



# Support Vector Machine

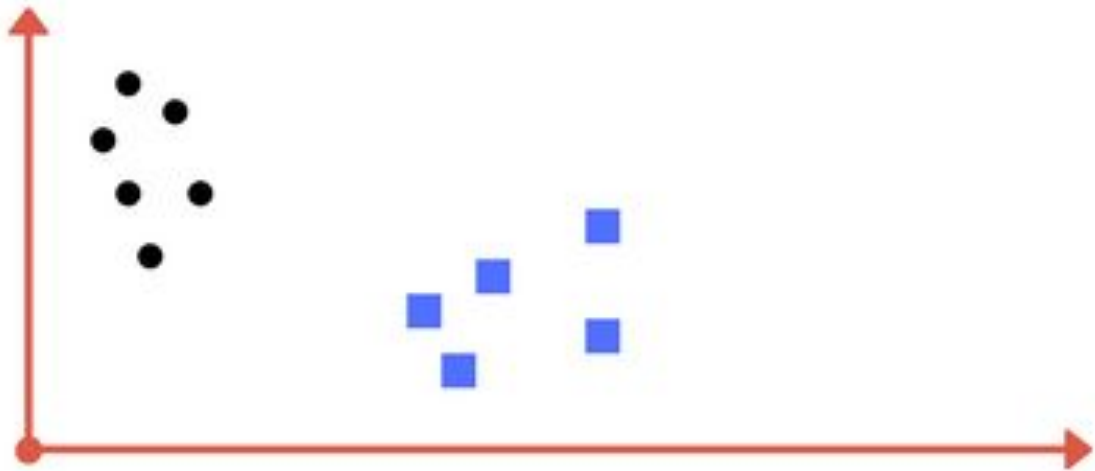
Classification , Regression and Outliers detection

# Introduction

SVM

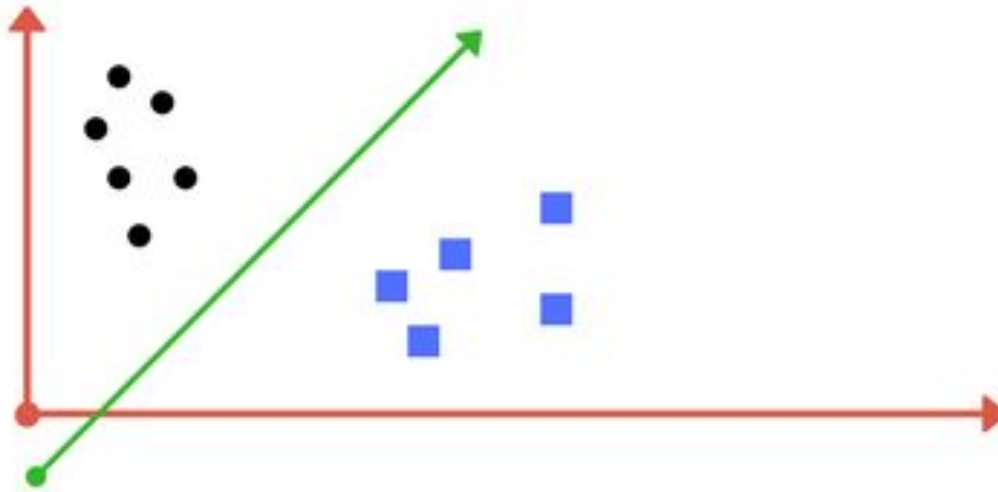
A Support Vector Machine (SVM) is a discriminative classifier which intakes training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

---



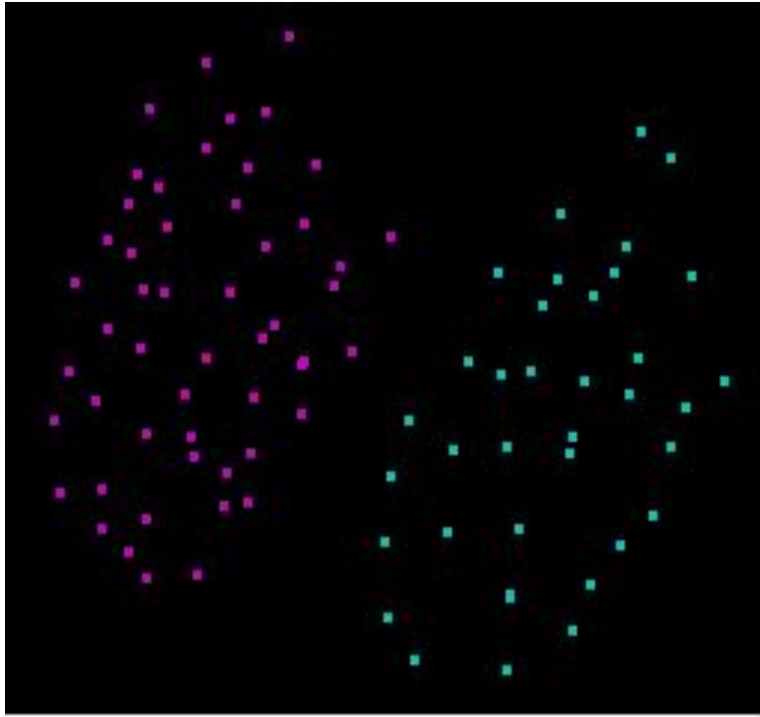
What could be drawn to classify the black dots from blue squares?

# Linearly separable data

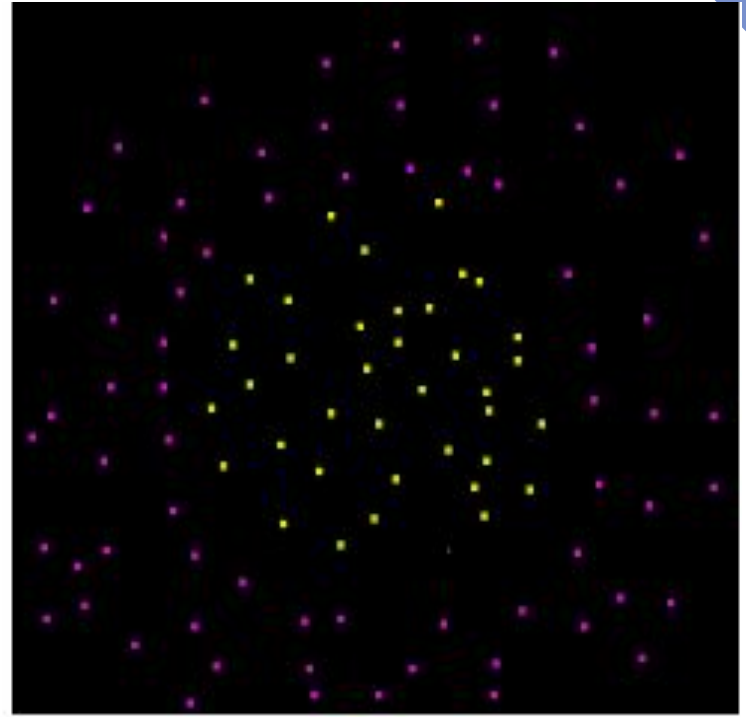


A line drawn between these data points classify the black dots and blue squares.

# Linear vs Nonlinear separable data

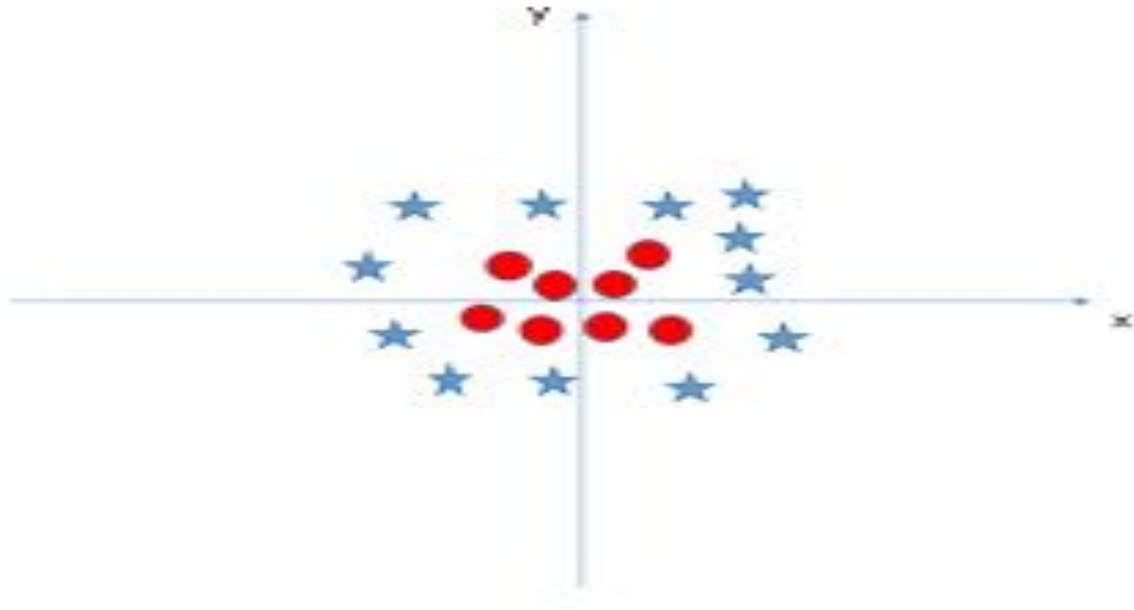


Linearly separable data



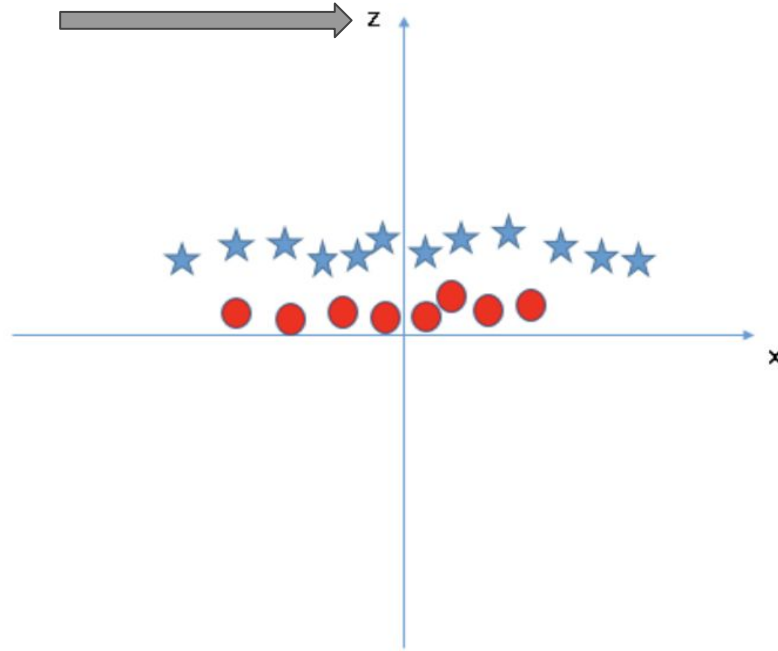
Non linearly separable data

# NonLinearly separable data



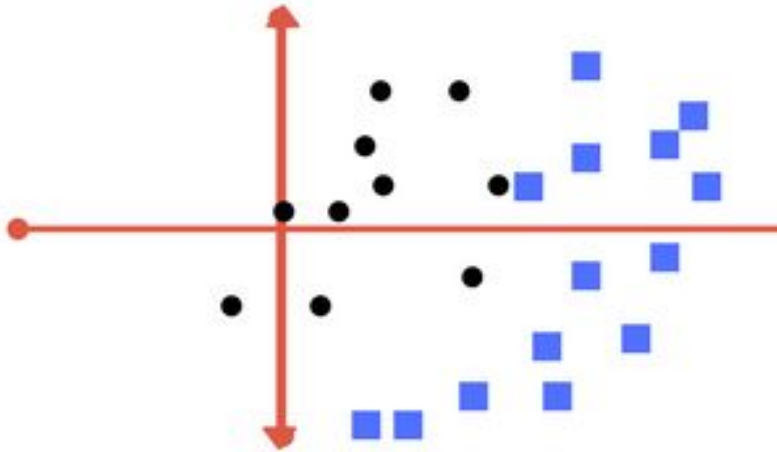
What could be drawn to classify these data points ( red dots from blue stars )?

# NonLinearly separable data

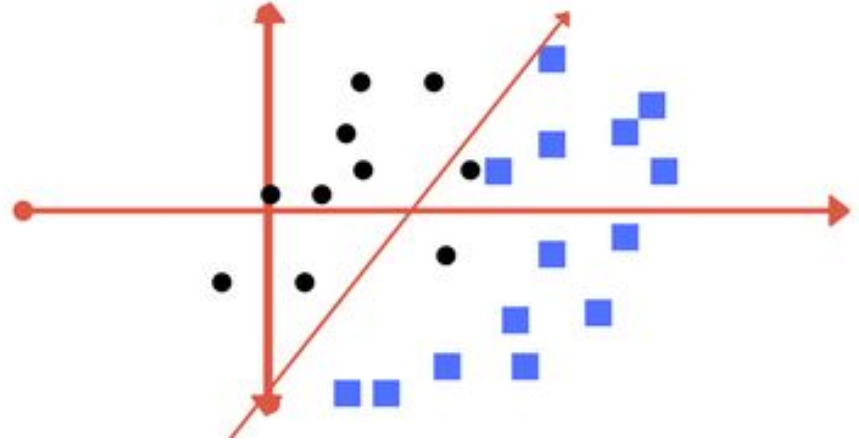


Here the hyperplane is a 2d plane drawn parallel to  $x$ -axis that is the separator.

## Non Linear data ( type 2 )



Raw Data



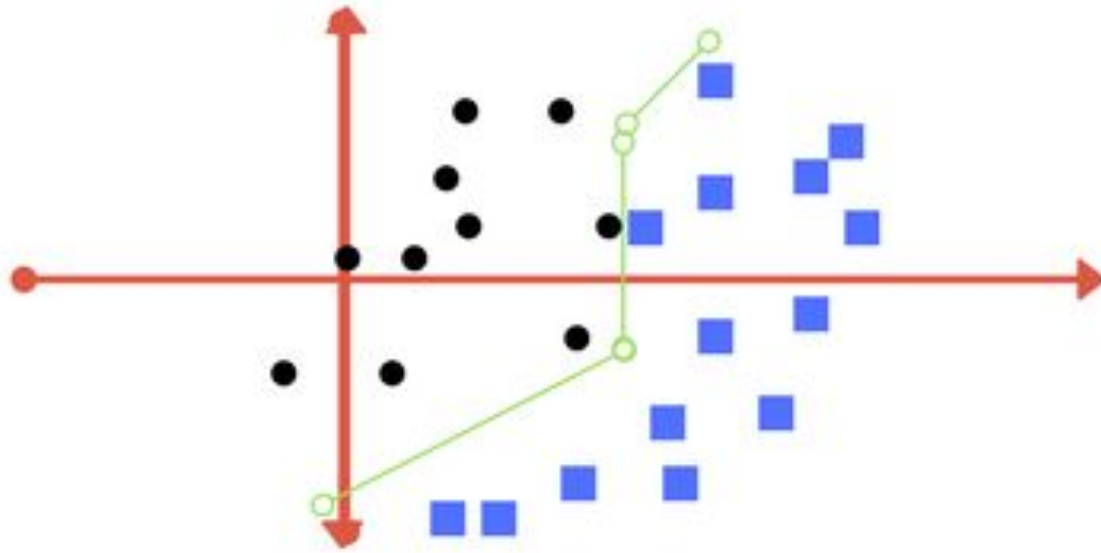
Line as Hyperplane





For the previous data the **line** , if used as a **Hyperplane**

- Two black dots also fall in category of blue squares
- Data separation is not perfect
- It tolerates some **outliers** in the classification



This type of separator best provides the classification.

But

- It is quite difficult to train a model like this .
- This is termed as **Regularisation parameter**.

# Tuning Parameters

SVM

1. Kernel
2. Regularization
3. Gamma
4. Margin

---

# Margin

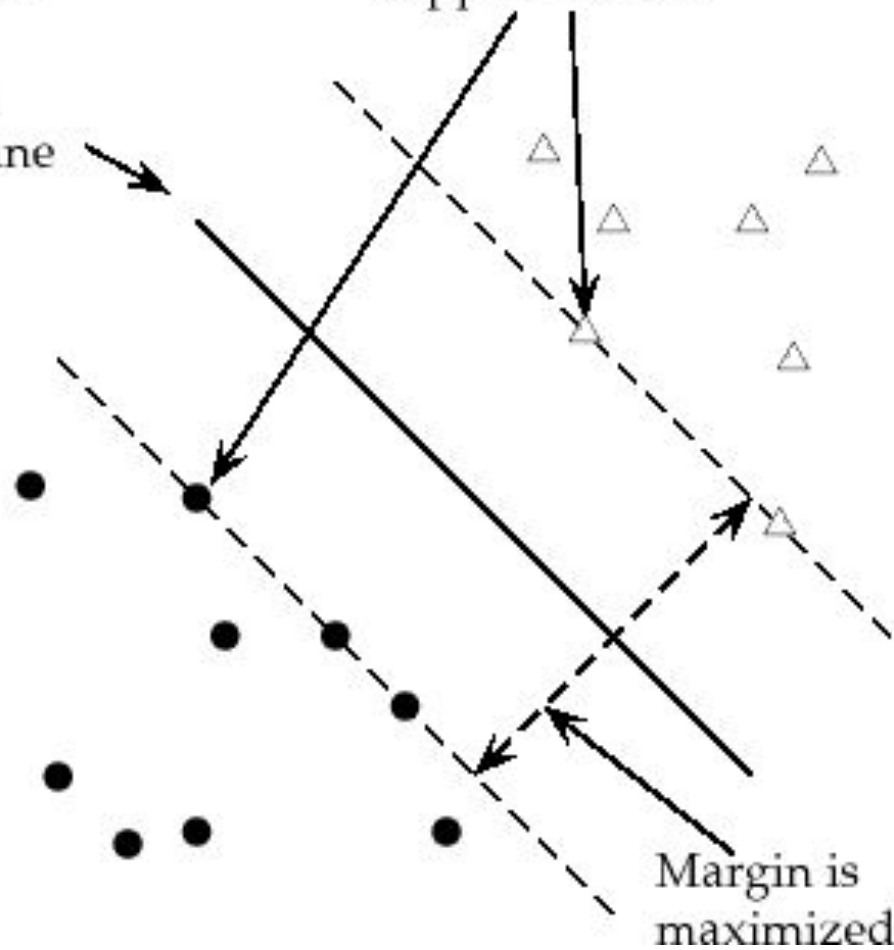
Margin is the perpendicular distance between the closest data points and the Hyperplane ( on both sides )

The best optimised line ( hyperplane ) with maximum margin is termed as Margin Maximal Hyperplane.

The closest points where the margin distance is calculated are considered as the support vectors.

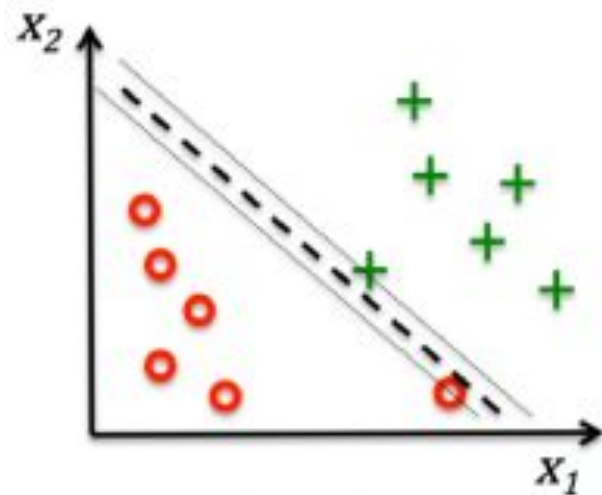
Maximum  
margin  
decision  
hyperplane

Support vectors

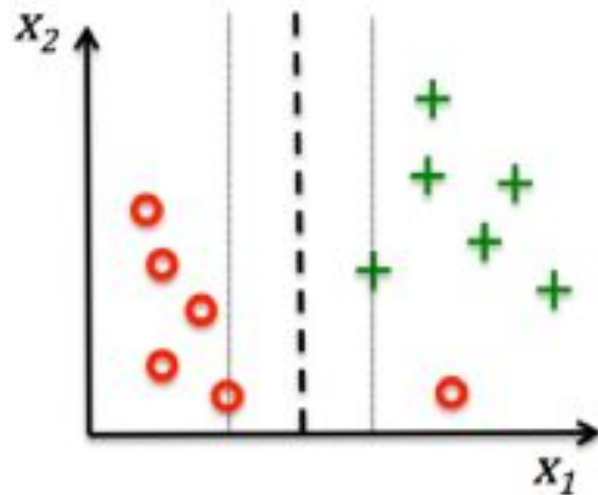


# Regularization

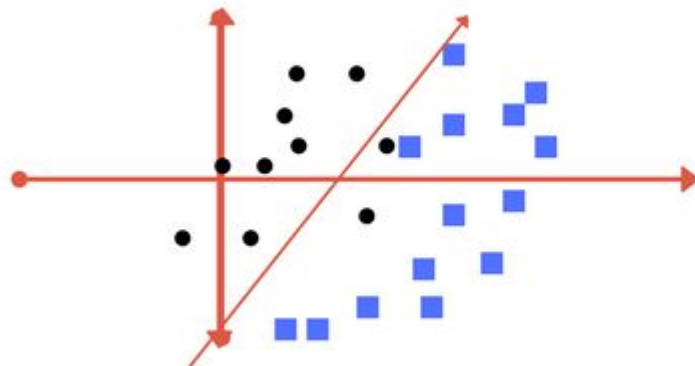
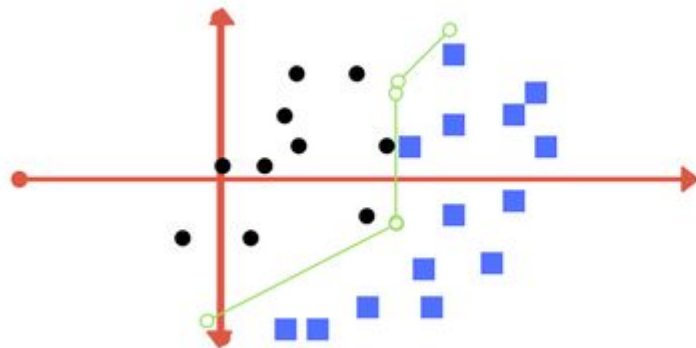
- Also the ' C ' parameter in Python's SkLearn Library
  - Optimises SVM classifier to avoid misclassifying the data.
  - $C \rightarrow \text{large}$                       Margin of hyperplane  $\rightarrow$  small
  - $C \rightarrow \text{small}$                       Margin of hyperplane  $\rightarrow$  large
    - misclassification(possible)
1.  $C \rightarrow \text{large}$  , chance of overfit
  2.  $C \rightarrow \text{small}$  , chance of underfitting



Large value for  
parameter  $C$



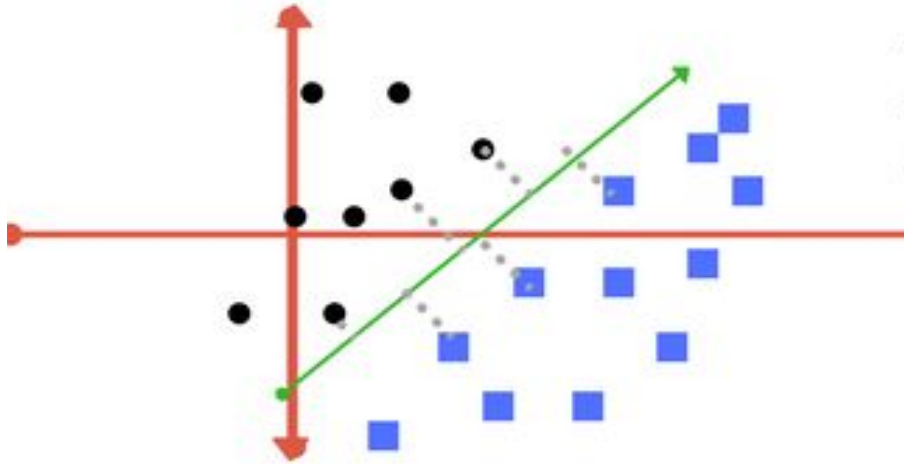
Small value for  
parameter  $C$



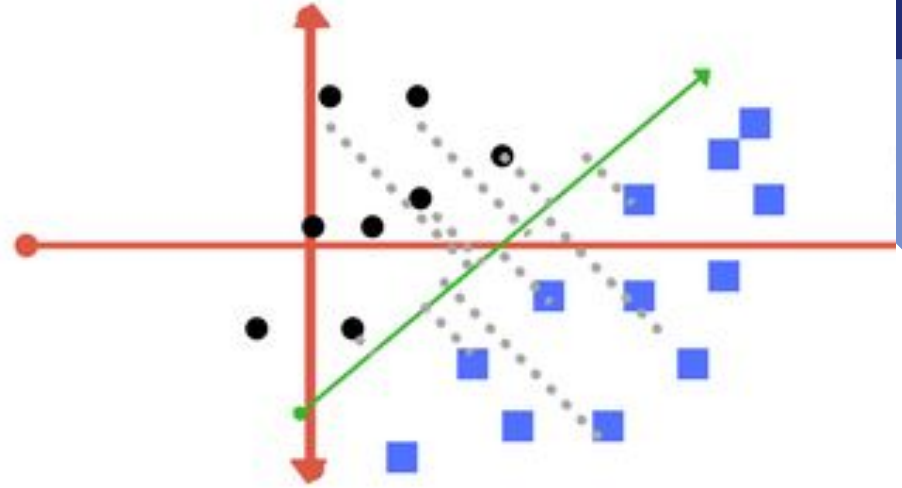
# Gamma

- Defines how far influences the calculation of of plausible line of separation.
- Low gamma ----> points far from plausible line are considered for calculation
- High gamma ----> points close to plausible line are considered for calculation





High Gamma Value



Low Gamma Value

# Kernels

- Mathematical functions for transforming data
- using some linear algebra
- Different SVM algorithms use different types of kernel functions

# Various kernels available

1. Linear kernel
2. Non - linear kernel
3. Radial basis function ( RBF )
4. Sigmoid
5. Polynomial
6. Exponential

# Example :

$K(x, y)$  =



Kernel function

$\langle f(x), f(y) \rangle$



dot product of n- dimensional inputs

# Mathematical representation

$$x = (x_1, x_2, x_3); y = (y_1, y_2, y_3)$$

$$f(x) = (x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3)$$

$$f(y) = (y_1y_1, y_1y_2, y_1y_3, y_2y_1, y_2y_2, y_2y_3, y_3y_1, y_3y_2, y_3y_3)$$

$$K(x, y) = (\langle x, y \rangle)^2$$

$$x = (1, 2, 3)$$

$$y = (4, 5, 6)$$

$$f(x) = (1, 2, 3, 2, 4, 6, 3, 6, 9)$$

$$f(y) = (16, 20, 24, 20, 25, 30, 24, 30, 36)$$

$$\langle f(x), f(y) \rangle = 16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324 = 1024$$

$$K(x, y) = (4 + 10 + 18)^2 = 1024 \text{ ----> Kernel function}$$

## Pros :

- It works really well with clear margin of separation
- It is effective in high dimensional spaces.
- It is effective in cases where number of dimensions is greater than the number of samples.
- It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

## Cons :

- It doesn't perform well, when we have large data set because the required training time is higher
- It also doesn't perform very well, when the data set has more noise i.e. target classes are overlapping
- SVM doesn't directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is related SVC method of Python scikit-learn library.



# Applications :

1. Face detection
2. Text and hypertext categorization
3. Classification of images
4. Bioinformatics
5. Handwriting recognition
6. Protein fold and remote homology detection
7. Generalized predictive control(GPC)



# Let's code now

Data used : Iris from Sklearn

Plots : Matplotlib

Kernels : Linear and rbf

File : svm\_final.py

Link to code : [Click here for code](#)

***Thank You***