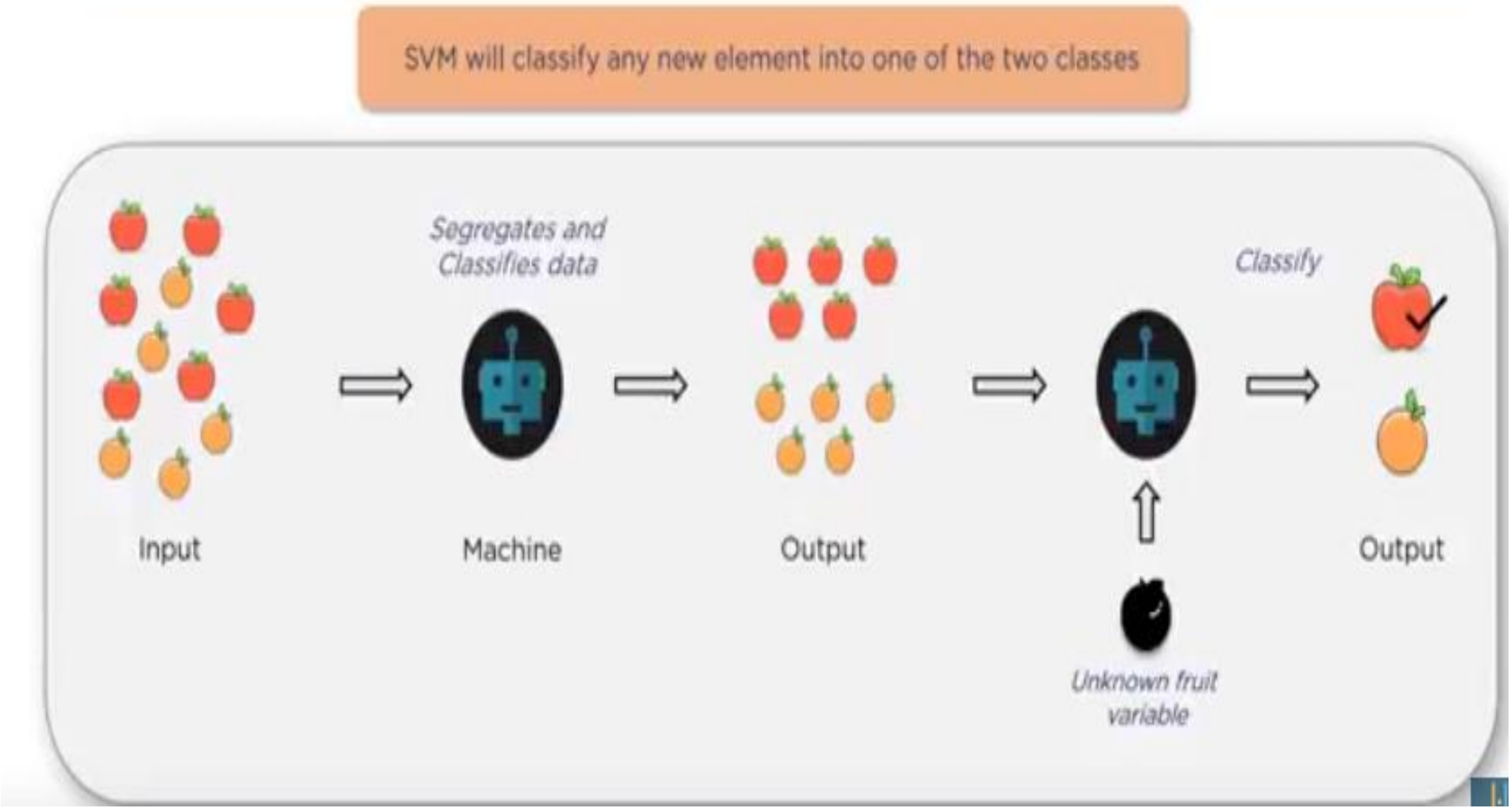
Unit2
Support Vector Machines

What is SVM?

- SVM is a type of classification algorithm which classifies data based on its features
- “Support Vector Machine” (SVM) is a supervised machine learning algorithm
- It is used for both classification or regression challenges
- SVM is a binary classifier
- If lot of features are there, then SVM will serve best

What is SVM?



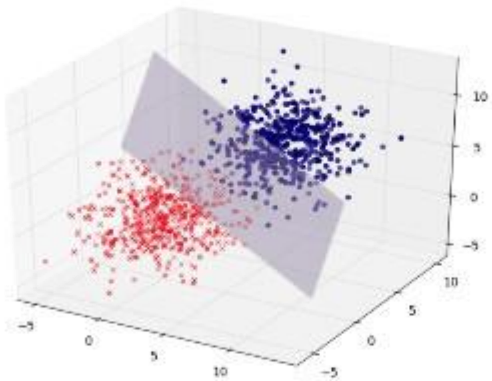
SVM will classify any new element into one of the two classes
Segregates and Classifies data
Classify
Input
Machine
Output
Output
Unknown fruit variable

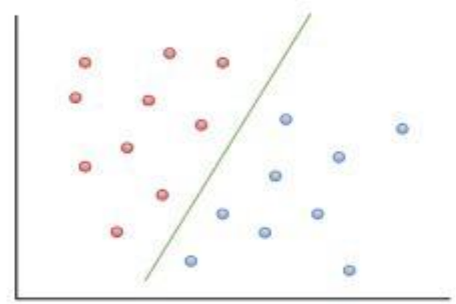
$$\mathbf{w}^T\mathbf{x} = 0$$

$$y = ax + b$$

Hyperplane

Line





Motivation

- ***Maximize margin***: we want to find the classifier whose decision boundary is furthest away from any data point.
- *Margin is the distance between the left hyperplane and right hyperplane.*

How does SVM work?

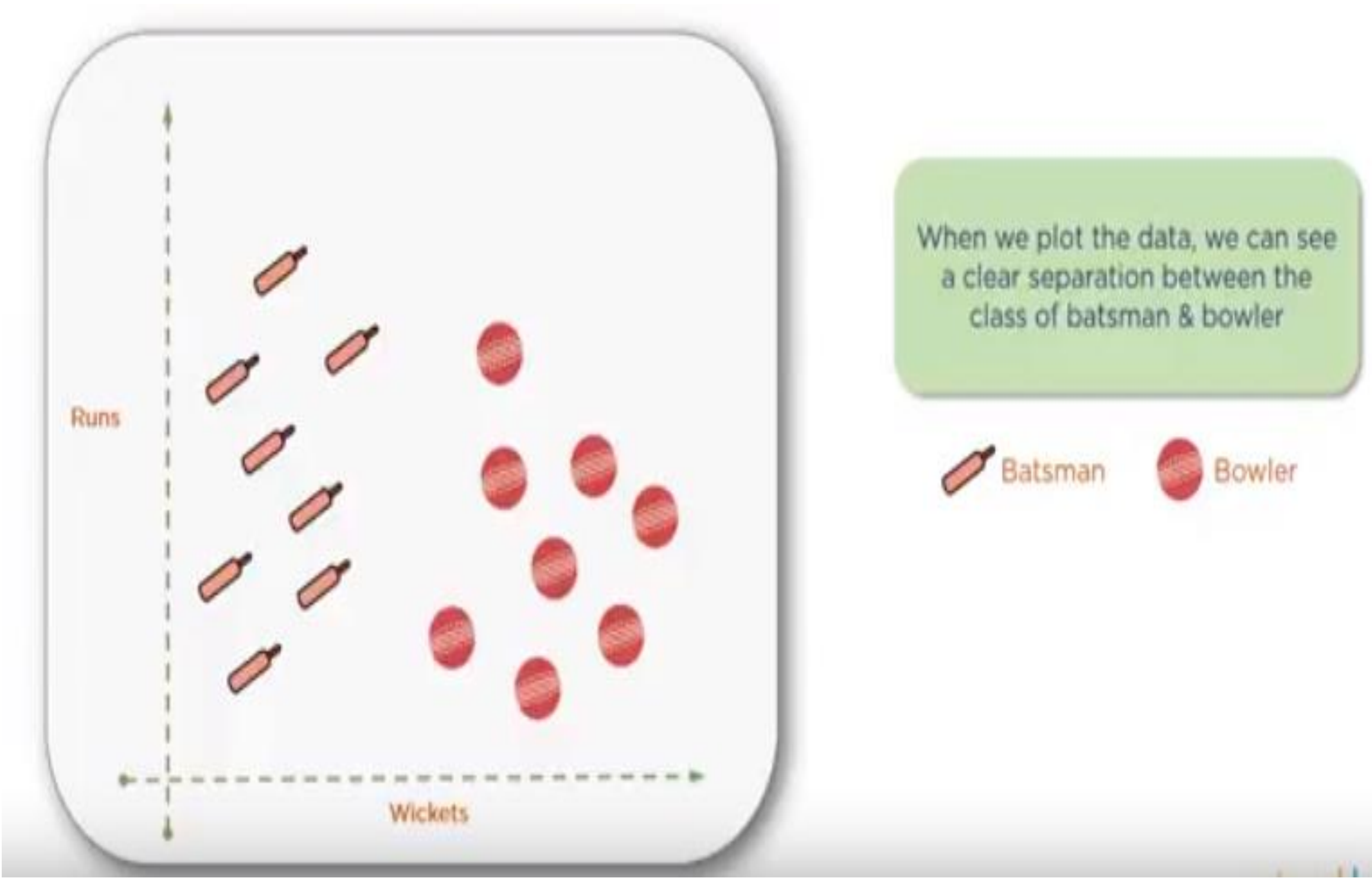
- Plot each data item as a point in n-dimensional space with the value of each feature being the value of a particular coordinate
- Perform classification by finding the hyper-plane that differentiate the two classes very well

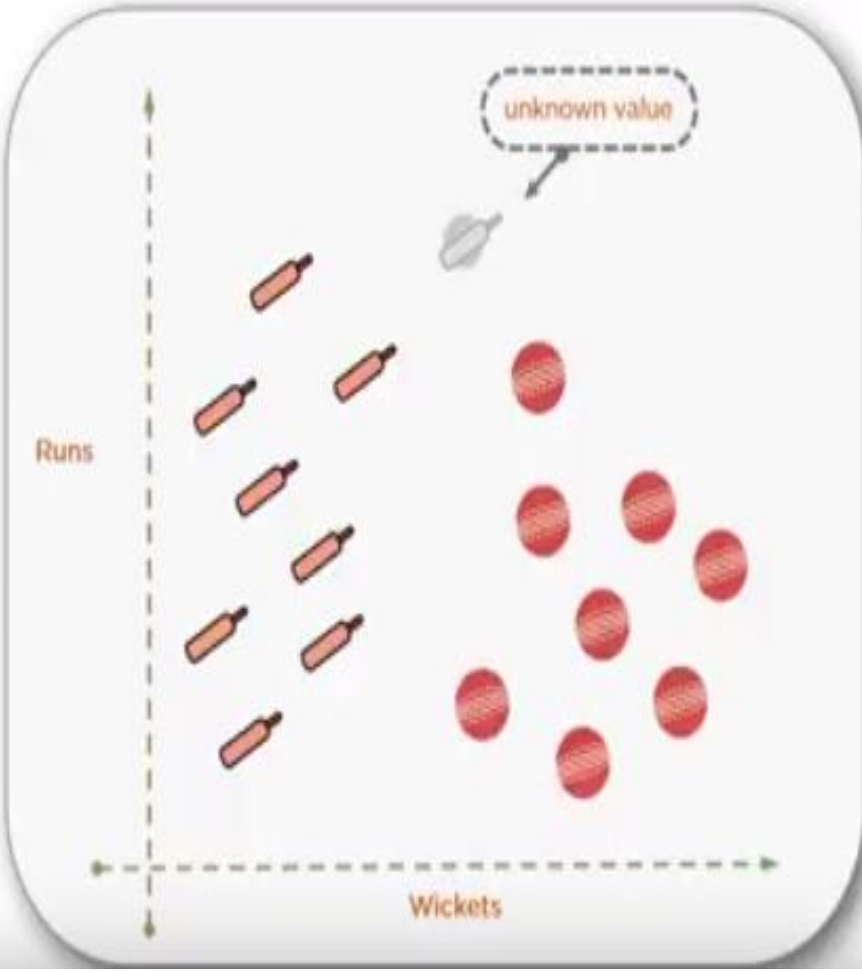
Here is an example…

Let's understand SVM with an example. We will classify cricket players into batsmen and bowlers using the runs to wicket ratio.

A player with more runs is a batsman

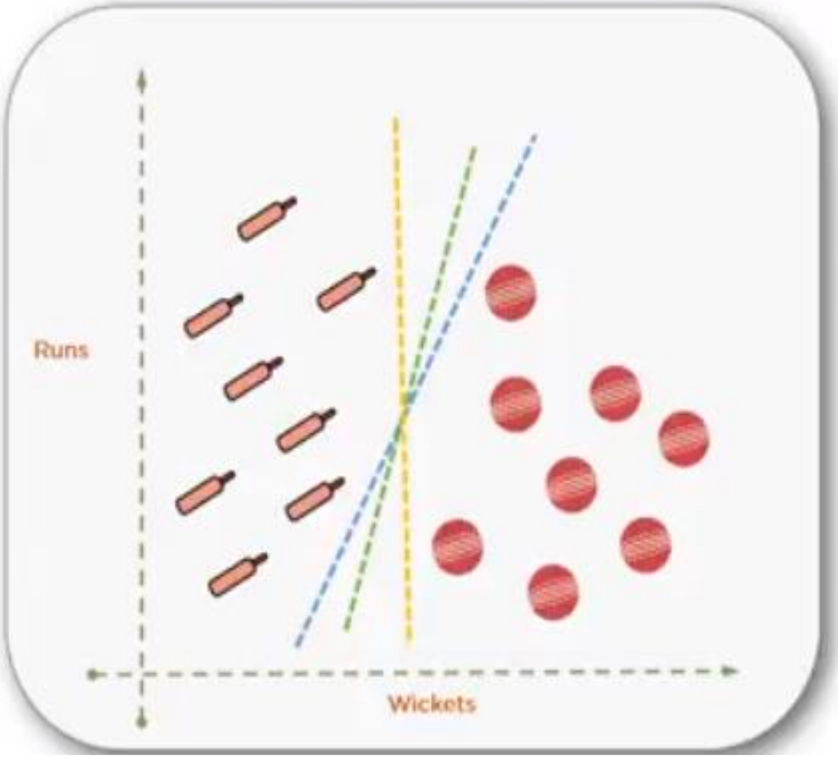
A player with more wickets is a bowler


Runs
Wickets
When we plot the data, we can see a clear separation between the class of batsman & bowler
Batsman
Bowler



Now, we want to classify a new player variable as a batsman or a bowler

A **decision boundary** is required in order to classify the new unknown variable



The decision boundary is a separation between the 2 classes

We can draw multiple lines here as decision boundaries

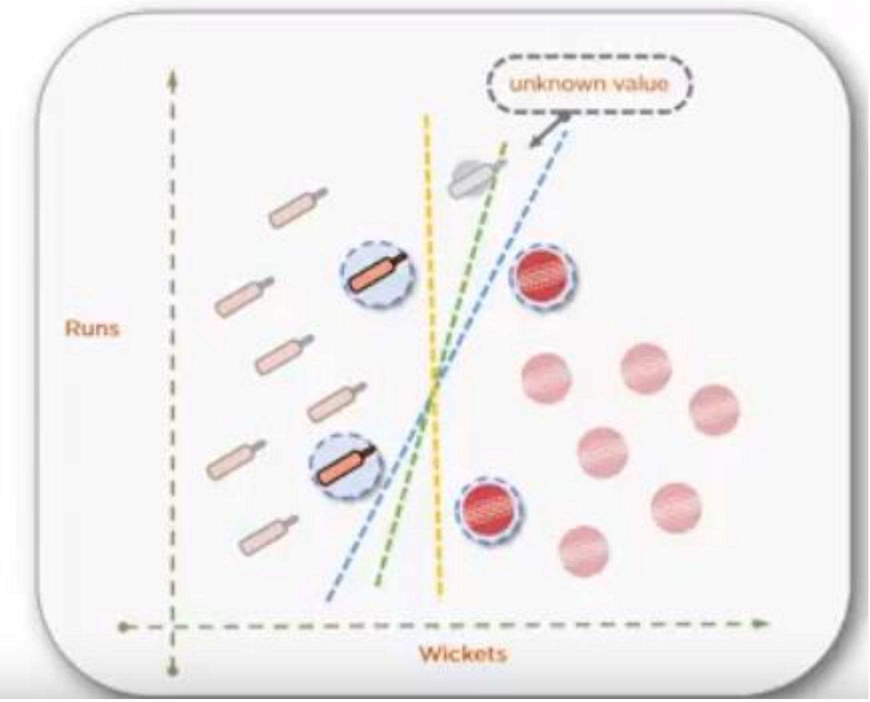
Linear regression – Line of best fit
SVM- Line of best separator

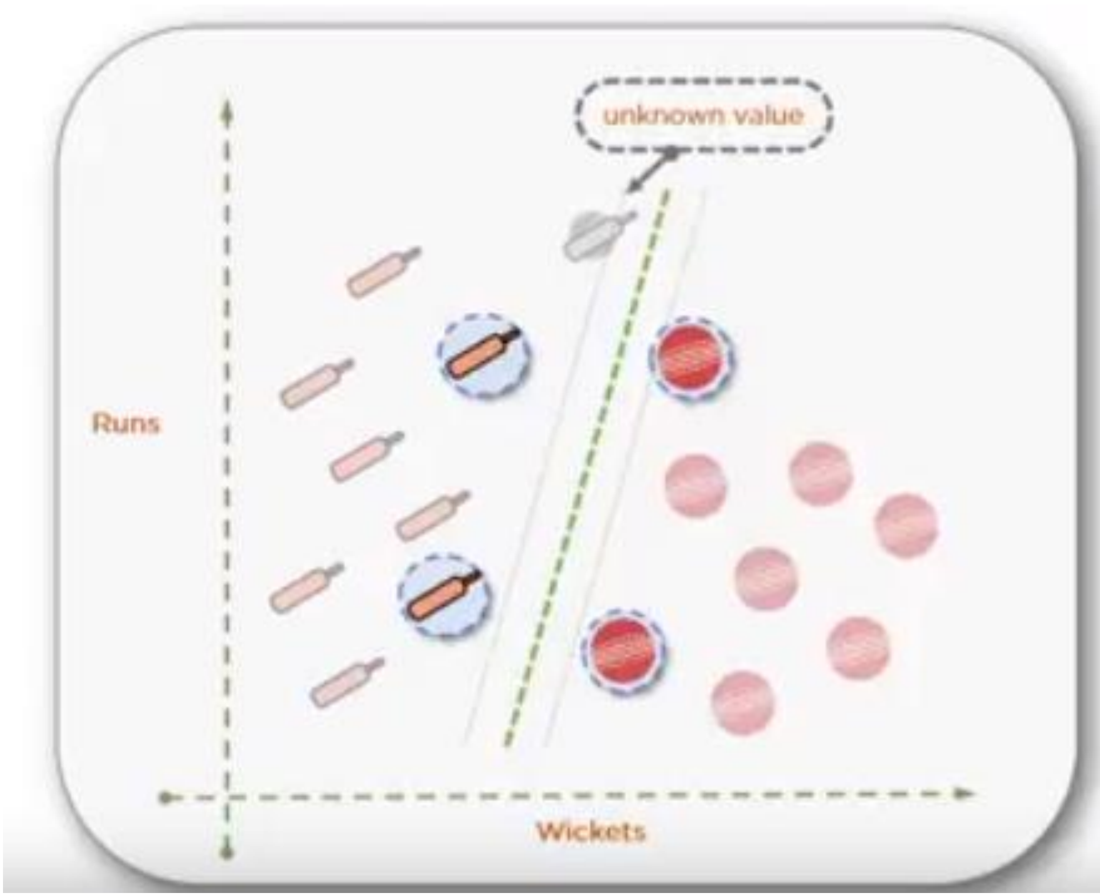
The best line is selected by **computing** the **maximum margin** from equidistant **Support Vectors**

But,
what exactly are **Support Vectors** here?

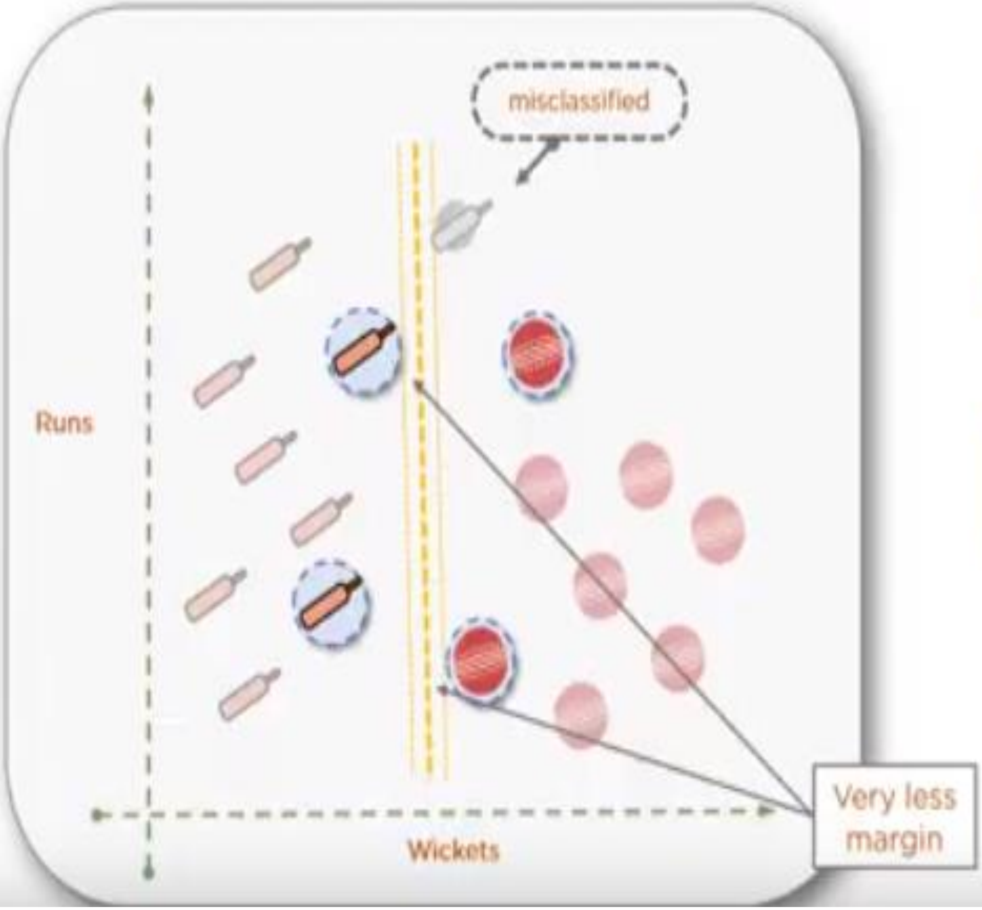
What are support vectors?

- Support vectors are the points which are very close to the dividing line
- Using the support vectors we can select the best line to divide the data




unknown value
Runs
Wickets

- Separator is called hyper plane
- Hyper plane has the maximum distance to the support vectors of any class
- D+ is the shortest distance to the closest positive point
- D- is the shortest distance to the closest positive point
- Distance margin – sum of D+ and D-



If the margin between the support vectors is not maximum, then data can get misclassified

Example, The player here is **misclassified** as a bowler

- This problem set is 2-dimensional because classification is only between 2 classes
- 2 dimensional applications of SVM are called linear SVM

- What is the equation of the separator?

$$y = \beta_0 + x^T \beta \triangleq 0$$

- If

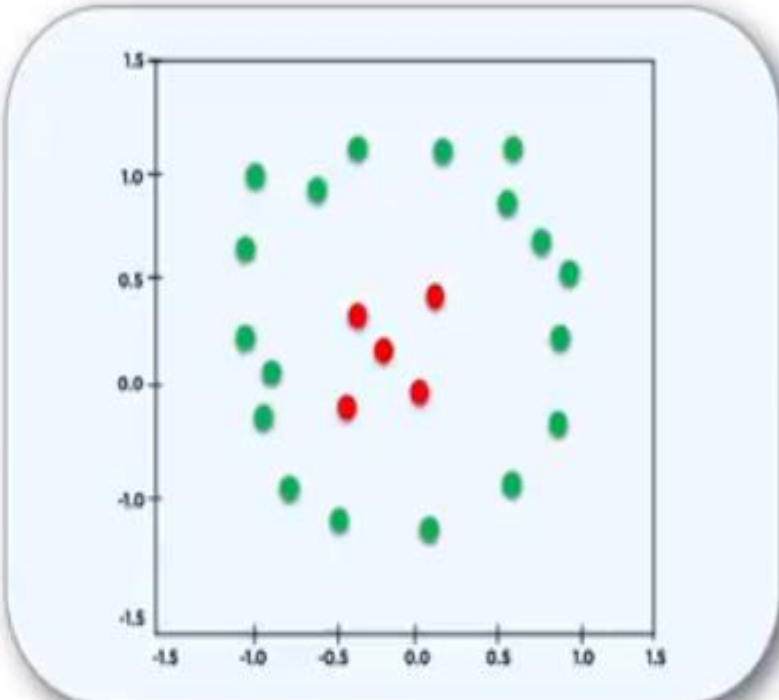
$$\beta_0 + \beta^T x < 0 \Rightarrow \text{class } `-1'$$

$$\beta_0 + \beta^T x > 0 \Rightarrow \text{class } +1$$

- With a Constraint of having Maximum Margin

$$y_i (x_i^T \beta + \beta_0) \geq M, \quad i = 1, \ldots$$

Linearly non-separable????



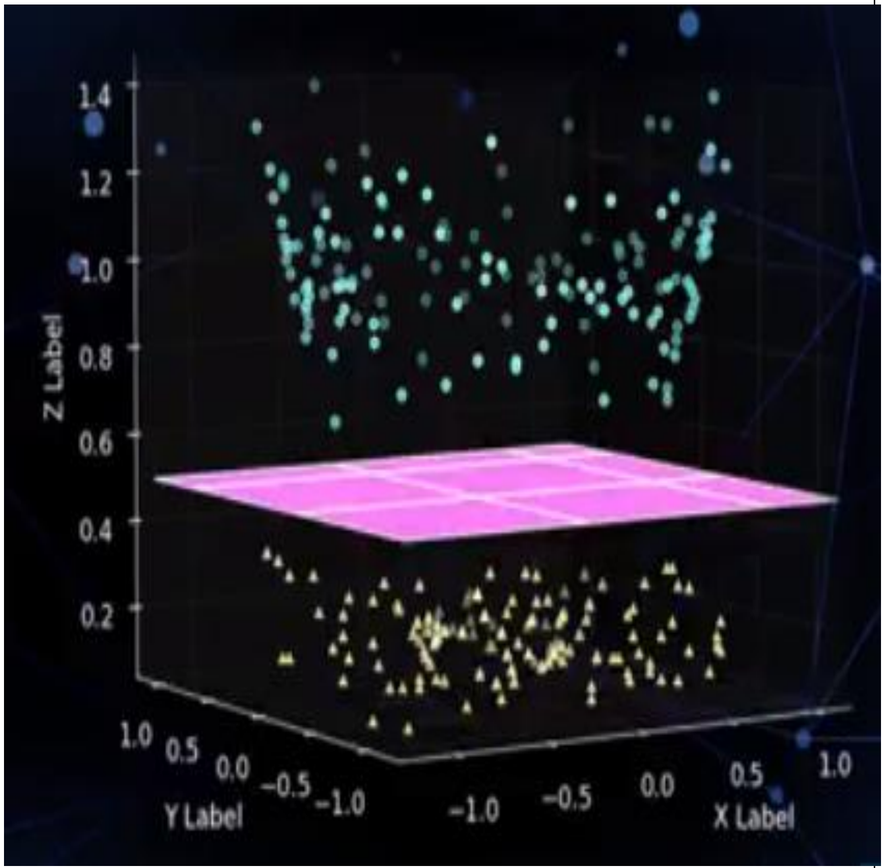
Sample 2D dataset

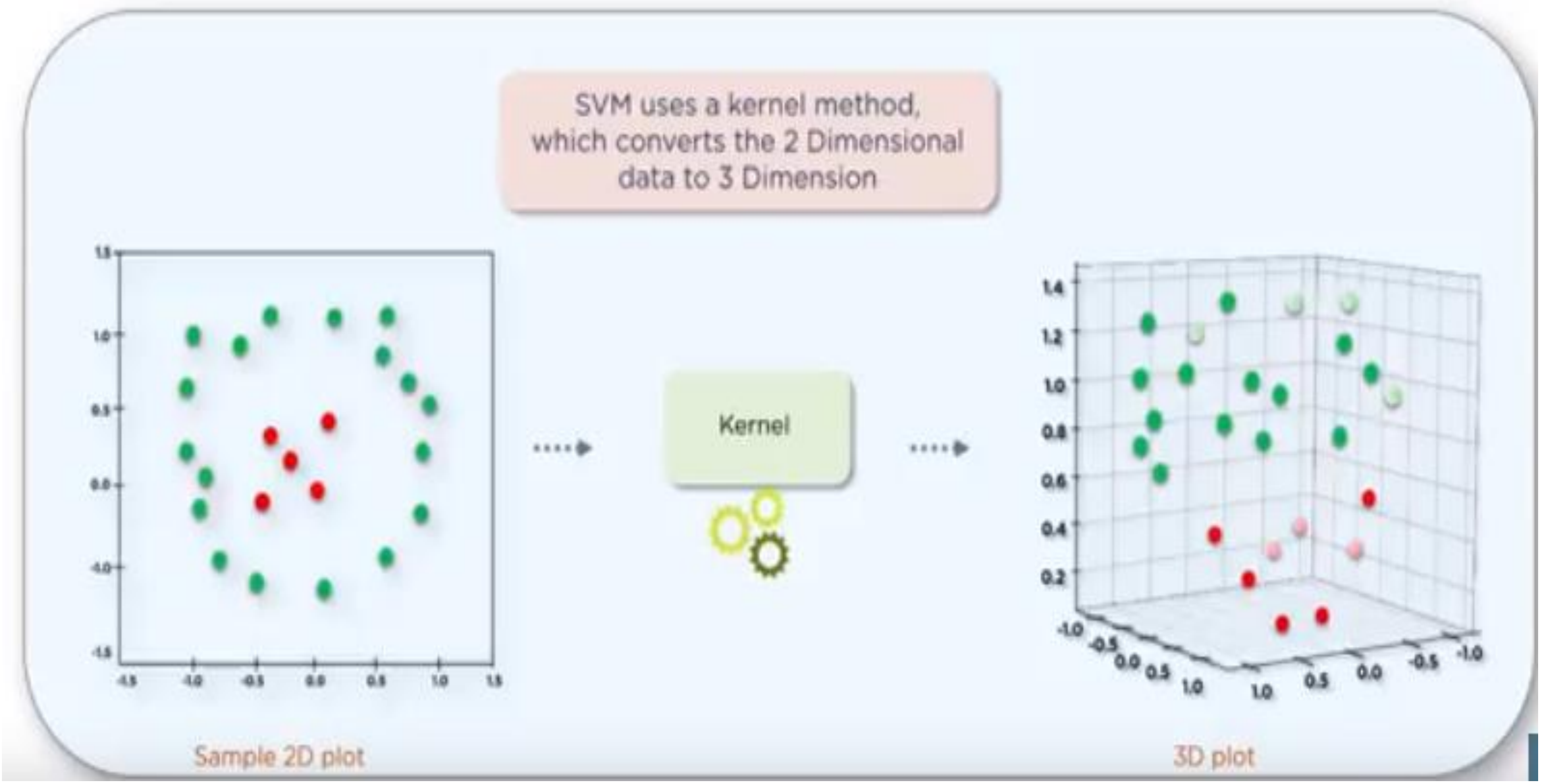
What if our 2 dimensional data looks like the given graph?

How will SVM work on such data?

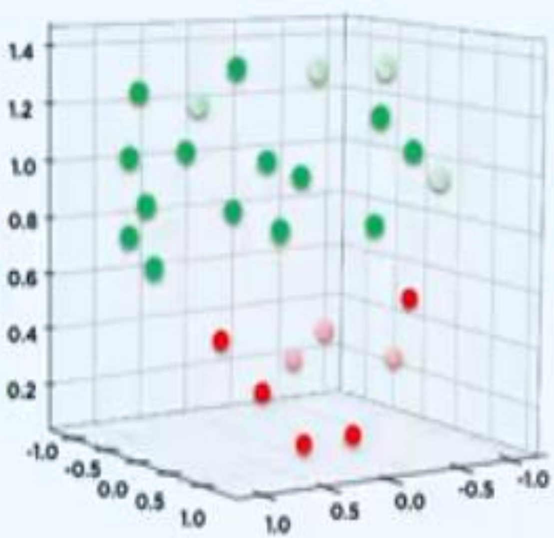
Kernels

- Process of making non-linearly separable data point to linearly separable data point is also known as **Kernel Trick**
- Most of the times, raw data are non-linearly separable. Then Kernel trick is applied to make it linearly separable
- Different kernels: Polynomial, radial, sigmoid, guassian etc…



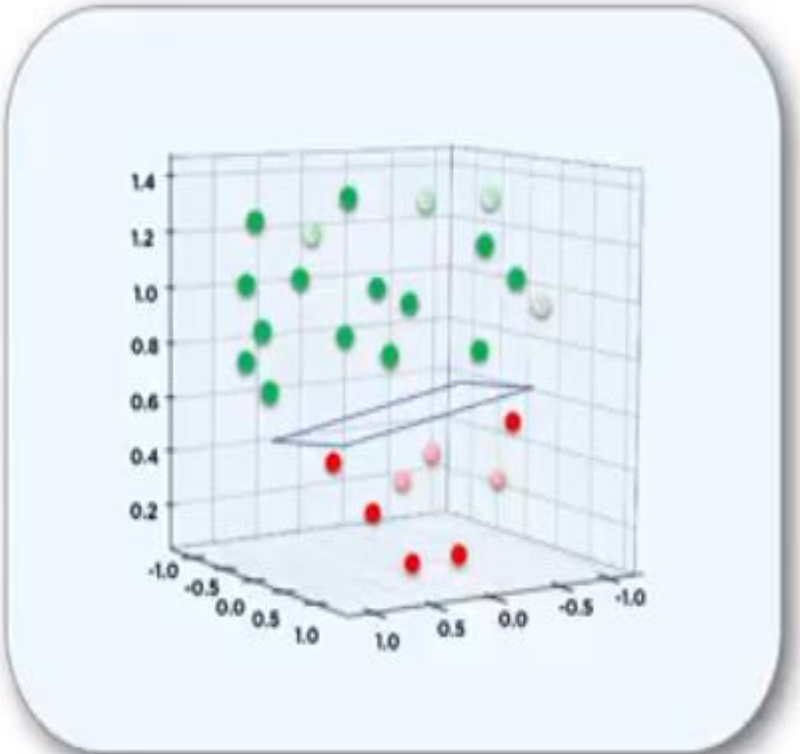


SVM uses a kernel method, which converts the 2 Dimensional data to 3 Dimension
Kernel
Sample 2D plot
3D plot



Let R be the number of dimensions of this data

The kernel converts a given R^2 dimension to R^3 dimension



Once the data is in 3 Dimensions, SVM separates the data in the graph using a 2D plane

Kernel functions

- Sigmoid $k(x, y) = \tanh(\alpha x^T y + c)$
- Polynomial $k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j} + 1)^d$
- Gaussian $k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2)$
- RBF

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

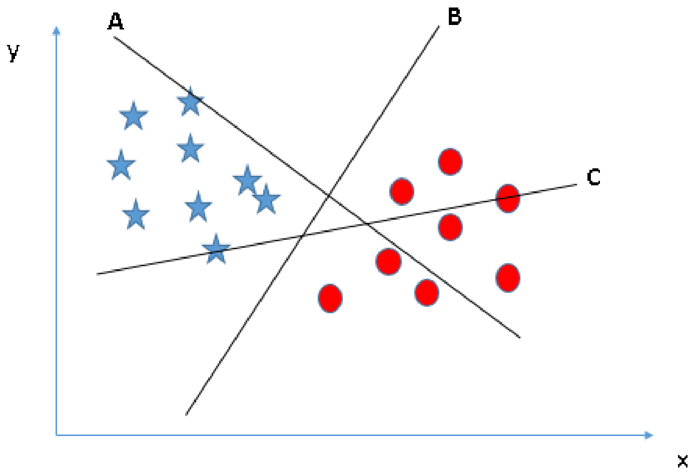
Applications of SVM

- Face detection
- Text categorization
- Image classification
- bioinformatics

How can we 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.

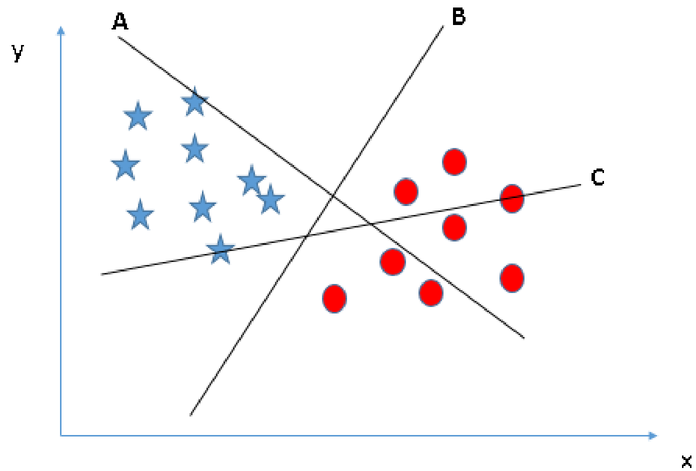


You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.

How can we identify the right hyper-plane?

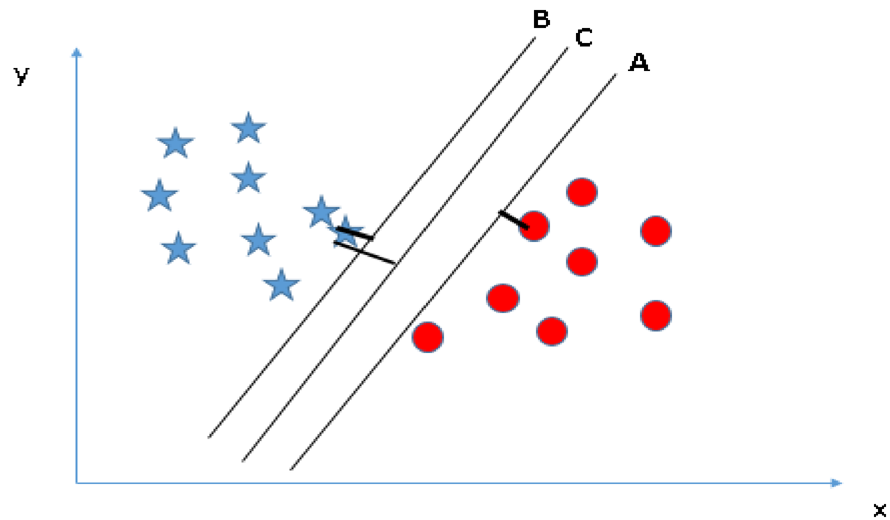
- Scenario-2

Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now. How can we identifv the right hyper-plane?



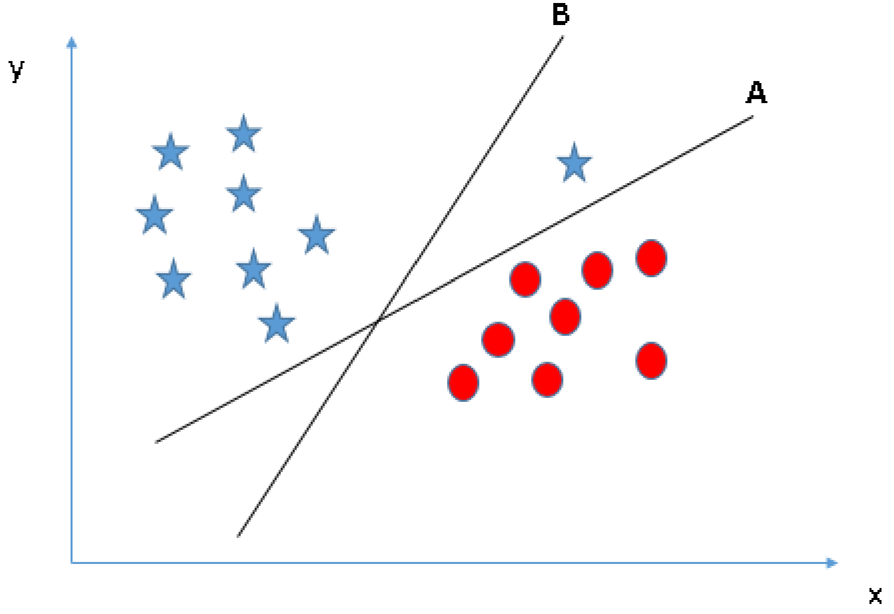
Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**.

How can we identify the right hyper-plane? Scenario-2 contd..



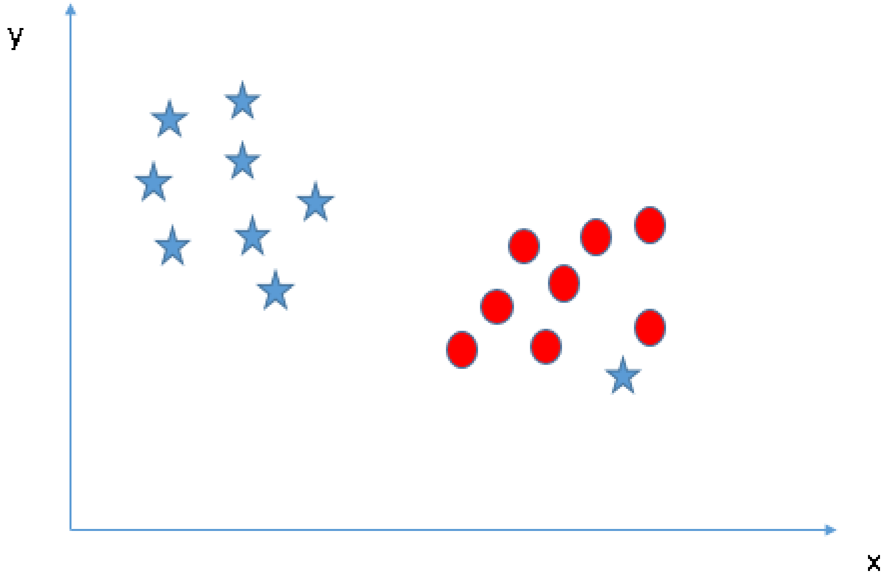
Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

- Scenario-3
  - Hint: Use the rules as discussed in previous section to identify the right hyper-plane



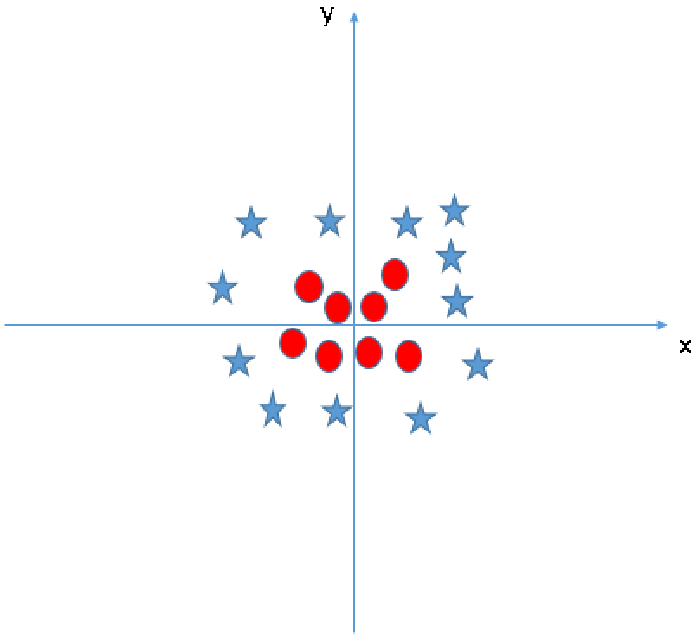
- Scenario-3
  - Hint: Use the rules as discussed in previous section to identify the right hyper-plane
  - Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A.** But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A.**

- 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.

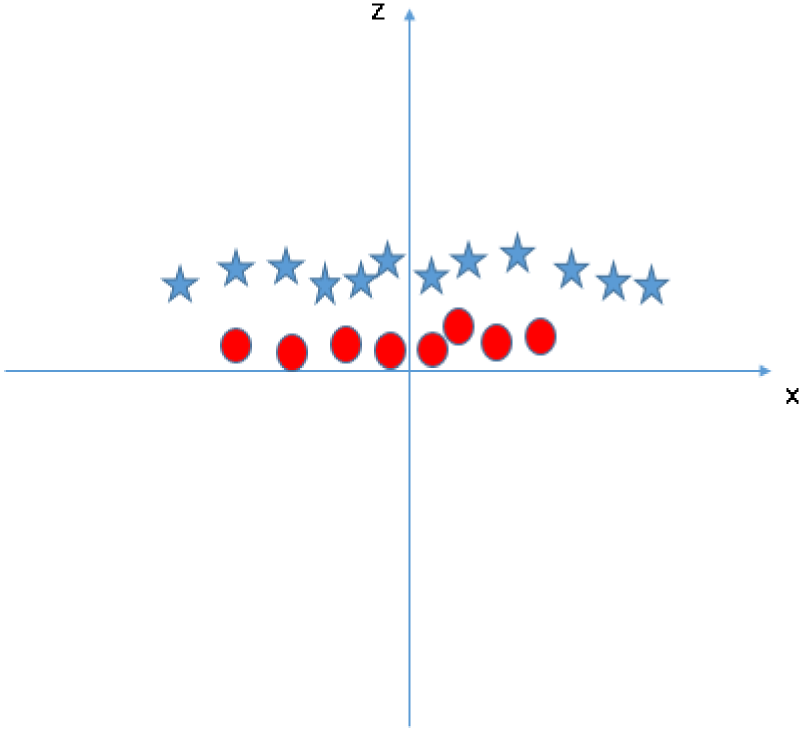


- one star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, we can say, SVM is robust to outliers.

- 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.



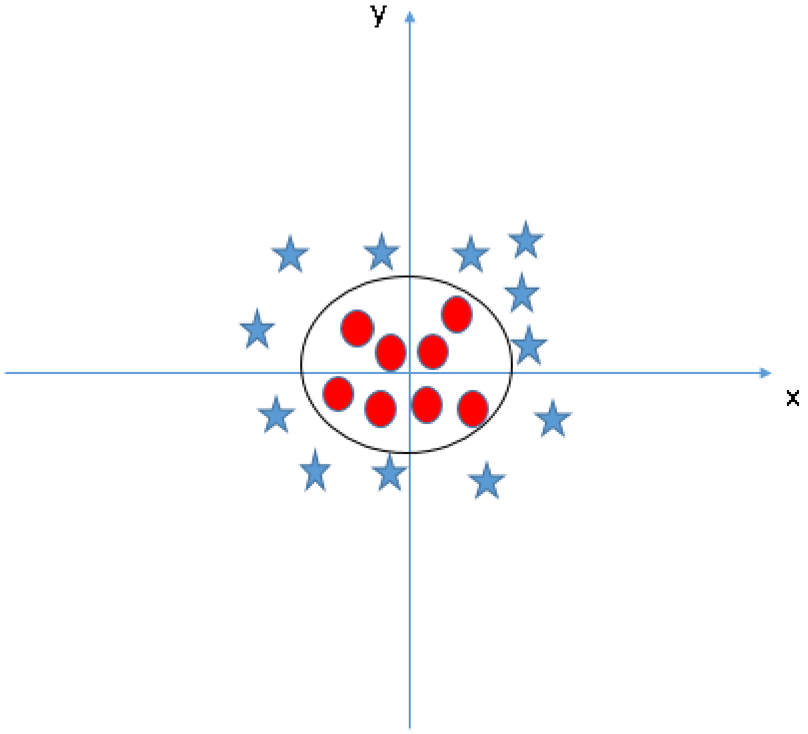
- SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:



- In above plot, points to consider are:
  - 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.
- In SVM, it is easy to have a linear hyper-plane between these two classes.
- should we need to add this feature manually to have a hyper-plane????
- No, 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
- it converts not separable problem to separable problem, these functions are called kernels.
- It is mostly useful in non-linear separation problem.
- 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:



Hard margin & Soft margin

- The hard **margin** is a one which clearly separate positive and negative points.
- **Soft margin** is also called as noisy linear **SVM** which includes some miss-classified points.
- Solution to the **soft margin** is approximation of points which are miss-classified in linear decision boundary.

Pros and Cons

- **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.

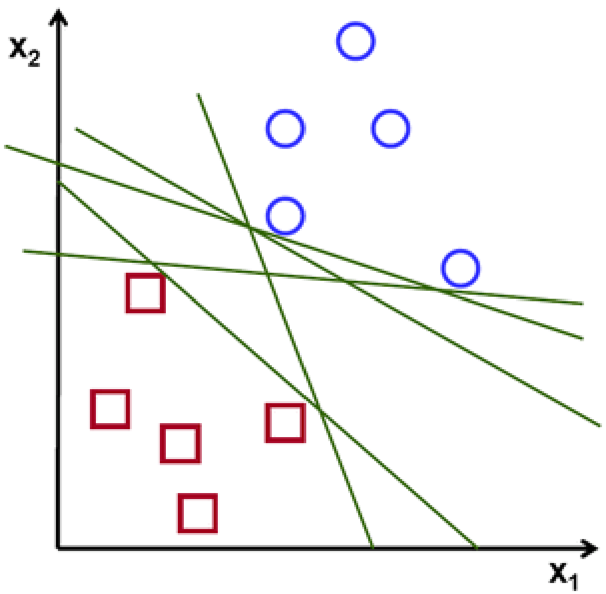
Pros and Cons

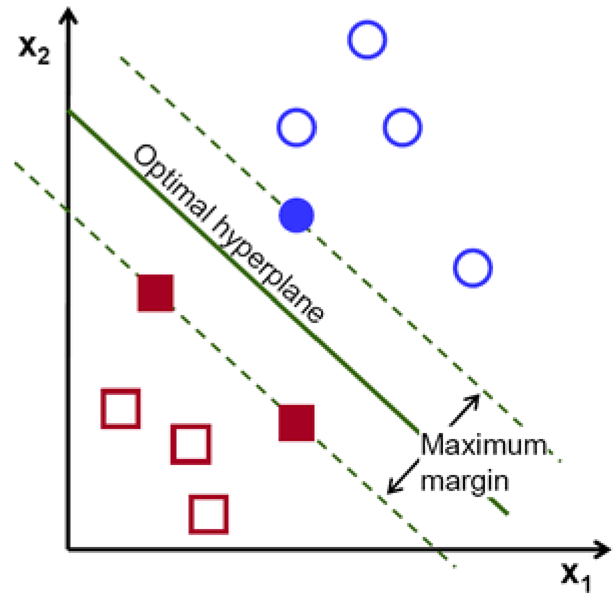
- **Cons:**
  - It doesn't perform well, when we have large data set because the required training time is higher
  - It also doesn't perform very well, when the data set has more noise i.e. target classes are overlapping
  - SVM doesn't directly provide probability estimates, these are calculated using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

Math behind SVM

SVM Objective

- The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data



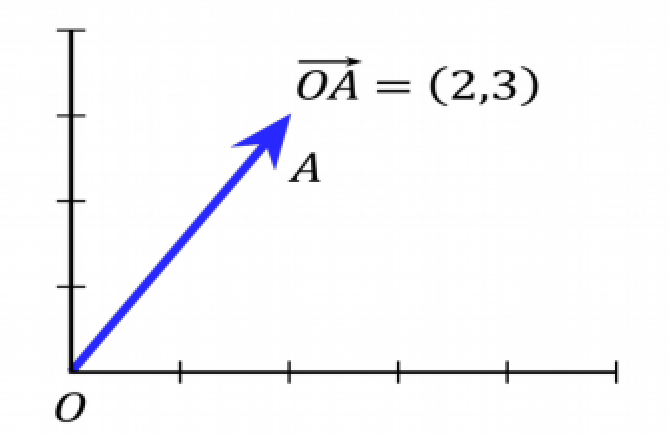


Vectors

- Vectors are mathematical quantity which has both magnitude and direction. A point in the 2D plane can be represented as a vector between origin and the point.

Fig.-1

$\vec{OA}$ is a vector and length between O and A is its magnitude.



Length of Vectors

- Length of vectors are also called as norms. It tells how far vectors are from the origin.

Length of vector $x(x_1, x_2, x_3)$ *is calculated as :*

$$||x|| = \sqrt{x_1^2 + x_2^2 + x_3^2}$$

Direction of Vector

Direction of vector $x(x_1, x_2, x_3)$ *is calculated as:*

$$\left\{ \frac{x_1}{\|x\|}, \frac{x_2}{\|x\|}, \frac{x_3}{\|x\|} \right\}$$

Dot Product

- Dot product between two vectors is a scalar quantity . It tells how to vectors are related.

Two vectors u and v and their dot product is calculated as:

Lengh of vector **u**,**v**

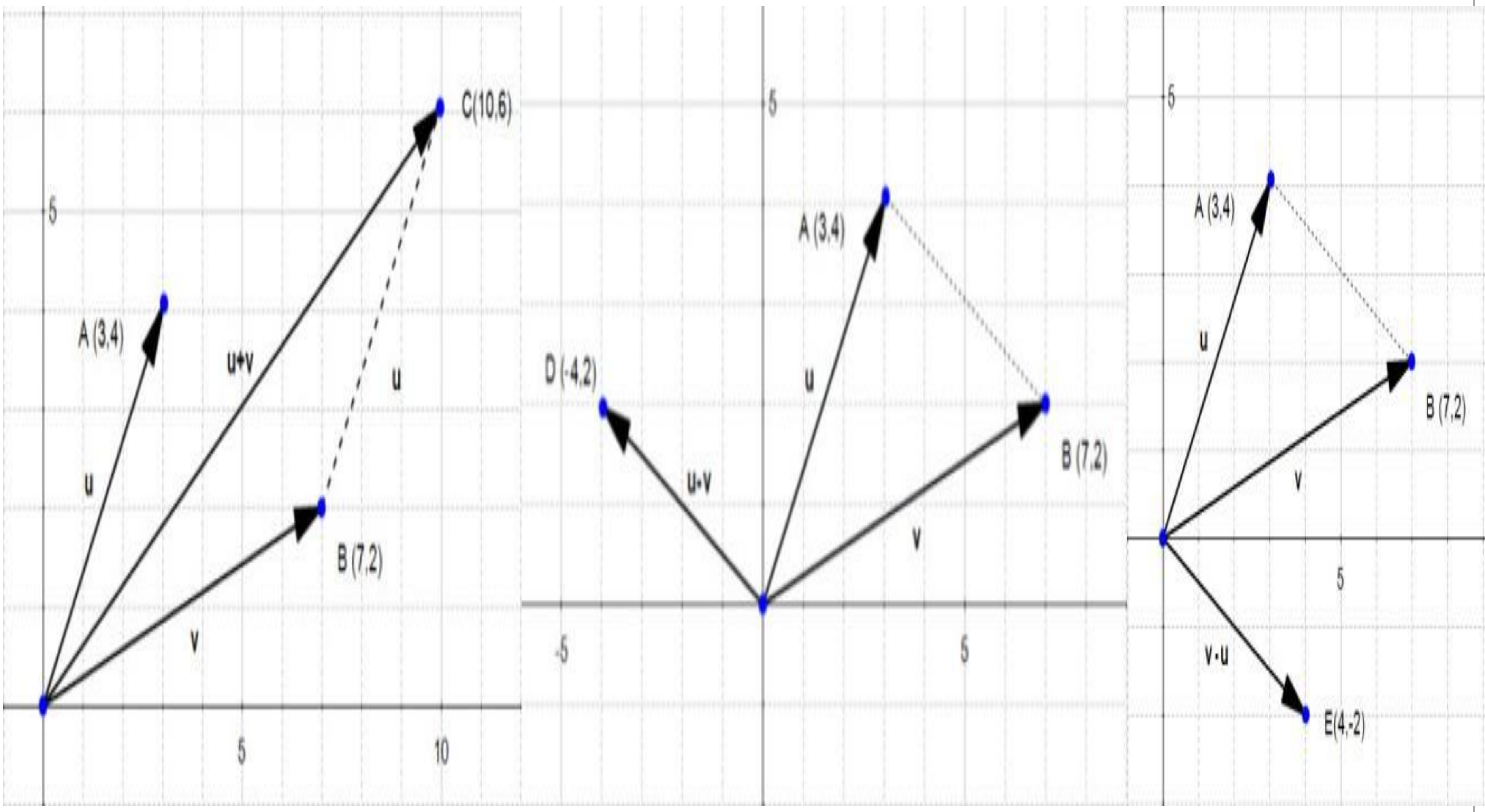
Symbol for inner product

Angle between **u** and **v**

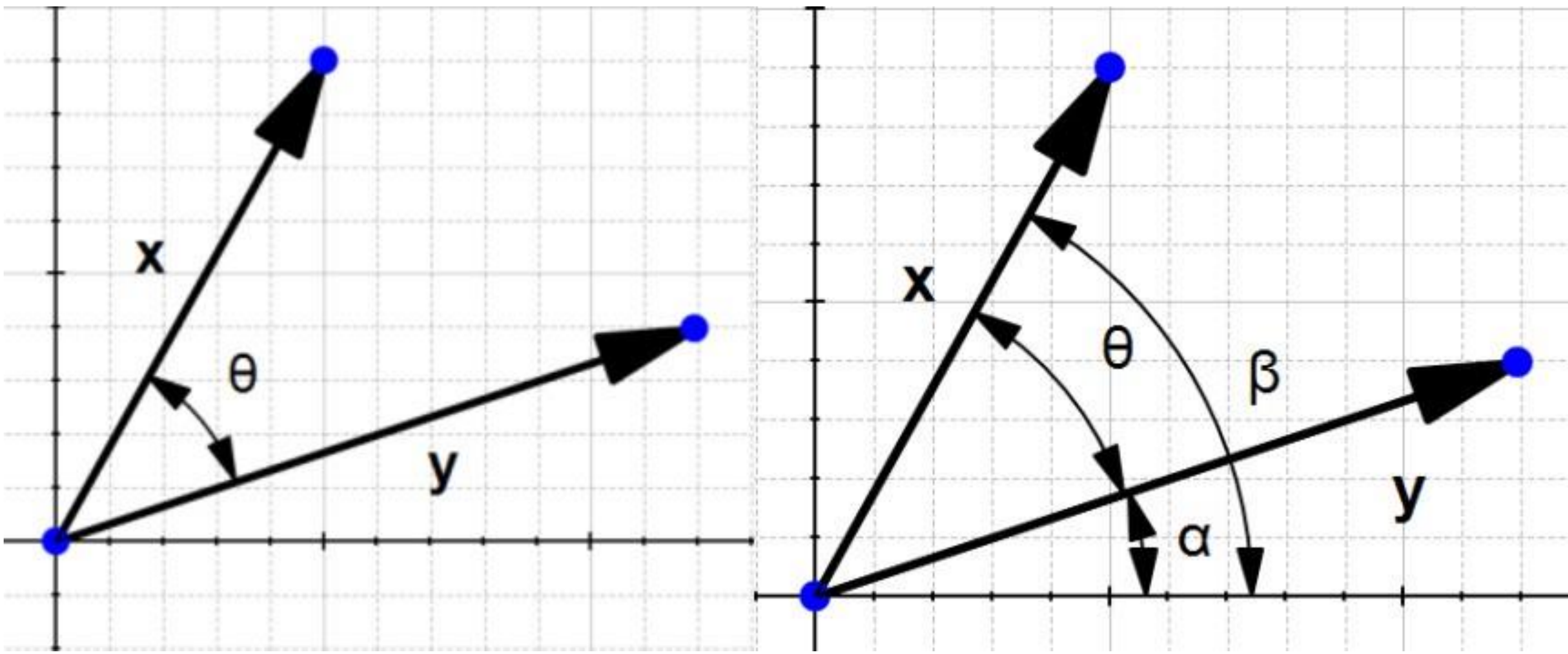
$$\mathbf{u} \bullet \mathbf{v} = |\mathbf{u}||\mathbf{v}|\cos(\theta) \tag{1}$$

$$= x_1 \times x_2 + y_1 \times y_2 \tag{2}$$

Addition & Subtraction of vectors



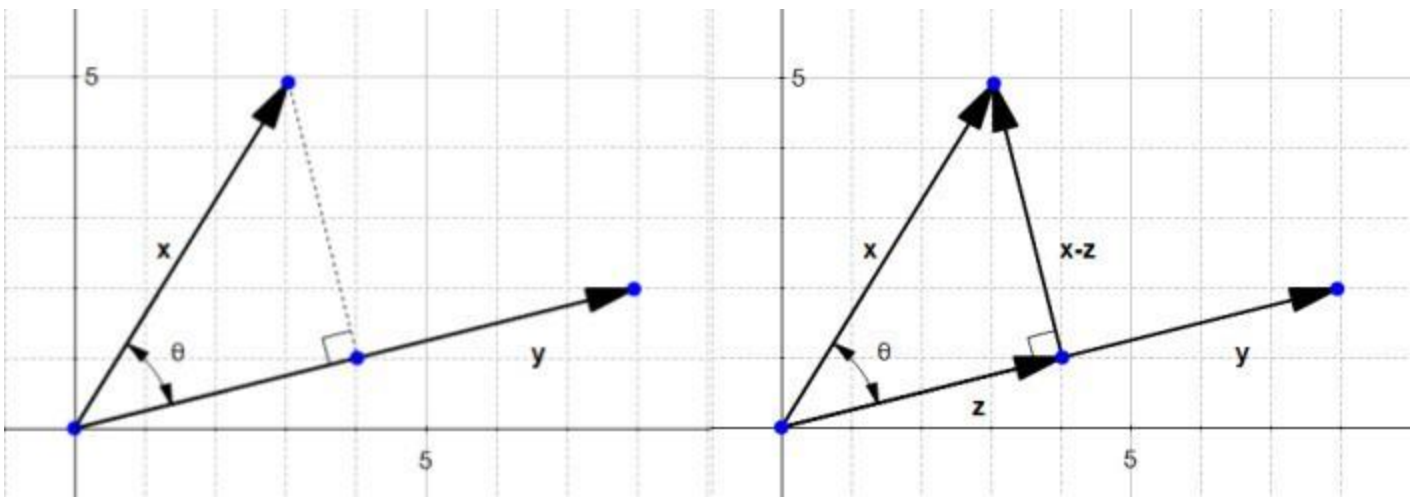
Dot product of vectors



$$\mathbf{x} \cdot \mathbf{y} = x_1 y_1 + x_2 y_2 = \sum_{i=1}^{2} (x_i y_i)$$

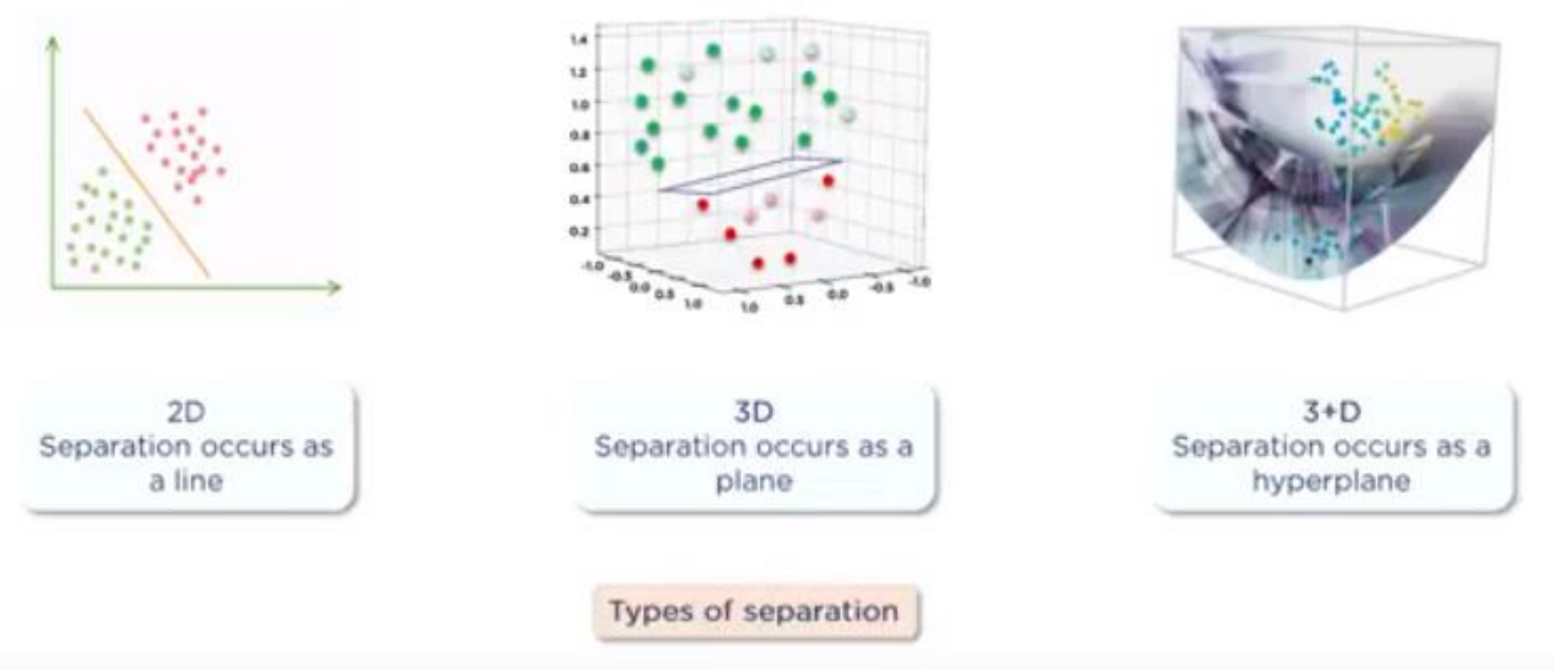
Orthogonal projection of vectors

- It allows us to compute the distance between **x** and the line which goes through **y** (x-z).

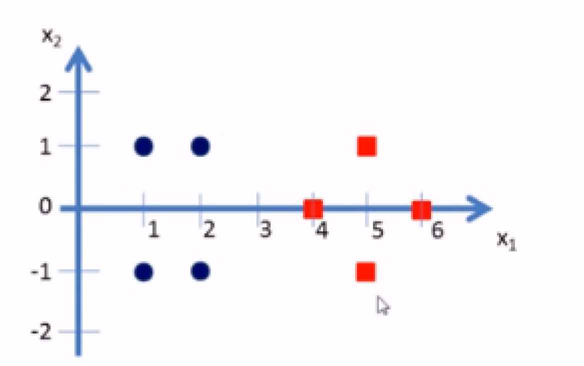


Hyper-plane

- It is plane that linearly divide the n-dimensional data points in two component. In case of 1D it is a point, In case of 2D, hyperplane is line, in case of 3D it is plane. It is also called as *n-dimensional line.*

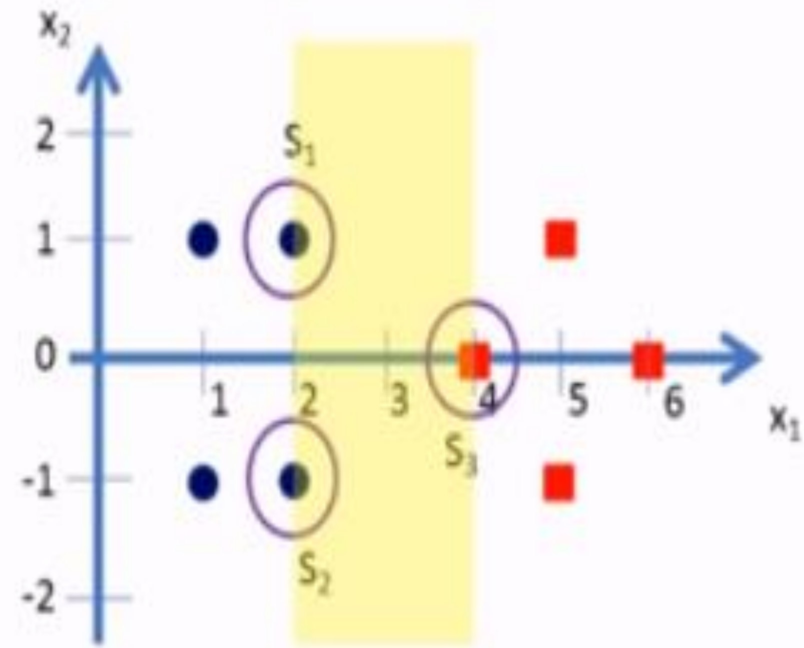


Let’s work a problem



X1	X2	class
1	1	Blue
1	-1	Blue
2	1	Blue
2	-1	Blue
4	0	Red
5	1	Red
5	-1	Red
6	0	Red

- Here we select 3 Support Vectors to start with.
- They are S_1, S_2 and S_3.



$$S_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

$$S_2 = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$

$$S_3 = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$$

- Here we will use vectors augmented with a 1 as a bias input, and for clarity we will differentiate these with an over-tilde. That is:

$$S_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \qquad \widetilde{S_1} = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$$

$$S_2 = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \qquad \widetilde{S_2} = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}$$

$$S_3 = \begin{pmatrix} 4 \\ 0 \end{pmatrix} \qquad \widetilde{S_3} = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

- Now we need to find 3 parameters α_1, α_2, and α_3 based on the following 3 linear equations:

$$\alpha_1 \widetilde{S_1} . \widetilde{S_1} + \alpha_2 \widetilde{S_2} . \widetilde{S_1} + \alpha_3 \widetilde{S_3} . \widetilde{S_1} = -1 \;\; (-ve\ class)$$

$$\alpha_1 \widetilde{S_1} . \widetilde{S_2} + \alpha_2 \widetilde{S_2} . \widetilde{S_2} + \alpha_3 \widetilde{S_3} . \widetilde{S_2} = -1 \; (-ve\ class)$$

$$\alpha_1 \widetilde{S_1} . \widetilde{S_3} + \alpha_2 \widetilde{S_2} . \widetilde{S_3} + \alpha_3 \widetilde{S_3} . \widetilde{S_3} = +1 \;\; (+ve\ class)$$

- Let's substitute the values for $\widetilde{S_1}$, $\widetilde{S_2}$ and $\widetilde{S_3}$ in the above equations.

$$\widetilde{S_1} = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \qquad \widetilde{S_2} = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \qquad \widetilde{S_3} = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} . \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} . \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} . \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} . \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} . \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} . \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} . \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} . \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} . \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} = +1$$

- After simplification we get:

$$6\alpha_1 + 4\alpha_2 + 9\alpha_3 = -1$$

$$4\alpha_1 + 6\alpha_2 + 9\alpha_3 = -1$$

$$9\alpha_1 + 9\alpha_2 + 17\alpha_3 = +1$$

- Simplifying the above 3 simultaneous equations we get: α_1=α_2= -3.25 and α_3= 3.5.

- The hyper plane that discriminates the positive class from the negative class is give by:

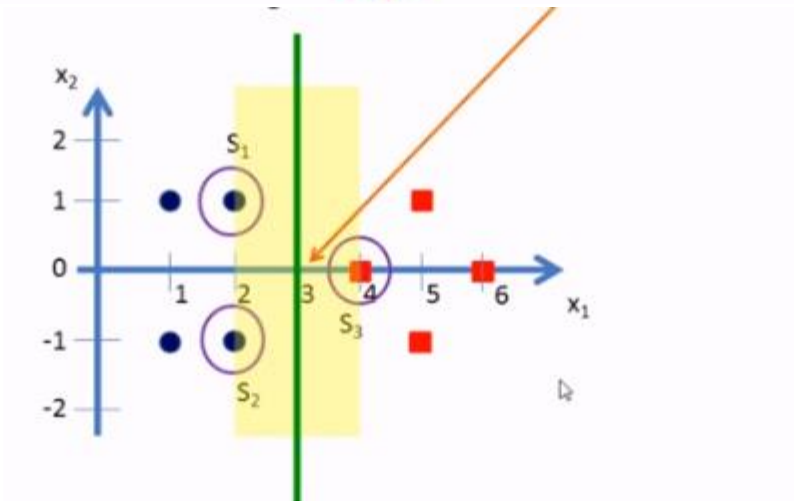
$$\tilde{w} = \sum_i \alpha_i \tilde{S}_i$$

- Substituting the values we get:

$$\tilde{w} = \alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

$$\tilde{w} = (-3.25).\begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + (-3.25).\begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + (3.5).\begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -3 \end{pmatrix}$$

- Our vectors are augmented with a bias.
- Hence we can equate the entry in $\widetilde{w}$ as the hyper plane with an offset b.
- Therefore the separating hyper plane equation $y = wx + b$ with $w = \binom{1}{0}$ and offset $b = -3$.

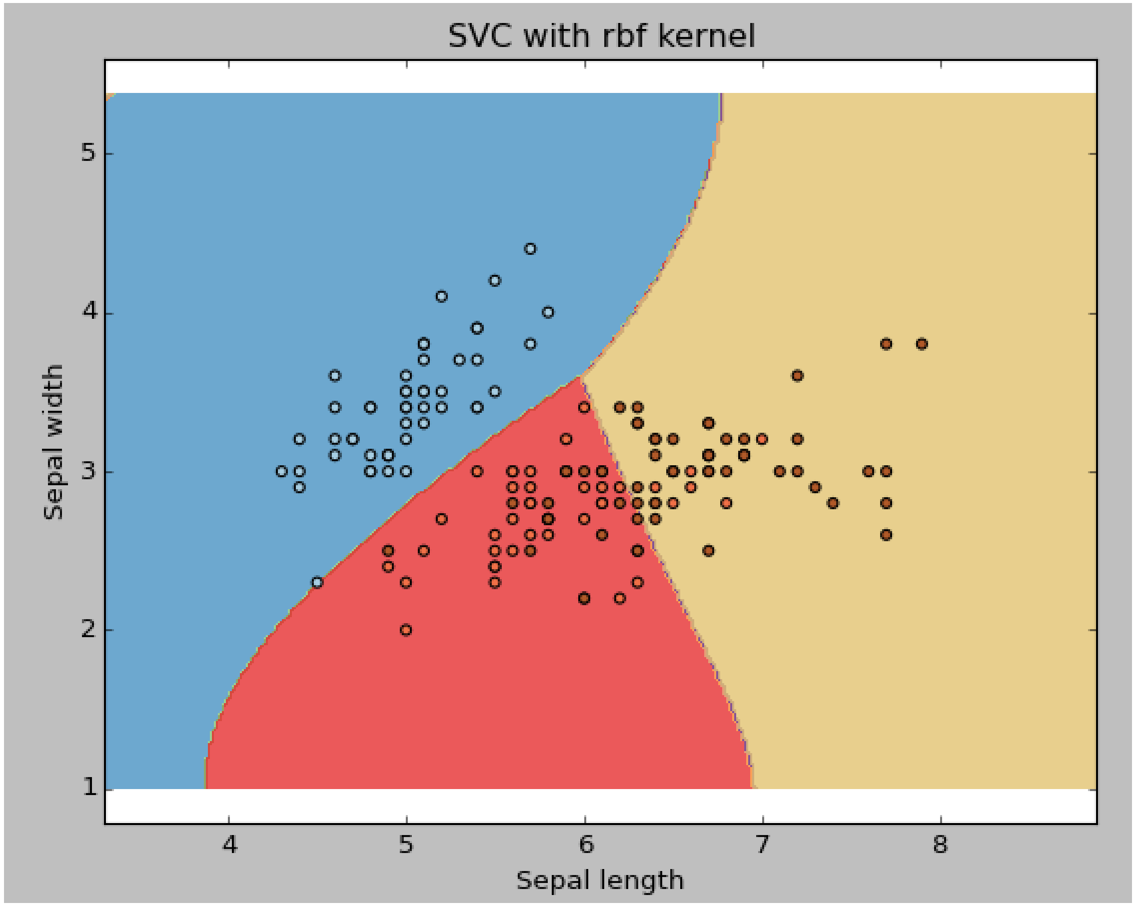


Hint: b value will be complemented. -3 should be considered as 3
W(1,0) plots a vertical line. W(0,1) plots a horizontal line .
(1,1) Any other value plot a slanting line

SVM in Python

- from sklearn.svm import SVC
- clf = SVC(kernel='linear')
- #The kernel parameter can be tuned to take ‘linear’, ‘poly’, ‘sigmoid’, ‘rbf’(radial basis function).
- # fitting x samples and y classes
- clf.fit(x_train, y_train)
- clf.predict(x_test)

Iris dataset



What is a slack variable

- In an optimization problem, a **slack variable** is a **variable** that is added to an inequality constraint to transform it into an equality. Introducing a **slack variable** replaces an inequality constraint with an equality constraint and a non-negativity constraint on the **slack variable**.

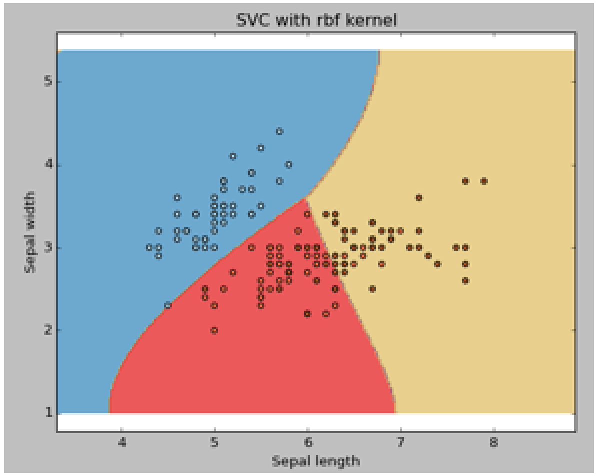
Tuning Parameters

- C and Gamma
- C – Cost / Error Term – soft margin cost function
  - Controls trade-off between smooth decision boundary and classifying training points correctly
- Gamma – Regularization parameter
  - Defines how far the influence of the single training example reaches
  - Low values- far
  - High values- close

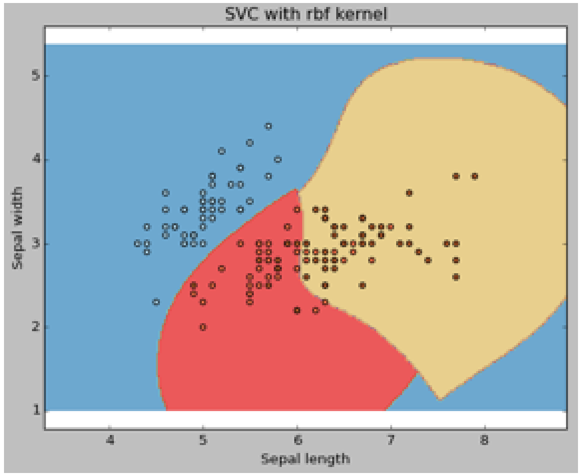
C

- A **large C** gives you **low bias and high variance**. Low bias because you penalize the cost of misclassification a lot. Large C makes the cost of misclassification high, thus forcing the algorithm to explain the input data stricter and potentially overfit.
- A **small C** gives you **higher bias and lower variance**. Small C makes the cost of misclassification low, thus allowing more of them for the sake of wider "cushion“

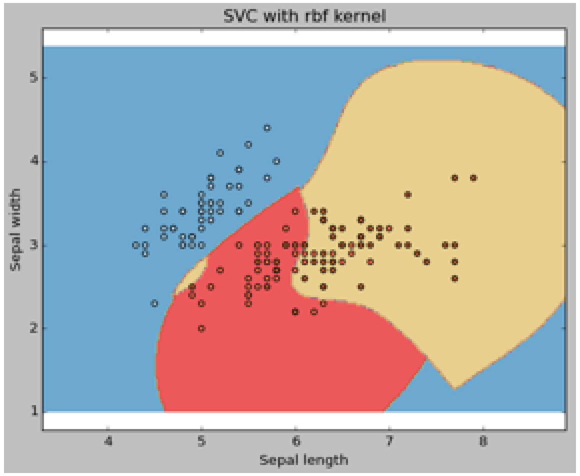
c =1



C =100



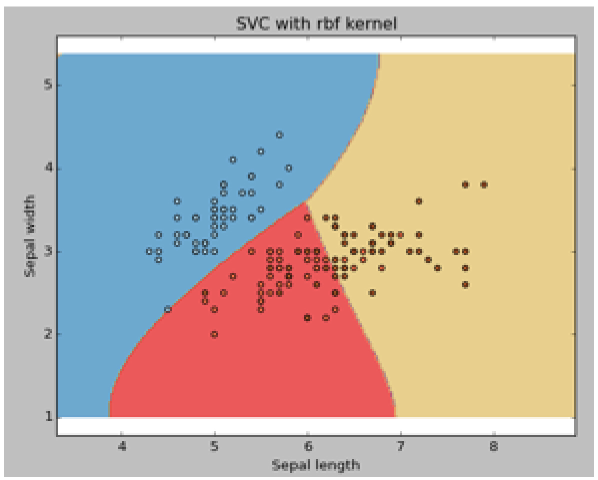
c =1000



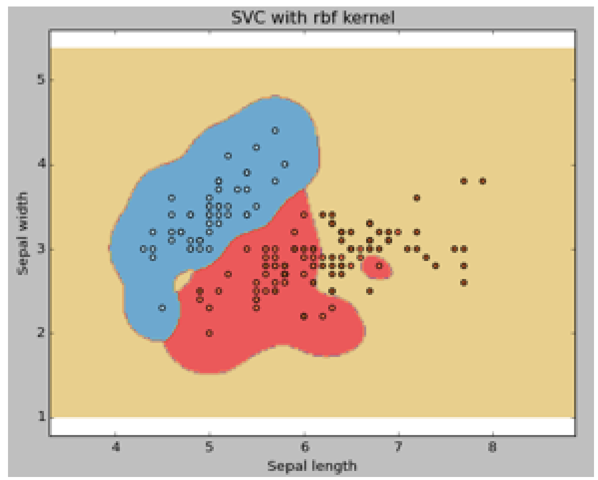
Gamma

- **Gamma** explains how far the influence of a single training example reaches. When gamma is very small, the model is too constrained and cannot capture the complexity or "shape" of the data.
- For a **low gamma**, the model will be **too constrained** and include all points of the training dataset, without really capturing the shape.
- For a **higher gamma**, the model will capture the shape of the dataset well and may overfit

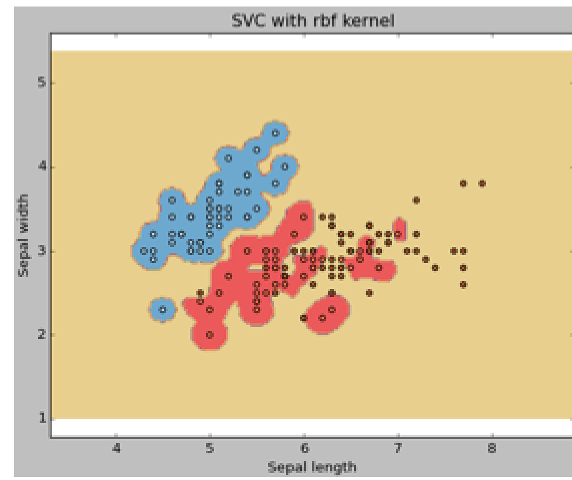
gamma =0



gamma =10



gamma =100



References

- **Problem solving** https://www.youtube.com/watch?v=LXGaYVXkGtg
- https://www.youtube.com/watch?v=QkAmOb1AMrY
- http://axon.cs.byu.edu/Dan/678/miscellaneous/SVM.example.pdf
- **Svm playground**
- http://macheads101.com/demos/svm-playground/
- https://cs.stanford.edu/~karpathy/svmjs/demo/
- **Ml playground**
- http://ml-playground.com/#
- https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/