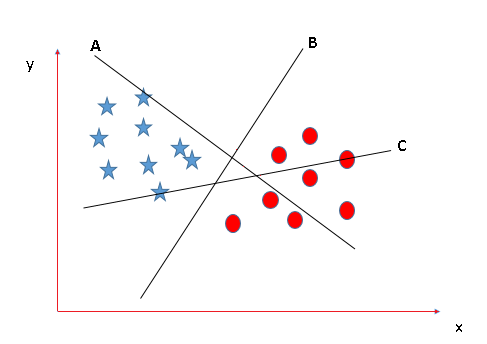
**Classifying data using Support Vector Machines(SVMs) in R**

In machine learning, Support vector machines (SVM) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. It is mostly used in classification problems. In this algorithm, each data item is plotted as a point in n-dimensional space (where n is a number of features), with the value of each feature being the value of a particular coordinate. Then, classification is performed by finding the hyper-plane that best differentiates the two classes.   
In addition to performing linear classification, SVMs can efficiently perform a non-linear classification, implicitly mapping their inputs into high-dimensional feature spaces.

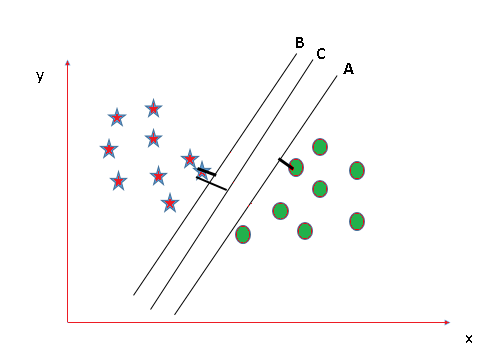
### How SVM works

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 that categorizes new examples.  
The most important question that arises while using SVM is how to decide the right hyperplane. Consider the following scenarios:

* **Scenario 1:**   
  In this scenario, there are three hyperplanes called A, B, C. Now the problem is to identify the right hyper-plane which best differentiates the stars and the circles.



* The thumb rule to be known, before finding the right hyperplane, to classify star and circle is that the hyperplane should be selected which segregate two classes better.  
  In this case, B classifies star and circle better, hence it is a right hyperplane.
* **Scenario 2:**   
  Now take another Scenario where all three planes are segregating classes well. Now the question arises of how to identify the right plane in this situation.



* In such scenarios, calculate the margin which is the distance between the nearest data point and hyper-plane. The plane has the maximum distance will be considered as the right hyperplane to classify the classes better.  
  Here C is having the maximum margin and hence it will be considered as a right hyperplane.

Above are some scenarios to identify the right hyper-plane.

# Importing the dataset

dataset = read.csv('Social\_Network\_Ads.csv')

dataset = dataset[3:5]

**Selecting columns 3-5**   
This is done for ease of computation and implementation (to keep the example simple).

# Taking columns 3-5

dataset = dataset[3:5]

**Encoding the target feature**

# Encoding the target feature as factor

dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))

**Splitting the dataset**

# Splitting the dataset into the Training set and Test set

install.packages('caTools')

library(caTools)

set.seed(123)

split = sample.split(dataset$Purchased, SplitRatio = 0.75)

training\_set = subset(dataset, split == **TRUE**)

test\_set = subset(dataset, split == **FALSE**)

**Feature Scaling**

# Feature Scaling

training\_set[-3] = scale(training\_set[-3])

test\_set[-3] = scale(test\_set[-3])

**Fitting SVM to the training set**

# Fitting SVM to the Training set

install.packages('e1071')

library(e1071)

classifier = svm(formula = Purchased ~ .,

                 data = training\_set,

                 type = 'C-classification',

                 kernel = 'linear')

**Predicting the test set result**

# Predicting the Test set results

y\_pred = predict(classifier, newdata = test\_set[-3])

**Making Confusion Matrix**

# Making the Confusion Matrix

cm = table(test\_set[, 3], y\_pred)

**Visualizing the Training set results**

# installing library ElemStatLearn

library(ElemStatLearn)

# Plotting the training data set results

set = training\_set

X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

grid\_set = expand.grid(X1, X2)

#expand.grid() - Create a data frame from all combinations of the supplied vectors or factors.

colnames(grid\_set) = c('Age', 'EstimatedSalary')

y\_grid = predict(classifier, newdata = grid\_set)

plot(set[, -3],

     main = 'SVM (Training set)',

     xlab = 'Age', ylab = 'Estimated Salary',

     xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = **TRUE**)

points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'coral1', 'aquamarine'))

points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))

**Visualizing the Test set results**  
set = test\_set

X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

grid\_set = expand.grid(X1, X2)

colnames(grid\_set) = c('Age', 'EstimatedSalary')

y\_grid = predict(classifier, newdata = grid\_set)

plot(set[, -3], main = 'SVM (Test set)',

     xlab = 'Age', ylab = 'Estimated Salary',

     xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = **TRUE**)

points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'coral1', 'aquamarine'))

points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))

Since in the result, a hyper-plane has been found in the Training set result and verified to be the best one in the Test set result. Hence, SVM has been successfully implemented in R.