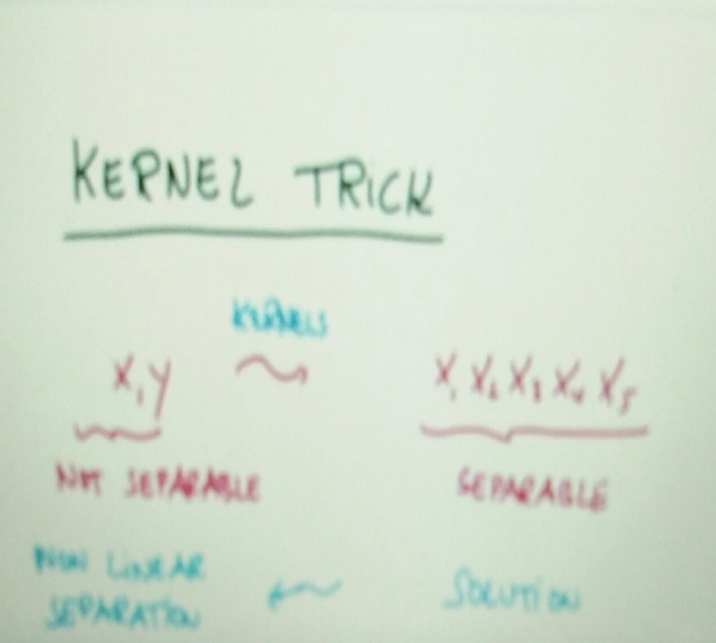
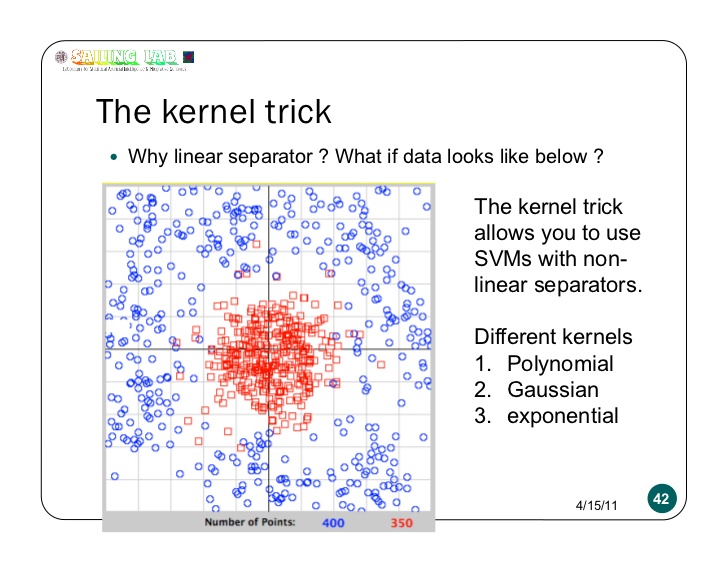
Kernal Support Vector Machine

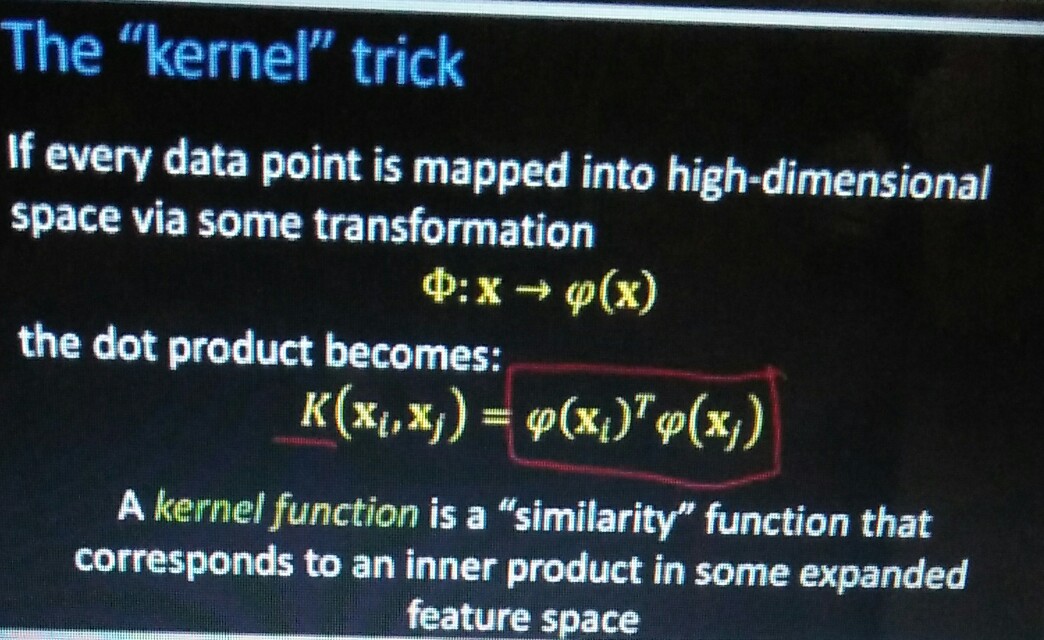
`

**Short note about Support Vector machine**:

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), support vector machines (SVMs, also support vector networks are [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) models with associated learning [algorithms](https://en.wikipedia.org/wiki/Algorithm) that analyze data used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis)..

**Kernal Trick-**

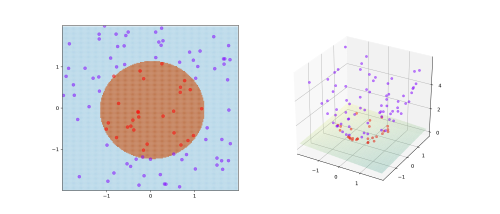
****

****

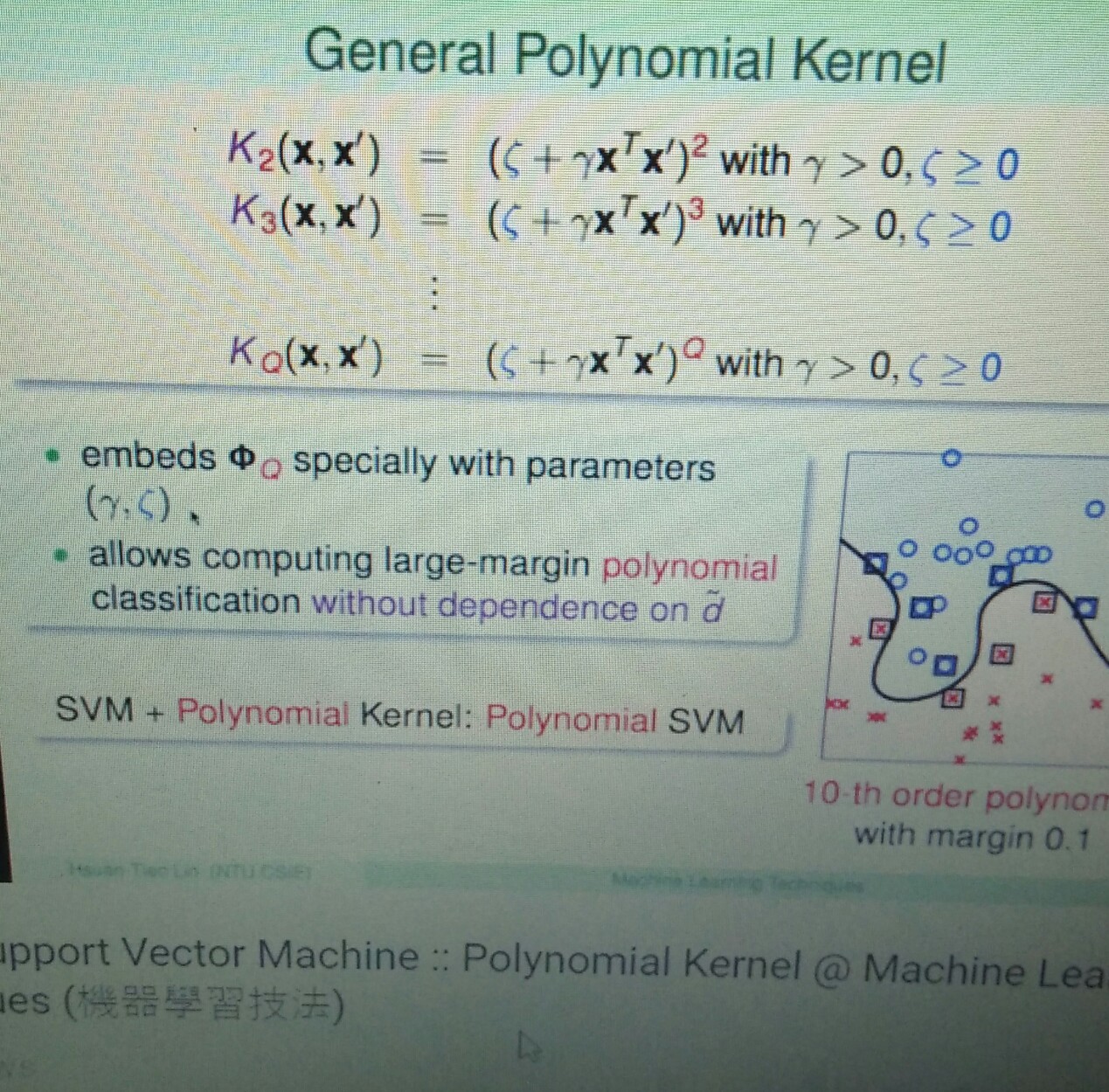
**Kernal Method-**

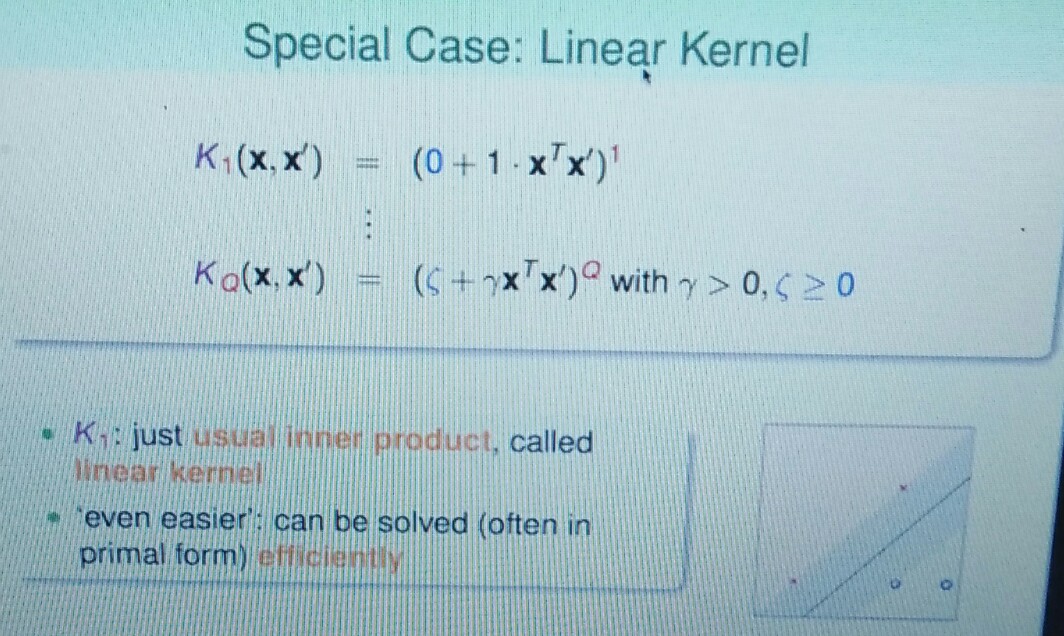
In machine learning, kernel methods are a class of algorithms for pattern analysis, whose best known member is the support vector machine (SVM). ... This approach is called the "kernel trick".

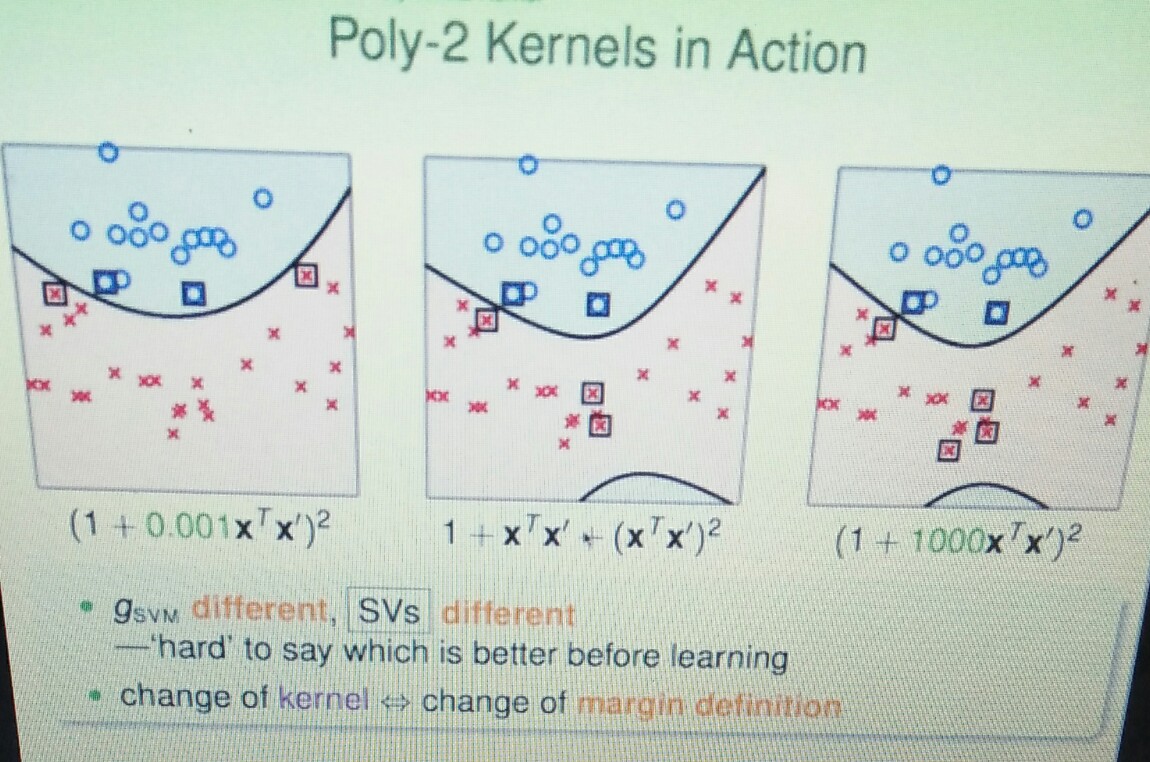
A Kernal function is a “similarity” function that corresponds to an inner product in some expanded featured space

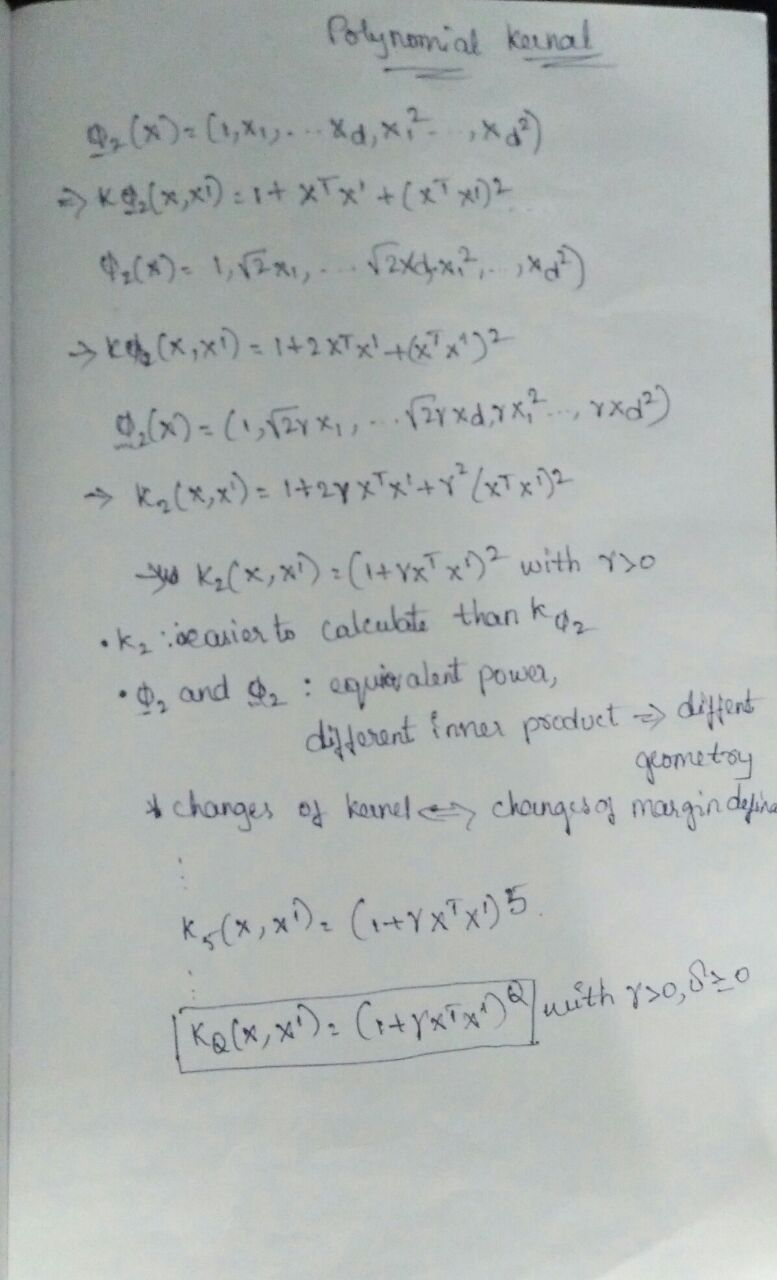


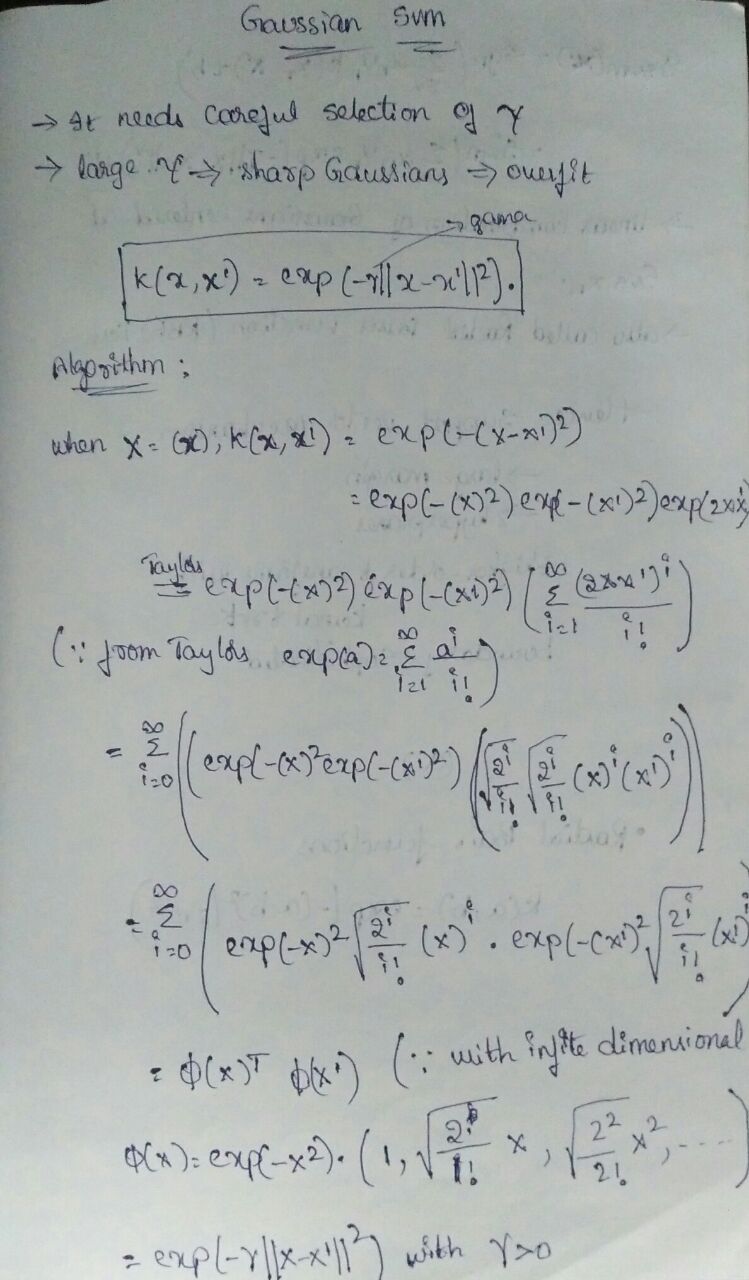
Decode Complex Algorithm

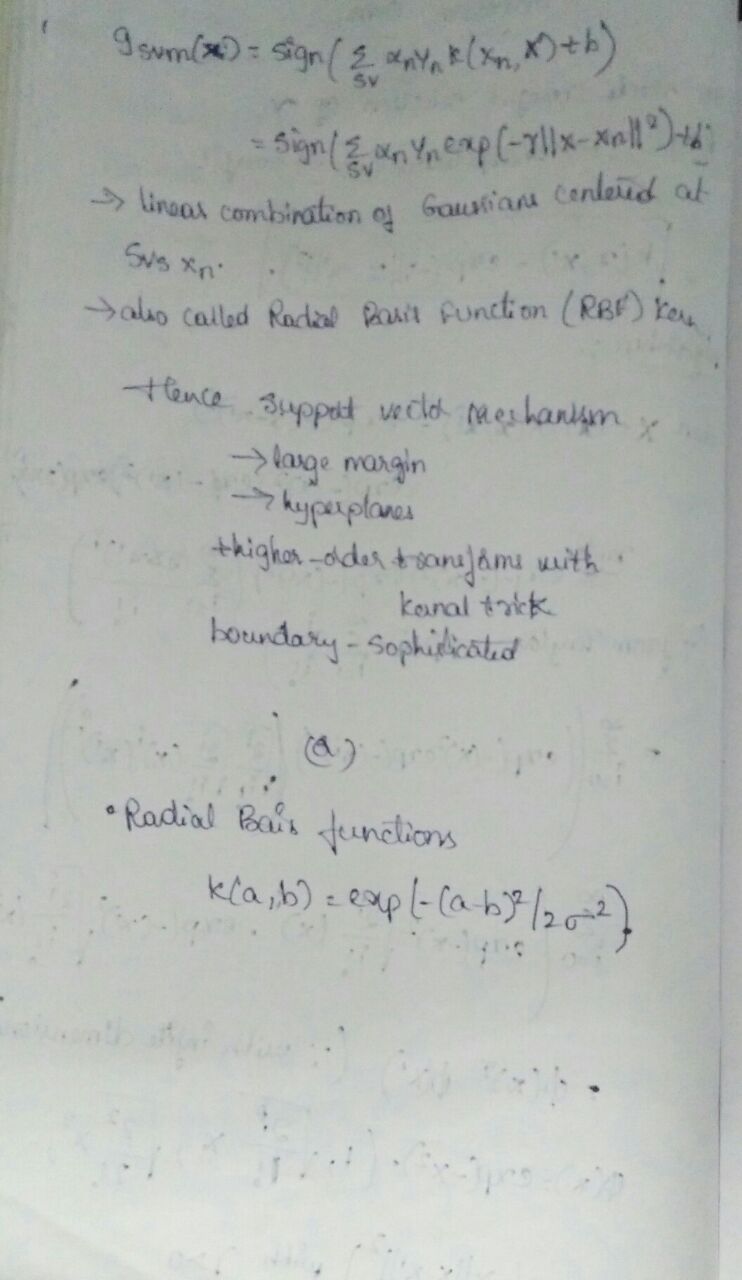




****

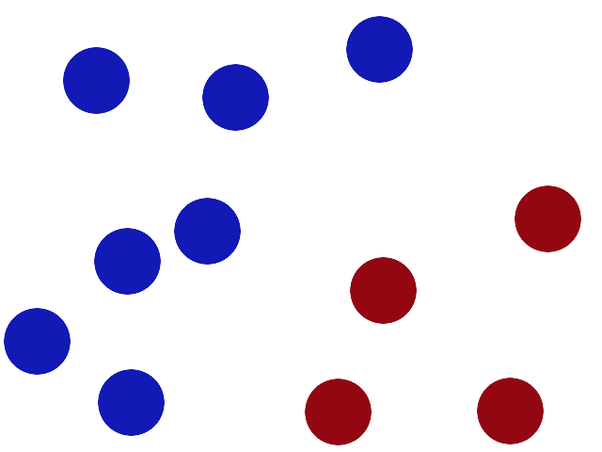
****

****

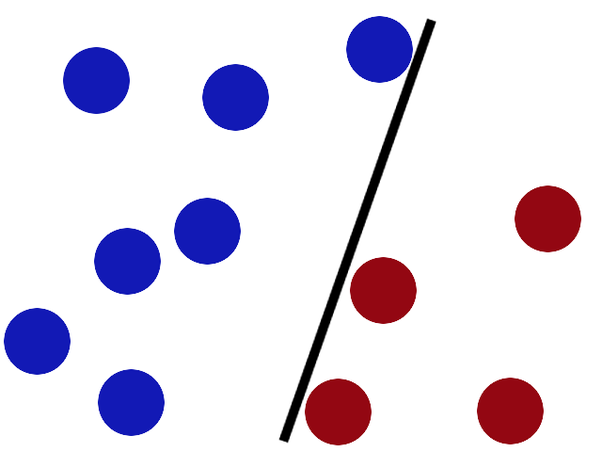
****

**Example-**

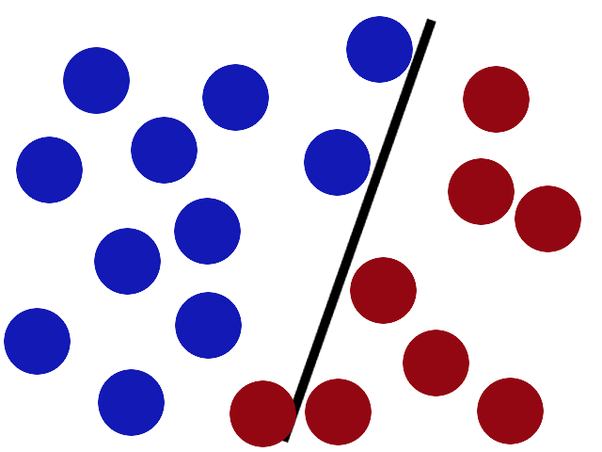
We have 2 colors of balls on the table that we want to separate.



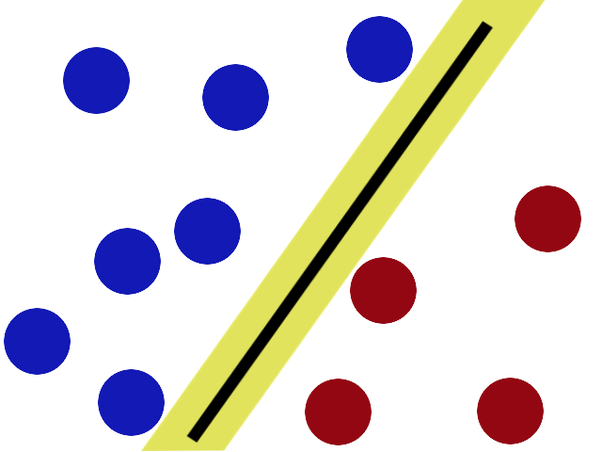
We get a stick and put it on the table, this works pretty well right?



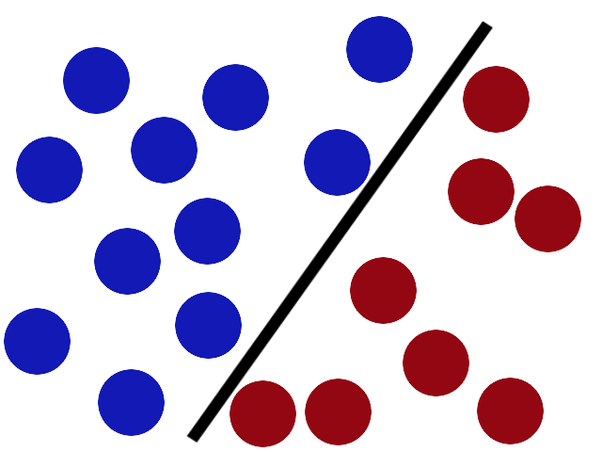
Some villain comes and places more balls on the table, it kind of works but one of the balls is on the wrong side and there is probably a better place to put the stick now.



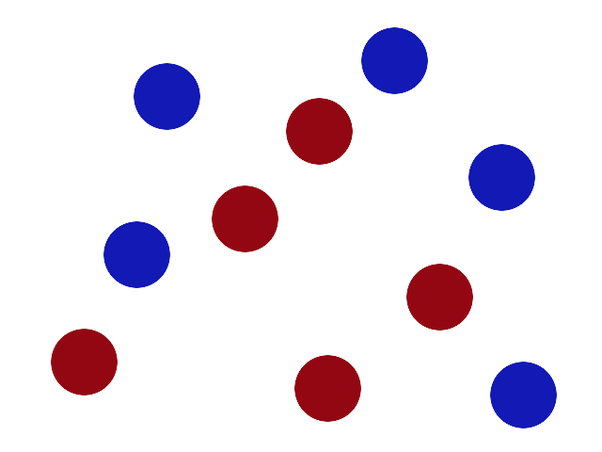
SVMs try to put the stick in the best possible place by having as big a gap on either side of the stick as possible.



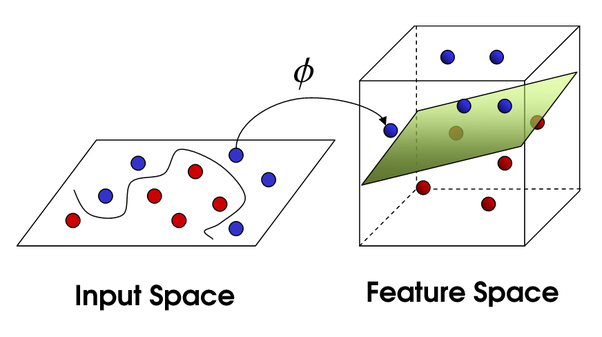
Nowwhen the villain returns the stick is still in a pretty good spot.



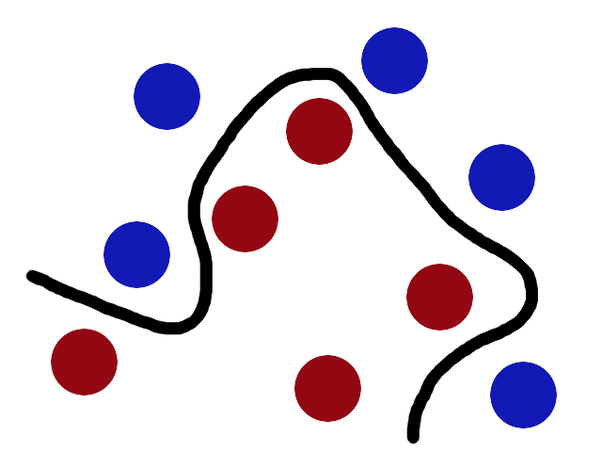
There is another trick in the SVM toolbox that is even more important. Say the villain has seen how good you are with a stick so he gives you a new challenge.

x

There’s no stick in the world that will let you split those balls well, so what do you do? You flip the table of course! Throwing the balls into the air. Then, with your pro ninja skills, you grab a sheet of paper and slip it between the balls.



Now, looking at the balls from where the villain is standing, they balls will look split by some curvy line.



Boring adults the call balls data, the stick a classifier, the biggest gap

trick optimization, call flipping the table kernelling and the piece of paper a hyperplane.

**Case Study-**

**UseCases-**

1. Finding Age and immunity
2. Finding temperature vs. Number of cones sold at ice cream store
3. Finding Population vs Food consumption
4. Finding quantity with yield
5. Finding the response of product based on features.
6. Finding the chances to get king while playing cards.
7. Finding obese person with adiposity prone to heart disease
8. Finding Product Price with respect to other vendors
9. Determining the chances to win cricket match .
10. Determining the chances of getting Jobs after Completing Graduation.
11. Speed and distance relationship
12. Finding rate of growth of the economy of a Institution

**Applications**

**Use Cases of kernel Method**

Application areas of kernel methods are diverse and include

**Geostatistics**: branch of statistics focusing on spatial or spatiotemporal datasets.

**Kriging**: A method of interpolation for which the interpolated values are modeled by a Gaussian process governed by prior covariances, as opposed to a piecewise-polynomial spline chosen to optimize smoothness of the fitted values

**Inverse distance weighting**: It is a type of deterministic method for multivariate interpolation with a known scattered set of points. The assigned values to unknown points are calculated with a weighted average of the values available at the known points.

**3D reconstruction:** It is the process of capturing the shape and appearance of real objects.

**Bioinformatics:** It is an interdisciplinary field that develops methods and software tools for understanding biological data.

**Chemoinformatic,Information extraction handwriting recognition** And many more.

**-Stock market investment prediction**

**-Clearly defining or recognizing the best fit line.**

**Eg. Hand Written digit classification and general recognizion**

**-Determines the ideal separation of two entities**

**R Code**-

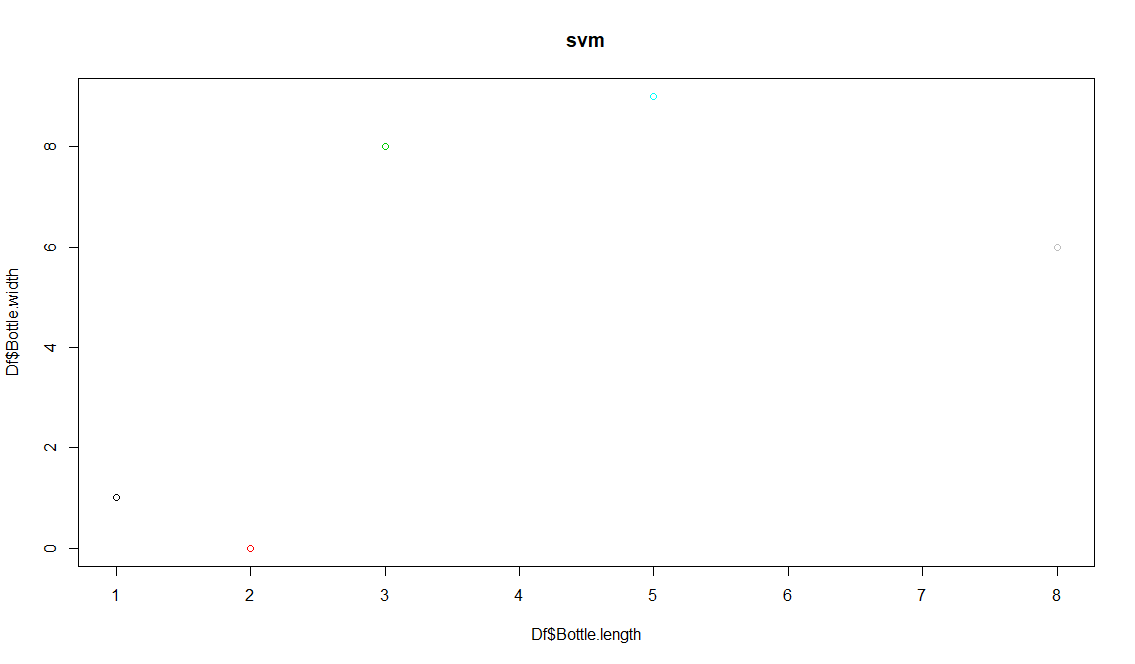
Df=data.frame(Bottle.length=c(1,2,3,5,8),Bottle.width=c(1,0,8,9,6))

install.packages("e1071")

library(e1071)

plot(Df)

plot(Df$Bottle.length,Df$Bottle.width,col=Df$Bottle.length,main="svm")



s=sample(1,1)

> svm.fit=svm(Df$Bottle.length ~Df$Bottle.width,data=Df,kernal="linear",cost=0.1,scale=FALSE,main="SVM")

> plot(svm.fit,Df)

> install.packages("e1071")

> s=sample(9,6)

> s

[1] 2 9 4 8 5 1

> svmfit=svm(Df$Bottle.width ~.,data=Df,kernal="linear",cost=10,scale=FALSE)

> plot(svmfit,Df)

> svmfit$index

[1] 1 2 3 4 5

> summary(svmfit)

Call:

svm(formula = Df$Bottle.width ~ ., data = Df, kernal = "linear", cost = 10,

scale = FALSE)

Parameters:

SVM-Type: eps-regression

SVM-Kernel: radial

cost: 10

gamma: 1

epsilon: 0.1

Number of Support Vectors: 5

**Python**

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.cross\_validation import train\_test\_split

# Importing the dataset

SVR = pd.read\_csv("C:\\Users\\Rama\\Documents\\bank data\\bank.csv")

print(SVR.head())

print(SVR.tail())

SVR

X = SVR.iloc[0:50,:1].values

y = SVR.iloc[0:50,8].values

print(X)

# Splitting the dataset into the Training set and Test set

from sklearn.cross\_validation import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Fitting K-NN Regression to the Training set

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 2, metric = 'minkowski', p = 2)

classifier.fit(X\_train, y\_train)

#Fitting SVR to the current Data

from sklearn.svm import SVR

regressor=SVR(kernel='rbf')

regressor.fit(X,y)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

#with warnings.catch\_warnings():

#import warnings

#from sklearn.exceptions import DataConversionWarning

#warnings.filterwarnings(action='ignore', category=DataConversionWarning)

#y\_predict=regressor.predict(6.5)

#null\_counts =SVR.isnull().sum()

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

#visualise the SVR results

plt.scatter(X,y,color='red')

plt.plot(regressor.predict(X),color='blue')

plt.title('SVR')

plt.xlabel('age')

plt.ylabel('balance')

