Project -3

Ubit:v33 ub# : 50292192

Introduction :

In this project, we train the model using 4 classification algorithms and combine the results of the classifers to form the final decision. We use MNIST to train the data set and test using both MNIST and usps data for each classifiers.

Data preprocessing :

MNIST data has handwritten images from 0-9 each image of size 28 x 28 where as in usps data set image size is 16 x 16 . So we enlarge the size of usps dataset to 28 x 28 for testing. we take

Logistic regression :

We used 1-of-K coding scheme t = [t1, ..., tK]. for our multiclass classification task. The model can be represented as:

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

where the activation a_k are given by $a_k = w^{\top}_k x + b_k$.

The gradient of error is given by

$$\nabla_{\mathbf{w}_j} E\left(\mathbf{x}\right) = \left(y_j - t_j\right)\mathbf{x}$$

The weights are updated by the following method

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

We used Mini batch SGD method to update the weights. The computation of gradient error can be done by matrix multiplication and dividing the data in batches largely out performs the speed of computing the gradient error

The input has 60000 x 784 features which has to be classified into 10 models

lambda LR\	0.001	0.01	0.1
0.01	0.8658	0.8658	0.8658
0.03	0.8688	0.8688	0.8688
0.09	0.8724	0.8724	0.8724
0.3	0.8816	0.8816	0.8816
0.9	0.8932	0.8935	0.8932

Fig 1

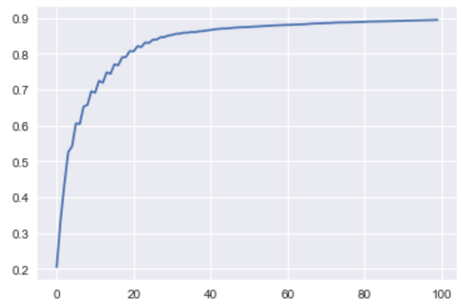
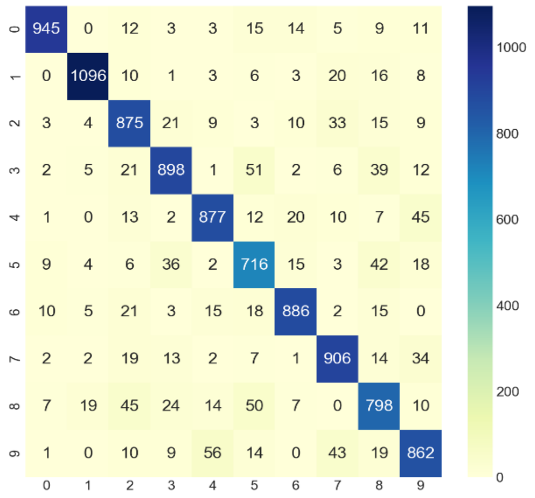


fig2 validation accuracy

We observe that the maximum accuracy obtained through softmax regression is 89.3. From fig 1 we can conclude that the best lambda and learning rate are 0.01 and 0.9 respectively.



	0	1	2	3	4	5	6	7	8	9
0	945	0	12	3	3	15	14	5	9	11
1	0	1096	10	1	3	6	3	20	16	8
2	3	4	875	21	9	3	10	33	15	9
3	2	5	21	898	1	51	2	6	39	12
4	1	0	13	2	877	12	20	10	7	45
5	9	4	6	36	2	716	15	3	42	18
6	10	5	21	3	15	18	886	2	15	0
7	2	2	19	13	2	7	1	906	14	34
8	7	19	45	24	14	50	7	0	798	10
9	1	0	10	9	56	14	0	43	19	862

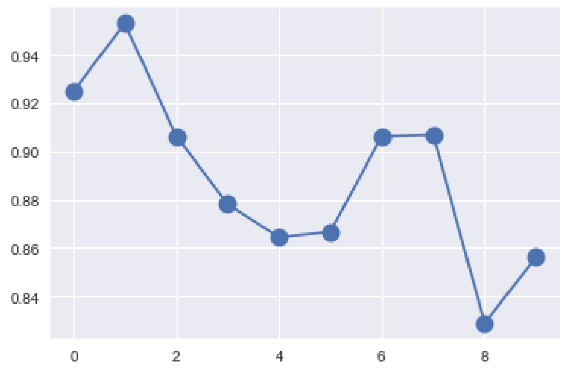


Fig3: Confusion matrix

Fig4: classwiseaccuracy

We observe that class “8” has the lowest test accuracy of 83.3 and class “1” has the highest accuracy of 95.5. On USPS data set we get an accuracy of 30.456.

Random forest :

Random forests are collection of decision trees.The models that are developed using decision trees require very simple heuristics. Decision trees are composed of a set of if-else statements done in a specific order. Thus, if the data set is changed to a new data set that might have some bias or difference in spread of underlying features compared to the previous set. The model will fail to be as accurate. This is because the data will not fit the model as well. So , we use bagging where in we sample the data and use individual sample to form n models .Essentially, these models run at the same time and predict on which hypothesis is most accurate . random forest classifier is basically bootstrap aggregation of decision trees

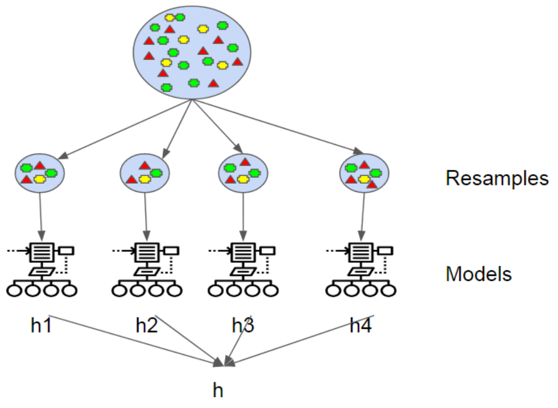
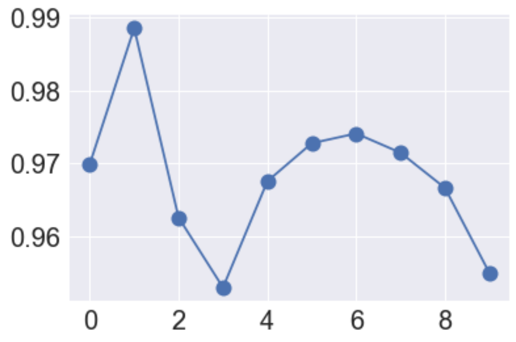
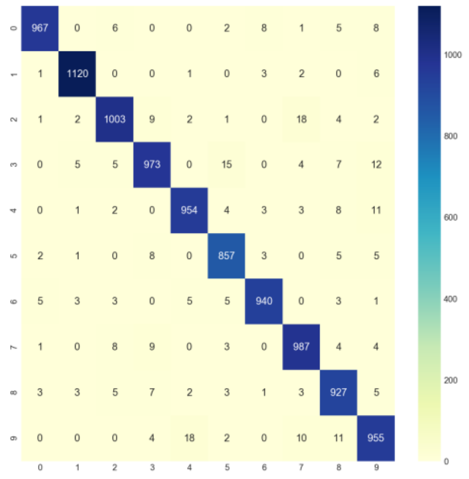


Fig -5

Accuracy of a random forest model mainly depends on two parameters 1. No of trees 2. Number of features each tree in the RF considers at each split. Default value of number of features is squrt(784)=28 We observe that as we increase the no of trees the accuracy increases rapidly at the beginning and accuracy growth variation is very small after 100 .

Max_feat→ No_of_est	**28**	**40**	**50**
10	0.8958	0.8976	0.8998
25	0.9275	0.9287	0.9266
50	0.9423	0.9473	0.9493
100	0.9686	0.9683	0.9687
125	0.9687	0.9683	0.9687





grid search accuracies   Class wise accuracy   confusion matrix

From the above graphs we observe that class "3" has the lowest accuracy of 95.2 and class "1" has the highest accuracy of 99. On usps dataset random forest classifier gives an accuracy of 41.

Neural Network:

We have implemented the feed forward with back propagation neural network. It works as follows:

A neural network has one input layer, one output layer and hidden layers. Weights are between input and hidden layer and other for the weights between hidden and output layer. And there are also weights between hidden layers. The product of input features and weights of layer 1 acts as input to the hidden layer where an activation function is applied to it. We have used relu as the activation function and softmax at the output layer.

The gradient of the error function is calculated as follows:

$$\frac{\partial E}{\partial w_{ji}^{(1)}} = \delta_j x_i, \qquad \frac{\partial E}{\partial w_{kj}^{(2)}} = \delta_k z_j$$

where

$$\delta_k = y_k - t_k$$

$$\delta_j = h'(z_j) \sum_{k=1}^{K} w_{kj} \delta_k$$

We use stochastic gradient descent algorithm to update the weights

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

Hyper Parameter Tuning (Lambda, Eta, Number of nodes,Number of layers):

For the single layer neural network, we need to tune certain hyper parameters like learning parameter eta, lambda and the number of nodes in the hidden layer. The typical values for eta and lambda lie between 0.01 to 0.1 and the number of nodes in the hidden layer should lie between the number of nodes in input layer and the number of nodes in the output layer.

We run a grid search for eta, lambda and number of nodes, train the neural network with the given parameters and test on the validation set and choose the parameter combination that gives the least classification error.

The values taken for eta for the grid search were: 0.01, 0.05, 0.1, 0.5

The values used for lambda for the grid search were: 0.01, 0.03, 0.1, 0.3

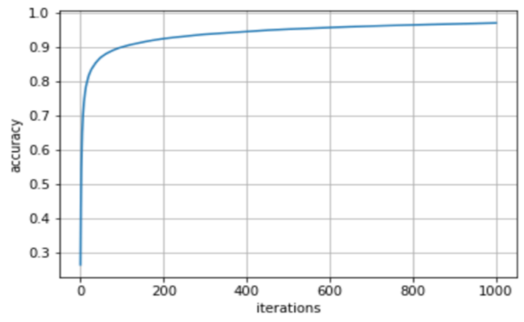
And, the number of nodes for the grid search were: 100, 150, 500, 700

Following are the accuracies on the validation data set for eta = 0.01 and various combinations of lambda and number of nodes

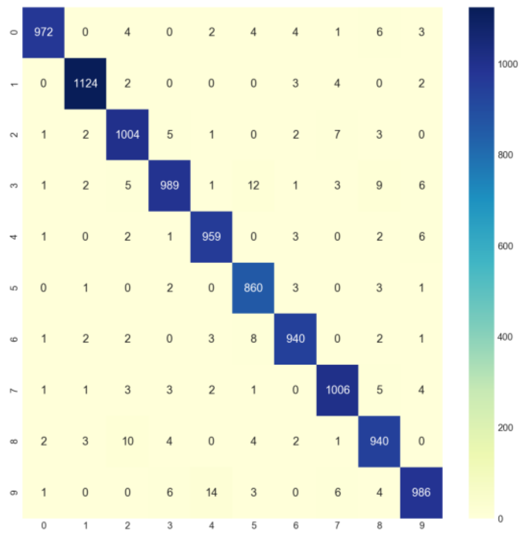
<--no. of nodes

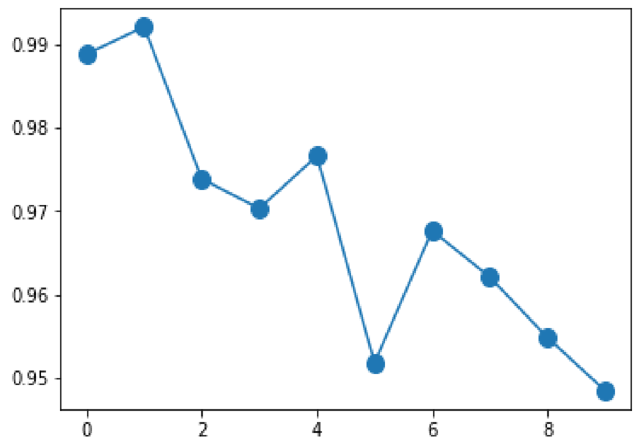
Lambda→	0.01	0.03	0.1	0.3
100	94.58	94.05	94.93	94.20
150	96.58	96.78	96.41	96.23
500	97.76	97.18	97.91	97.15

Grid search accuracies



validation accuracy





Confusion matrix for neural networks

Classwise accuracy for neural networks

After running the grid search for various values of eta, the optimal values obtained were:

Eta = 0.1, Lambda = 0.01, Number of hidden layer nodes = 300

Training data accuracy: 97.89%

Testing accuracy on MNIST dataset: 97.44%

Testing accuracy on USPS dataset: 45.80%

SUPPORT VECTOR MACHINE:

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In two dimentional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

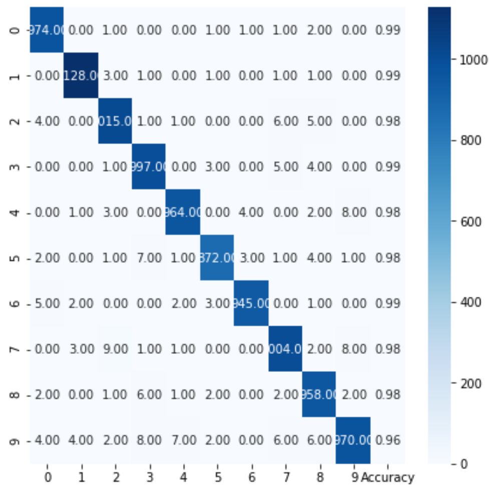
For radial basis function kernel we have 2 regulariztion parameters C and gamma . A standard SVM seeks to find a margin that separates all positive and negative examples. However, this can lead to poorly fit models if any examples are mislabeled or extremely unusual. To overcome this we use soft margin which allows us to ignore the data points which are on the other side of the line. C controls the infulence of support vector

Gamma

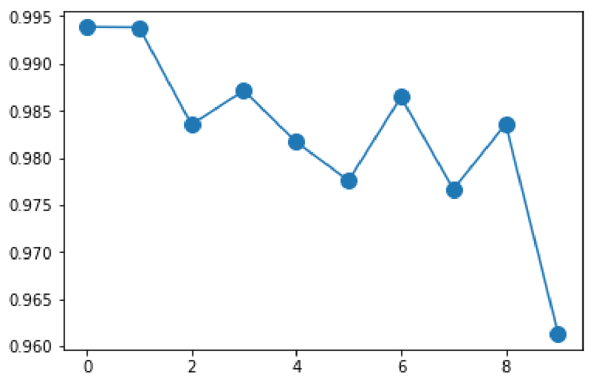
standard SVM is a type of linear classification using dot product. But it is possible to model more complicated relationships by replacing each dot product with a nonlinear kernel function (such as a Gaussian radial basis function or Polynomial kernel). Gamma is the free parameter of the Gaussian radial basis function.

$$K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2), \gamma > 0$$

A small gamma means a Gaussian with a large variance so the influence of x_j is more, i.e. if x_j is a support vector, a small gamma implies the class of this support vector will have influence on deciding the class of the vector x_i even if the distance between them is large**.** If gamma is large, then variance is small implying the support vector does not have wide-spread influence. Technically speaking, large gamma leads to high bias and low variance models, and vice-versa



Confusion matrix for SVM



Classwise accuracy for SVM

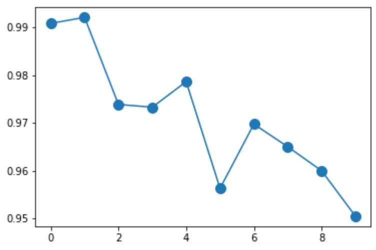
Linear kernel	Default	0.92
RBF	Gam=0.05,C=2	0.98
RBF	Gam = 1, C=1	0.96

Majority voting rule ensemble classifier:

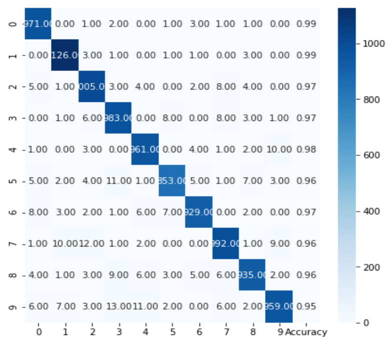
Now, we will implement a simple EnsembleClassifier class that allows us to combine the three different classifiers. We define a predict method that let's us simply take the majority rule of the predictions by the classifiers. E.g., if the prediction for a sample is

- classifier 1 -> class 1
- classifier 2 -> class 1
- classifier 3 -> class 2

we would classify the sample as "class 1."



Class wise acc



conf matrix

conclusion:

The free lunch theorem states that any two optimization algorithms are equivalent when their performance is averaged across all possible problems. In terms of machine learning, a highly optimized model might not work well for any set of new data. For this project assignment we developed models for hand written digit recognition using MNIST dataset as input and techniques like multiclass logistic regression, single layer neural network and convolutional neural networks with training accuracies of 89.15%, 97.06% and 99.89% respectively. However, these models give only 36.58%, 39.80% and 40.8% accuracy on USPS dataset. This proves the free lunch theorem.