Machine Learning

Support Vector Machine

Module Goals

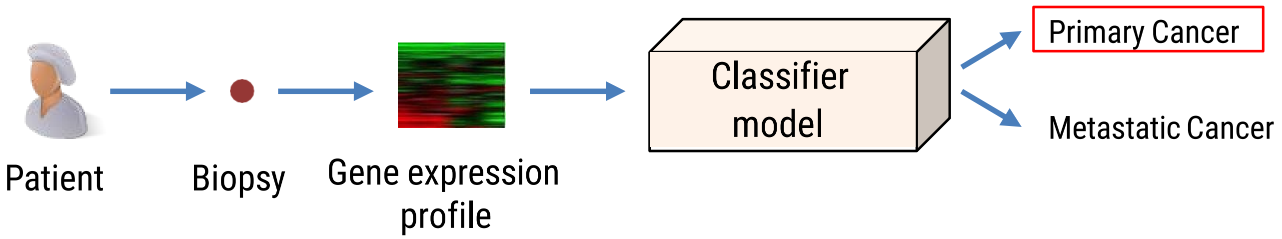
After completing this module, you should be able to:

- Formulate null and alternative hypotheses for applications involving
  - a single population mean from a normal distribution
  - a single population proportion (large samples)
- Formulate a decision rule for testing a hypothesis
- Know how to use the critical value and p-value approaches to test the null hypothesis (for both mean and proportion problems)
- Know what Type I and Type II errors are
- Assess the power of a test

Digital Vidya

Data-analysis problems of interest

1. Build computational classification models (or "*classifiers*") that assign patients/samples into two or more classes.
   - Classifiers can be used for diagnosis, outcome prediction, and other classification tasks.
   - E.g., build a decision-support system to diagnose primary and metastatic cancers from gene expression profiles of the patients:

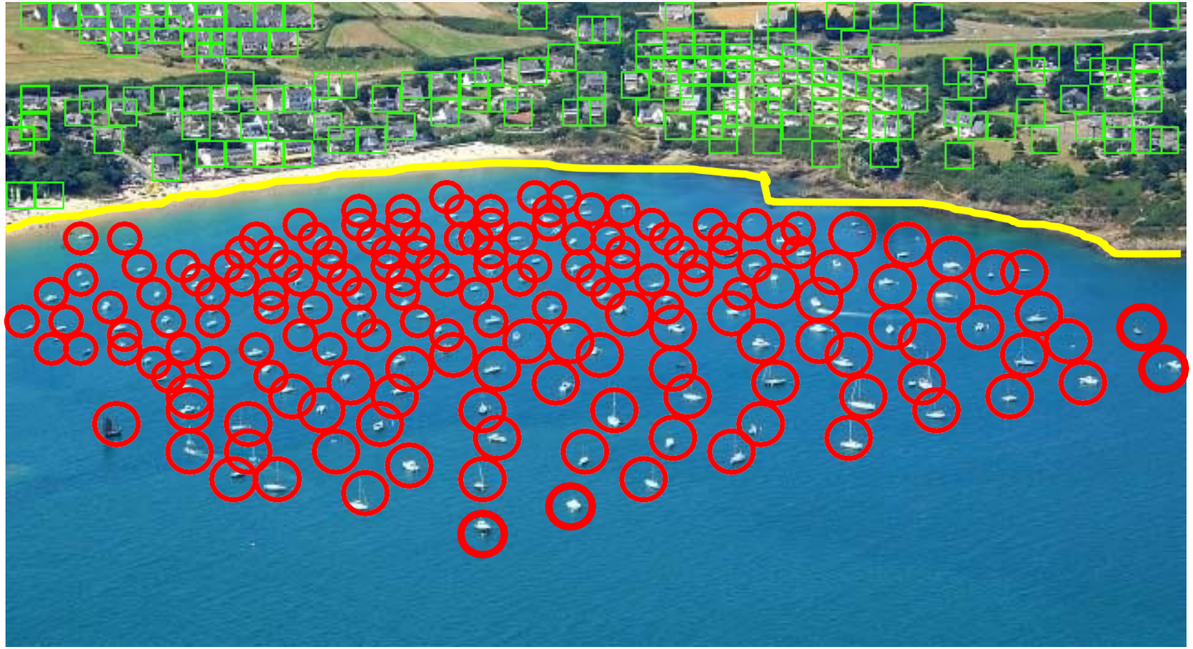


Basic principles of classification

- Want to classify objects as boats and houses.

Digital Vidya

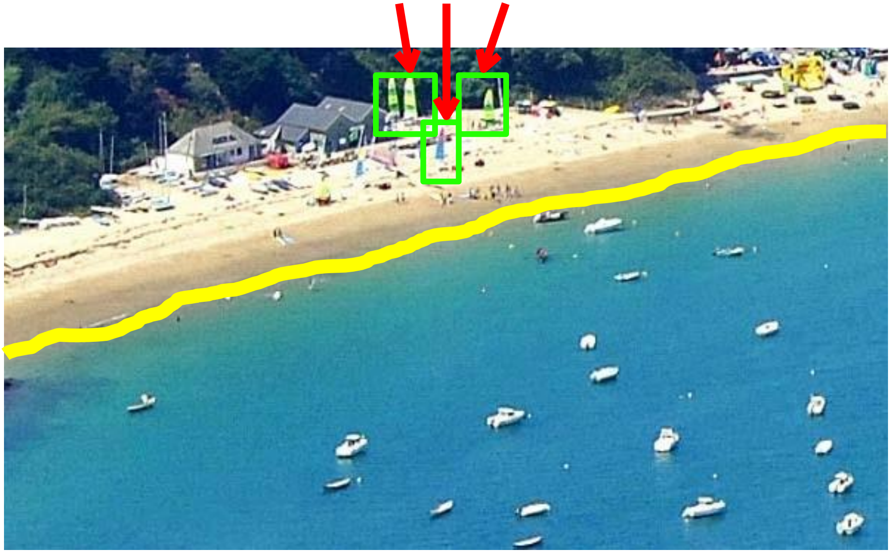
Basic principles of classification



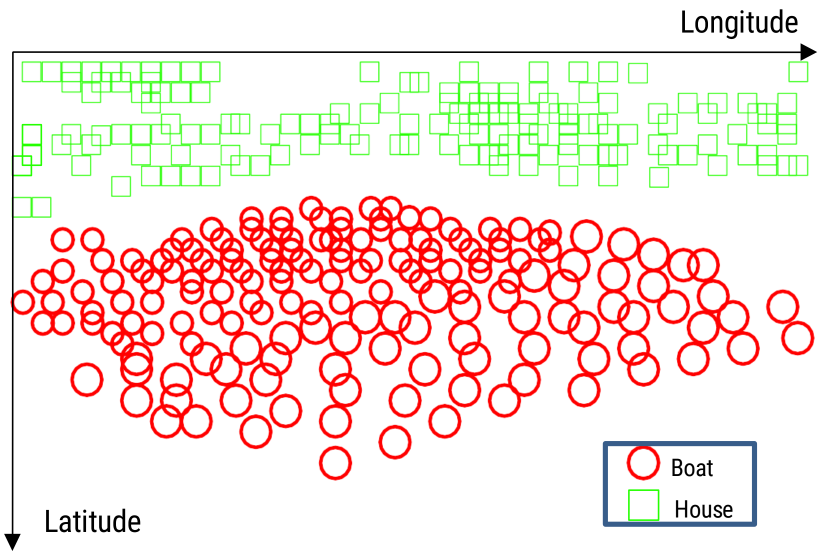
- All objects before the coast line are boats and all objects after the coast line are houses.
- Coast line serves as a *decision surface* that separates two classes.

Basic principles of classification

These boats will be misclassified as houses

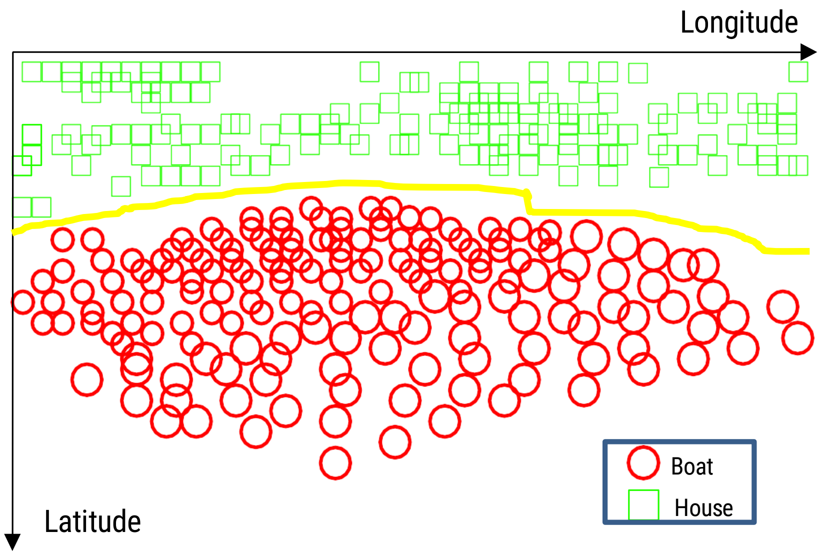


Basic principles of classification



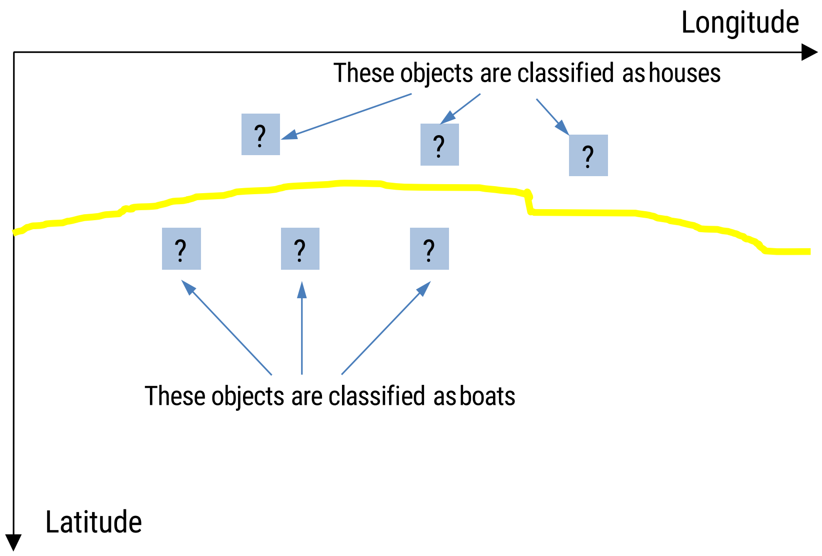
- The methods that build classification models (i.e., "*classification algorithms*") operate very similarly to the previous example.
- First all objects are represented geometrically.

Basic principles of classification



Then the algorithm seeks to find a decision surface that separates classes of objects

Basic principles of classification



Unseen (new) objects are classified as “boats” if they fall below the decision surface and as “houses” if the fall above it

Digital Vidya

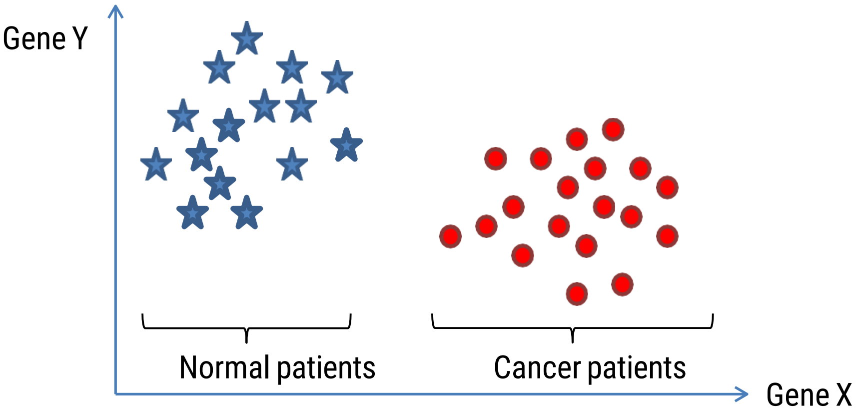
The Support Vector Machine (SVM) approach

- Support vector machines (SVMs) is a binary classification algorithm

- SVMs are important because of (a) theoretical reasons:
  - Robust to very large number of variables and small samples
  - Can learn both simple and highly complex classification models
  - Employ sophisticated mathematical principles to avoid overfitting

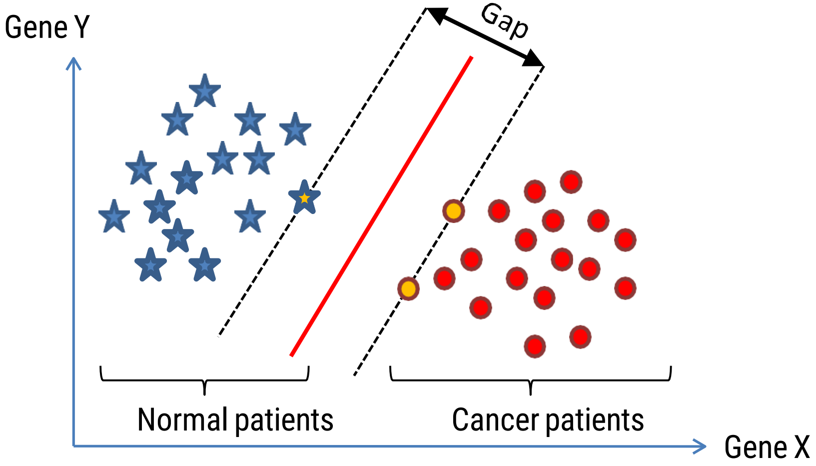
  and (b) superior empirical results.

Main ideas of SVMs



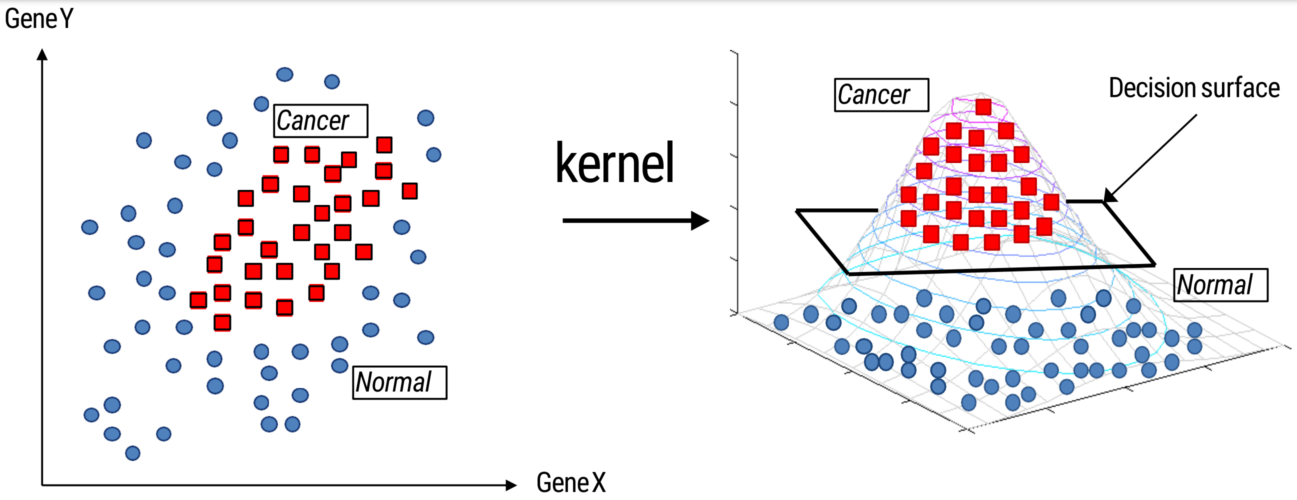
- Consider example dataset described by 2 genes, gene X and gene Y
- Represent patients geometrically (by "vectors")

Main ideas of SVMs



- Find a linear decision surface ("hyperplane") that can separate patient classes <u>and</u> has the largest distance (i.e., largest "gap" or "margin") between border-line patients (i.e., "support vectors");

Main ideas of SVMs



- If such linear decision surface does not exist, the data is mapped into a much higher dimensional space (“feature space”) where the separating decision surface is found;
- The feature space is constructed via very clever mathematical projection (“kernel trick”).

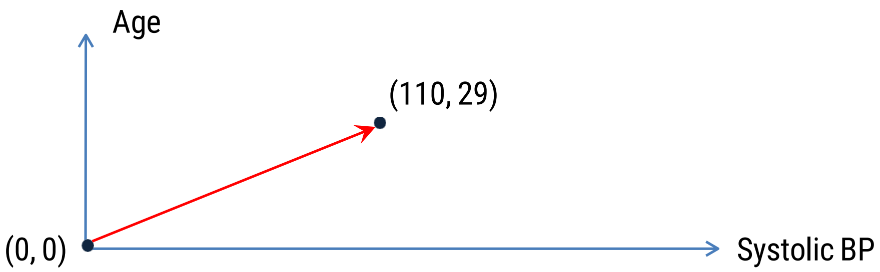
Necessary mathematical concepts

How to represent samples geometrically? Vectors in n-dimensional space ($\mathbb{R}^n$)

- Assume that a sample/patient is described by *n* characteristics ("features" or "variables")
- **Representation:** Every sample/patient is a vector in $\mathbb{R}^n$ with tail at point with 0 coordinates and arrow-head at point with the feature values.
- **Example:** Consider a patient described by 2 features:

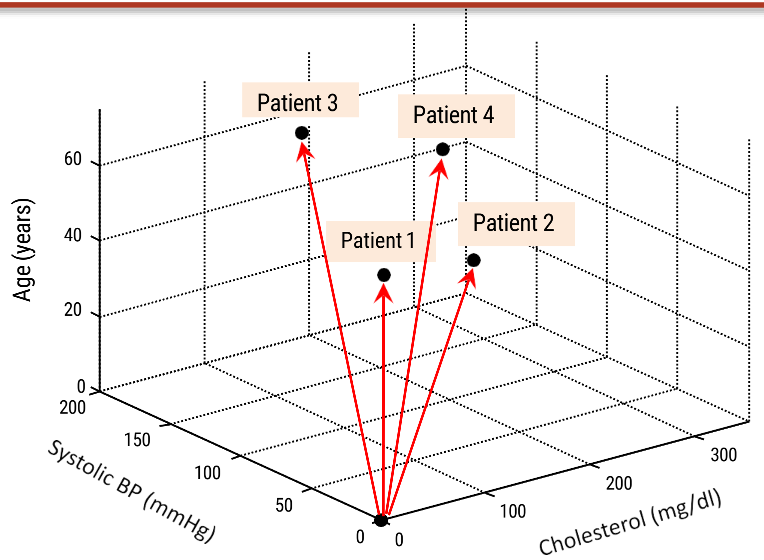
  Systolic BP = 110 and *Age* = 29.

  This patient can be represented as a vector in $\mathbb{R}^2$:



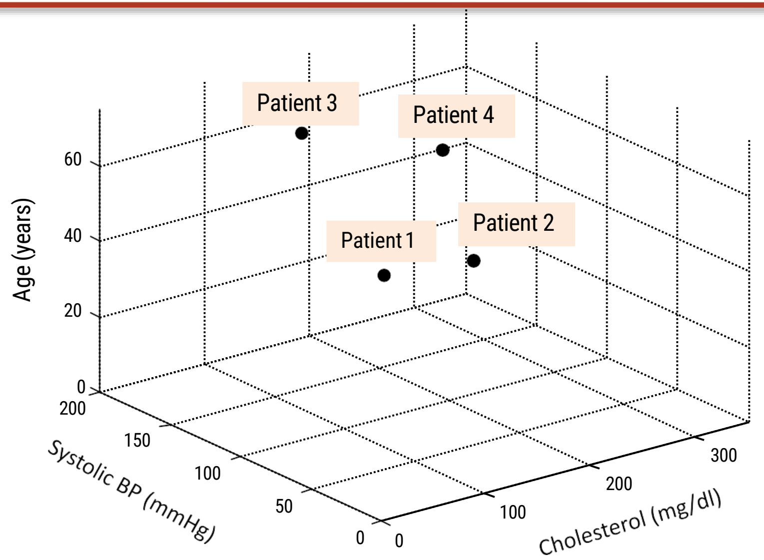
Digital Vidya

How to represent samples geometrically? Vectors in n-dimensional space ($\mathbb{R}^n$)



Patient id	Cholesterol (mg/dl)	Systolic BP (mmHg)	Age (years)	Tail of the vector	Arrow-head of the vector
1	150	110	35	(0,0,0)	(150, 110, 35)
2	250	120	30	(0,0,0)	(250, 120, 30)
3	140	160	65	(0,0,0)	(140, 160, 65)
4	300	180	45	(0,0,0)	(300, 180, 45)

How to represent samples geometrically? Vectors in n-dimensional space ($\mathbb{R}^n$)



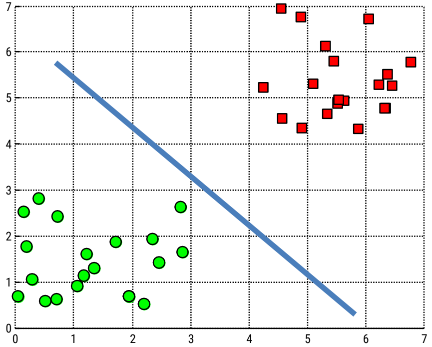
Since we assume that the tail of each vector is at point with 0 coordinates, we will also depict vectors as points (where the arrow-head is pointing).

Digital Vidya

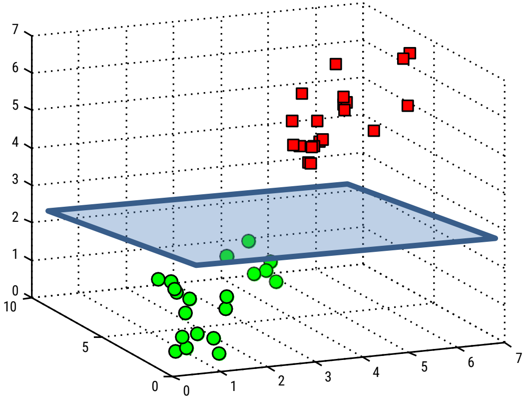
Purpose of vector representation

- Having represented each sample/patient as a vector allows now to geometrically represent the decision surface that separates two groups of samples/patients.

A decision surface in $\mathbb{R}^2$



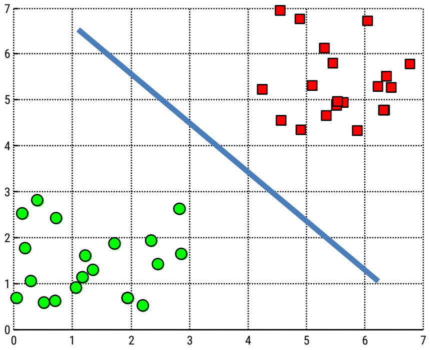
A decision surface in $\mathbb{R}^3$



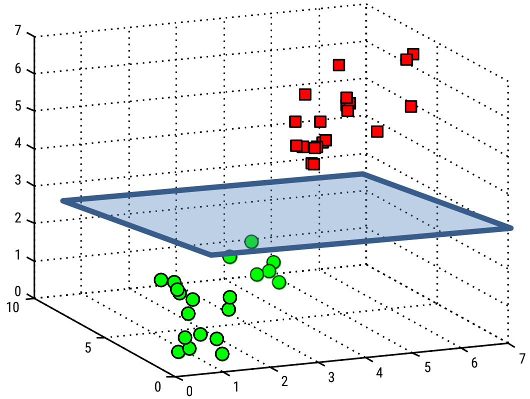
Hyperplanes as decision surfaces

- A hyperplane is a linear decision surface that splits the space into two parts;
- It is obvious that a hyperplane is a binary classifier.

A hyperplane in $\mathbb{R}^2$ is a line



A hyperplane in $\mathbb{R}^3$ is a plane

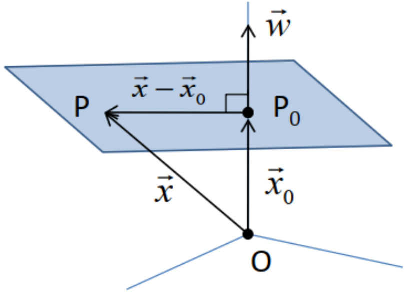


A hyperplane in $\mathbb{R}^n$ is an n-1 dimensional subspace

Digital Vidya

Equation of a hyperplane

Consider the case of $\mathbb{R}^3$:



An equation of a hyperplane is defined by a point (P_0) and a perpendicular vector to the plane ($\vec{w}$) at that point.

Define vectors: $\vec{x}_0 = \overrightarrow{OP_0}$ and $\vec{x} = \overrightarrow{OP}$, where *P* is an arbitrary point on a hyperplane.

A condition for *P* to be on the plane is that the vector $\vec{x}-\vec{x}_0$ is perpendicular to $\vec{w}$:

$$\vec{w}\cdot(\vec{x}-\vec{x}_0)=0 \quad \text{or}$$

$$\vec{w}\cdot\vec{x}-\vec{w}\cdot\vec{x}_0=0 \quad \text{define } b=-\vec{w}\cdot\vec{x}_0$$

$$\boxed{\vec{w}\cdot\vec{x}+b=0}$$

The above equations also hold for $\mathbb{R}^n$ when n>3.

Equation of a hyperplane

<u>Example</u>

$\vec{w} = (4,-1,6)$

$P_0 = (0,1,-7)$

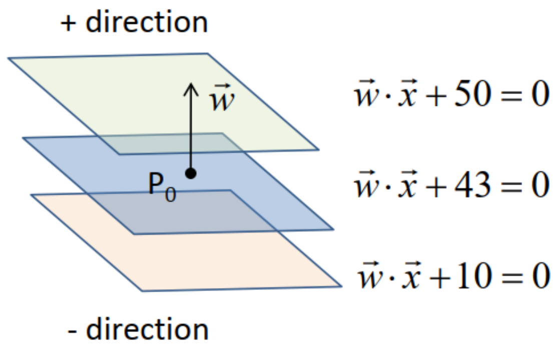
$b = -\vec{w} \cdot \vec{x}_0 = -(0-1-42) = 43$

$\Rightarrow \vec{w} \cdot \vec{x} + 43 = 0$

$\Rightarrow (4,-1,6) \cdot \vec{x} + 43 = 0$

$\Rightarrow (4,-1,6) \cdot (x_{(1)}, x_{(2)}, x_{(3)}) + 43 = 0$

$\Rightarrow 4x_{(1)} - x_{(2)} + 6x_{(3)} + 43 = 0$

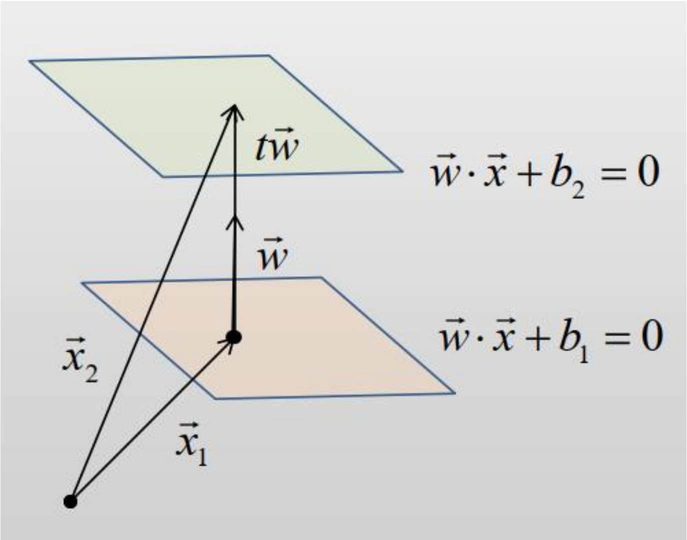


What happens if the *b* coefficient changes?
The hyperplane moves along the direction of $\vec{w}$.
We obtain "parallel hyperplanes".

Distance between two parallel hyperplanes $\vec{w} \cdot \vec{x} + b_1 = 0$ and $\vec{w} \cdot \vec{x} + b_2 = 0$ is equal to $D = |b_1 - b_2| / \|\vec{w}\|$.

Digital Vidya

(Derivation of the distance between two parallel hyperplanes)



$$\vec{x}_2=\vec{x}_1+t\vec{w}$$

$$D=\|t\vec{w}\|=|t|\|\vec{w}\|$$

$$\vec{w}\cdot\vec{x}_2+b_2=0$$

$$\vec{w}\cdot(\vec{x}_1+t\vec{w})+b_2=0$$

$$\vec{w}\cdot\vec{x}_1+t\|\vec{w}\|^2+b_2=0$$

$$(\vec{w}\cdot\vec{x}_1+b_1)-b_1+t\|\vec{w}\|^2+b_2=0$$

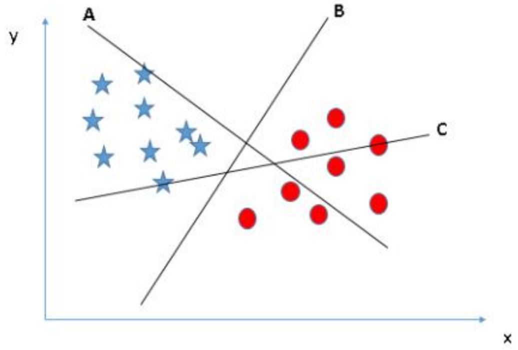
$$-b_1+t\|\vec{w}\|^2+b_2=0$$

$$t=(b_1-b_2)/\|\vec{w}\|^2$$

$$\boxed{\Rightarrow D=|t|\|\vec{w}\|=|b_1-b_2|/\|\vec{w}\|}$$

SVM Hyperplane

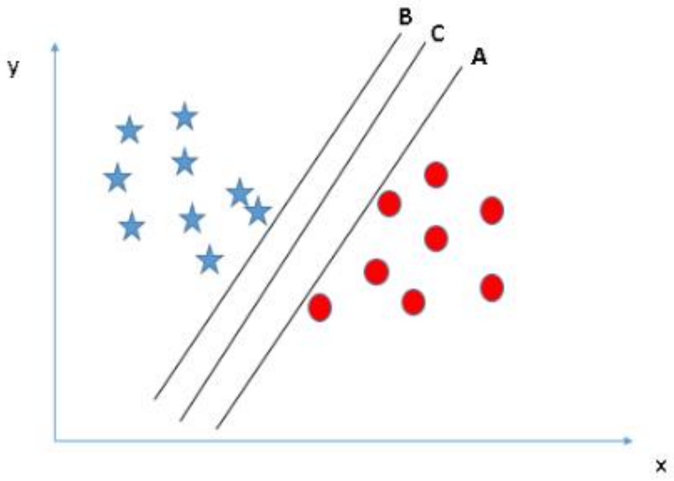
- Identify the right hyper-plane (Scenario-1): Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.



- You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.

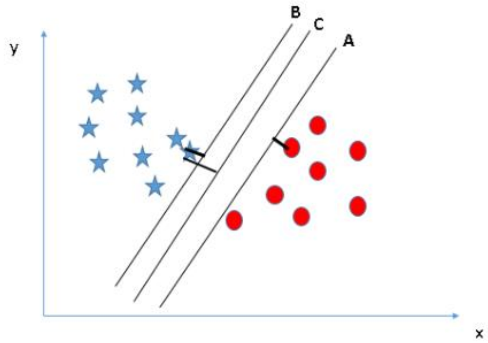
SVM Hyperplane

- **Identify the right hyper-plane (Scenario-2):** Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?



SVM Hyperplane

- Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**.

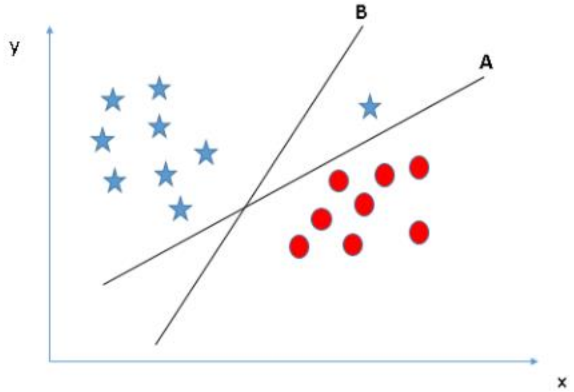


- Above, you can see that the margin for hyper-plane C is high as compared to both A and B.
- Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

Digital Vidya

SVM Hyperplane

- Identify the right hyper-plane (Scenario-3):



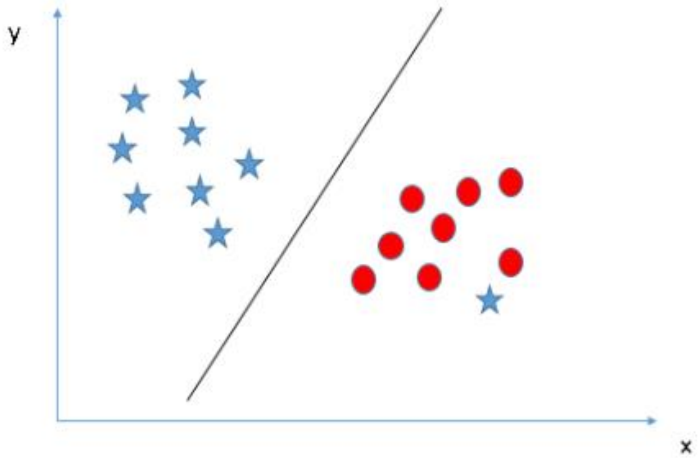
- You may select the hyper-plane **B** as it has higher margin compared to **A.** But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A.**

SVM Hyperplane

- **Can we classify two classes (Scenario-4)?:** Below, I am unable to segregate the two classes using a straight line, as one of star lies in the territory of other(circle) class as an outlier.

SVM Hyperplane

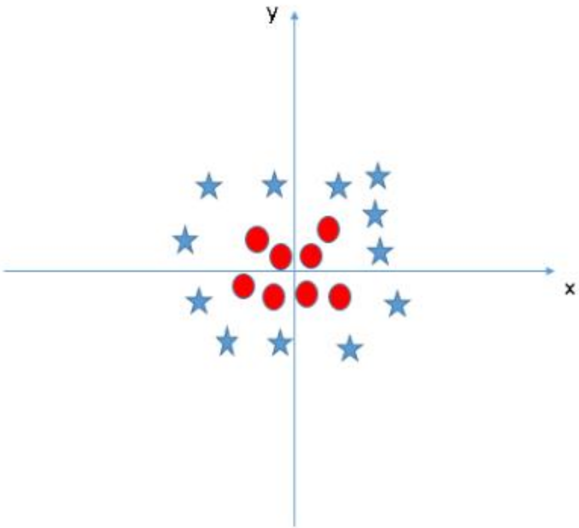
- One star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, we can say, SVM is robust to outliers.



Digital Vidya

SVM Hyperplane

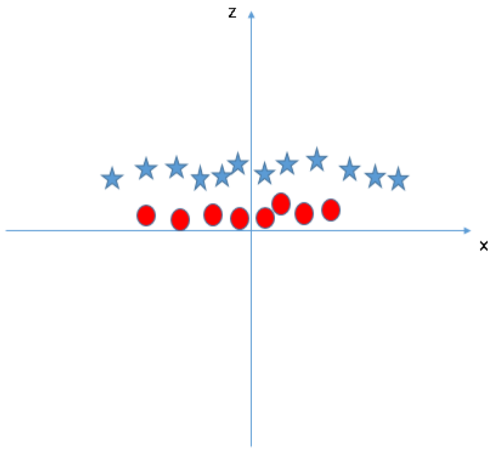
- **Find the hyper-plane to segregate to classes (Scenario-5):** In the scenario below, we can’t have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.



Digital Vidya

SVM Hyperplane

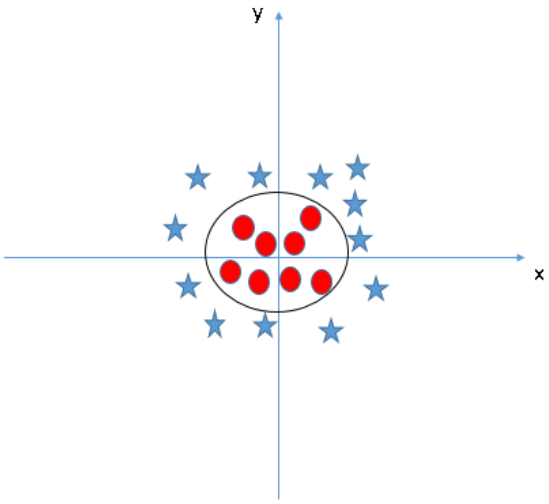
- SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:



- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

SVM Kernel

- SVM has a technique called the kernel trick.
- These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels.
- It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs you've defined.
- When we look at the hyper-plane in original input space it looks like a circle

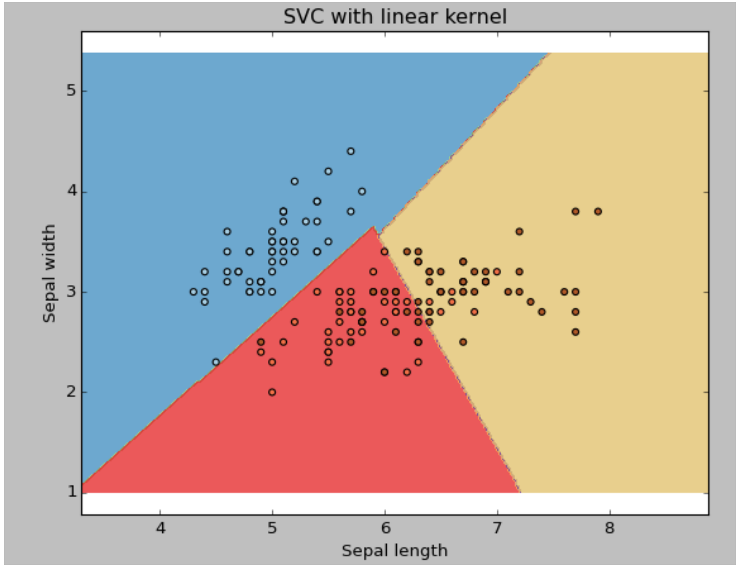


SVM Python

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets
# import some data to play with
iris = datasets.load_iris()
X = iris.data[:, :2] # we only take the first two features. We could
 # avoid this ugly slicing by using a two-dim dataset
y = iris.target
# we create an instance of SVM and fit out data. We do not scale
our
# data since we want to plot the support vectors
C = 1.0 # SVM regularization parameter
svc = svm.SVC(kernel='linear', C=1,gamma=0).fit(X, y)
# create a mesh to plot in
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
h = (x_max / x_min)/100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
np.arange(y_min, y_max, h))
plt.subplot(1, 1, 1)
Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('SVC with linear kernel')
plt.show()
```

SVM Python

- SVM Linear Kernel

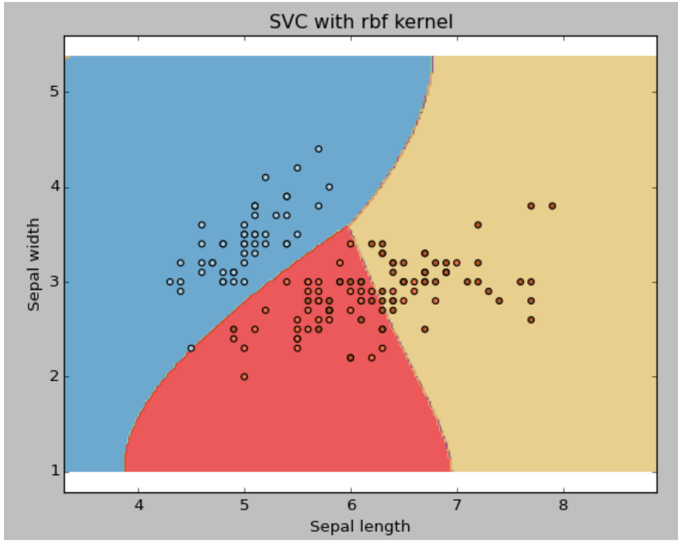


Digital Vidya

SVM Python

- SVM RBF Kernel

svc = svm.SVC(kernel='rbf', C=1,gamma=0).fit(X, y)

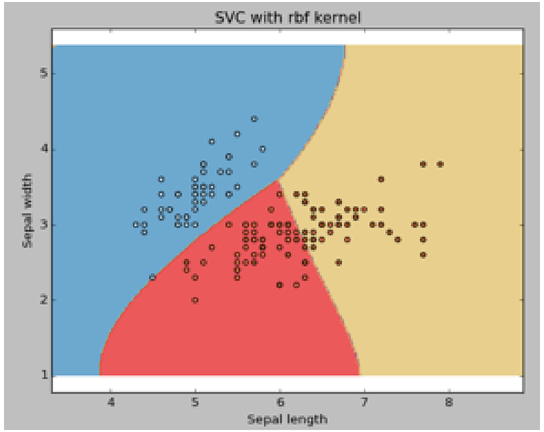


SVM Python

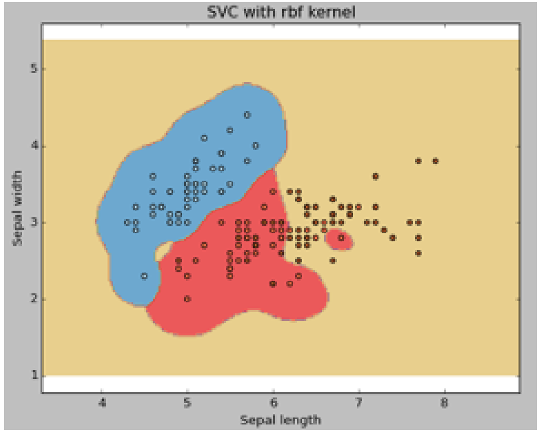
- SVM Gamma

svc = svm.SVC(kernel='rbf', C=1,gamma=0).fit(X, y)

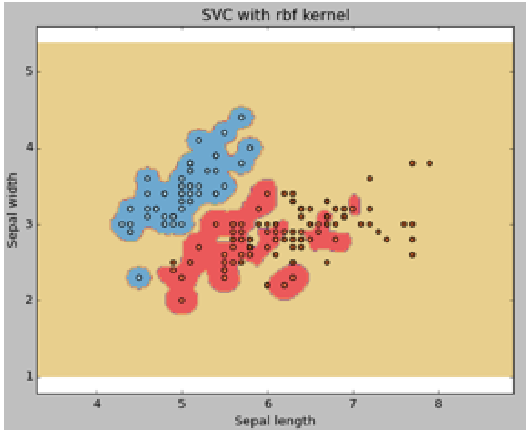
gamma =0



gamma =10



gamma =100

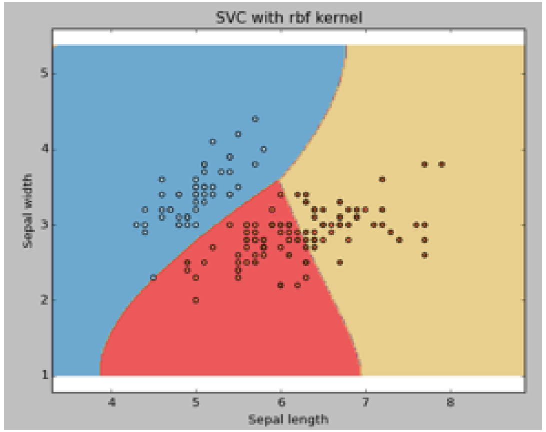


SVM Python

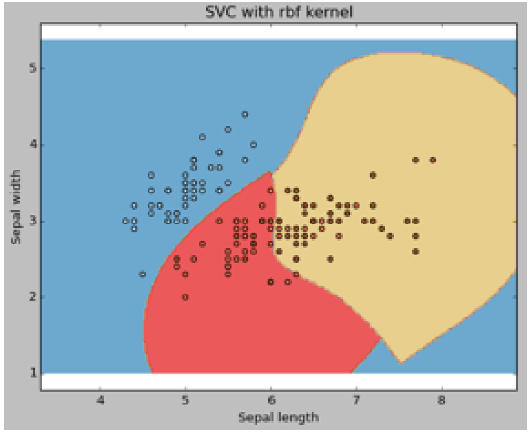
- ## SVM Penalty parameter

C: Penalty parameter C of the error term. It also controls the trade off between smooth decision boundary and classifying the training points correctly.

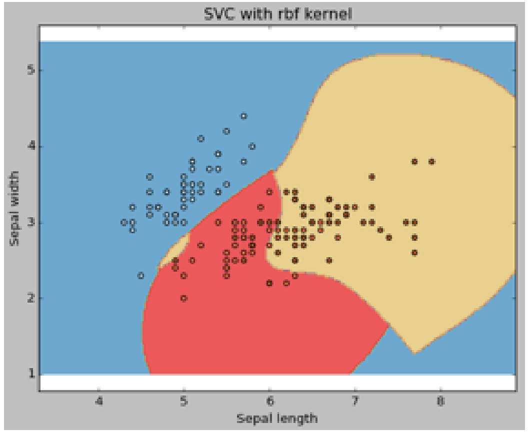
c =1



C =100



c =1000



Thank You

Digital Vidya