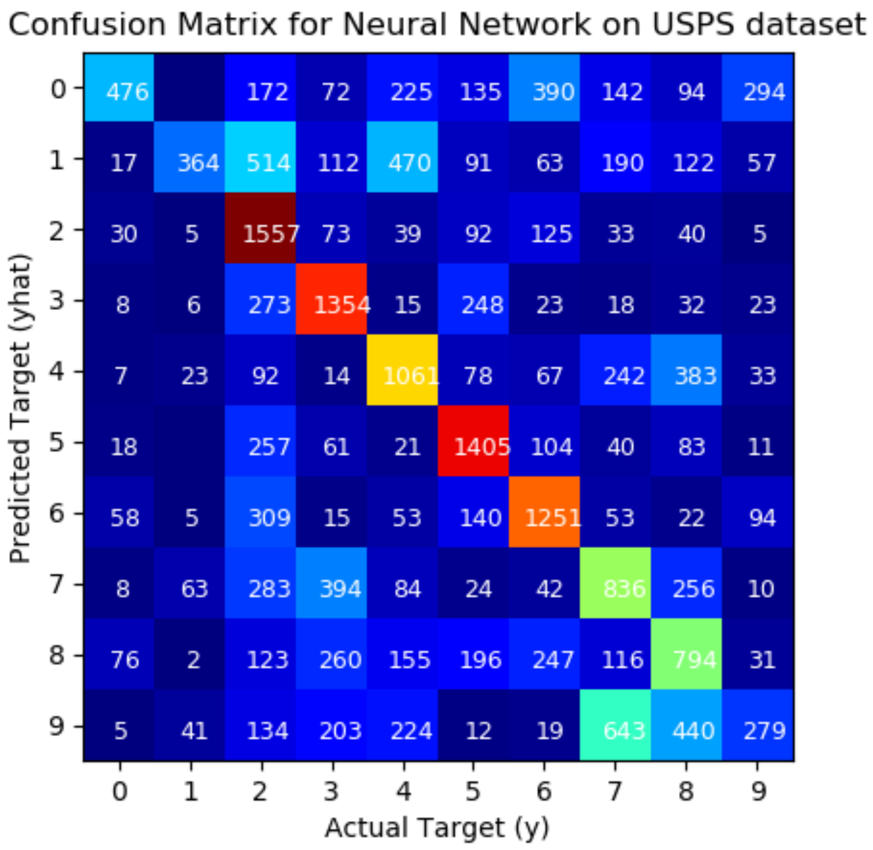


Fig 7 – Neural Network on MNIST dataset using Sigmoid

We see a drop in the accuracy. We observe similar results by using other activation functions such as tanh. We hence decide to stick with ReLU.

Activation Function	Testing on	Accuracy
ReLU	USPS dataset	46.89 %



	0	1	2	3	4	5	6	7	8	9
0	476		172	72	225	135	390	142	94	294
1	17	364	514	112	470	91	63	190	122	57
2	30	5	1557	73	39	92	125	33	40	5
3	8	6	273	1354	15	248	23	18	32	23
4	7	23	92	14	1061	78	67	242	383	33
5	18		257	61	21	1405	104	40	83	11
6	58	5	309	15	53	140	1251	53	22	94
7	8	63	283	394	84	24	42	836	256	10
8	76	2	123	260	155	196	247	116	794	31
9	5	41	134	203	224	12	19	643	440	279

Fig 8 – Neural Network on USPS dataset

Like in Logistic Regression, we clearly see poor accuracy when testing our model on the USPS dataset. By looking at the confusion matrix, we can deduce that several numbers in the USPS dataset are classified incorrectly by our model that has been trained over the MNIST dataset. Some notable inaccuracies are 9 being predicted as 0, seven being predicted as one or nine, three being predicted as seven or nine etc. As a matter of fact, we notice that seven is being incorrectly predicted as one almost as many times as it is being correctly predicted as seven. Moreover, nine is being incorrectly predicted as zero, almost three times that of it being correctly predicted as nine.

Activation Function	Testing on	Accuracy
ReLU	MNIST + USPS datasets	63.88 %

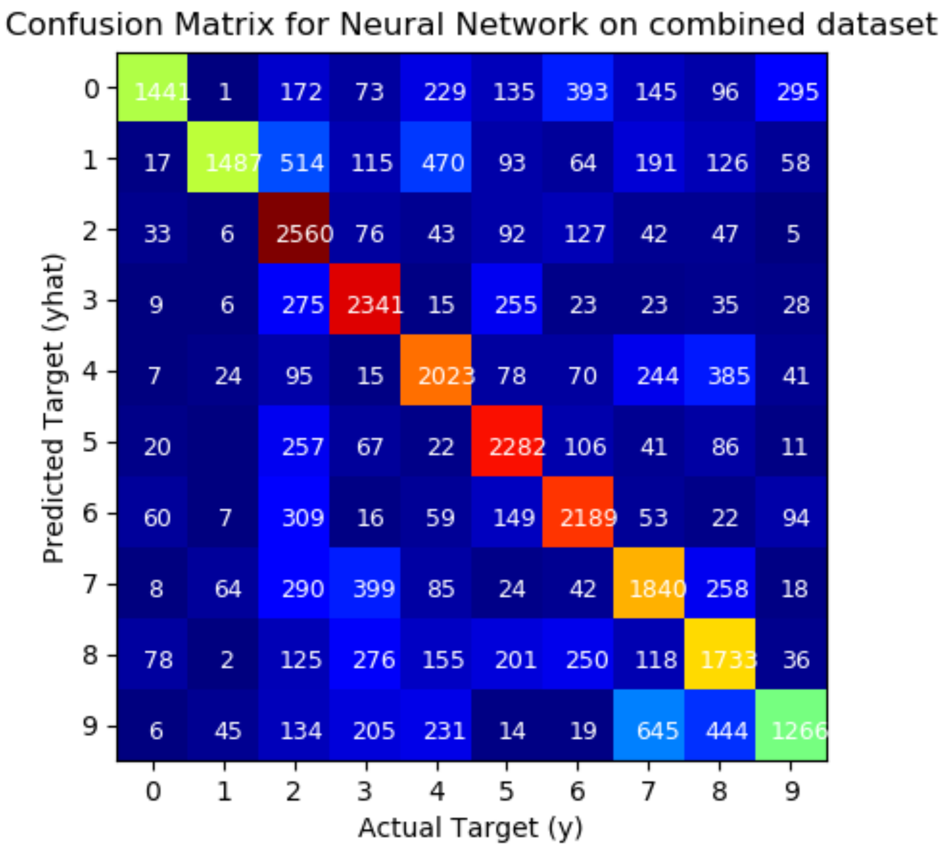
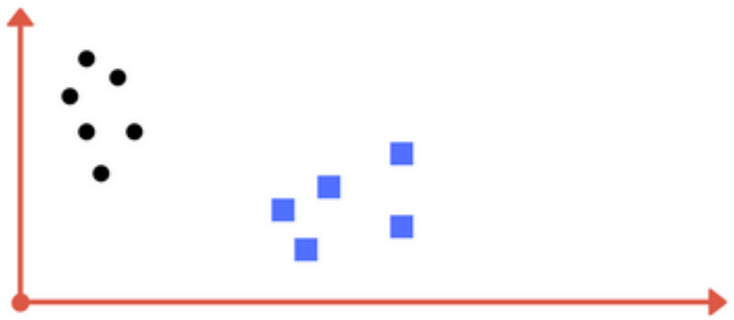


Fig 9 – Neural Network on Combined datasets

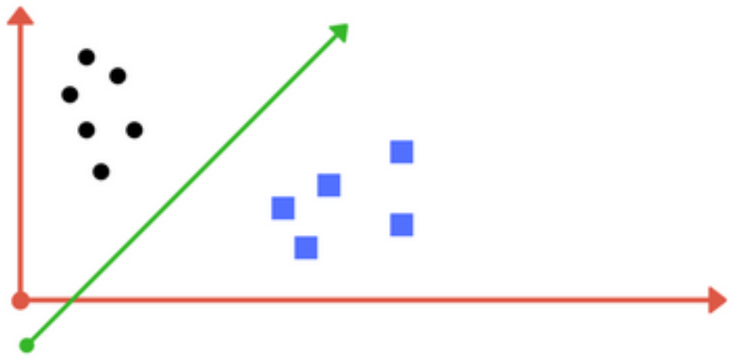
As expected, the accuracy results on combined dataset is better than that of testing on USPS dataset alone, but worse than testing on MNIST dataset alone. Like in Logistic Regression, we also notice that the accuracy is more inclined towards that of the accuracy when testing on USPS dataset alone.

Support Vector Machine (SVM)

"A 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 dimentional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side."



We need to separate the two classes shown in the image above. An approach to this is to draw a line between them as shown in the image below:



Any point that is left of line falls into black circle class and on right falls into blue square class. "Separation of classes", that's what SVM does. It finds out a line/hyper-plane (in multidimensional space that separate outs classes)

Tuning Parameters of SVM

1. 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.

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:

$f(x) = B(0) + sum(ai * (x,xi))$

This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B0 and ai (for each input) must be estimated from the training data by the learning algorithm.

The polynomial kernel can be written as

$K(x,xi) = 1 + sum(x * xi)\hat{}d$

and exponential as

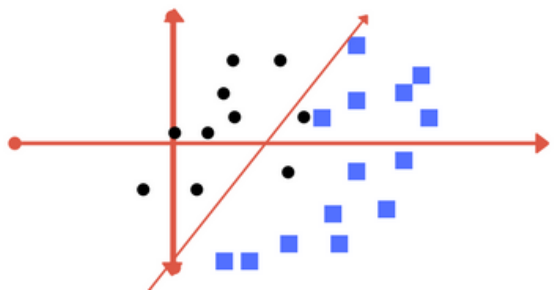
$K(x,xi) = exp(-gamma * sum((x — xi^2)).$

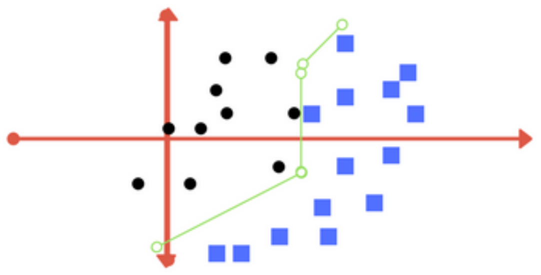
2. Regularization:

The Regularization parameter (often termed as C parameter in python’s sklearn library) tells the SVM optimization how much you want to avoid misclassifying each training example.

For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

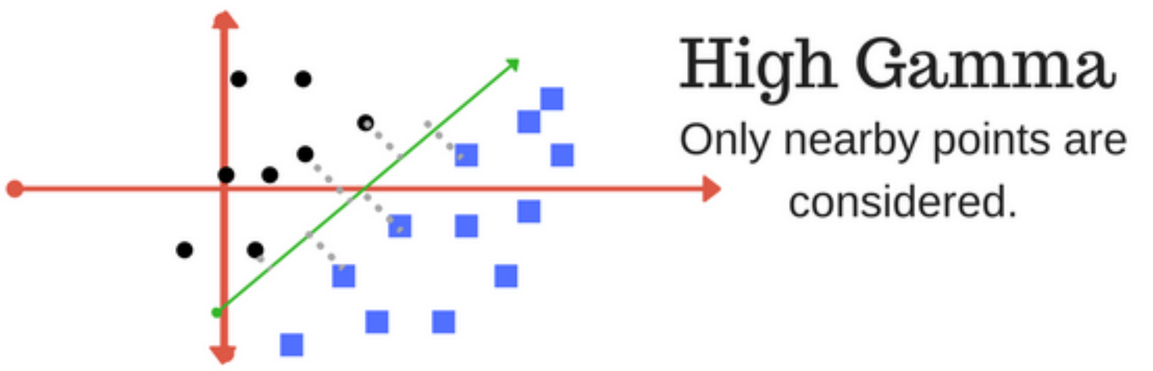
The images below are example of two different regularization parameter. Left one has some misclassification due to lower regularization value. Higher value leads to results like right one.

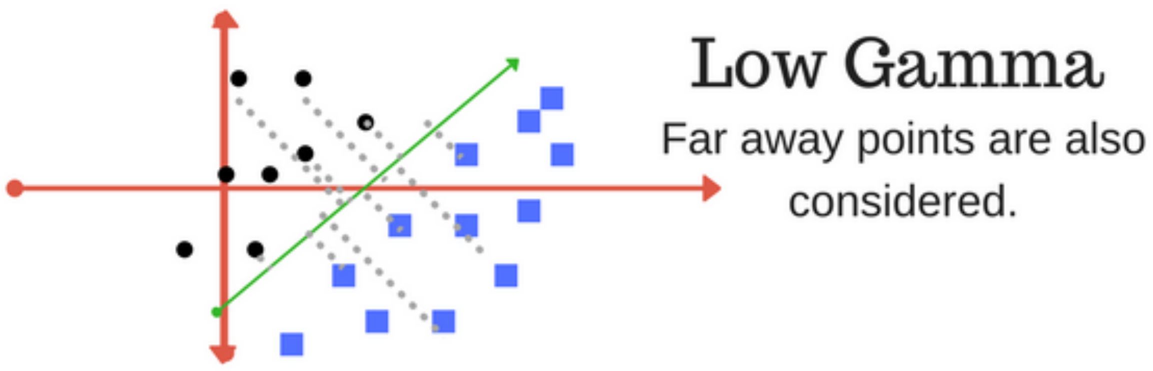




3. <u>Gamma</u>

   The gamma parameter defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’. In other words, with low gamma, points far away from plausible seperation line are considered in calculation for the seperation line. Where as high gamma means the points close to plausible line are considered in calculation.

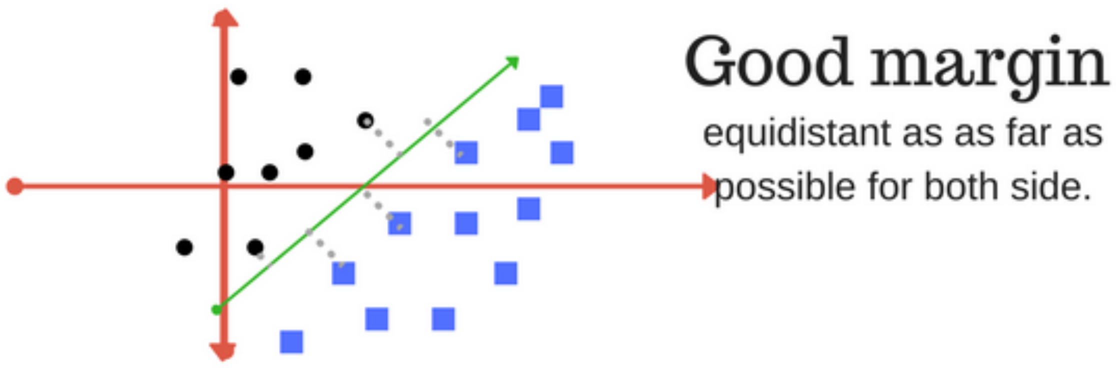


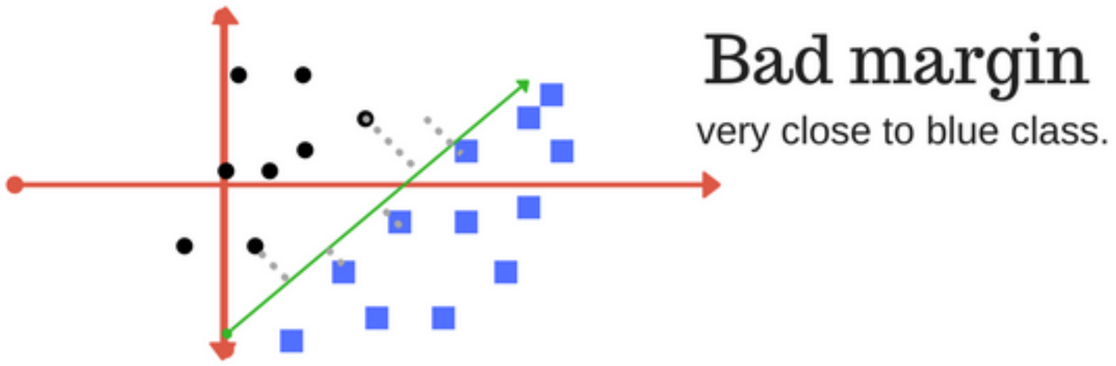


4. <u>Margin</u>

   A margin is a separation of line to the closest class points.

   A good margin is one where this separation is larger for both the classes. Images below gives to visual example of good and bad margin. A good margin allows the points to be in their respective classes without crossing to other class.


Good margin
equidistant as as far as possible for both side.


Bad margin
very close to blue class.

SVM Observations

With *C* as 1.0 and using RBF (Radial Basisi Function) as the kernel, we get the following results performing SVM on our datasets:

Testing on	**Accuracy**
MNIST dataset	96.13 %

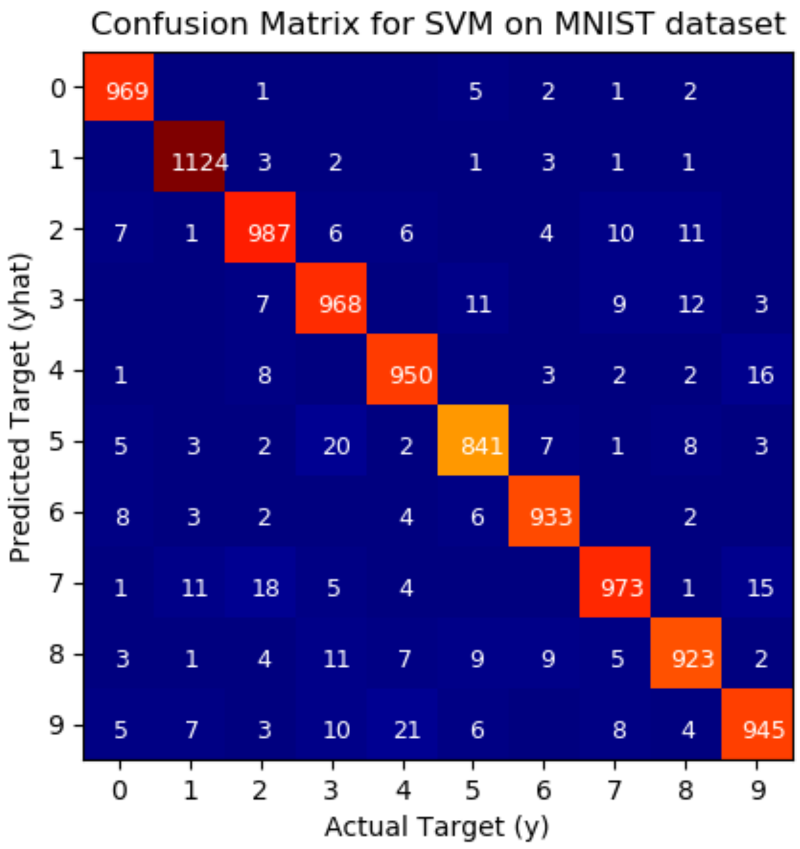
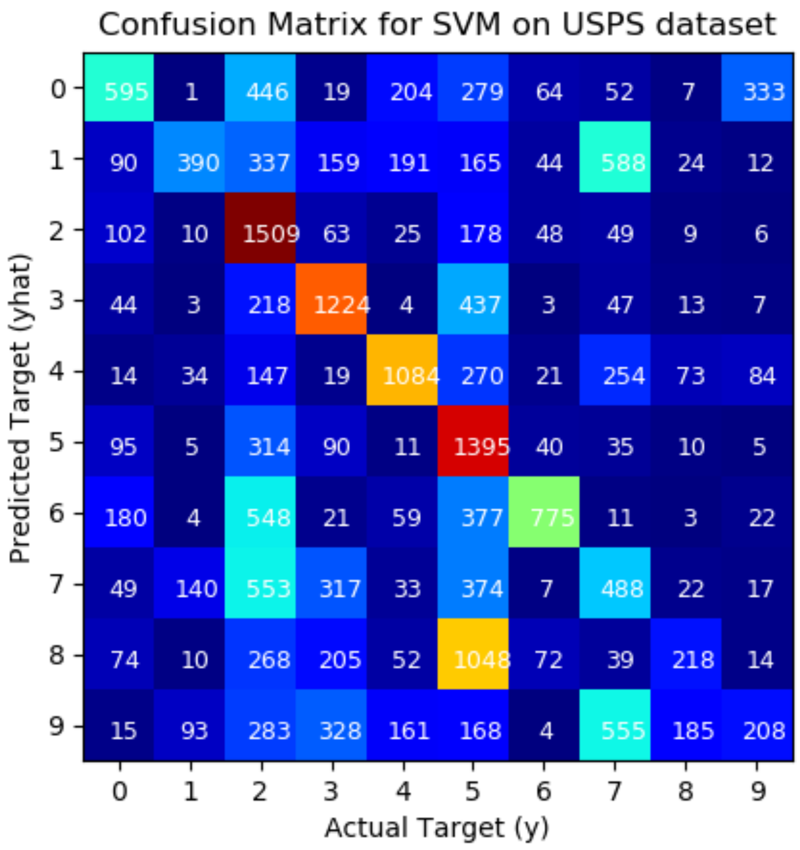


Fig 10 – SVM on MNIST dataset

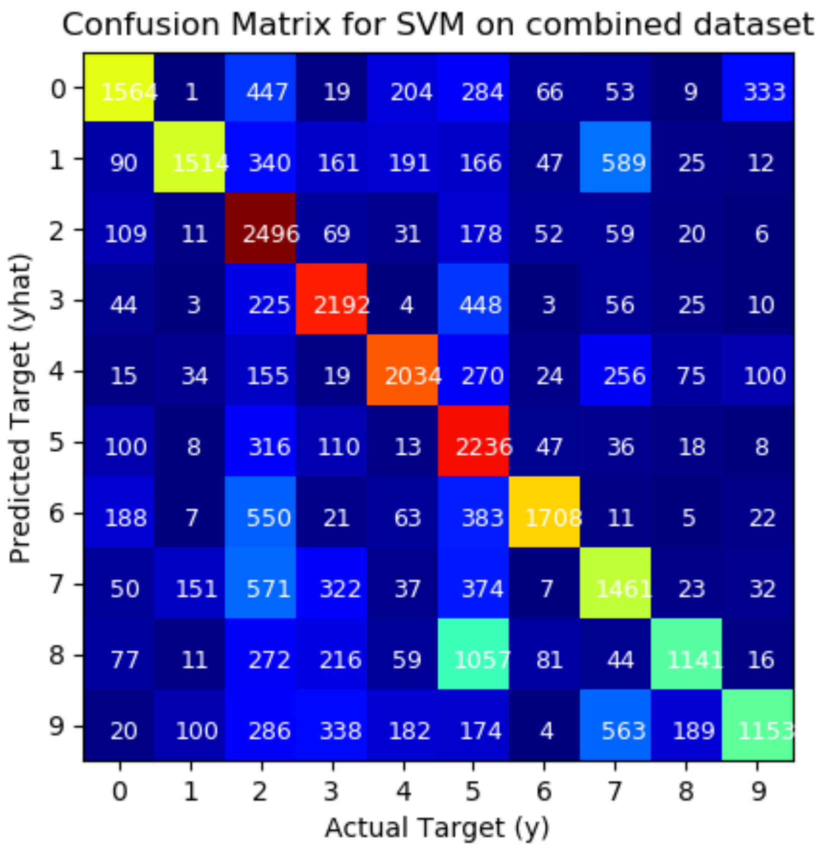
Testing on	Accuracy
USPS dataset	39.43 %



	0	1	2	3	4	5	6	7	8	9
0	595	1	446	19	204	279	64	52	7	333
1	90	390	337	159	191	165	44	588	24	12
2	102	10	1509	63	25	178	48	49	9	6
3	44	3	218	1224	4	437	3	47	13	7
4	14	34	147	19	1084	270	21	254	73	84
5	95	5	314	90	11	1395	40	35	10	5
6	180	4	548	21	59	377	775	11	3	22
7	49	140	553	317	33	374	7	488	22	17
8	74	10	268	205	52	1048	72	39	218	14
9	15	93	283	328	161	168	4	555	185	208

Fig 11 – SVM on USPS dataset

Testing on	**Accuracy**
MNIST + USPS datasets	58.33 %



	0	1	2	3	4	5	6	7	8	9
0	1564	1	447	19	204	284	66	53	9	333
1	90	1514	340	161	191	166	47	589	25	12
2	109	11	2496	69	31	178	52	59	20	6
3	44	3	225	2192	4	448	3	56	25	10
4	15	34	155	19	2034	270	24	256	75	100
5	100	8	316	110	13	2236	47	36	18	8
6	188	7	550	21	63	383	1708	11	5	22
7	50	151	571	322	37	374	7	1461	23	32
8	77	11	272	216	59	1057	81	44	1141	16
9	20	100	286	338	182	174	4	563	189	1153

Fig 12 – SVM on Combined datasets

Random Forest

Random Forest is a supervised learning algorithm. Like the name suggests, the algorithm creates a forest and makes it somehow random. The "forest" it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result.

To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. I will talk about random forest in classification, since classification is sometimes considered the building block of machine learning. Below you can see how a random forest would look like with two trees:

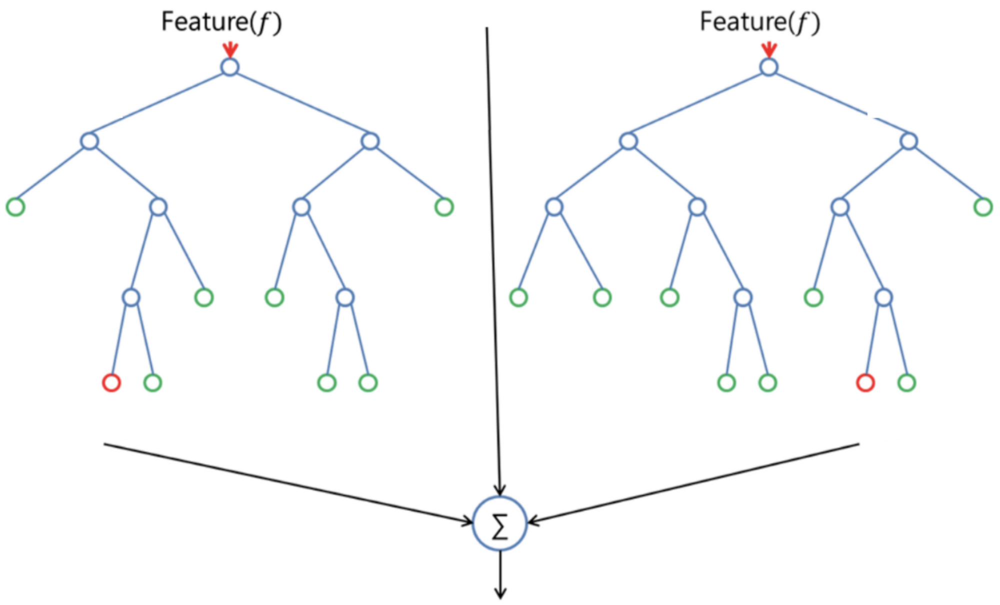


Fig 13 – Random Forest

Random Forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, you don't have to combine a decision tree with a bagging classifier and can just easily use the classifier-class of Random Forest. [3]

Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in Random Forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. [3]

Random Forest Observations

With *n_estimators* as 100, *max_depth* as 20 and *random_state* as 0, we get the following results performing Random Forest on the MNIST dataset.

Testing on	**Accuracy**
MNIST dataset	96.74 %

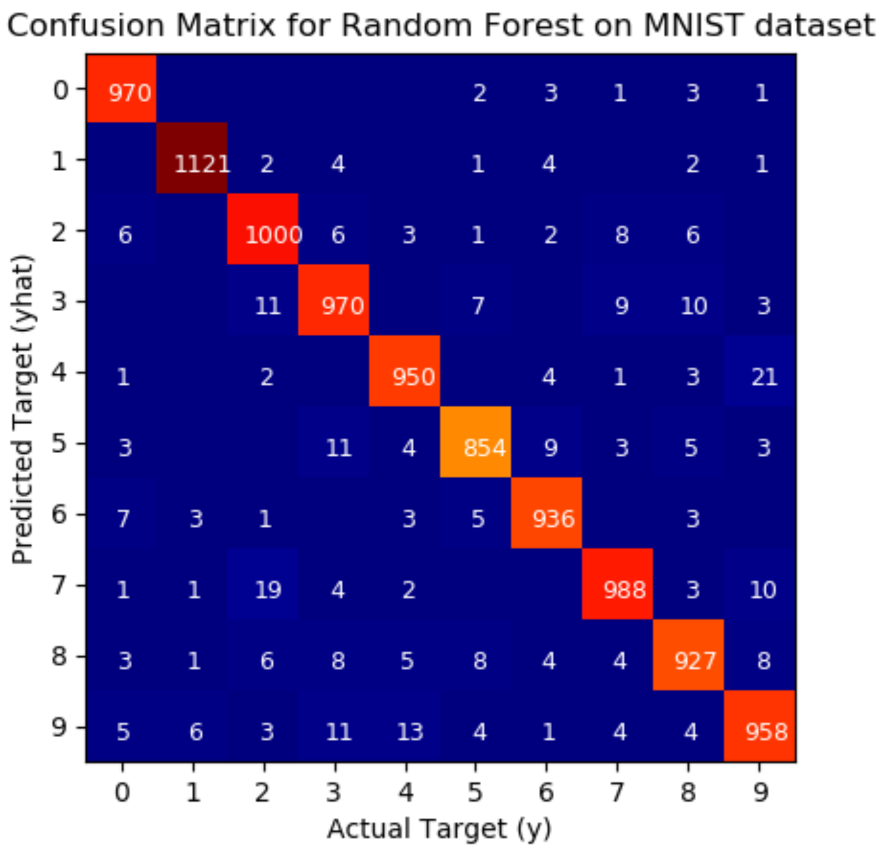
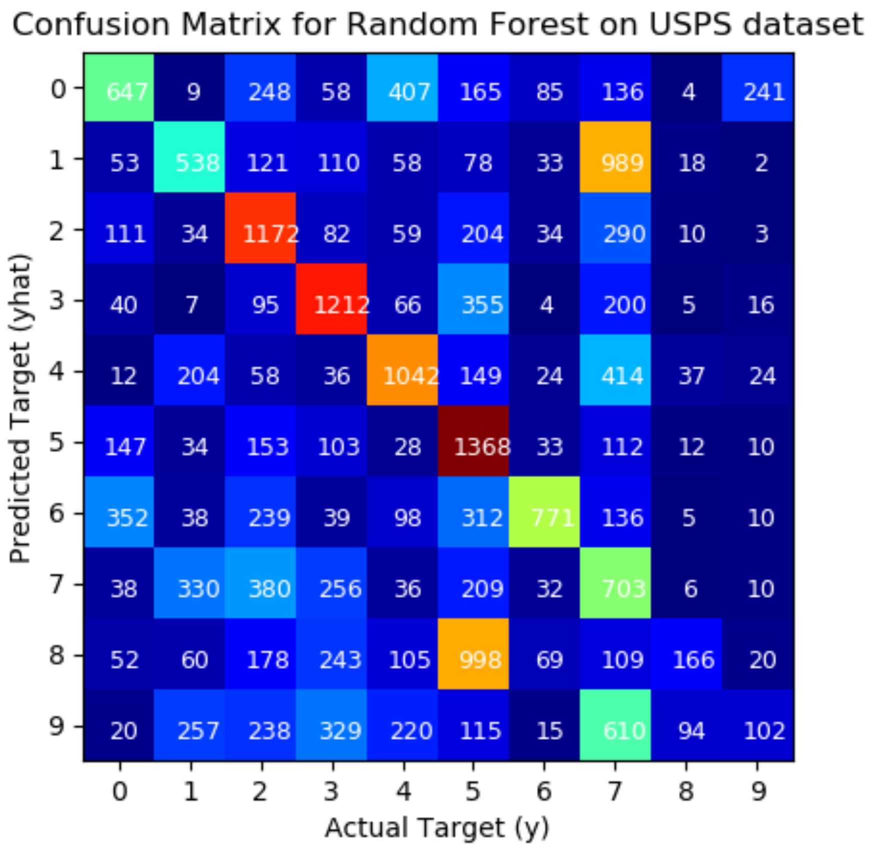


Fig 14 – Random Forest on MNIST dataset

Testing on	Accuracy
USPS dataset	38.89 %



	0	1	2	3	4	5	6	7	8	9
0	647	9	248	58	407	165	85	136	4	241
1	53	538	121	110	58	78	33	989	18	2
2	111	34	1172	82	59	204	34	290	10	3
3	40	7	95	1212	66	355	4	200	5	16
4	12	204	58	36	1042	149	24	414	37	24
5	147	34	153	103	28	1368	33	112	12	10
6	352	38	239	39	98	312	771	136	5	10
7	38	330	380	256	36	209	32	703	6	10
8	52	60	178	243	105	998	69	109	166	20
9	20	257	238	329	220	115	15	610	94	102

Fig 15 – Random Forest on USPS dataset

Testing on	Accuracy
MNIST + USPS datasets	58.18 %

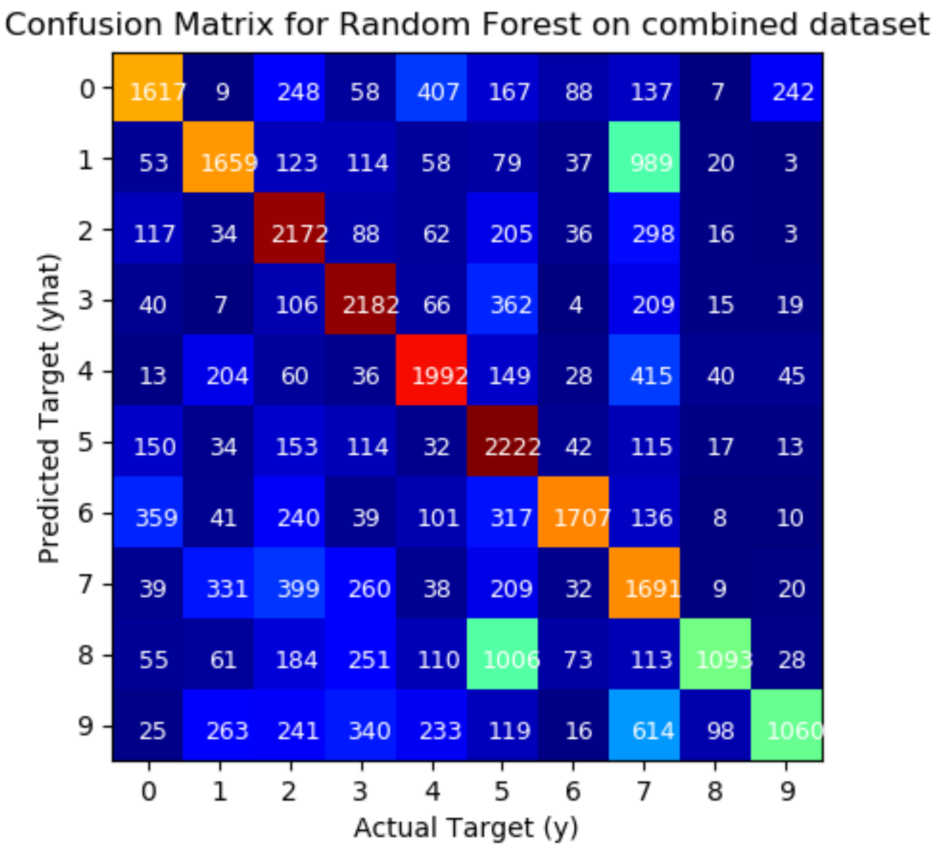


Fig 16 – Random Forest on Combined datasets

Ensemble of Classifiers

We combine our models using the concept of majority voting. In simple words, we find the ‘mode’ of the classifier predictions for each test sample.

Ensemble Observations

Testing on	Accuracy
MNIST dataset	97.48 %

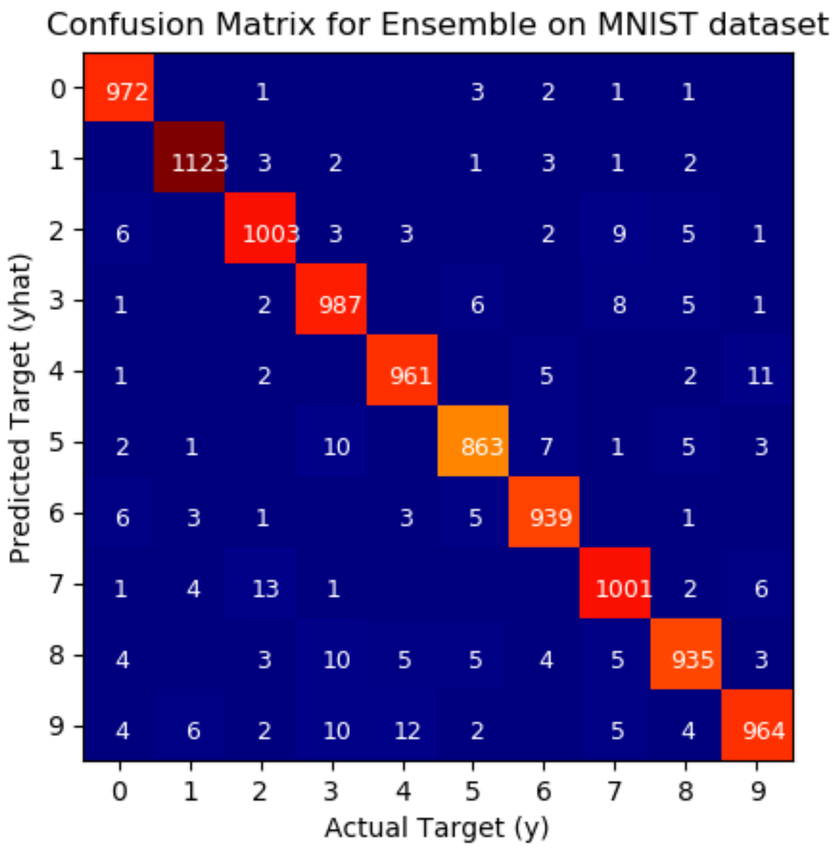


Fig 17 – Ensemble on MNIST dataset

The accuracy is very similar to that of Neural Network, but slightly lower than that of Neural Network.

Testing on	Accuracy
USPS dataset	44.29 %

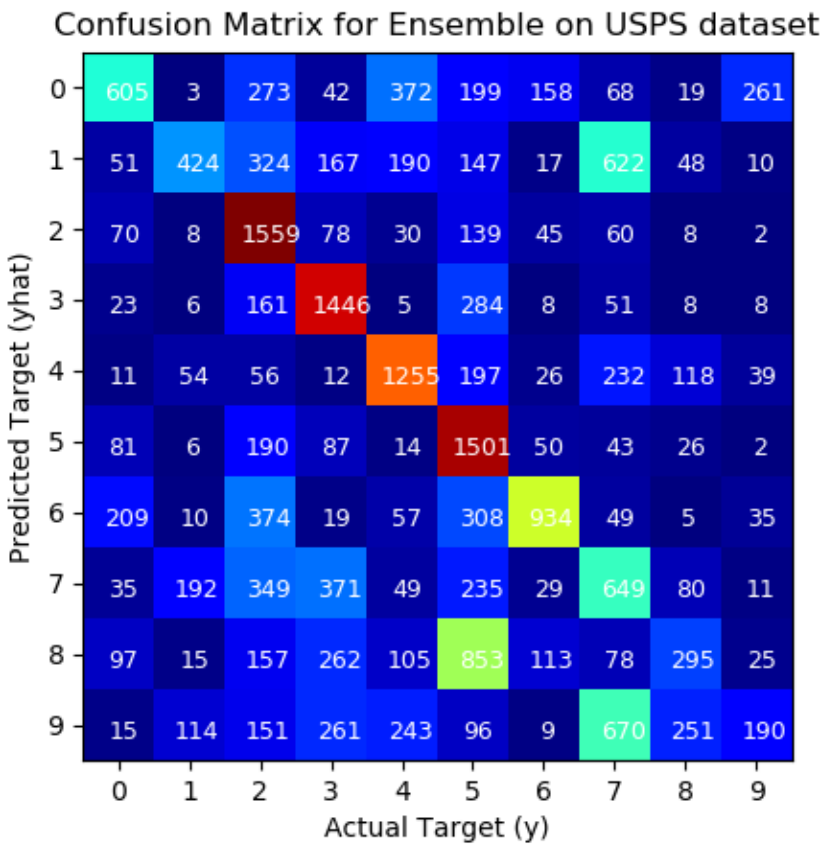
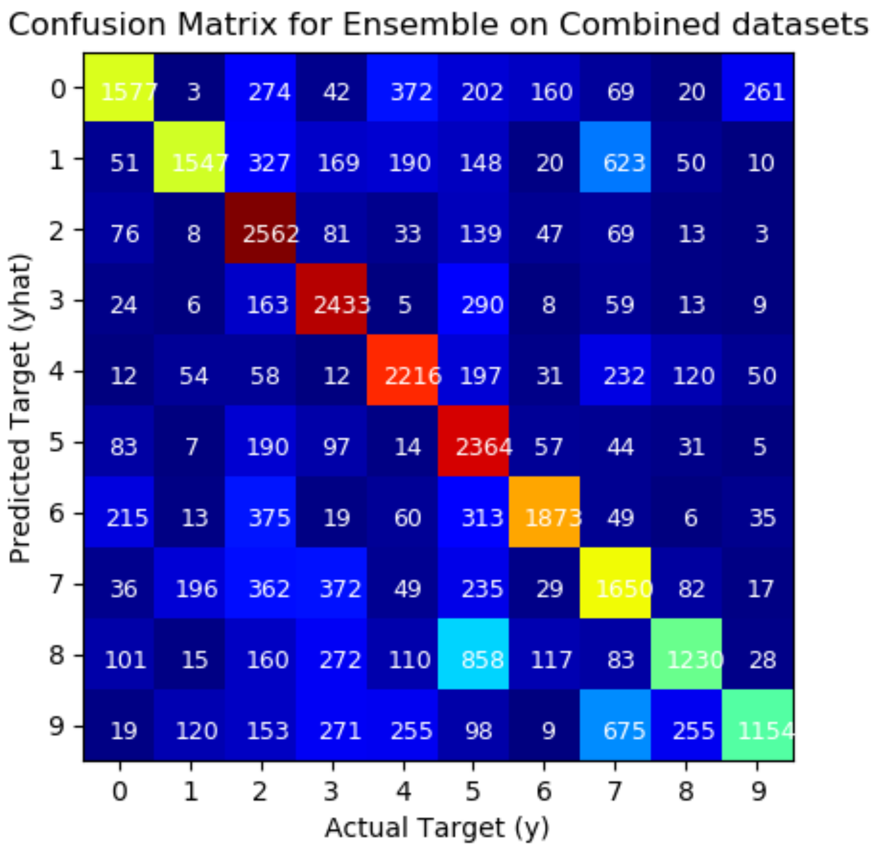


Fig 18 – Ensemble on USPS dataset

Like when tested on MNIST dataset, the accuracy of ensemble classifier when tested on USPS dataset is very similar to, but lower than that of Neural Network, which is the model with the highest accuracy on our test samples.

Testing on	Accuracy
MNIST + USPS datasets	62.02 %



	0	1	2	3	4	5	6	7	8	9
0	1577	3	274	42	372	202	160	69	20	261
1	51	1547	327	169	190	148	20	623	50	10
2	76	8	2562	81	33	139	47	69	13	3
3	24	6	163	2433	5	290	8	59	13	9
4	12	54	58	12	2216	197	31	232	120	50
5	83	7	190	97	14	2364	57	44	31	5
6	215	13	375	19	60	313	1873	49	6	35
7	36	196	362	372	49	235	29	1650	82	17
8	101	15	160	272	110	858	117	83	1230	28
9	19	120	153	271	255	98	9	675	255	1154

Fig 19 – Ensemble on MNIST + USPS datasets

Inferences

Classifier / Test on	MNIST dataset	USPS dataset	MNIST + USPS dataset
Logistic Regression	85.95 %	25.12 %	45.39 %
Neural Network	97.85 %	46.89 %	63.88 %
SVM	96.13 %	39.43 %	58.33 %
Random Forest	96.74 %	38.61 %	57.99 %
Ensemble	97.48 %	44.29 %	62.02 %

Table 1 – Accuracy Table

- We clearly observe that Logistic Regression has the poorest performance compared to the other three models. We can hence deduce that Logistic Regression is not a good choice for a model when we are required to perform multi-class regression.

- We observe that our models perform well when tested with MNIST datasets, however their accuracies drop considerably when tested on USPS dataset. Hence, we can deduce that the MNIST and the USPS datasets are quite different from each other, and training a model over only one of them, will result in lower accuracies when tested over the other.

- When we test on a combination of MNIST and USPS models, as we would expected, the accuracy results is better than that of testing on USPS dataset alone, but worse than testing on MNIST dataset alone. We also notice that the accuracy is more inclined towards that of the accuracy when testing on USPS dataset alone. This is simply because the USPS dataset has twice the number of samples than that of the MNIST dataset. Hence the average is weighed more towards the USPS dataset testing accuracy.

- One of the major inconsistencies we can observe in the confusion matrices of our models is the considerably high count of four being wrongly predicted as nine, and vice versa. Another similar inaccuracy is observed between classifying two and seven.
  We can hence deduce that the digits four and nine, and two and seven, look somewhat similar when handwritten.

- We observe that Neural Network has the best performance, both, on each test dataset individually and overall. Hence, we can say that the results we obtained, do not support the "No Free Lunch" Theorem. [^]

- Our ensemble of classifiers performs better than Logistic Regression, SVM and Random Forest classifiers, however is slightly poorer than that of Neural Network's performance. This holds true for all the three test scenarios i.e. testing on MNIST samples, testing on USPS samples and testing on a combination of MNIST and USPS samples.

  One possible explanation to this could be that the other classifiers (i.e. Logistic Regression, SVM and Random Forest) tend to predict incorrectly more often than Neural Network (as can be seen by their individual accuracies) but moreover, also tend to make the same incorrect prediction. Hence, they have a stronger vote in deciding the prediction of the ensemble.

---

^ The "No Free Lunch" Theorem states that if one algorithm performs better than another for some class of problems, then it must perform worse in the remaining problems.

References

1. Sayali Sonawane
   “What is the difference between Linear Regression and Logistic Regression”
   https://stackoverflow.com/questions/12146914/what-is-the-difference-between-linear-regression-and-logistic-regression

2. Savan Patel
   “Machine Learning 101 – Chapter 2”
   https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

3. Niklas Donges
   “The Random Forest Algorithm”
   https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd