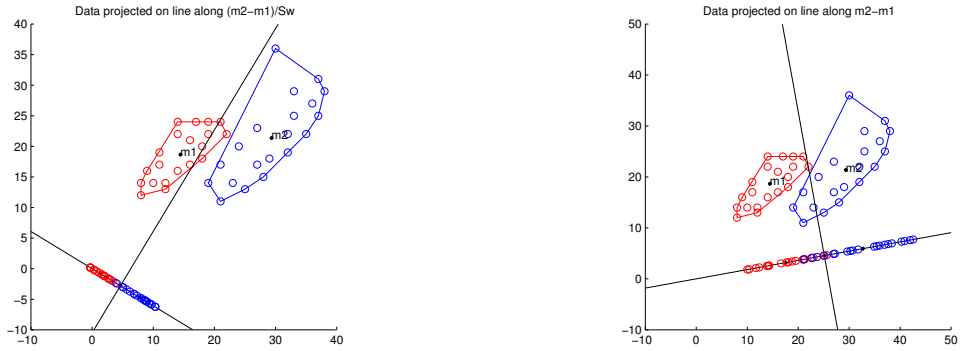
Q.1 To determine whether $w_o = {S_W}^{-1}(m_1 - m_2)$ or $w_o = (m_1 - m_2)$ should be used as solution, so as to maximize $J(w) = \frac{w^T S_B w}{w^T S_W w}$, I conducted the following experiment. A data set was obtained which is shown in Fig. 1. The sample mean M_1 and M_2 and sample covariance marix S_1 and S_2 were calculated. $S_W = S_1 + S_2$ is the within class covariance matrix. $S_B = (M_1 - M_2)^T(M_1 - M_2)$ is the between class variance.

(1) $w_o = {S_W}^{-1}(m_1 - m_2)$

Fig. 1(a) shows the line on which data set was projected alongwith the projected means of the two classes. The direction of this line was along $w_o = {S_W}^{-1}(M_1 - M_2)$. The Euclidean distance between the two projected means was 7.3673 and the error in classification was 2.5%.

(2) $w_o = (m_1 - m_2)$

Fig. 1(b) shows the line along $w_o = (M_1 - M_2)$, on which the data was projected. The Euclidean distance between two projected means was 15.389 and the error in classification was 10%



(a) ${S_W}^{-1}(M_1 - M_2)$ (b) $(M_1 - M_2)$

Figure 1: Fisher's Linear Discriminant Function

From the above figures it is clear that higher classification accuracy can be achieved if within class variance(S_W) is also considered while maximizing J(w) function. The reason for this is, it may be possible that the sample means of the two classes be far apart in one direction but at the same time the overlapping area of the projected samples of both classes may also be very large as shown in Figure 1(b). Due to this, evenif the means are far apart, it may not be possible to accurately classify the data points lying in the overlapping region. To ensure better accuracy, the overlapping area of the two classes, in the projected plane, should be as small as possible while the distance between two means should be as large as possible. The overlapping area can be determined from the within class variance S_W. Thus both S_W and $(M_1 - M_2)$ must be considered while determining the direction of line on which the data should be projected.

Q.2 The data in Q.2 and Q.3 was obtained from

http://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits.

Data set description: The data is a set of handwritten digits from 0 to 9. There are 16 features for each digit. For my experiment I selected data and feature vector corresponding to digits 0 and 1 so as to have two class classification problem. The data was divided into two parts - training set and testing set.

(1) 16 dimensional data

(a) SVM

Two different codes for Support vector machine were used. First one was SVMlight obtained from http://svmlight.joachims.org/ and the second one was Matlab's inbuilt function. SVM-light is an implementation of Vapnik's Support Vector Machine for the problem of pattern recognition, for the problem of regression, and for the problem of learning a ranking function. It uses various standard kernels defined, to obtain support vectors during the training of support vector machine. The optimization algorithms used in this software can be obtained from Thorsten Joachims, Learning to Classify Text Using Support Vector Machines and Dissertation, Kluwer, 2002 and T. Joachims, 11 in: Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schlkopf and C. Burges and A. Smola (ed.), MIT Press, 1999. The data set was first converted in the format identified by SVMlight. This code gives accuracy achieved in learning and classification process as output. It has two file- one for learning SVM and other for classifying the test data based on learning parameters obtained.

The following results were obtained on training set:

Error = 2.44%
Accuracy = 97.56%
Number of kernel evaluations = 17540
Number of SV = 45

The following results were obtained on test set:

Error = 2.34%
Accuracy = 97.66%

(b) Neural Network

The source code for neural network was obtained from

http://www.philbrierley.com/main.html?code/index.html&code/codeleft.html.

It uses feedforward and back propagation technique to train the network. Backpropagation if done using gradient descent method. As the code only trained the network, a code was written to test the trained network. There are 3 layers- input, hidden and output. Number of neurons in input layer depends on the size of feature vectors. Number of neurons in hidden layer can be specified by user. The output have only one neuron making it a NN for classifying two classes. If the output < 0 then it belongs to class 1 and if output ≥ 0 then is belongs to class 2. Table 1 shows the training and testing error with learning rate = 0.1 as a function of number of neurons in hidden layer. A function tanh() was used at hidden layer for

Table 1: Training and testing error as a function of number of neurons in hidden layer

Number of neurons	Training Error(percent)	Testing error(percent)
1	5.97	8.53
5	6.20	8.67
10	5.03	9.63

transforming input values to output values. The bias is provided at input by concatinating the input feature vector with a 1.

Table 1 shows that, increasing number of neurons in hidden layer causes decrease in the accuracy of the network. This behaviour can be accounted to overfitting of training data set. Overfitting may achieve small training error but may increase the testing error. So the optimum number of neurons in hidden layer can be decided based on experiments or if there is some prior knowledge of the data.

Table 2 shows the training and testing error as a function of learning rate. Smaller value of learning rate causes slow convergence of the network while larger value causes fast convergence. While it is not possible to show it here, it was observed that for larger values of learning rate, the error in output fluctuated by large values and for smaller values of learning rate, the error in the output changed by small values before settling to the final value.

Table 2: Training and testing error as a function of learning rate with number of neurons in hidden layer = 5

Learning rate	Training Error(percent)	Testing error(percent)
0.05	8.42	8.39
0.1	6.20	8.67
0.5	1.29	6.88

Thus the learning rate = 0.5 achieves better classifier accuracy.

(c) Comparing SVM and NN

From the above given results it is clear that SVM performs better than NN on this data set. As SVM uses different kernels in estimating the partition hypercurve, it optimizes SVM and selects the best kernel for obtaining higher accuracy. As against that, the NN uses only single kernel tanh() and is the most general form of NN. More complicated and data specific NN can be designed to achieve better accuracy in classifying data.

(2) 2-Dimensional data

Another experiment was performed with data set having only two feature vectors. This data set was obtained from the original data set by taking two feature vectors that affected the decision, the most. Table 3 shows the error in classification using SVM. Two different kernels were used - linear and quadratic. Fig. and Fig. show the data and partition hypercurve obtained. Table 4 shows the error in classification using NN. The number of neurons in hidden layer was 1 and the results for two different learning rates is shown in the table.

Table 3: Classification error as a function of kernel in SVM

Kernel	Error(percent)
Linear	5.88
Quadratic	1.96

Table 4: Error as a function of learning rate in NN

Learning rate	Error(percent)
0.1	3.92
0.05	1.96

Figure 2: SVM with linear kernel

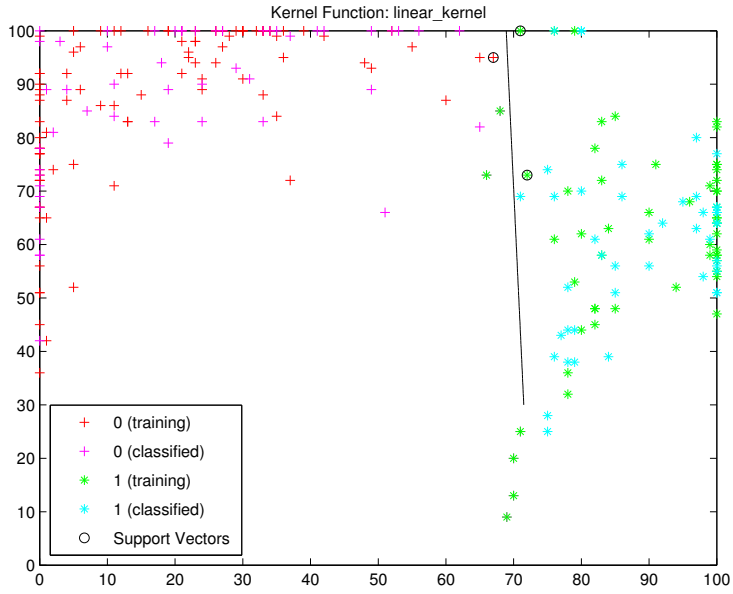
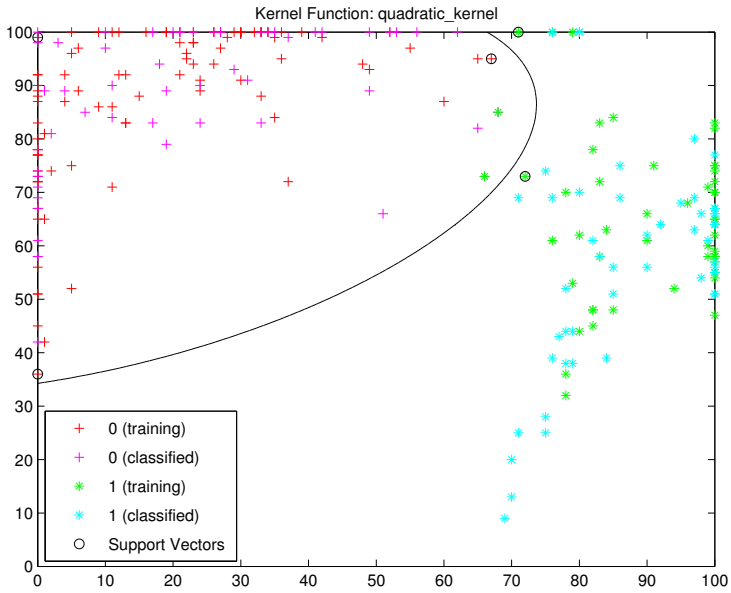


Figure 3: SVM with quadratic kernel



Conclusion

It can be observed that in two dimensions, the accuracy of SVM and NN is almost same. Though the number of observations were decreased, still it can be seen that having more features doesnot imply higher accuracy. The feature that has maximum influence in deciding the class of the data should be selected and if desired accuracy is not achieved, then only higher dimensions should be considered.

(Q.3) Nonparametric classifiers

(1) Parzen Window

The data set for this question was same as in question 2 with two features. In Parzen window the size of the window is kept fixed and the number of training data belonging to each class within that window are counted. The new data is classified as belonging to the class which has maximum number of training data present in the window centered at the new data point. A square window was considered for this experiment. The training data set size for each class was 200 observations. And this was tested on 100 test data observations having equal number of data belonging to both the classes. Table 5 shows the classification error as a function of window size.

Table 5: Classification error as a function of window size in Parzen window classifier

Window size(hn)	Error(percent)
20	2.33
30	1.67
40	1.33
50	1.33
70	2.00

(2) k-nearest neighbor

In k-nearest neighbor classifier, we find the k-nearest neighbors in training data set from the new data point which we intend to classify. Then the class which has maximum data points from training data set, is assigned to the new data set. For this experiment too, I took same data as above. First the distance to all the points in training data set was found and sorted in ascending order. Then top k distances and data points corresponding to them were taken. Out of them, the class that had maximum occurance, was assigned to the new unclassified data point. Table 6 shows the classification error as the function of number of neighbors considered.

Table 6: Classification error as a function of number of neighbors considered

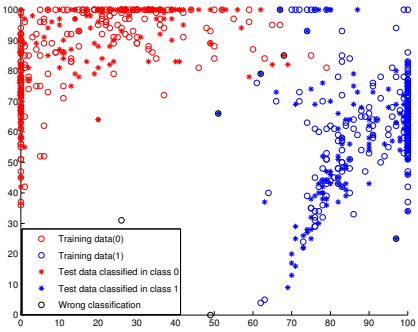
Number of neighbors(kn)	Error(percent)
10	1.33
20	0.67
30	1
50	1.33
70	1.33

(3) Nearest Neighbor

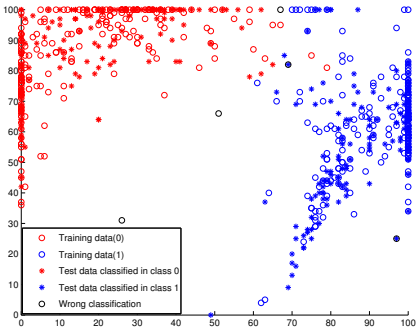
In nearest neighbor classifier, the distance between unclassified data point and all the training set data is found. I considered Euclidean distance as I had 2-dimensional data. The class of the data point having smallest distance is assigned to the unclassified data point. Using same data as in part (1) and (2), I got error of 2% with this classifier. Figure 6 shows the classifier output.

(4) Conclusion

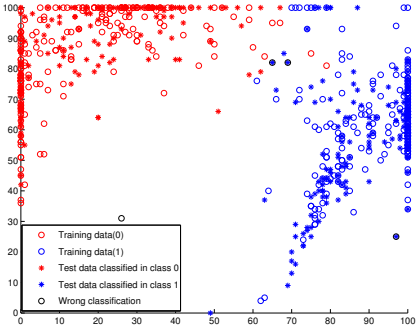
From the above reults it can be concluded that k-nearest neighbors gives then highest accuracy, next is Parzen window with optimum window size determined. Nearest neighbor has the least accuracy when data set is not well separated as in this case. Also for Parzen window, increasing the size of the window doesnot ensure higher accuracy. Too small and too large window sizes give more error as compared to mid-sized windows. Similarly for k-nearest neighbors, increasing the number of nearest neighbors considered for classification, doesnot guarantee higher classification accuracy. In fact after some value of kn, increasing kn doesnot have any effect on the classification which can be seen from kn = 50 and kn = 70.



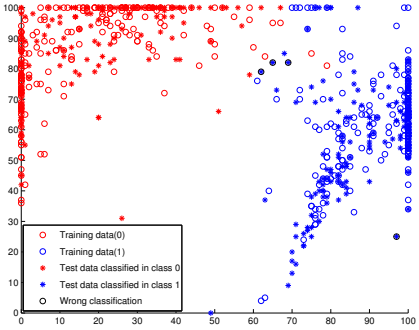
(a) hn = 20



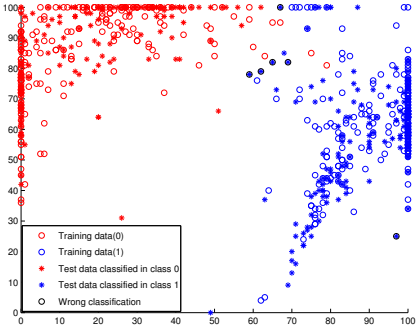
(b) hn = 30



(c) hn = 40

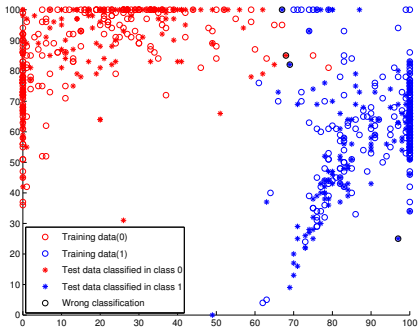


(d) hn = 50

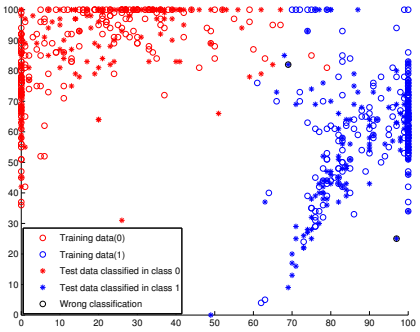


(e) hn = 70

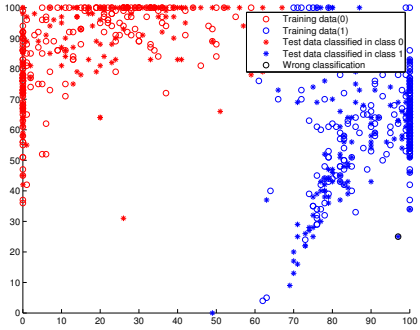
Figure 4: Parzen Window classifier with different window sizes



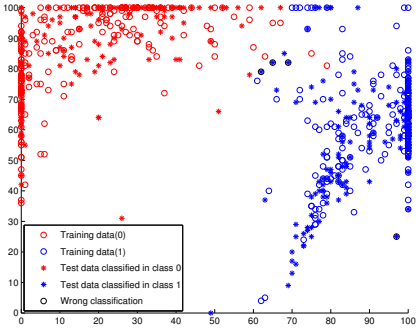
(a) kn = 10



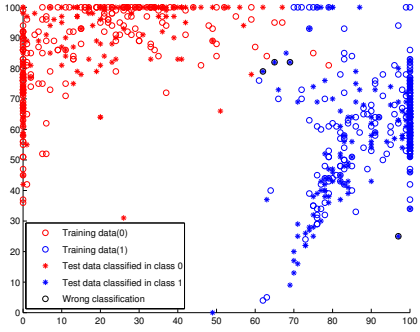
(b) kn = 20



(c) kn = 30



(d) kn = 50



(e) kn = 70

Figure 5: k-nearest neighbors classifier with different k values

Figure 6: Nearest neighbor classifier