Name: Pratik Mohan Ramdasi

ECE 656: Machine Learning and Adaptive Systems

Computer Assignment 3

Date: 11/17/2015

Title: Pattern Classification using SVM

SECTION 1: Introduction

The purpose of this computer assignment is to design a Support Vector Machine (SVM) for pattern classification. The MATLAB neural network (NN) toolbox is used to train and test SVM. The data for the study is the 'wine-recognition' dataset.
The dataset is the result of a chemical analysis of wines grown in a particular region in Italy but derived from three different cultivars (or classes). The analysis determined the quantities of 13 constituents found (i.e. 13-D feature space) in each of the three types of wines.
The attributes are:

1. Alcohol content
2. Malic acid
3. Ash
4. Alkalinity of Ash
5. Magnesium
6. Total Phenol
7. Flavonoid
8. Non-flavonoid Phenols
9. Proanthocyanins
10. Color intensity
11. Hue
12. OD280/OD315 of diluted wines
13. Proline

In Machine Learning, Support Vector Machines are supervised learning models used for classification and regression analysis. Given a set of training examples, each marked with the labels of the corresponding class, SVM builds a model that assigns new examples into the two classes, making it non-probabilistic binary linear classifier.
Neural Networks such as back propagation and RBF use MSE as a global measure and does not consider data structure or generalization ability however, SVM is based on Structural Risk Minimization (SRM) which defines an upper bound on generalization error rate.
With our 'wine' dataset, SVM will be suitable choice for classification as the number of training samples is small. Since, SVM is a 2 class classifier, we have to use it twice to separate dataset into 3 classes.
Computer Assignment is organized as follows:
Section 2: Architecture of SVM
Section 3: Stepwise Procedure
Section 4: Results
Section 5: Discussions and Comments
Section 6: Conclusion
Section 7: References

SECTION 2: Architecture of SVM

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.
A support vector machine constructs a 'hyperplane' or set of hyperplanes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. A good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (called as functional margin), since in general the larger the margin the lower the generalization error of the classifier. If such hyperplane exists, it is known as the 'maximum-margin hyperplane'.

Types of SVMs:

1. *Hard- Margin SVM*: these are linearly separable in feature space and they maximize generalization ability.
2. *Soft- Margin SVM*: They are not separable in feature space and they minimize classification error and maximize generalization ability.

Linear SVM:

Linearly separable classes:
Given training data as a sequence of l labeled points in R^n , the task is to find an "optimal" hyper-plane separating them. There is a factor here that we have to take into account: linear separability of training data.
For the linearly separable case, the support vector algorithm simply looks for the separating hyper-plane with the largest margin, where margin = $2d$ and d is the distance from the hyper-plane to the nearest positive or negative example. It turns out that the margin is inversely proportional to the absolute value of hyper-plane's normal $||w||$. This reduces the original problem to the following optimization problem:

$$\text{Minimize}||w|| \text{ subject to } \quad y_i(w.x_i + b) - 1 \geq 0$$

Where (x_i, y_i) are the training examples and b is the bias term.
$y_i \in \{-1, 1\}$ so that we define

$$x \in C1 \ for\ y_i = 1 \quad then\ (w.x_i + b) \geq 1$$

and

$$x \in C2 \ for\ y_i = -1 \quad then\ (w.x_i + b) \leq 1$$

Support Vectors are the samples for which inequalities become equalities. The margin of separation is given by $\rho = 2/||w||$.
Where $||w||$ represents norm of w.
SVM solution for the linearly separable problem is shown in the following figure.

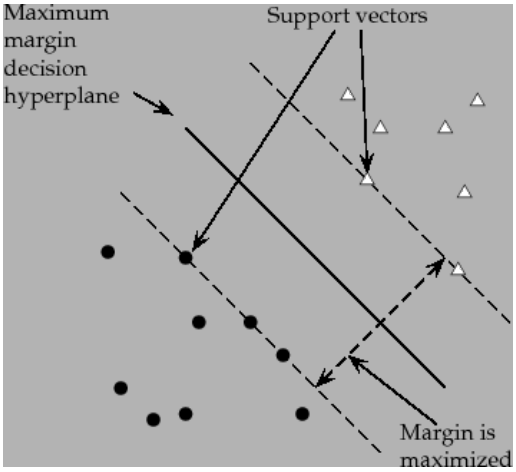


Figure 1. SVM for linearly separable case

The solid line is the solution hyperplane, margin is the distance between the parallel lines and the points on the parallel lines are marked “Support Vectors”.
The number of support vectors are small compared to the training dataset. It is also obvious that the solution hyperplane is defined only by the support vectors i.e. if we remove all the other points, the hyperplane will not be affected.

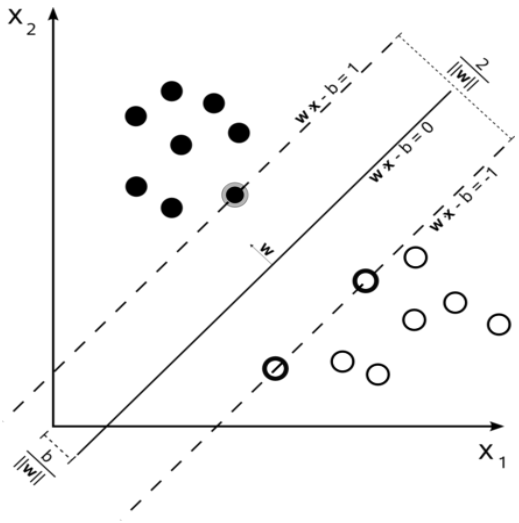


Figure 2. separation of two classes

Primal form:

The optimization problem presented above is difficult to solve as it depends on $||w||$, the norm of w which involves a square root. So, the minimization objective is altered by substituting $||w||$ with $\frac{||w||^2}{2}$ without changing the solution and the resulting Lagrangian cost function is:

$$J(w, b, x) = \frac{||w||^2}{2} - \sum_{i=1}^{n} \alpha_i \left[y_i(w.x_i + b) - 1 \right]$$

where the first term is a convex function in w and constraints are linear in w and $\alpha_i > 0$ are called *Lagrange Parameters.* The solutions are the *saddle points.* The saddle point has to be minimized wrt $w \; and \; b$ but maximized wrt α.

Solving the partial derivatives, we obtain:

$$w = \sum_{i=1}^{n} \alpha_i \, y_i \, x_i$$

and $$\sum_{i=1}^{n} \alpha_i \, y_i = 0$$

At solution, for each Lagrange multiplier, we have $\alpha_i \, [\, y_i(w.x_i + b) - 1] = 0$, called as *Kuhn-Tucker condition.*

Dual Form:

Writing the classification rule in its unconstrained dual form reveals that the *maximum-margin hyperplane* and therefore the classification task is only a function of the *support vectors*, the subset of the training data that lie on the margin.
Maximize objective function:

$$Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \sum_{i,j} \alpha_i \, \alpha_j y_i \, y_j \, x_i^t x_j$$

$$= \sum_{i=1}^{n} \alpha_i - \sum_{i,j} \alpha_i \, \alpha_j y_i \, y_j \, k(x_i \, , x_j \,)$$

subject to $\alpha_i \geq 0$ and to the constraint from the minimization in b, $\sum_{i=1}^{n} \alpha_i \, y_i = 0$
Here, the kernel is defined by $k(x_i \, , x_j \,) = x_i \, . x_j$ and $w = \sum_{i=1}^{n} \alpha_i \, y_i \, x_i$

For simplicity reasons, sometimes it is required that the hyperplane pass through the origin of the coordinate system. Such hyperplanes are called *unbiased*, whereas general hyperplanes not necessarily passing through the origin are called *biased*.

Linearly non-separable classes:

Our goal here is to find an optimal hyperplane for non- linearly separable patterns that minimizes probability of misclassification error over the training set.
The margin of separation is said to be '*soft*' if a data point violates condition

$$y_i(w.x_i + b) - 1 \geq 0$$

this can happen in one of the two cases:

- Case 1: samples between canonical hyperplanes but on the right side of the decision surface. No misclassification.
- Case2: samples between canonical hyperplanes but on the wrong side of the decision surface. Cause misclassification.

The method introduces non-negative *Slack Variables*,$\xi_i > 0$, measure of deviation of a sample from the ideal condition of separability,

$$y_i(w.x_i + b) \geq 1 - \xi_i, \forall \, i \in [1, n]$$

Primal form:

To minimize the classification error, we use the metric, $\psi(\xi) = \sum_{i=1}^{n} \xi_i$ which takes higher values for more misclassifications.
The objective function is then increased by a function which penalizes non-zero ξ_i, and the optimization becomes a trade off between a large margin and a small error penalty. If the penalty function is linear, the optimization problem becomes:

Find $w\ and\ b$ such that

$$\psi(w, \xi) = \frac{||w||^2}{2} + C \sum_{i=1}^{n} \xi_i$$

is minimized subject to

$$y_i(w.x_i + b) \geq 1 - \xi_i \qquad and \qquad \xi_i > 0 \qquad \forall\, i \in [1.n]$$

where C is the user specified parameter that presents a trade off between misclassification error and maximum margin. Thus the modified objective function becomes

$$J(w, b, \alpha, \xi) = \frac{||w||^2}{2} + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i [\, y_i(w.x_i + b) - 1 + \xi_i \,] - \sum_{i=1}^{n} \mu_i \xi_i$$

Taking partial derivatives wrt $w, b\ and\ \xi_i\ (at\ saddle\ points)$ and setting to zero gives $w = \sum_{i=1}^{n} \alpha_i\, x_i \alpha_i,\ \sum_{i=1}^{n} \alpha_i\, y_i = 0$ and $\mu_i + \alpha_i = C$

The free coefficient b can be found from

$$\alpha_i [\, y_i(w.x_i + b) - 1 + \xi_i \,] = 0$$

The *Kuhn-Tucker condition* in this case can be given by

$$\alpha_i [\, y_i(w.x_i + b) - 1 + \xi_i \,] = 0$$

Dual form:

Dual form is same as before except the second condition is changed to,

$$0 \leq \alpha_i \leq C \ , \forall\, i \in [1, n]$$

Non-linear SVM:

The treatment of the most interesting, non-linear, case is based on the idea of injecting the data points into a higher-dimensional Hilbert space via some non-linear mapping, and using the linear support vector algorithm there to separate the training examples. This makes use of the fact that the data appears in the training problem only in the form of dot products.
Suppose there was a map $\Phi: R^d \rightarrow$ H, from our 'data space' into a higher-dimensional Hilbert space H. If there were a function evaluating dot products in H inexpensively, $K(x_i, x_j) = \Phi(x_i).\Phi(x_j)$, then we could train our machine in H and would never need an explicit form of Φ.
In our algorithm for linear SVM, we substitute all dot products $x_i . x_j$ with $K(x_i, x_j)$ the *kernel* function, and we will end up with a machine that does a linear separation but in a different Hilbert space. Our classifier then will be concerned with the sign of $f(x)$, where

$$f(x) = \sum_{i=1}^{n} \alpha_i \ y_i \Phi(s_i).\Phi(\mathrm{x}) + \mathrm{b} = \sum_{i=1}^{n} \alpha_i \ y_i K(s_i, x) + b$$

s_i are the support vectors, n is the number of support vectors.

Some of the kernel functions used in the pattern recognition problems are listed in the following table:

$K(x, y) = (x.y + 1)^p$	results in classifier that is polynomial of degree p
$K(x, y) = e^{-\frac{\lvert\lvert x-y \rvert\rvert^2}{2\sigma^2}}$	gives a Gaussian Radial Basis function classifier
$K(x, y) = \tanh(\beta x.y + \delta)$	β *and* δ are user specified values. It results in two-layer sigmoidal neural network.

SECTION 3: Stepwise Procedure

1. Get the “wine recognition” dataset.
2. Divide the data into training, validation and testing sets with equal number of samples in each class.
3. Data is preprocessed before training (standardization).
4. Use two soft margin SVM’s for this three class problem with different kernels like Gaussian RBF, polynomial, 2-layer BPNN etc. Compare the decision surfaces and confusion matrix.
5. Cross validation method is used to find optimum C value and kernel parameter for each machine. Define the number of folds for cross validation.
   Cross validation procedure, in general, is as follows:
   - In case of Gaussian RBF kernel, there is problem of finding optimal $C\ and\ \sigma(sigma)$ as there values are unknown beforehand.
   - The goal is to identify good (C, σ) so that the classifier can accurately predict unknown data (i.e. testing data).
   - In *k-fold cross validation*, we first divide the training set into *k* subsets of equal size. Sequentially one subset is tested using the classifier trained on the remaining *k-1* subsets. Thus, each instance of the whole training set is predicted once, so the cross validation accuracy is the percentage of the data correctly classified.
   - Cross-validation can prevent the overfitting problem.

6. Test the machine on the testing data set and compute classification rate by plotting confusion matrix.
7. Compare SVM results with those of BPNN and give comments.

SECTION 4: Results

In this computer assignment, we will design support vector machine using various kernels for a given wine dataset to classify it into the three classes.
The wine recognition dataset is from the website- http://archive.ics.uci.edu/ml/datasets/Wine.
As mentioned earlier, there are total 178 samples with 13 attributes each. These are divided into 3 classes such that 59 samples represent class1, 71 samples represent class2 and remaining 48 samples represent class3.
We will be using equal number of samples for each class. Number of samples from each class can be made equal by adding random white Gaussian noise with zero mean and unity variance. Hence, our new wine data set contains:
Class1 = 60 samples
Class2 = 60 samples
Class3 = 60 samples
Total number of samples will be 180.
Out of these 180 samples, we select 144 samples (48 samples from each class). In this way, dataset contains 144 samples having 13 features each. Among these, 120 samples are used for training and validation and remaining 24 samples are considered for testing of our SVM classifier.
Training, validation and testing datasets are standardized for future processing. The procedure is given below.

Pre-processing:
Since the attribute values are very different from each other, it is necessary to do preprocessing on the dataset before using it for training.
Procedure for standardizing the dataset:

- 13 Attributes corresponding to total 180 samples are arranged column wise. Calculate the mean and standard deviation value of each attribute.
- Subtract the mean value from each value and divide by the standard deviation.

The operation can be shown mathematically as,

$$x = (x - \mu)/\sigma$$

Where, $x = input\ sample,\ \mu = mean\ value,\ \sigma = std\ deviation$
Newly formed dataset is used for training and testing.

C and σ ($sigma$) after cross-validation:
Various (C, σ) pairs are tried and the one with the best cross-validation accuracy is selected. It is found that trying exponentially growing sequences of C and σ is a practical method to identify good parameters. Some of the naïve methods are used such as 'grid search' for this purpose as it is fast (for two parameters) and gives satisfactory results.
By trying various combinations we have selected $C = 10$ and $\sigma = 1.5$.
Number of folds (k) $= 5$
In MATLAB, there is a procedure to find the optimal C and σ for the RBF kernel. This procedure is attached in the form of code at the end (references).

SVM design:

MATLAB Neural Network Toolbox is used for designing the SVM for training and testing the wine dataset. As mentioned earlier, SVM is a binary classifier therefore, at a time it can separate only 2 classes. Steps are given below:
Step1: Separate class1 from class2 and class3 combined.
Step2: Separate class2 from class3.

"ClassificationLearner" tool from MATLAB allows us to import the dataset and use all 13 features to plot scatter plot and calculate classification rate. This tool helps in selecting the 2 features out of original 13 which will distinguish the classes as correctly as possible.
By trying different combinations of feature pairs, we arrive at the best possible results for the pair **'feature 3 and feature 4'**.
Classification rates for different SVMs considering feature 3 and feature 4 is shown in the following table:

Type of SVM	Classification Rate (%)
Linear SVM	96.5
Quadratic SVM	97.9
Cubic SVM	95.8
Medium Gaussian SVM	98.6

Scatter plot of the data:

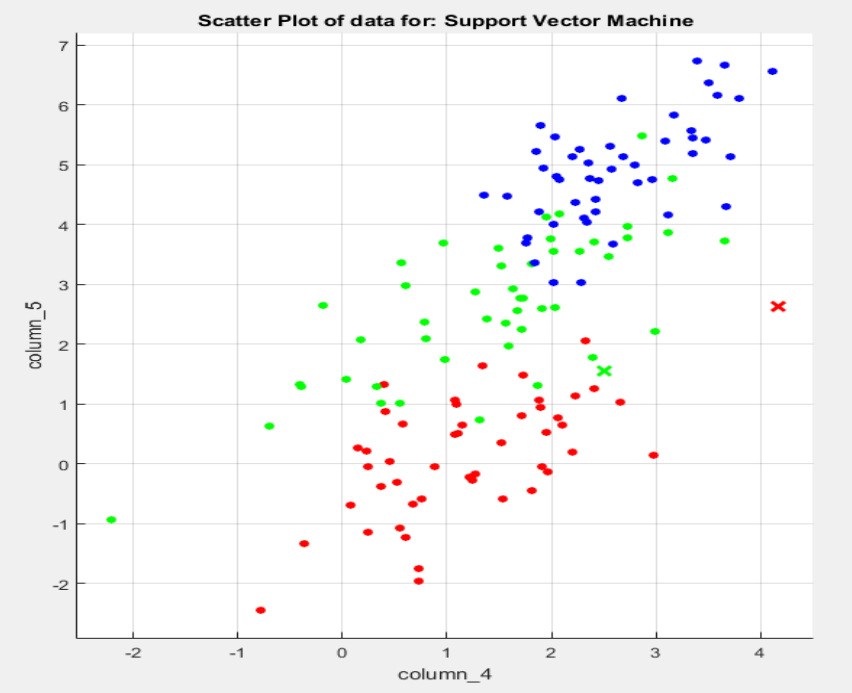


Figure 1: scatter plot of the entire dataset

X-axis is the feature 3 and Y-axis is feature 4 (first column of the data contains targets (1,2 and 3)).

Linear SVM

Step 1: separating class 1 from class 2 & class 3 combined

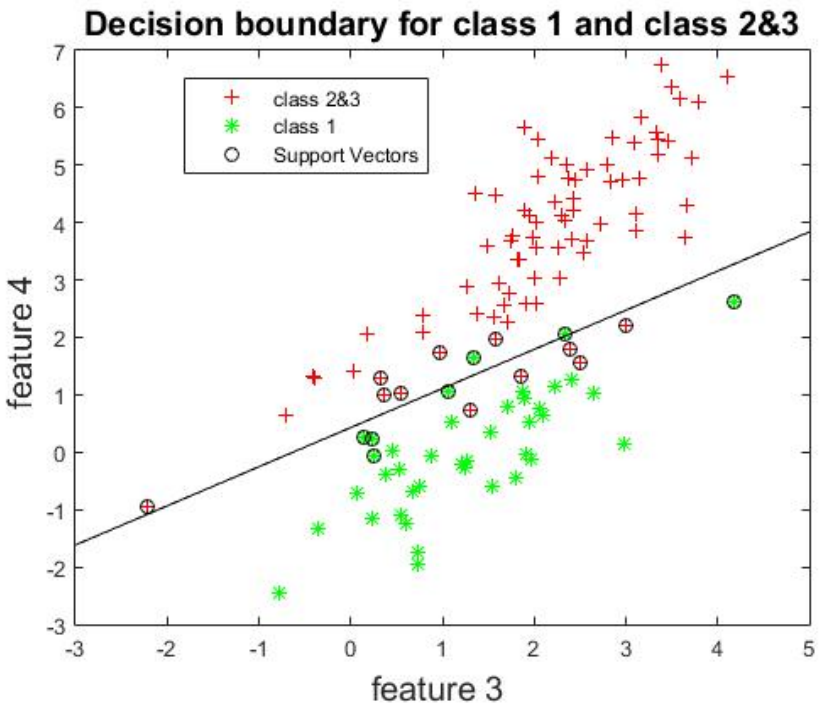


Figure 2: Decision boundary separating class 1

Step 2: separating class 2 from class 3

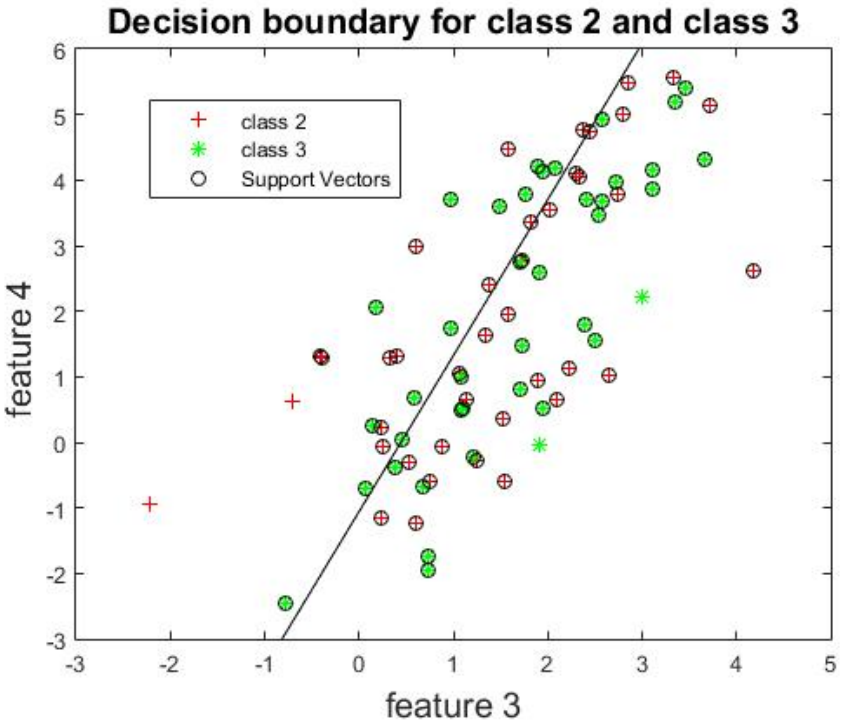


Figure 3: Decision boundary separates class 2 and class 3

Confusion matrix: overall

All class confusion matrix

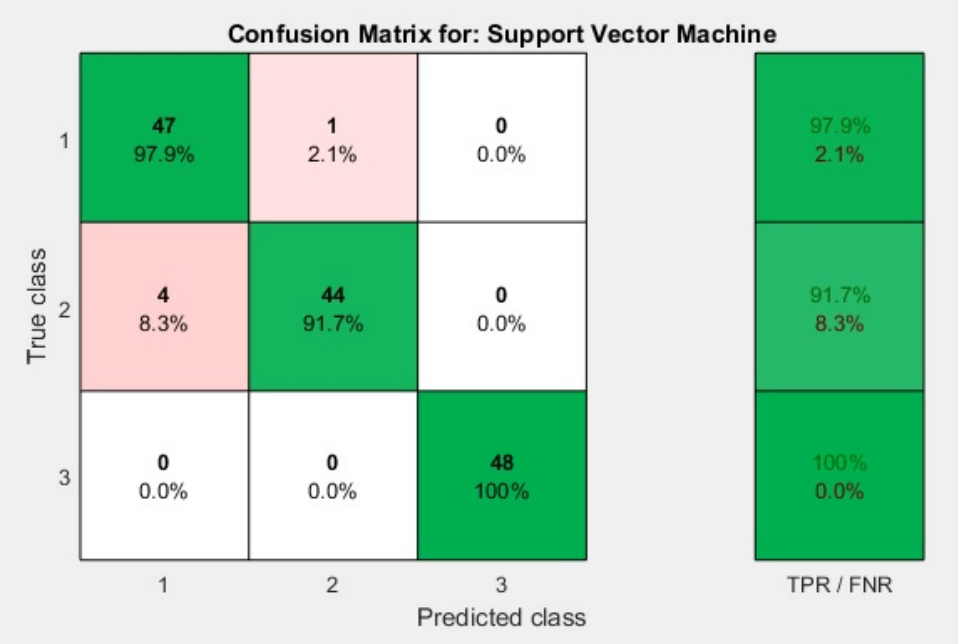


Figure 4: Overall confusion matrix for linear SVM

Observations:

- Given three class training data is classified using linear support vector machine.
- In the first step, class 1 is separated from class 2 and class 3 combined. The decision boundary is shown in the figure1.
- Observing the separation, we can see that the linear function provides a good separation between the classes. Circled elements are called as 'support vectors' and as can be seen, they define the decision boundary, i.e. almost all of them are present on the decision boundary.
- However, as we see in the second step, the separation is not as good as first. The primary reason for it will be the number of training samples considered and large number of support vectors spread out all over the place. However, the result is still considerable.
- The hyperplane separating the two classes is optimal hyperplane and it maximizes the margin of separation between class 1 and class 2&3 combined.
- Using linear SVM and our two step procedure, we get the overall classification rate as shown in the table below:

Class	Classification Rate (%)
Class 1	97.9
Class 2	91.7
Class 3	100

- As we see, only 5 samples are mis classified from class 2 and class 1 combined.

Kernel 1: Polynomial with polynomial order = 3

Step 1: separating class 1 from class 2 & class 3 combined

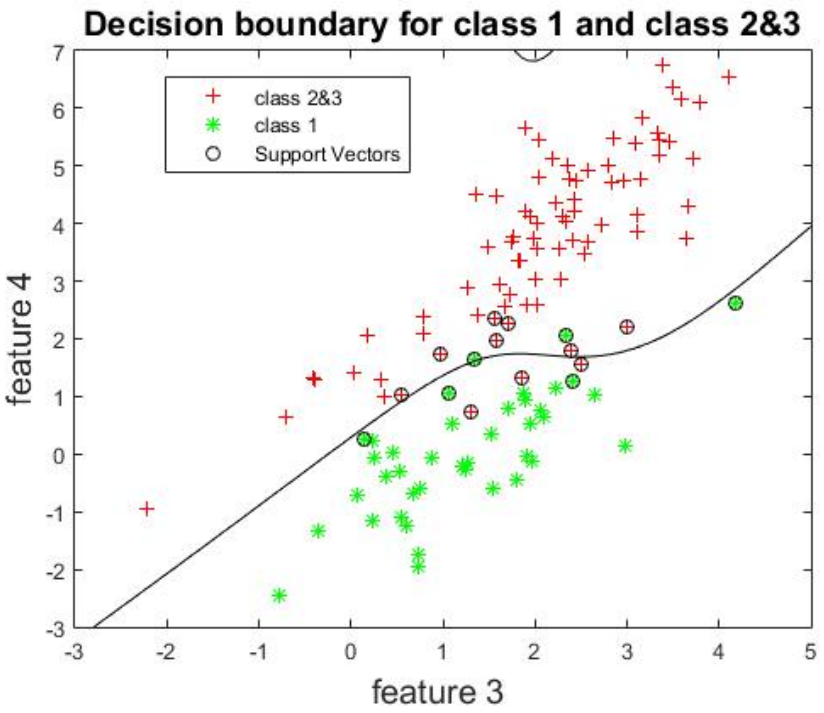


Figure 5: Decision boundary separating class 1

Step 2: separating class 2 from class 3

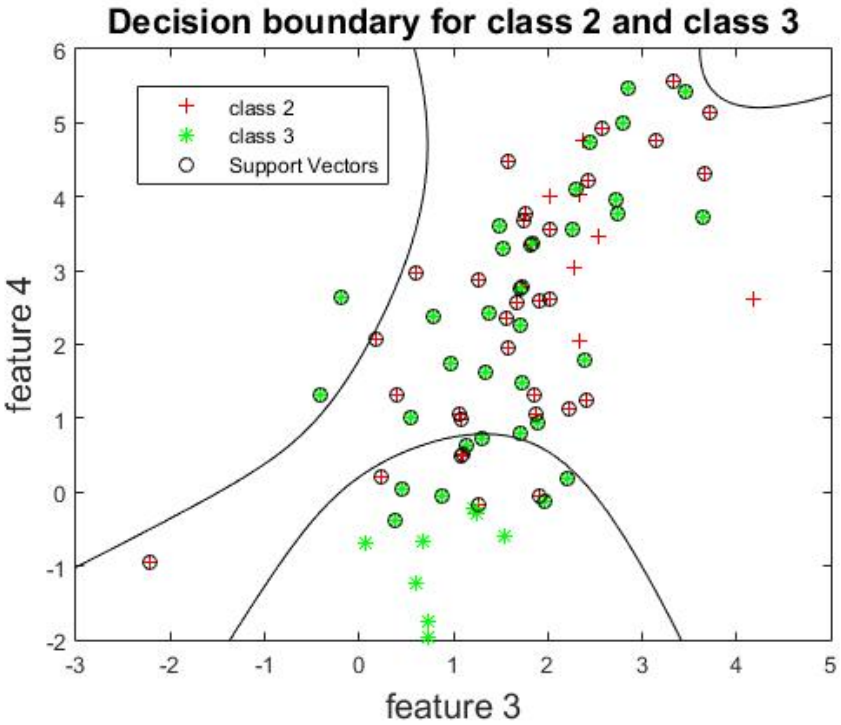


Figure 6: Decision boundary separating class 2 from class 3

Confusion matrix: overall

All class confusion matrix

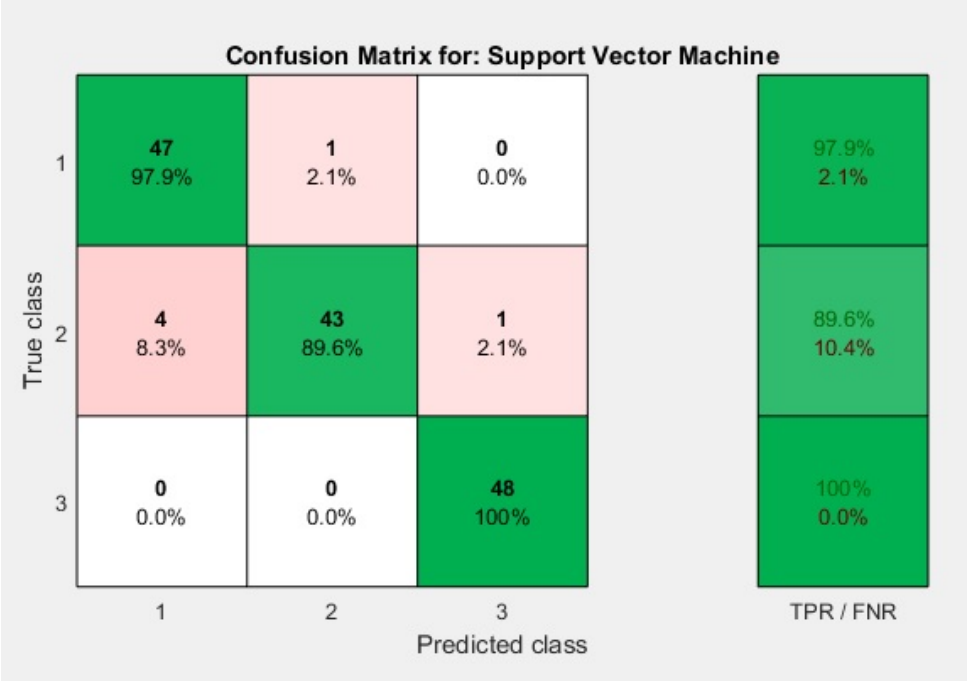


Figure 7: Overall confusion matrix for polynomial SVM

Observations:

- Given three class training data is classified using support vector machine with polynomial kernel.
- Pattern classification depends on the order of the polynomial considered. Polynomial order is a kernel parameter in this case. Most common order is 2 however, as the polynomial order increases, classification rate improves as well as separation is better.
- Here we have used order of the polynomial 3 which gives better separation results as compared to linear SVM as well as other lower degree polynomial kernels. Circled elements are called as 'support vectors' and as can be seen, they define the decision boundary, i.e. almost all of them are present on the decision boundary.
- Class 1 separation (step 1) is very good as compared to the class 2 and class 3 separation (step 2). However, overall misclassification rate is high.
- Using polynomial SVM and our two step procedure, we get the overall classification rate as shown in the table below:

Class	Classification Rate (%)
Class 1	97.9
Class 2	89.6
Class 3	100

- Class 2 classification rate is lesser than linear SVM. This shows that higher order polynomials may cause some problems and features should be properly selected.

Kernel 3: Gaussian RBF SVM

Step 1: separating class 1 from class 2 & class 3 combined

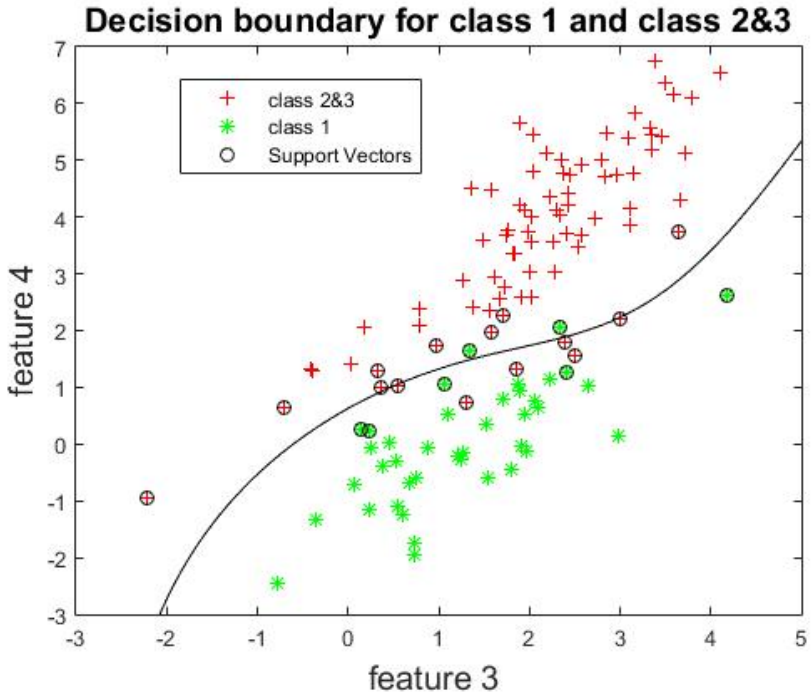


Figure 8: Decision boundary separating class 1

Step 2: separating class 2 from class 3

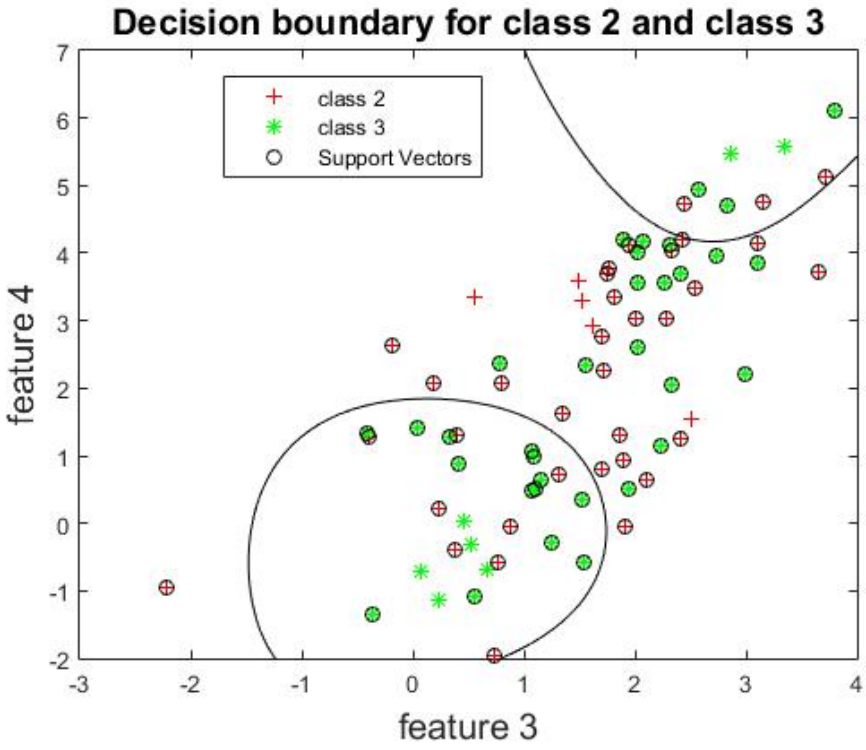


Figure 9: Decision boundary separating class 2 from class 3

Confusion matrix: overall

All class confusion matrix

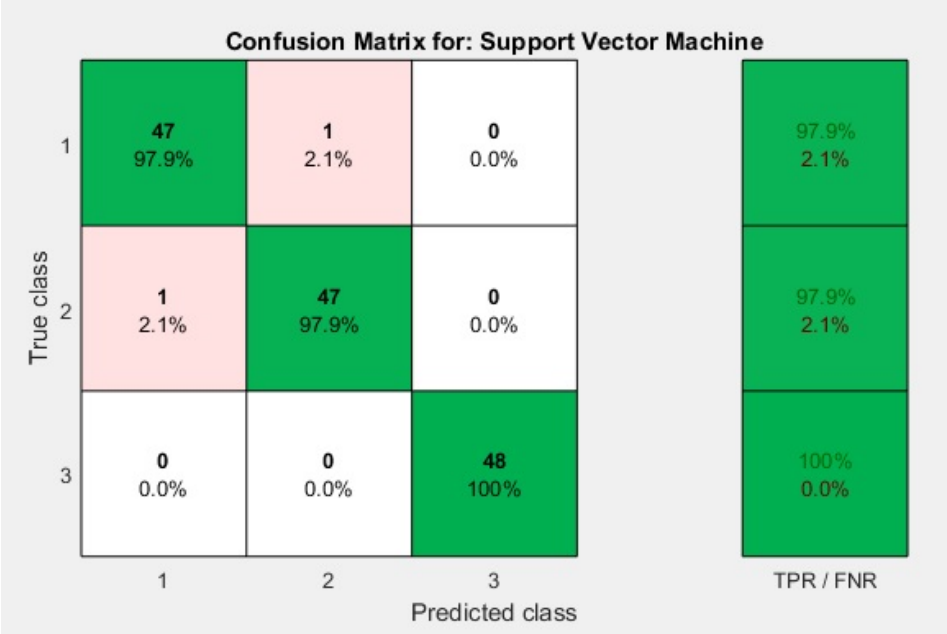


Figure 10: Overall confusion matrix for Gaussian RBF kernel SVM

Observations:

- By selecting $C = 10$ and $\sigma = 1.5$, we obtained the decision boundary as shown in figures 8 and 9. The learned decision boundary in the original space is non-linear.
- Numbers of support vectors define the number of RBF's.
- Solution of SVM problem is the linear combination of the RBF kernels that sit on the support vectors.
- If σ is very small, RBF kernel is very wide. In this case, optimization will be difficult as decision function of all data points will be changed as the kernel is wide. On the other hand, if σ is very large, RBF kernel is very narrow and all the training samples will eventually be support vectors which is not desirable.
- We have chosen the value $\sigma = 1.5$ that gives satisfactory results as can be seen from the decision boundaries in figure 8 and 9.
- Using Gaussian RBF kernel for SVM and our two step procedure, we get the overall classification rate as shown in the table below:

Class	Classification Rate (%)
Class 1	97.9
Class 2	97.9
Class 3	100

- We see that all the classes have much higher classification rates than Linear and polynomial SVM. This shows why this kernel is widely used for pattern classification using SVM.

SECTION 5: Discussion and Comments

Generalization ability of Support Vector Machine (SVM):

- Generalization ability of SVM is the performance of SVM for future unknown samples.
- SVM minimizes the bound on generalization error by maximizing the margin of separation between the two classes.
- For a given kernel the SVM automatically finds the classifier with the best generalization ability by tuning the parameters of the hyperplane and adjusting the number of support vectors. Then, to find the best mapping, we have to train our machine several times for a set of kernel parameters and select the value that minimizes the generalization error.
- To find the optimal C and σ parameters, cross validation approach is used.
- In *k-fold cross validation*, we first divide the training set into *k* subsets of equal size. Sequentially one subset is tested using the classifier trained on the remaining *k-1* subsets. Thus, each instance of the whole training set is predicted once, so the cross validation accuracy is the percentage of the data correctly classified.
- In this project, the optimal values we selected are, $C = 10$ and $\sigma = 1.5$ for $k = 5$ fold cross validation. These values are used for testing the dataset and corresponding results are shown in the previous section.

Generalization ability of the Back Propagation Neural Networks (BPNN):

- To assess the generalization performance of our BPNN algorithm trained on different training set sizes, we conducted some experiments varying the size of the training set. The results are depicted in the following table:
  For a single hidden layer network (2-layer BPNN):

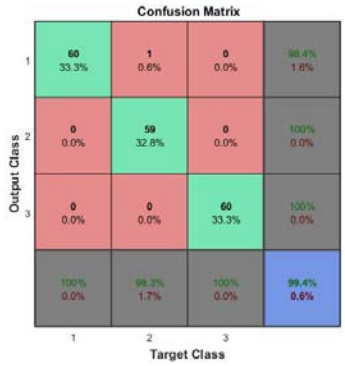
Training	Validation	Testing	MSE
40%	30%	30%	0.0203
50%	30%	20%	0.012
60%	20%	20%	0.0053
70%	15%	15%	$5.74 * 10^{-4}$
80%	10%	10%	$6.8 * 10^{-5}$
90%	5%	5%	$3.98 * 10^{-7}$

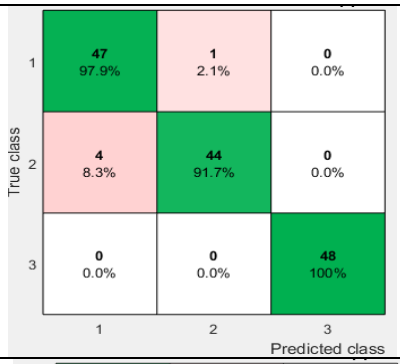
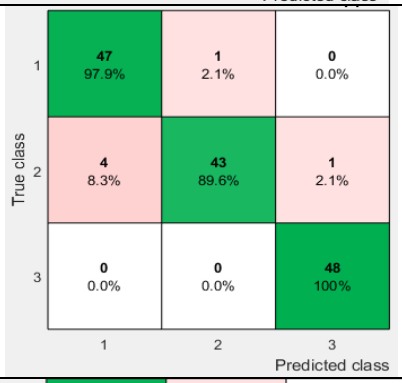
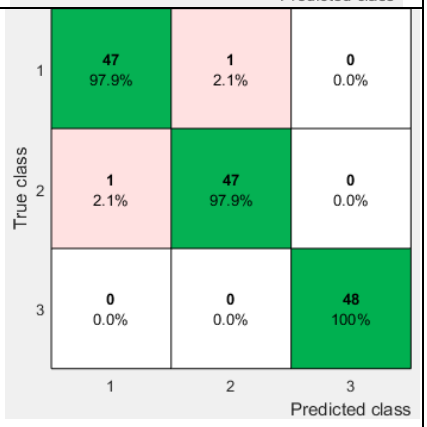
- Table shows the generalization MSE versus the size of the training set for a single hidden layer network with 30 neurons in it.
- As we can see, the generalization error goes on decreasing as the size of training data set goes on increasing and the minimum error will occur when the size of the training data set will be very high (90% or 100%). Perhaps one of the reason why this could be so is that the full training set is more representative of the problem space that the altered variations of it.
- In reality, we do not need to train our network with large amount of training data. Various experiments have been conducted to observe this and it has been found out that beyond certain size of training data, generalization do not improve, pointing that an excess of training patterns is no gain.
- As we can see from the table, we selected 70% training data as the minimum MSE is lower than

maximum allowable error and we have achieved good results based on our choice of training data size. This also gave us sufficient data for validation and testing our algorithm. The same trend on MSE follows for 3-layer BPNN network.

Comments

- In this study, we have applied Support Vector Machine for the pattern classification problem.
- We gave generated the plots showing the classification rate and decision boundaries. Below is the table showing the comparison between SVM and 2 & 3-layer BPNN used in the previous assignment based on several factors.
  These factors include:
  1. Correct Classification rate
  2. Associated confusion matrix
  3. Complexity

Type of Neural Network	Associated Learning rule or Kernel	Complexity	Correct classification rate	Associated confusion matrix
2- Layer BPNN	Generalized delta	Less complex, easy to implement	99.4	
3 – Layer BPNN	Levenberg-Marquardt	More complex than 2-layer, Moderate for implementation	99.4	

SVM	Linear	Easy to implement, good results	96	
SVM	Polynomial kernel (order of the polynomial =3)	Moderate/ Complex (depends on the order) to implement, optimal results	95	
SVM	Gaussian RBF kernel	Moderate implementation, optimal results, high efficiency	99	

Advantages/Disadvantages of Support Vector Machines:

Advantages:

- It has a regularization parameter that can be used to avoid overfitting.
- Tt uses the kernel trick, so you can build in expert knowledge about the problem via engineering the kernel.
- SVMs provide a good out-of-sample generalization, if the parameters C and σ (in the case of a Gaussian kernel) are appropriately chosen. This means that, by choosing an appropriate generalization grade, SVMs can be robust, even when the training sample has some bias.
- SVMs deliver unique solution, as the optimality problem is convex.
- It approximates a bound on the test error rates.
- Optimal design for multiclass problems.

Disadvantages

- Lack of transparency of results.
- Very high dimensions can be a problem.
- Kernel models can be quite sensitive to over-fitting the model selection criterion.
- Slow approach in the testing phase.
- High algorithmic complexity.
- Large memory requirements.

SECTION 6: Conclusion

- In this computer assignment, Support Vector Machine neural network is applied to the pattern classification problem.
- Support Vector Machine neural network is successfully constructed using Matlab Neural Network toolbox.
- Wine recognition dataset is used as the input data to train the network and classification is performed.
- Linear and non-linear SVMs are constructed.
- For non-linear SVM, polynomial with order 3 and Gaussian RBF kernels are used to perform pattern classification.
- From the results, we see that Gaussian RBF kernel gives better results in terms of classification rate as compared to Linear SVM and polynomial kernel function.
- It is important to choose proper values of C and σ to get optimal results in case of Gaussian RBF kernel.
- From the comparison table of SVM and BPNN, we see that SVM performs faster and gives optimal results as compared to 2 and 3-layer BPNN network. However, the classification rate is almost same (close to 99%) for all the mentioned approaches.
- Comparison with respect to several characteristics for both the networks is tabulated and generalization ability of the networks is discussed.

SECTION 7: References

- Eugene A. Borovikov, "An Evaluation of Support Vector Machines as a Pattern Recognition Tool"
- A. I. Belousov, S. A. Verzakov and J. von Frese. "Applicational Aspects of Support Vector Machines "
- Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. "A Practical Guide to Support Vector Classication".
- Piyush Rai. "Kernel Methods and Non-Linear Classification" http://www.cs.utah.edu/~piyush/teaching/15-9-print.pdf
- Asa Ben-Hur, Jason Weston. "A User's Guide to Support Vector Machines"
- Laura Auria, Rouslan A. Moro. "Support Vector Machine as a Technique for Solvency Analysis"
- Ming Chang Lee, Chang To. "Comparison of Support Vector Machine and Back Propagation Neural Network in Evaluating the Enterprise Financial Distress"
- www.researchgate.net , www.stackexchange.com
- MATLAB Mathworks , MATLAB NN toolbox.
- Lecture notes.
- Wikipedia
- MATLAB codes are attached for reference.

SVM MATLAB Code:

```
clear all;
clc;

 %% Separate class 1 from 2 and 3
x1range = 'B:N';
data = xlsread('wine_data.xlsx',x1range);
class1 = data((1:59),:);
class2 = data((60:130),:);
class3 = data((131:178),:);
% get 48 samples from every class
class1_new = class1((1:48),:);
class2_new = class2((1:48),:);
data = [class1_new;class2_new;class3];

% normalize the data
s=[];
for i =1:size(data,2)
    A = data(:,i);
    s(i)=std(A);
end
for j = 1:size(data,2)
    for i=1:size(data,1)
        data(i,j)= (data(i,j)- mean(data(:,j)))/s(j);
    end
end

% training and validation set
pre_training_1 = [data((1:40),(3:4)); data((49:88),(3:4));data((97:136),(3:4))];
pre_training_targets_1 = [ones(1,40) 2*ones(1,40) 3*ones(1,40)]';
p = randperm(length(pre_training_1));
training_1 = pre_training_1(p,:);
training_targets_1 = pre_training_targets_1(p,:);

% conversion for binary classification
indclass2 = [];
indclass3 = [];
for i = 1:length(training_targets_1)
    if training_targets_1(i)== 2
       class2_indx = i;
       indclass2 = [indclass2 class2_indx];
       training_targets_1(i)= 0;
    elseif training_targets_1(i)== 3
       class3_indx = i;
       indclass3 = [indclass3 class3_indx];
       training_targets_1(i)= 0;
    end
end

% train SVM
C = 10;
```

```
% linear SVM
figure;
svmstruct_123 =
svmtrain(training_1,training_targets_1,'boxconstraint',C,'kernel_function','linear'
,...
    'showplot','True');
legend('class 2&3','class 1','Support Vectors');
xlabel('feature 3 ','fontsize',15);ylabel('feature 4 ','fontsize',15);
title('Decision boundary for class 1 and class 2&3','fontsize',15);

% gaussian RBF
figure;
svmstruct_123 =
svmtrain(training_1,training_targets_1,'boxconstraint',C,'kernel_function','rbf','r
bf_sigma',1.5,...
'showplot','True');
legend('class 2&3','class 1','Support Vectors');
xlabel('feature 3 ','fontsize',15);ylabel('feature 4','fontsize',15);
title('Decision boundary for class 1 and class 2&3','fontsize',15);

% polynomial
figure;
svmstruct_123 =
svmtrain(training_1,training_targets_1,'boxconstraint',C,'kernel_function',...
    'polynomial','polyorder',3,'showplot','True');
legend('class 2&3','class 1','Support Vectors');
xlabel('feature 3 ','fontsize',15);ylabel('feature 4','fontsize',15);
title('Decision boundary for class 1 and class 2&3','fontsize',15);

%% separate class 2 and 3

indx = [indclass2 indclass3]';
training_2 = zeros(80,2);
for i = 1:length(indx)
    training_2(i,:) = data(indx(i),(3:4));
end
training_targets_2 = [zeros(1,40) ones(1,40)]';

% train SVM
figure;
svmstruct_23 =
svmtrain(training_2,training_targets_2,'boxconstraint',C,'kernel_function','linear'
,...
    'showplot','True');
legend('class 2','class 3','Support Vectors');
xlabel('feature 3','fontsize',15);ylabel('feature 4 ','fontsize',15);
title('Decision boundary for class 2 and class 3','fontsize',15);

% gaussian RBF
figure;
svmstruct_23 =
svmtrain(training_2,training_targets_2,'boxconstraint',C,'kernel_function','rbf','r
bf_sigma',1.5,...
'showplot','True');
legend('class 2','class 3','Support Vectors');
xlabel('feature 3','fontsize',15);ylabel('feature 4','fontsize',15);
title('Decision boundary for class 2 and class 3','fontsize',15);
```

```
% polynomial
figure;
svmstruct_23 =
svmtrain(training_2,training_targets_2,'boxconstraint',C,'kernel_function',...
    'polynomial','polyorder',3,'showplot','True');
legend('class 2','class 3','Support Vectors');
xlabel('feature 3','fontsize',15);ylabel('feature 4','fontsize',15);
title('Decision boundary for class 2 and class 3','fontsize',15);
```

CROSS VALIDATION Code

(Number of iterations in the loop are varied to find the optimal value. In the following code, 25 iterations are mentioned.)

```
% Z is an input parameter [rbf_sigma ; boxconstraint]

% cross validation
c = cvpartition(120,'KFold',5);

% serach optimal parameters
opts = optimset('TolX',5e-4,'TolFun',5e-4);

% search global minimum
m = 25; % number of iterations = 25
fval = zeros(m,1);
z = zeros(m,2);
for j = 1:m;
    [searchmin fval(j)] = fminsearch(minfn,randn(2,1),opts);
    z(j,:) = exp(searchmin);
end
z = z(fval == min(fval),:);
disp(z);
```