Support Vector Machines

SVM

Support Vector Machines

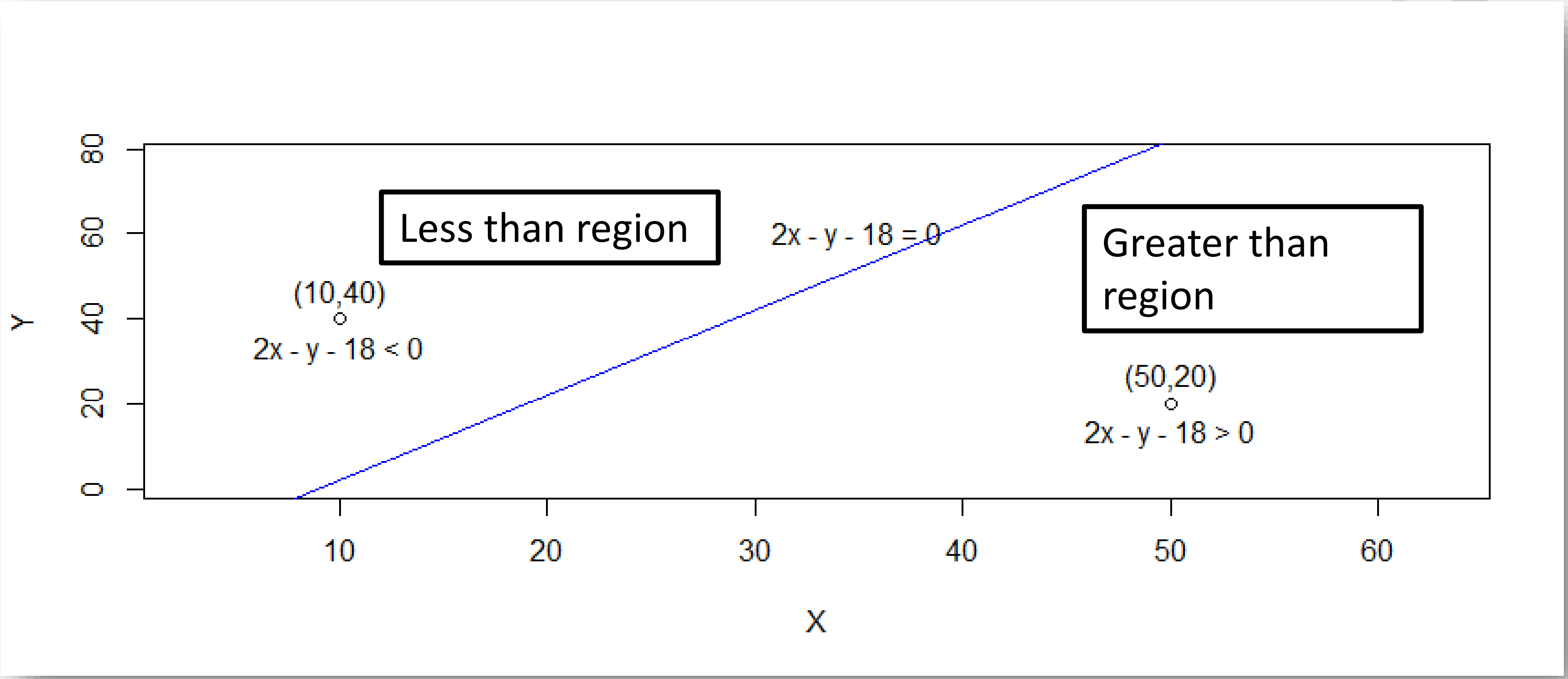
- Support Vector Machines can be used for classification as well as regression
- Usage of SVM is popular for classification than for regression
- We will be covering SVM for classification.

Understanding SVM

- SVM is a generalization of a simple classifier *maximum margin classifier*.
- The concept of *maximum margin classifier* can be extended to that of ***support vector classifier*** and **support vector machines**.

Straight Line Fundamentals (Revision)

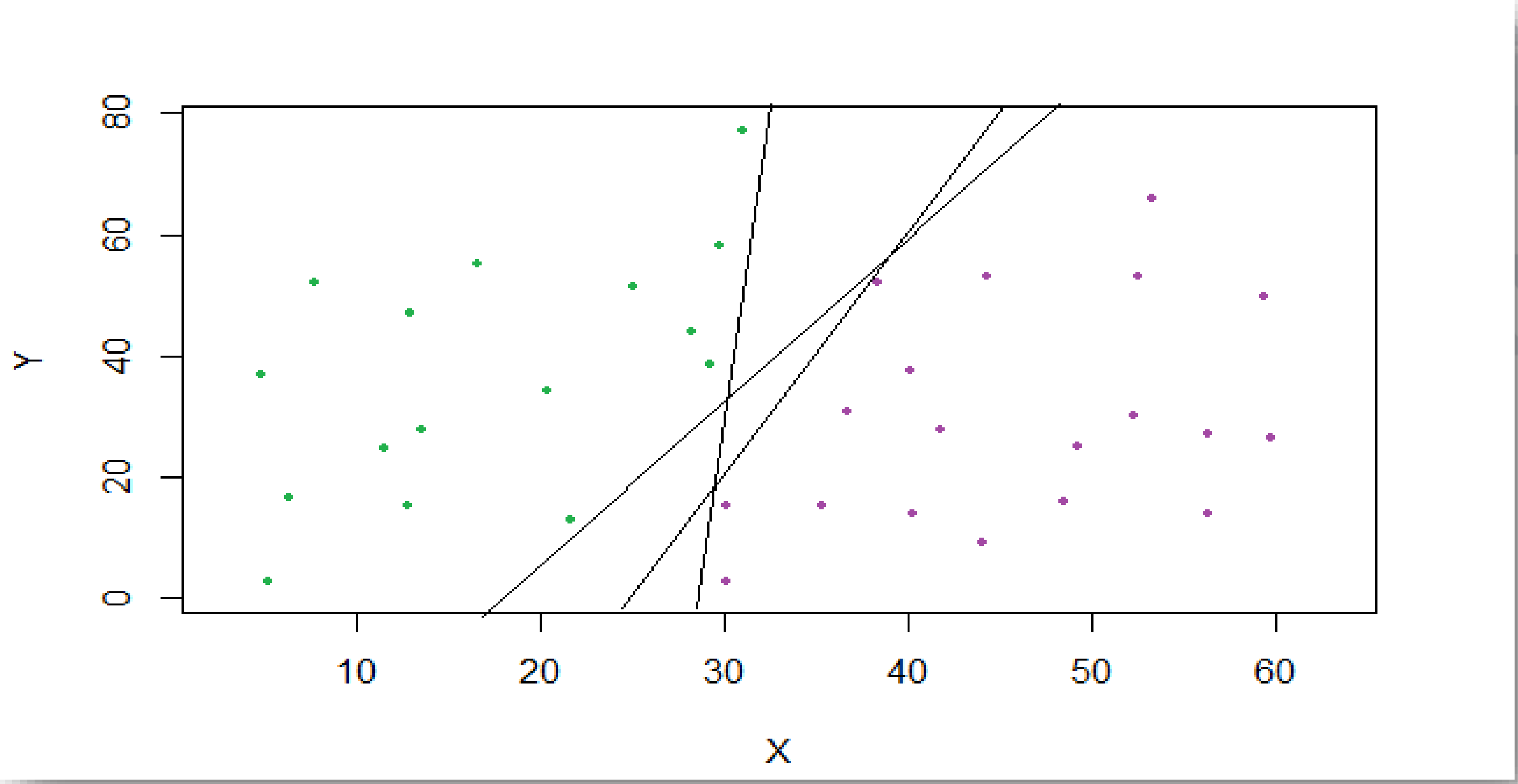
- Consider a line with equation, ax + by + c = 0
- Any point (x1,y1) which is lying on the line satisfies the equation of the line i.e. we can write ax1 + by1 + c = 0.



Separating Hyperplanes

- Let us understand the concept on 2-dimensional plane which can be further extended to multi-dimensional hyperplane
- Suppose that, it is possible to have three hyperplanes for a data

Separating Hyperplanes

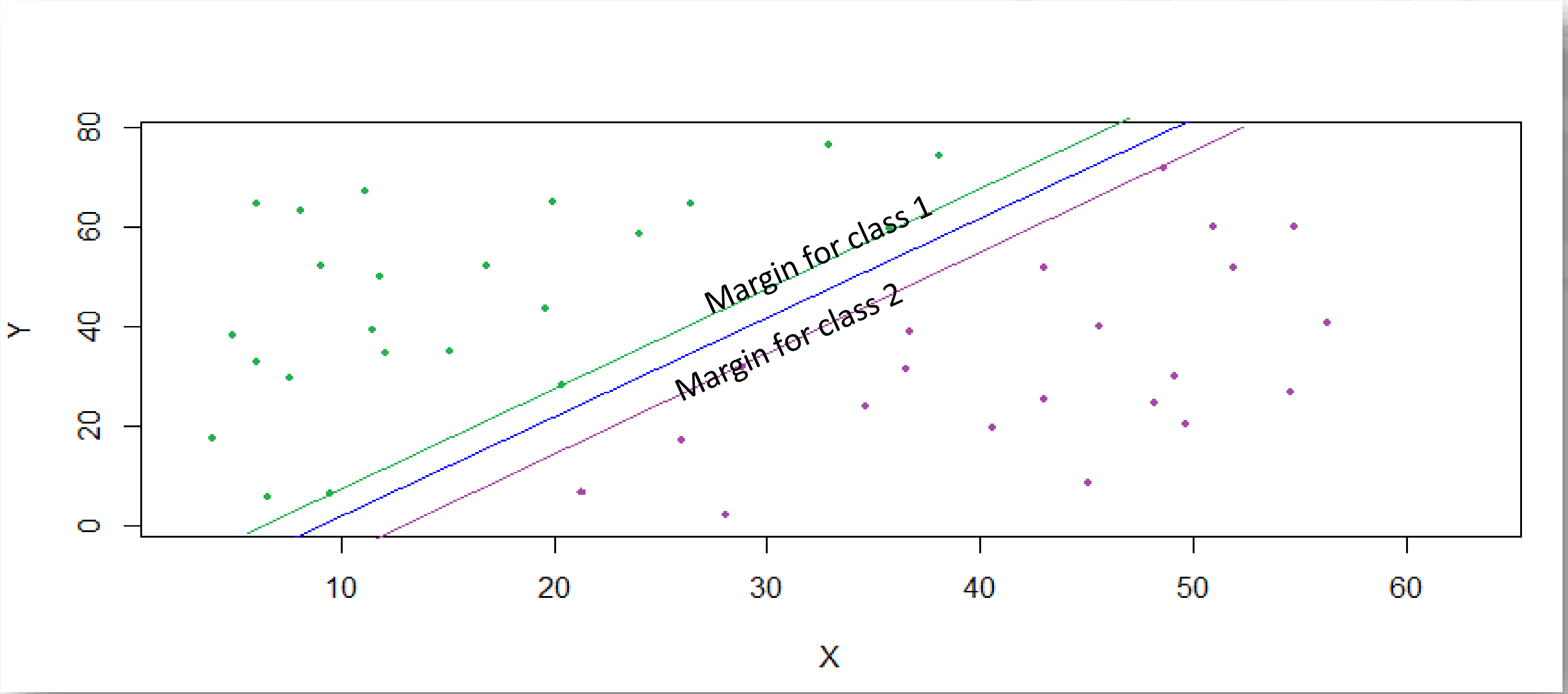


- We observe here that, three hyperplanes have separated the data. Any point which lies in the region of green points can be classified as category of green and similarly with purple.

Maximum Margin Classifier

- If our data can be perfectly separated using a hyperplane, then there will in fact exist an infinite number of such hyperplanes which will separate different categories in our response variable
- This can be made possible with a given separating hyperplane shifted a tiny bit up or down, or rotated, without coming into contact with any of the observations
- Hence we can imagine a separating hyperplane which has maximum distance from any nearest point in the data. This is called *maximum margin classifier*.

Maximum Margin Classifier



- We observe that 5 observations are equidistant from maximal margin hyperplane. These points are called *support vectors*.
- These points are called “support” in the sense that if these points were moved slightly then the maximal margin hyperplane would move as well.

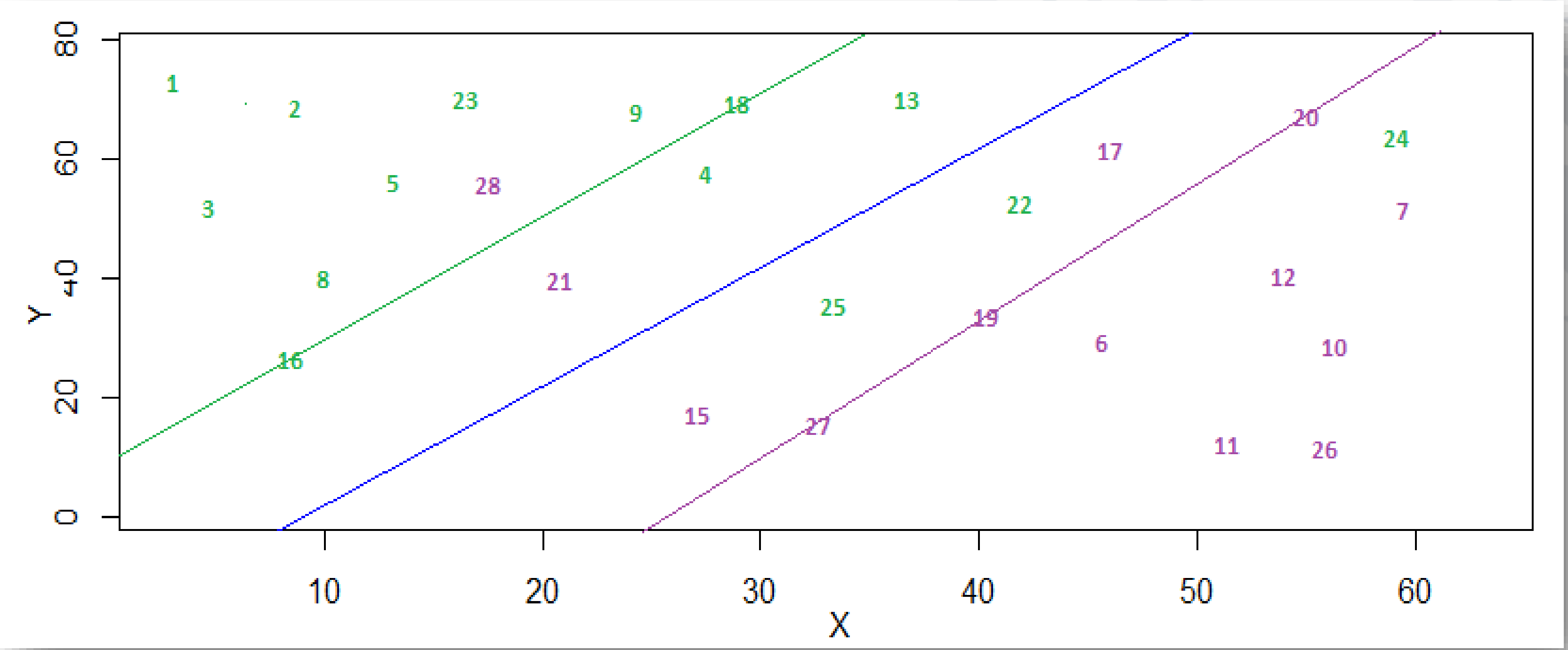
Non-Separable Case

- In case, if a separating hyperplane is not available then we cannot exactly separate two classes
- Instead, we can find a hyperplane that almost separates the two classes
- A generalization of the maximal margin classifier to the non-separable case is called as the *support vector classifier*

Support Vector Classifier

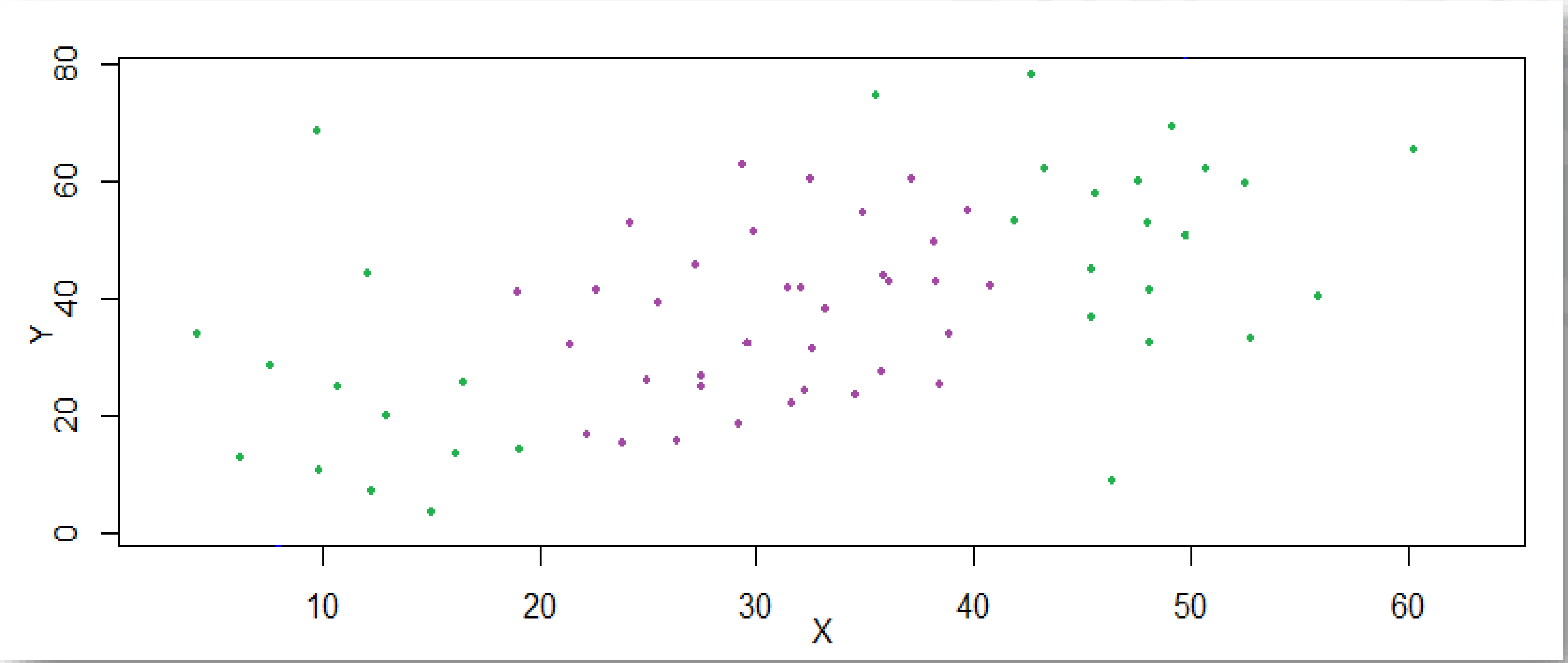
- In case, if a separating hyperplane is not available then a classifier can be considered which exactly does not separate the two classes but classifies most of the training set observations correctly
- In this case, some observations can be allowed to be on the incorrect side of the margin or also incorrect side of separating hyperplane
- This separating hyperplane can also be called as soft margin classifier as it can allow some violations

Illustration : SV Classifier



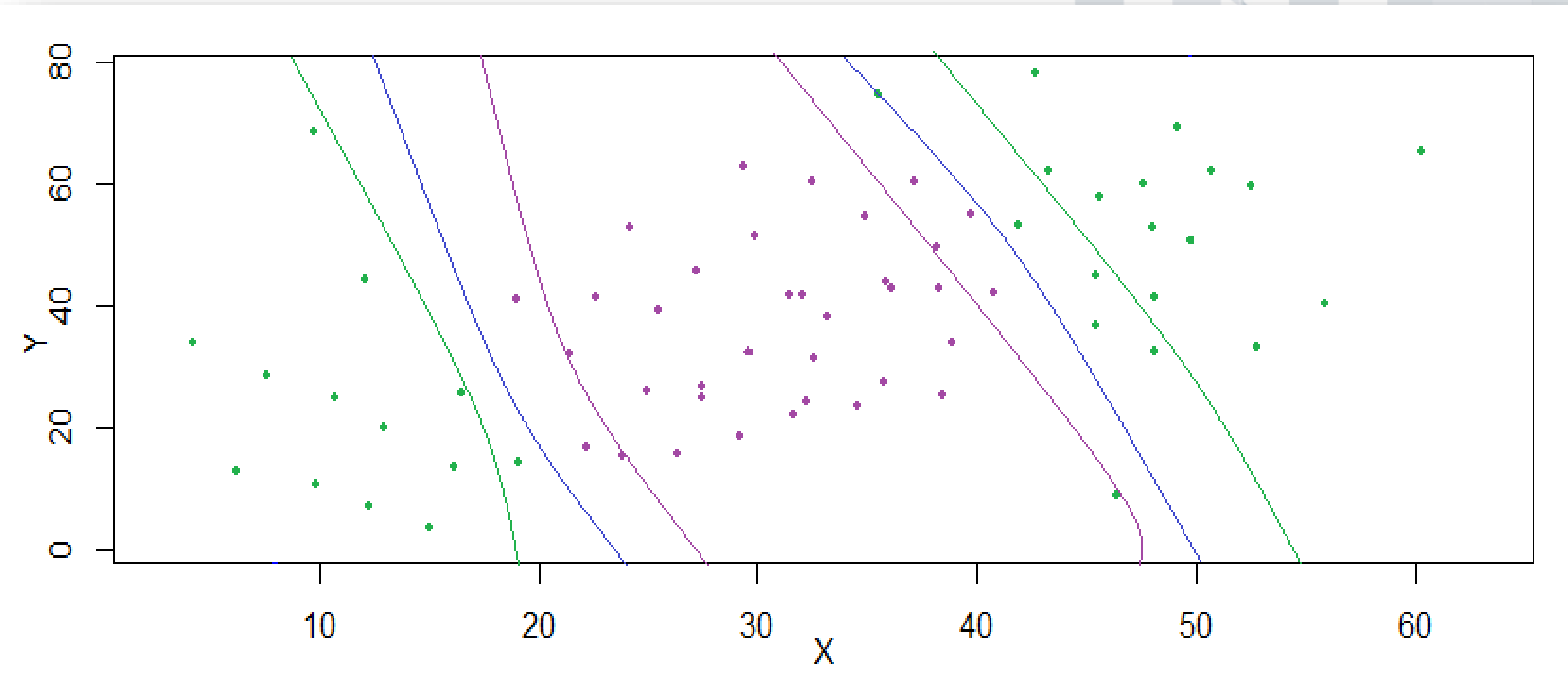
- Consider that the above diagram represents a support vector classifier fitted to a small dataset with 27 observations
- Observations 16, 18, 20, 19, 27 are on the margin
- Observations 4, 13, 15, 17 are on the wrong side of their respective margins
- Observations 21, 28, 25, 22, 24 are not only on the wrong side of their respective margins but also on the wrong side of the separating hyperplane

Classification with Non-Linear Decision Boundaries



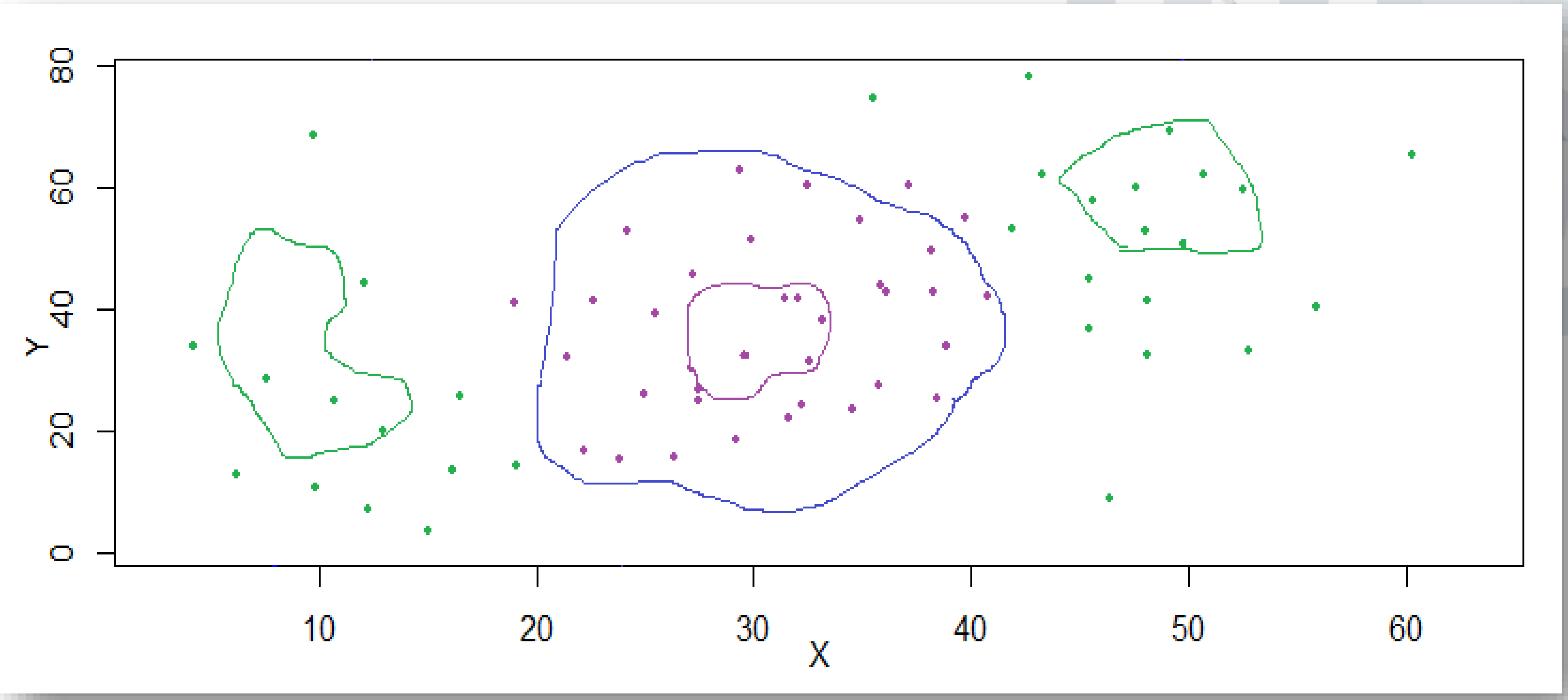
- When the class boundaries are non-linear, then the feature space (predictors) is enlarged with non-linear components in it.
- **Support Vector Machine** is an extension of support vector classifier that is constructed from enlarging feature space in a specific way using kernel functions

Possible Solutions



Polynomial Kernel

Possible Solutions



Radial Kernel

SVM – More than Two Classes

- There are two approaches most popular approaches for SVM with more than two classes:
  - One – Versus – One Classification
  - One – Versus – All Classification

One – Versus – One Classification

- Suppose there are K (K>2) classes for a SVM problem
- This approach considers $\binom{K}{2}$ SVMs comparing a pair of classes with each combination
- A test observation is classified by tallying the assignments to each of the K classes
- The final classification is decided by the majority assignments to a particular class

One – Versus – All Classification

- K (K>2) SVMs are fitted each time comparing one of the K classes to the remaining K-1 classes
- A test observation is assigned to that class out of K classes for which function of the estimated parameters is highest.

SVM in R

- SVM can be implemented in R using function *svm()* from package ***e1071***. There are other alternatives also like package ***LiblineaR***
- For K (K>2) classes *svm()* implements One-Versus-One Classification

Syntax : svm(formula, data, type, kernel, degree, gamma, …)

Where

formula : formula for model

data : training data frame

type : C – Classification, eps – Regression, etc.

kernel : kernel used, Can be any of “linear”, “polynomial”, “radial”, “sigmoid”

degree : Applicable for “polynomial” kernel

gamma : Applicable for “radial” kernel

Example: Riding Mowers

- A riding-mower manufacturer **MOW-EASE** took part in a Industrial Exhibition in which it got an opportunity to show a demo of its product to 180 different audience.
- The land owned by each of the audience and their approximate income have been recorded in the file RidingMowers.csv

Example: Riding Mowers

- The Data contains two predictors Area Owned (Lot_Size) and Income with response variable as "Bought" and "Not Bought" values

	Income	Lot_Size	Response
1	34	26	Not Bought
2	34	40	Not Bought
3	34	46	Not Bought
4	34	48	Not Bought
5	34	53	Not Bought
6	34	58	Not Bought
7	34	59	Not Bought
8	34	63	Not Bought
9	34	64	Not Bought
10	34	66	Bought
11	35	41	Not Bought

R Program & Output

```r
library(e1071)
fit.svm <- svm(Response~., type="C",data=training, kernel="linear")
svm.pred <- predict(fit.svm, newdata=validation)
svm.perf <- table(svm.pred, validation$Response, dnn=c("Predicted","Actual"))
confusionMatrix(svm.perf)
```

```
Confusion Matrix and Statistics

            Actual
Predicted     Bought Not Bought
  Bought          18          1
  Not Bought       3         31

               Accuracy : 0.9245
                 95% CI : (0.8179, 0.9791)
    No Information Rate : 0.6038
    P-Value [Acc > NIR] : 1.498e-07

                  Kappa : 0.8396
 Mcnemar's Test P-Value : 0.6171

            Sensitivity : 0.8571
            Specificity : 0.9688
         Pos Pred Value : 0.9474
         Neg Pred Value : 0.9118
             Prevalence : 0.3962
         Detection Rate : 0.3396
   Detection Prevalence : 0.3585
      Balanced Accuracy : 0.9129

       'Positive' Class : Bought
```

SVM Object

```
> fit.svm

Call:
svm(formula = Response ~ ., data = training, type = "C", kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.5

Number of Support Vectors:  26
```

- The above information indicates that for the model generated :
  - Cost = 1
  - Gamma = 0.5
  - Number of Support Vectors = 26

Visualizing the Output

- *plot.svm()* function generates a scatter plot of the input data of a SVM fit for classification models by highlighting the classes and support vectors.
- Optionally, draws a filled contour plot of the class regions.

Syntax : plot(svmObj, y ~ x, ...)

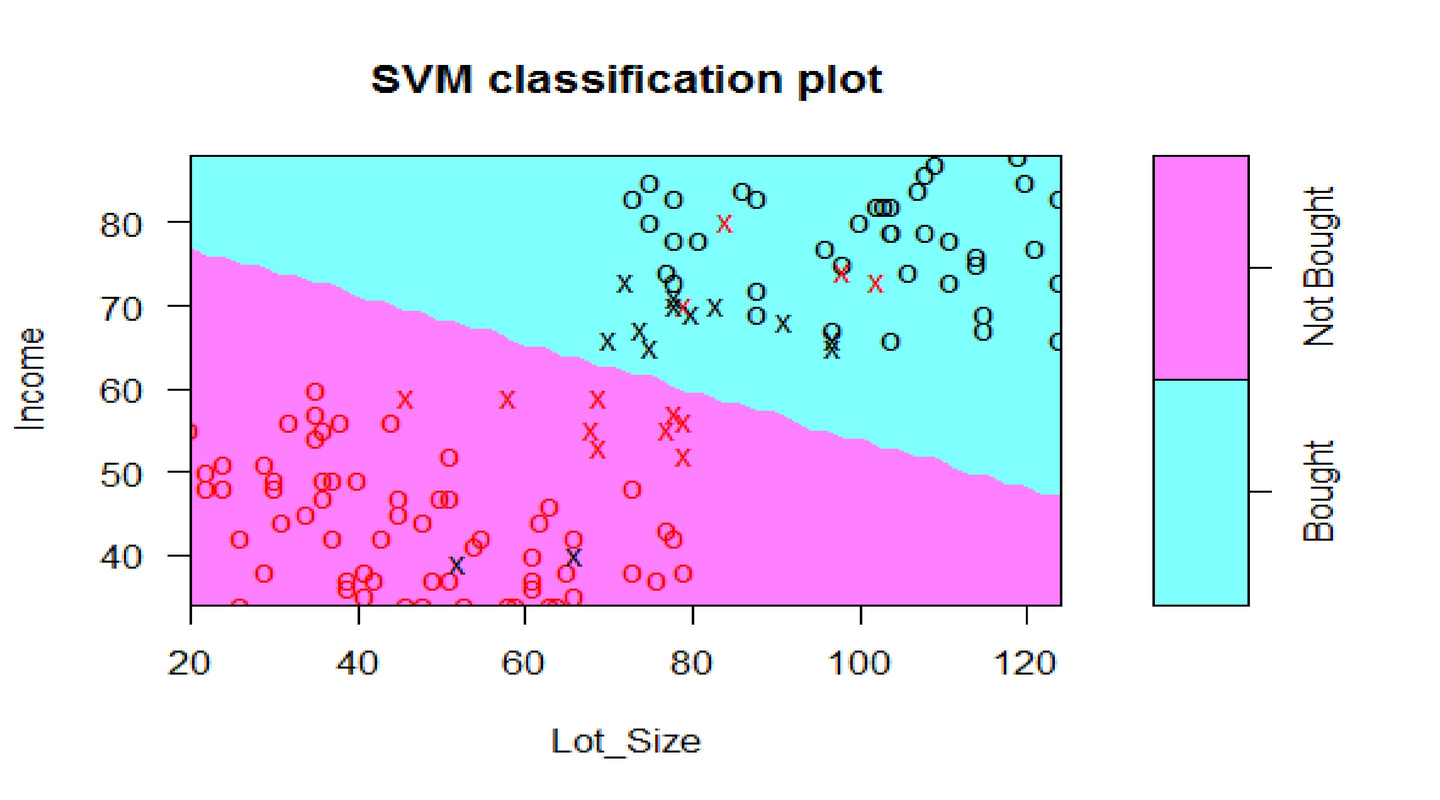
Where

svmObj : Object generated from function call of *svm()*

y , x : y and x variables respectively to be dispalyed on the plot

Plot

```
plot(fit.svm, training, Income~Lot_Size)
```



Tuning SVM

- The SVM model can be tuned with the function *tune()*
- The function tune() gives us the error values for different tuning parameters and also the best model among the inputted parameter list

Syntax : tune(method, formula, data, …)

Where

method : function to be tuned

formula, data : with their usual meaning

Tuning SVM - Example

```
> tune.out <- tune(svm,Response~.,data = training, kernel="linear",
+ ranges=list(cost=c(0.001,0.002,0.005,0.007,0.008,0.01,0.1,1,2,3,4)))
> summary(tune.out)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost
 0.01

- best performance: 0.0474359

- Detailed performance results:
    cost      error dispersion
1  0.001 0.41089744 0.07025009
2  0.002 0.41089744 0.07025009
3  0.005 0.11025641 0.09189525
4  0.007 0.07115385 0.06766794
5  0.008 0.06346154 0.07121830
6  0.010 0.04743590 0.05466078
7  0.100 0.04743590 0.05466078
8  1.000 0.04743590 0.05466078
9  2.000 0.04743590 0.05466078
10 3.000 0.04743590 0.05466078
11 4.000 0.04743590 0.05466078
```

Tuning Non-Linear

- For kernel = “polynomial”, `degree` argument can be tried for various values, as degree being actually degree of polynomial
- For kernel = “radial”, `gamma` argument can be tried for various values

Example : Kyphosis

- The kyphosis (package = ***rpart***) data frame has 81 rows and 4 columns. representing data on children who have had corrective spinal surgery
- Attributes:
  - **Kyphosis** : a factor with levels absent present indicating if a kyphosis (a type of deformation) was present after the operation.
  - **Age** : in months
  - **Number** : the number of vertebrae involved
  - **Start** : the number of the first (topmost) vertebra operated on

R Program & Output

```
> fit.svm

Call:
svm(formula = Kyphosis ~ ., data = training, type = "C", kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.3333333

Number of Support Vectors:  26
```

```
tune.out <- tune(svm,Kyphosis~.,data = training, kernel="radial",
               ranges=list(gamma=c(0.001,0.002,0.005,0.007,0.008,0.01,0.1,1,2,3,4),
                           cost=c(0.001,0.002,0.005,0.007,0.008,0.01,0.1,1,2,3,4)))
```

```
> summary(tune.out)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
   0.1    3

- best performance: 0.12
```