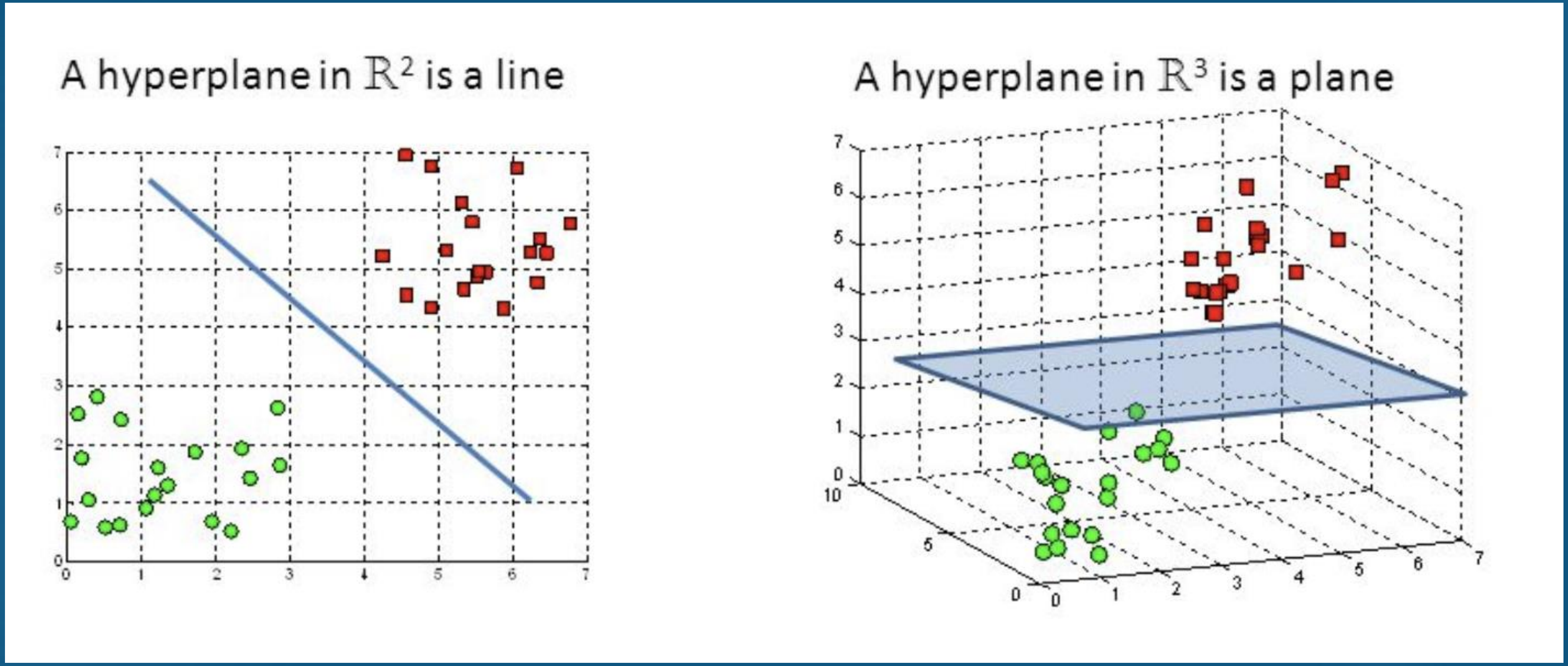
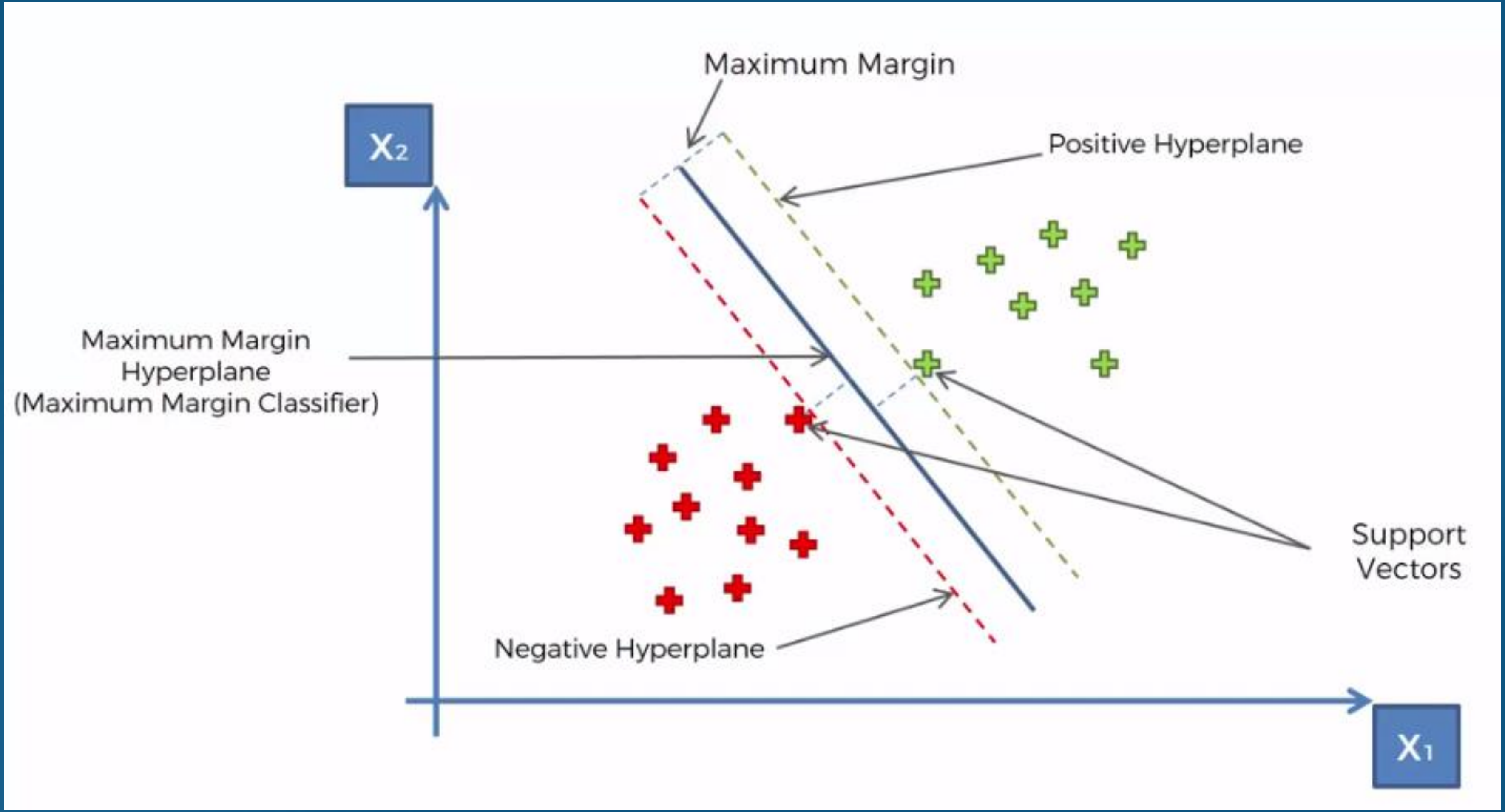
SUPPORT VECTOR MACHINES (SVM)

Support vector machines (SVM)

- **Support Vector Machines (SVM)** are supervised learning models used for classification and regression analysis.
- SVMs are based on the idea of finding a **hyperplane** that best divides a dataset into two classes



Support vector machines (SVM)



The goal is to choose a hyperplane with the greatest possible margin

Linear Classifiers

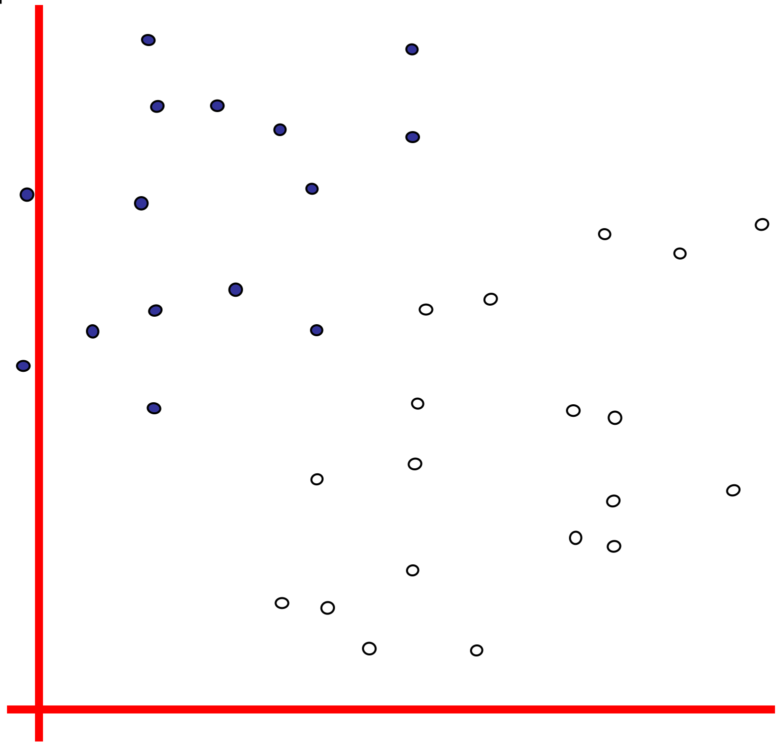
$$\boldsymbol{x} \longrightarrow \boxed{f} \longrightarrow y^{est}$$

Estimation:

$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

● denotes +1

○ denotes -1

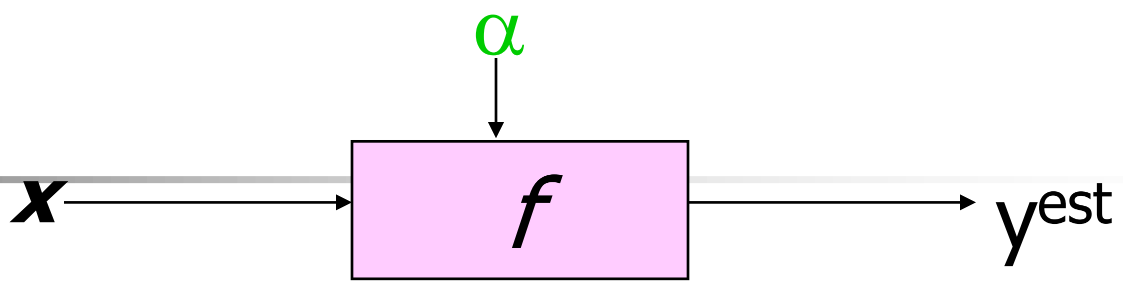


w: weight vector

x: data vector

How would you classify this data?

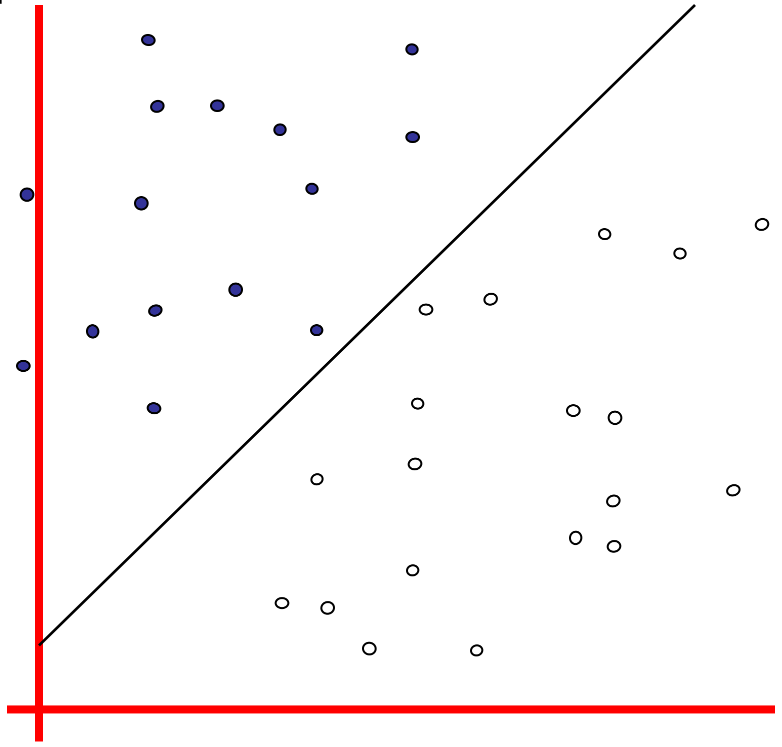
Linear Classifiers



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

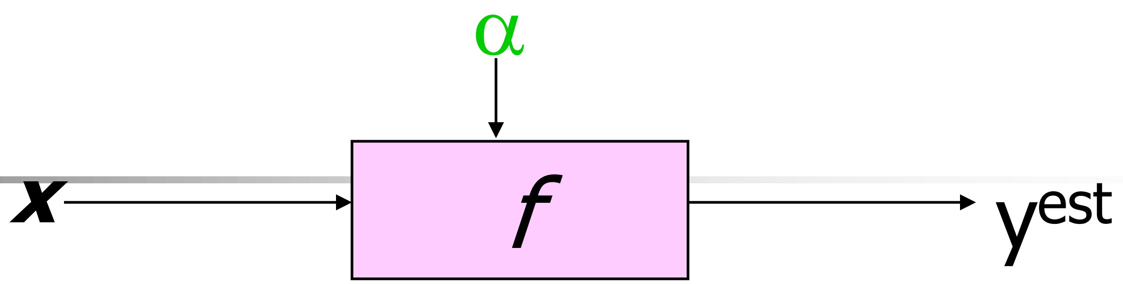
● denotes +1

○ denotes -1



How would you classify this data?

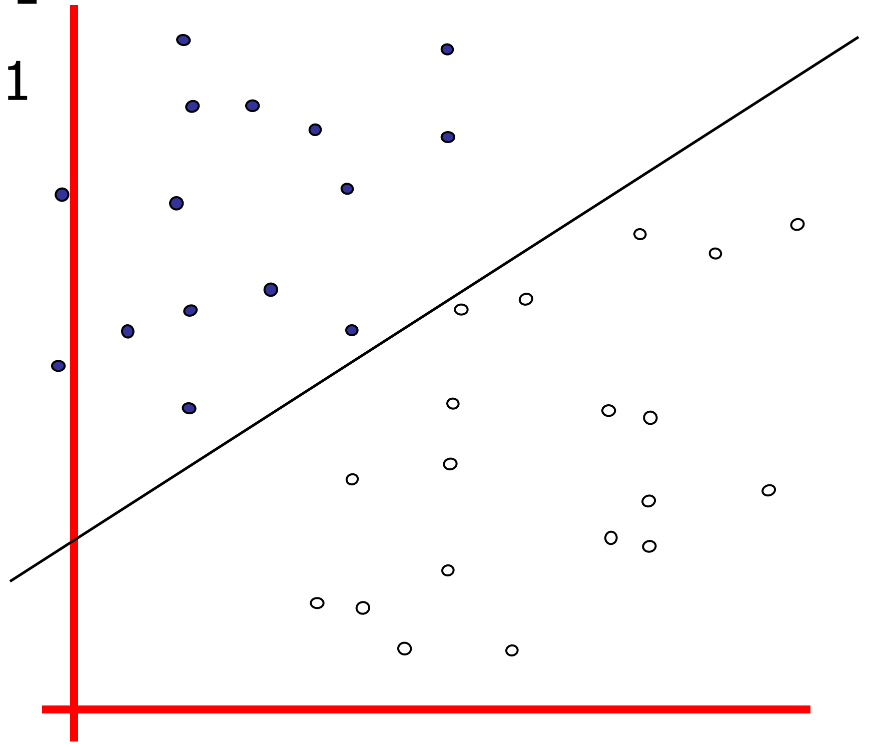
Linear Classifiers



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

● denotes +1

○ denotes -1

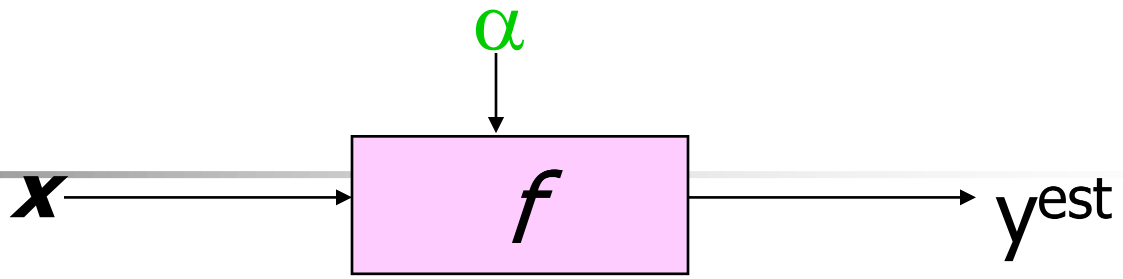


How would you classify this data?

Linear Classifiers

$$\mathbf{x} \longrightarrow \boxed{f} \longrightarrow y^{est}$$

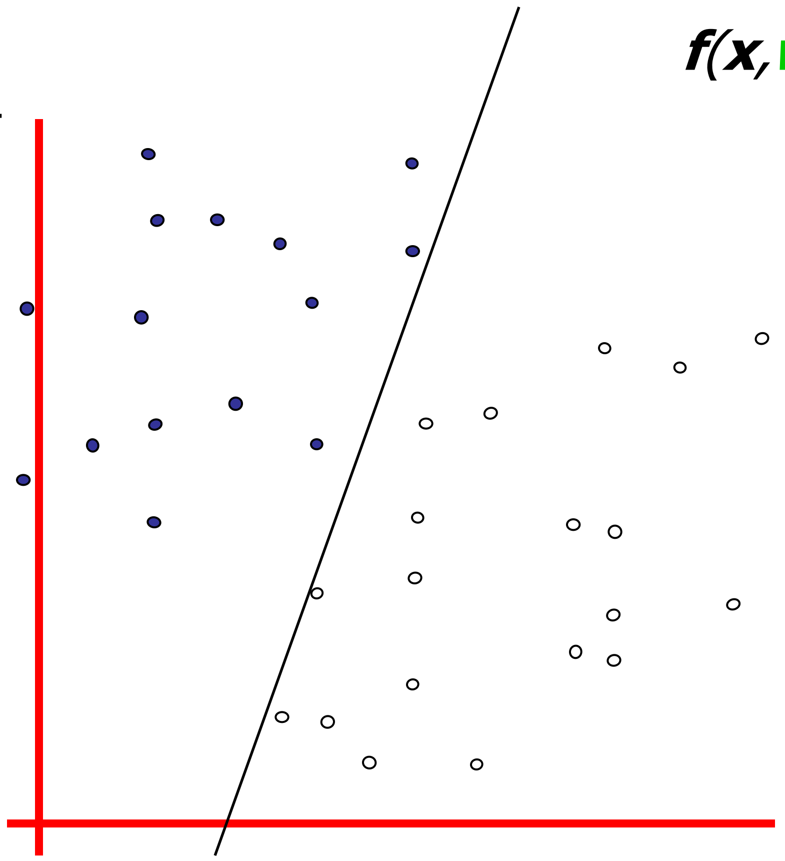
with α as input to f from above.



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

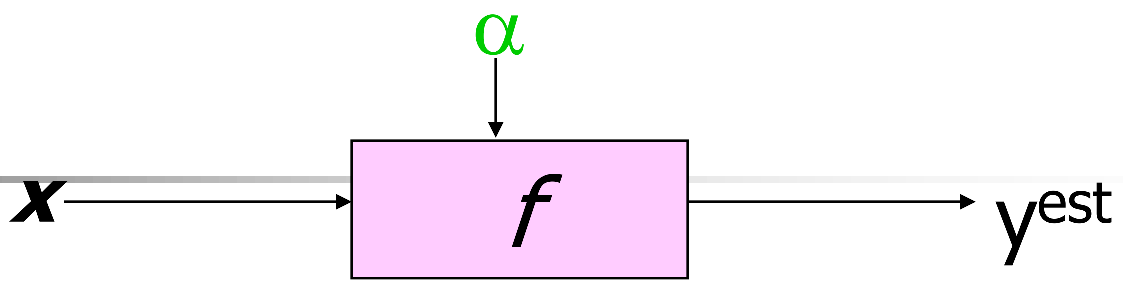
● denotes +1

○ denotes -1



How would you classify this data?

Linear Classifiers

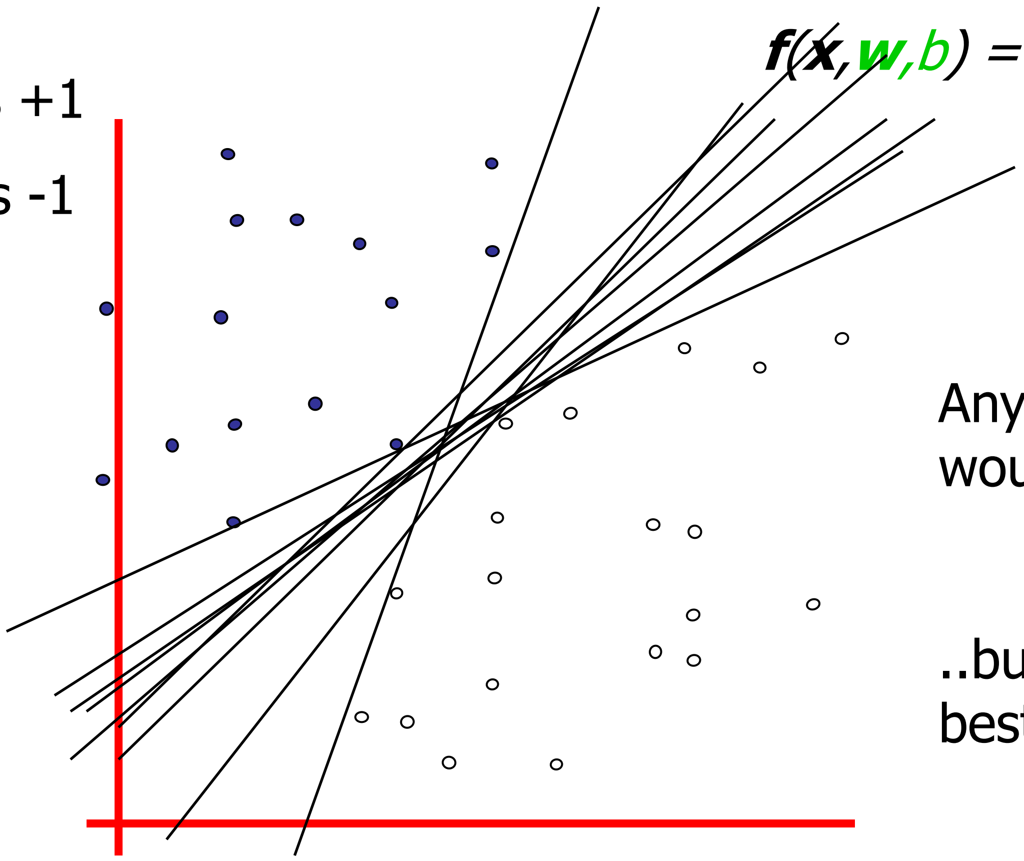


$$\begin{tikzcd} & \alpha \arrow[d] & \\ \boldsymbol{x} \arrow[r] & \boxed{f} \arrow[r] & y^{est} \end{tikzcd}$$

$f(\boldsymbol{x},\boldsymbol{w},b) = sign(\boldsymbol{w}\boldsymbol{.}\ \boldsymbol{x} - b)$

● denotes +1

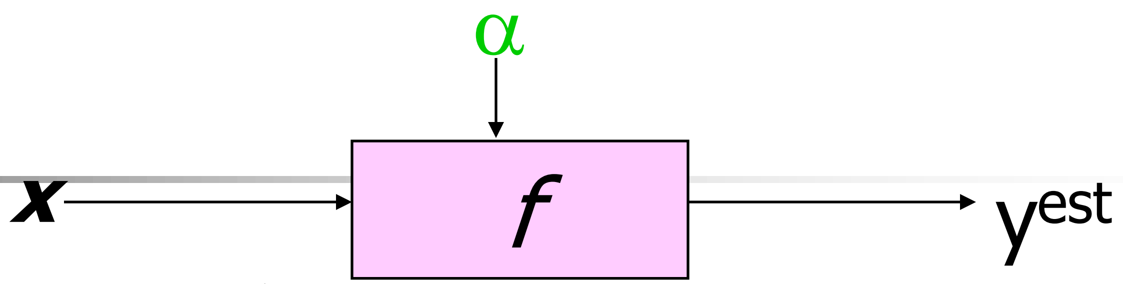
○ denotes -1



Any of these would be fine..

..but which is best?

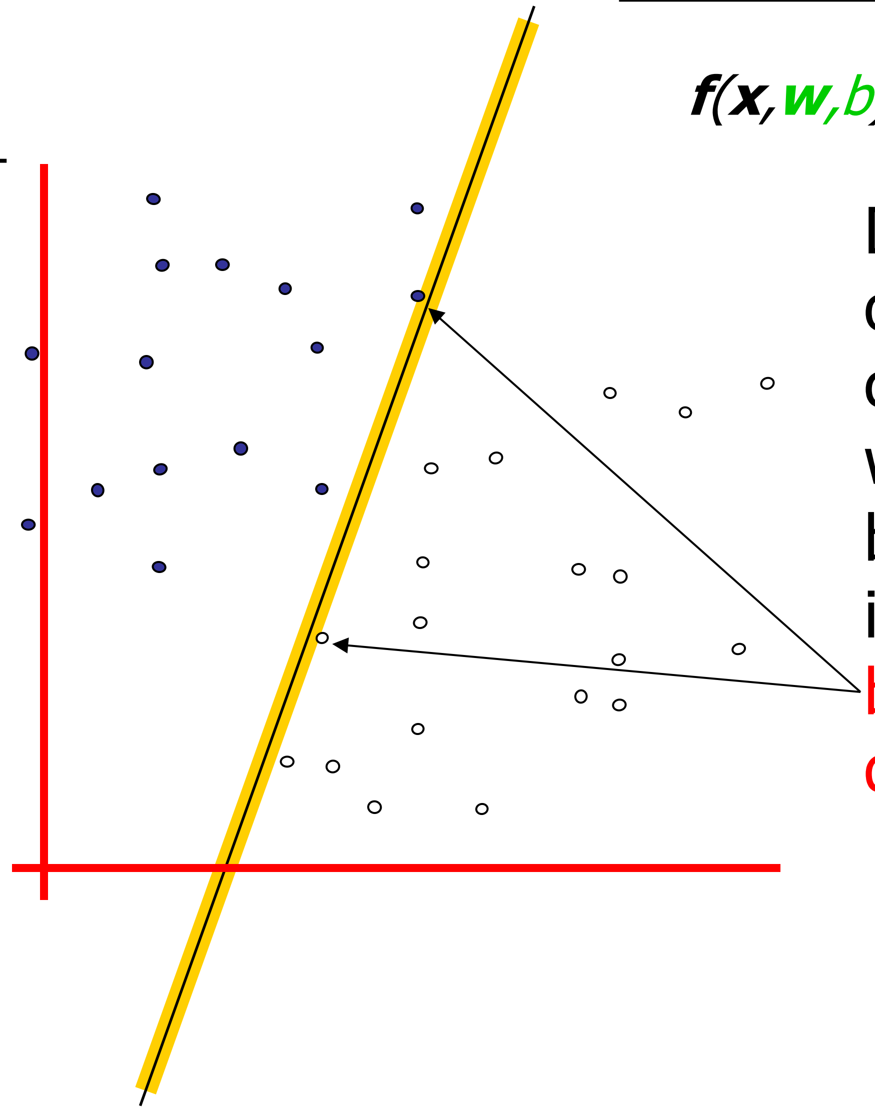
Classifier Margin



$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{w},b) = sign(\boldsymbol{w}.\ \boldsymbol{x} - b)$

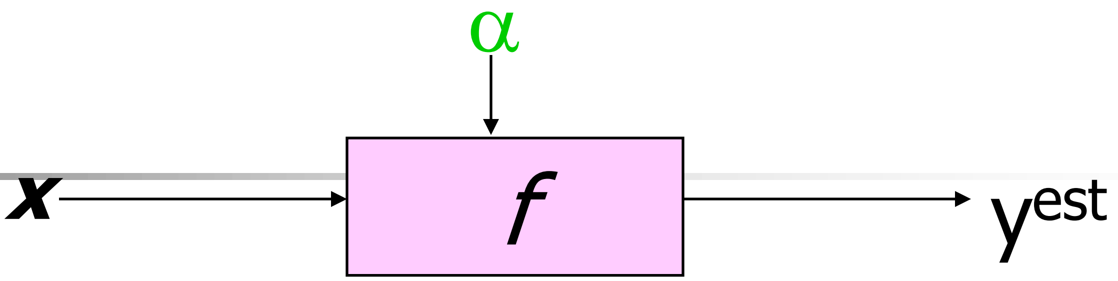
● denotes +1

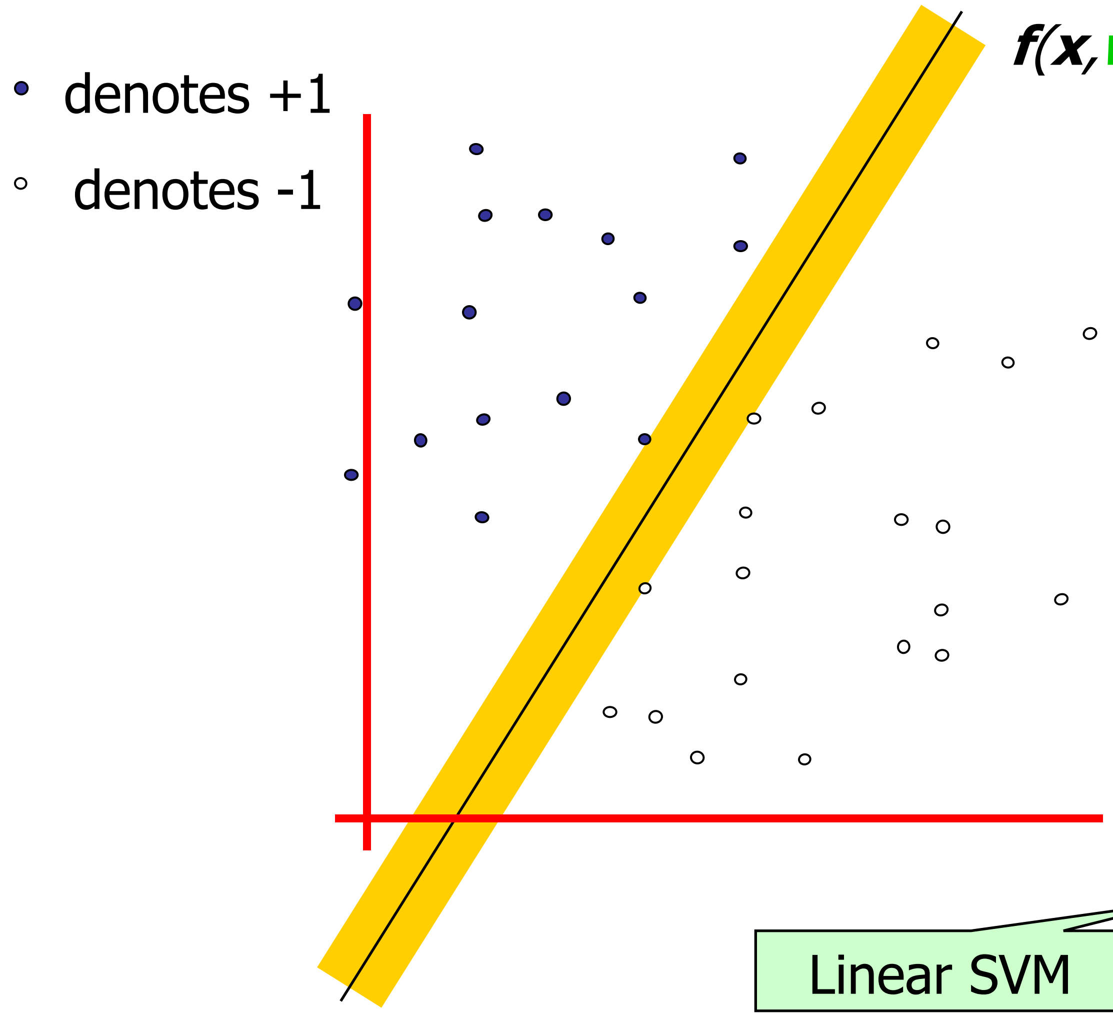
○ denotes -1



Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin





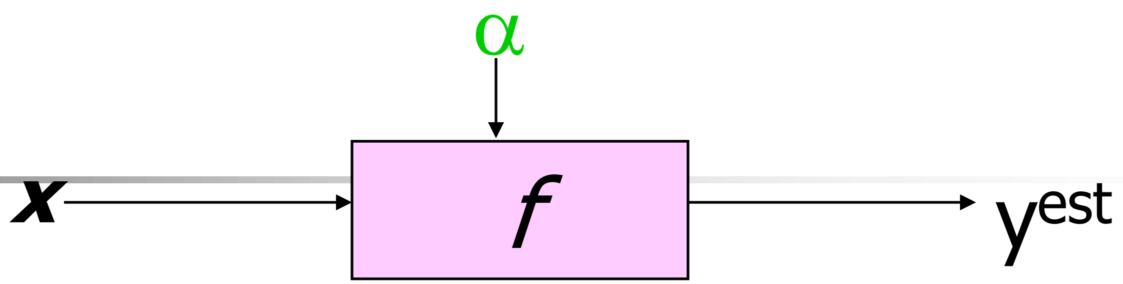
$f(\boldsymbol{x},\boldsymbol{w},b) = sign(\boldsymbol{w}. \boldsymbol{x} - b)$

The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

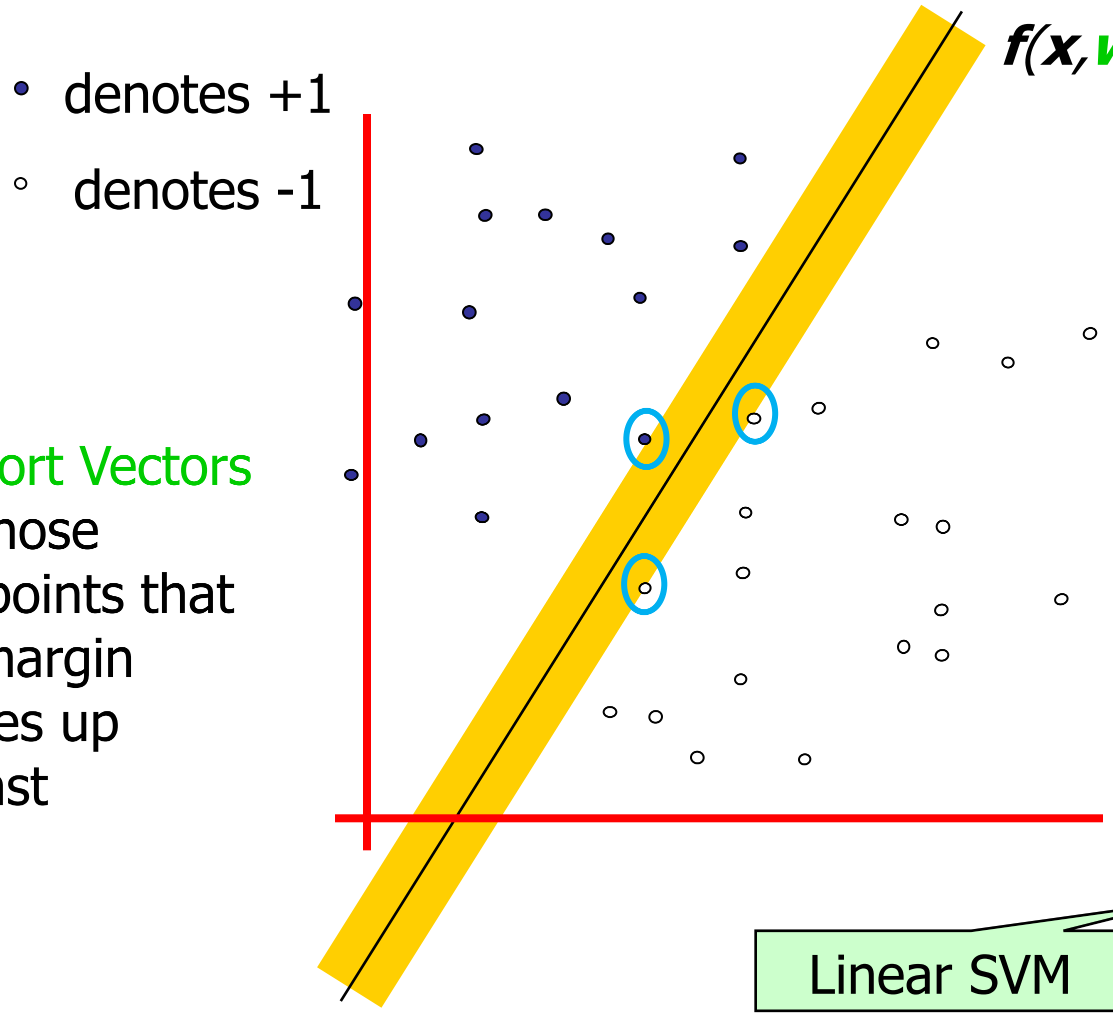
This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Maximum Margin



$f(\boldsymbol{x},\boldsymbol{w},b) = sign(\boldsymbol{w}.\ \boldsymbol{x} + b)$



Support Vectors are those datapoints that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

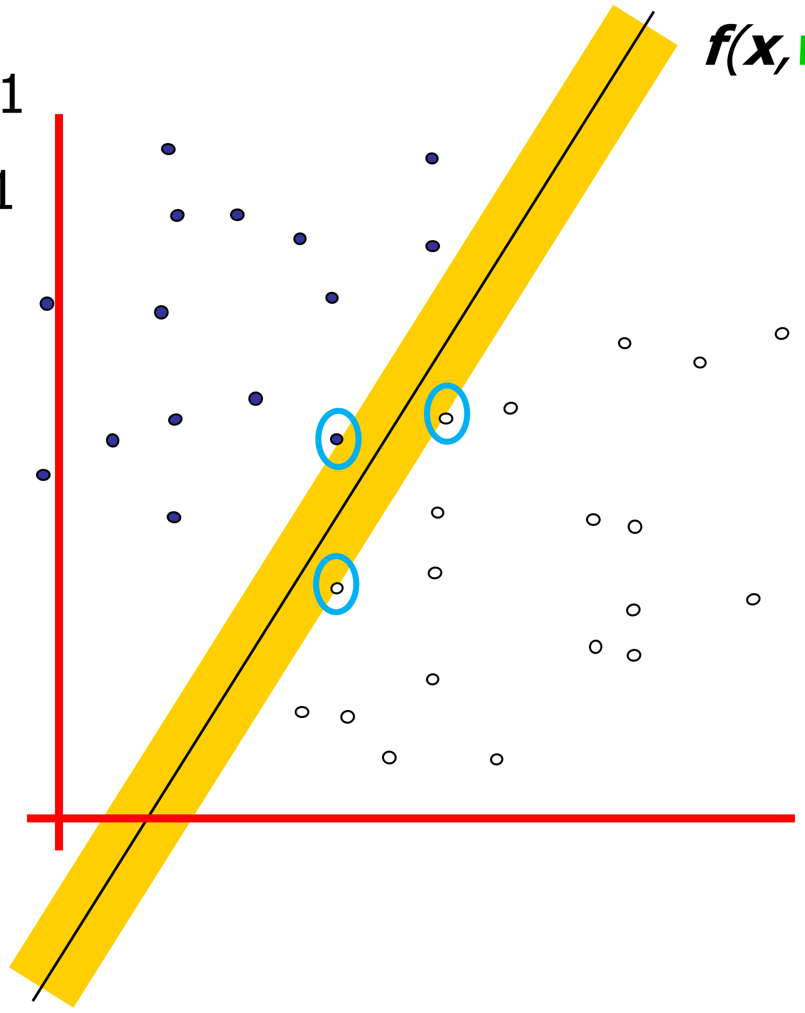
Linear SVM

Why Maximum Margin?

● denotes +1

○ denotes -1

$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

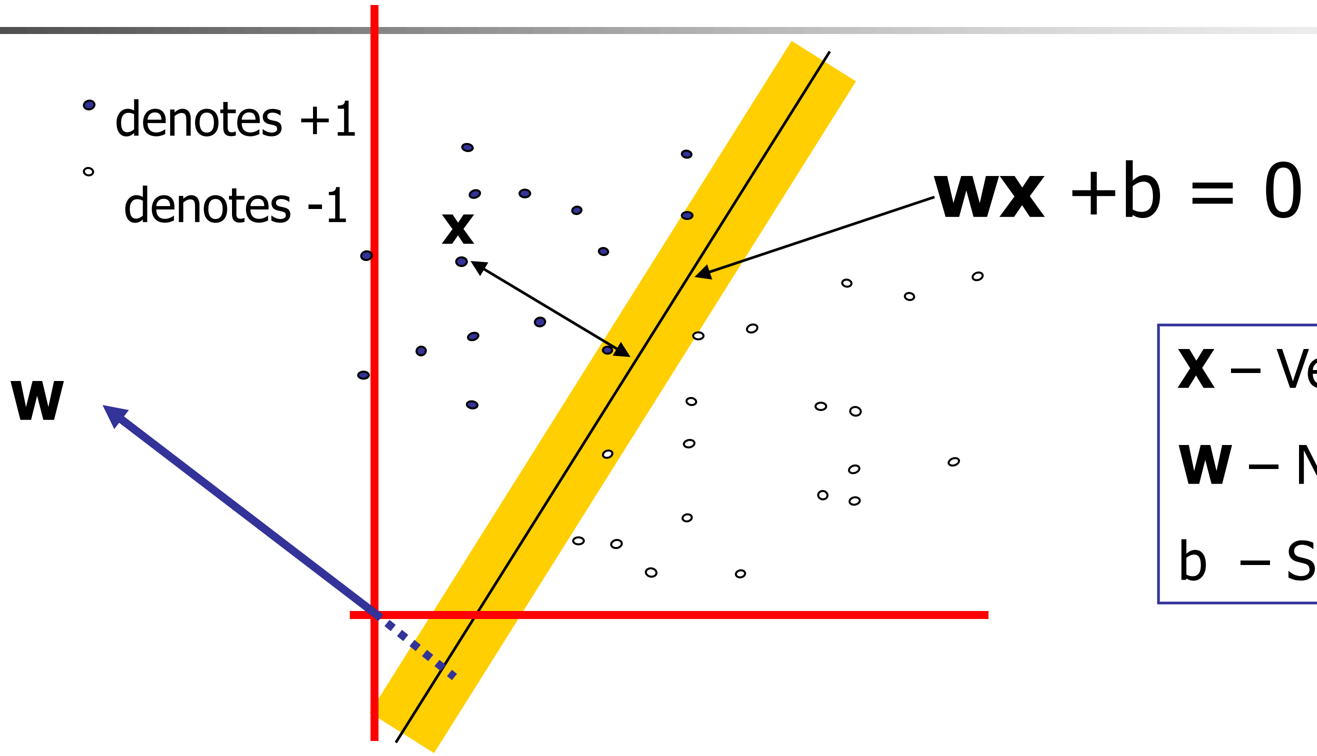


Support Vectors are those datapoints that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the maximum margin.

This is the simplest kind of SVM (Called an LSVM)

How to calculate the distance from a point to a line?



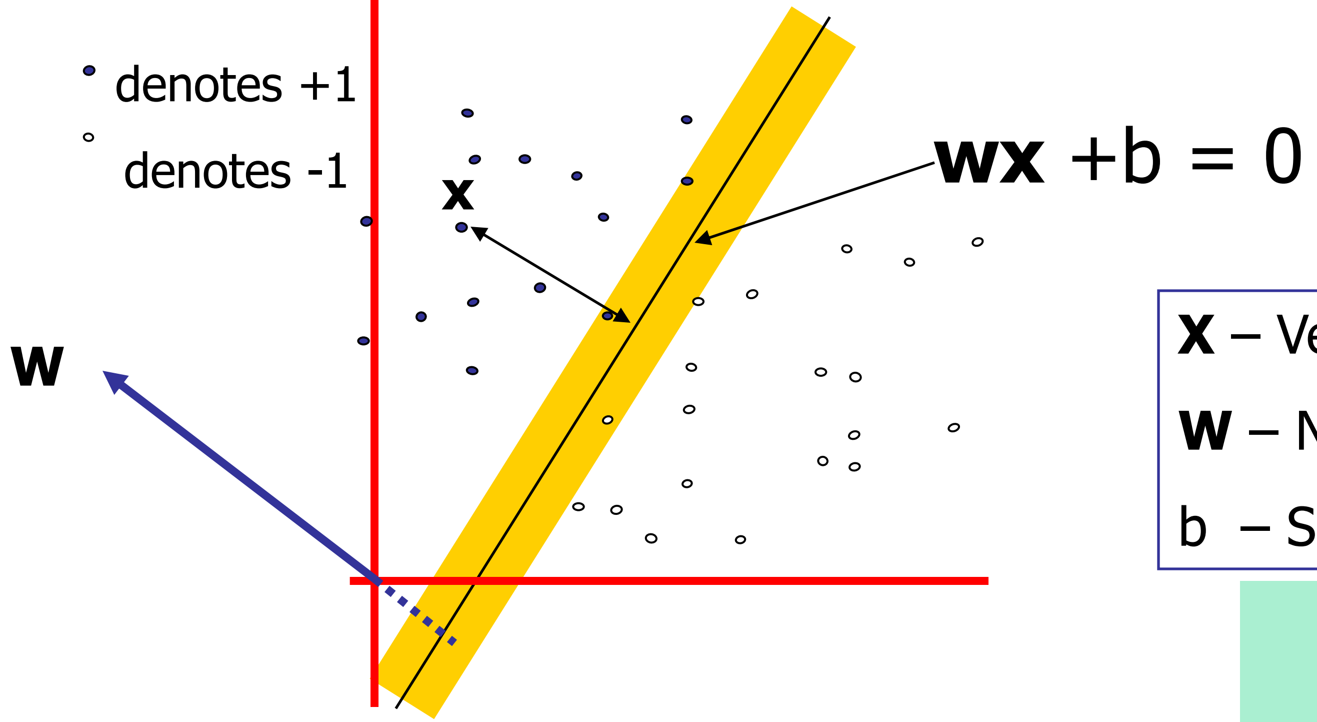
X – Vector

W – Normal Vector

b – Scale Value

- http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html
- In our case, $w_1 * x_1 + w_2 * x_2 + b = 0$,
- thus, $\mathbf{w} = (w_1, w_2)$, $\mathbf{x} = (x_1, x_2)$

Estimate the Margin



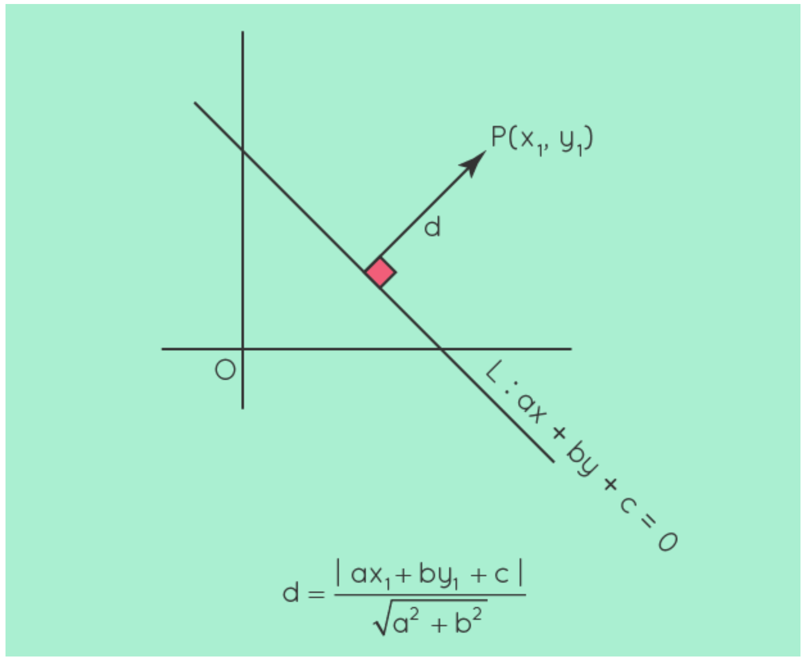
X – Vector

W – Normal Vector

b – Scale Value

- What is the distance expression for a point **x** to a line **wx**+b= 0?

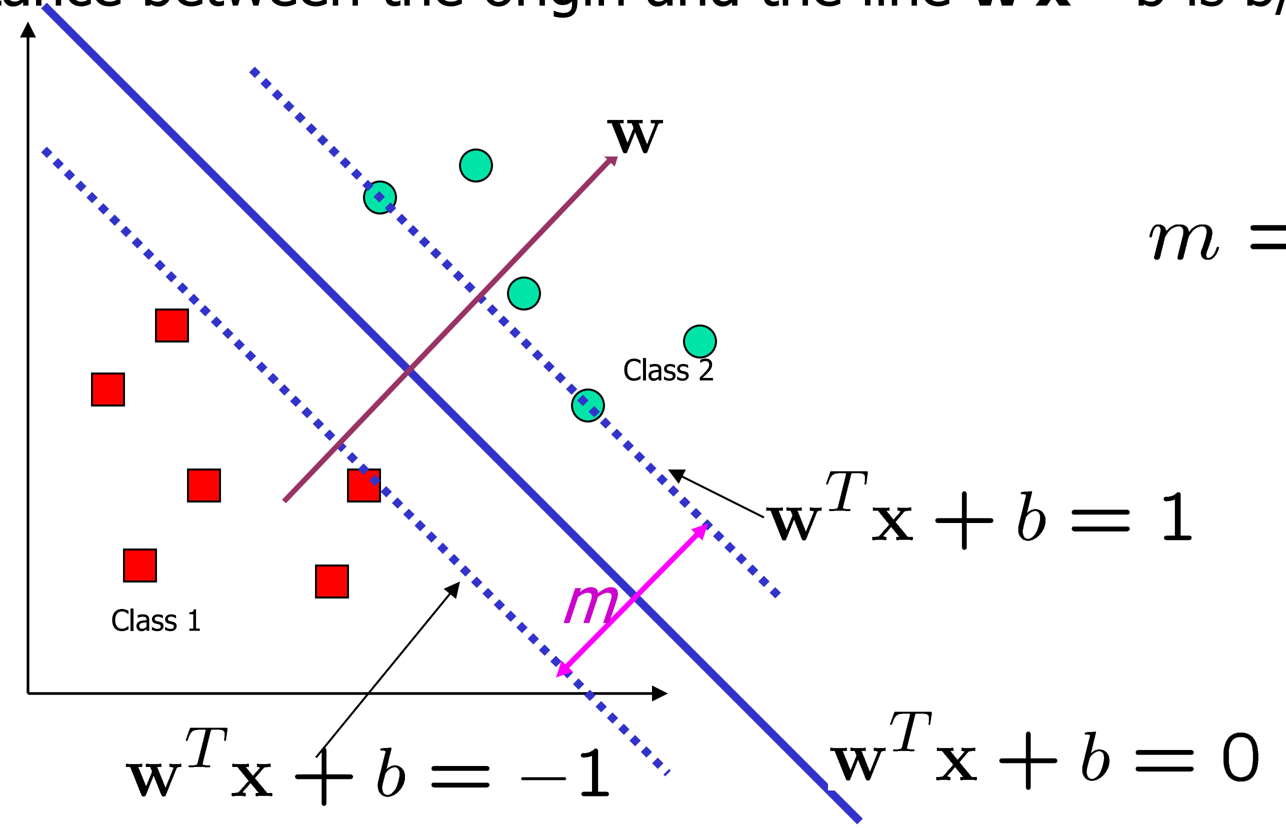
$$d(\mathbf{x}) = \frac{|\mathbf{x}\cdot\mathbf{w}+b|}{\sqrt{\|\mathbf{w}\|_2^2}} = \frac{|\mathbf{x}\cdot\mathbf{w}+b|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$



$$d = \frac{|ax_1 + by_1 + c|}{\sqrt{a^2 + b^2}}$$

Large-margin Decision Boundary

- The decision boundary should be as far away from the data of both classes as possible
  - We should maximize the margin, *m*
  - Distance between the origin and the line $\mathbf{w}^t\mathbf{x}=-b$ is $b/||\mathbf{w}||$



$\mathbf{w}$
Class 2
Class 1
m
$\mathbf{w}^T\mathbf{x}+b=1$
$\mathbf{w}^T\mathbf{x}+b=0$
$\mathbf{w}^T\mathbf{x}+b=-1$

$$m = \frac{2}{||\mathbf{w}||}$$

Finding the Decision Boundary

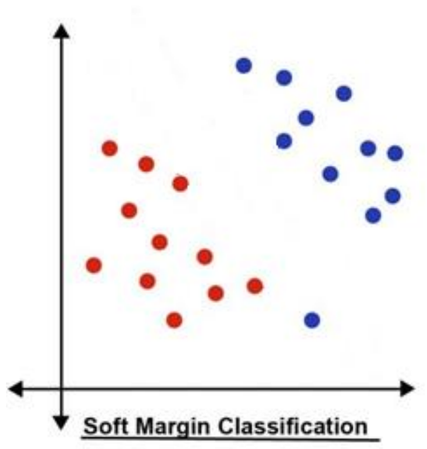
- Let $\{x_1, \ldots, x_n\}$ be our data set and let $y_i \in \{1,-1\}$ be the class label of x_i
- The decision boundary should classify all points correctly $\Rightarrow$

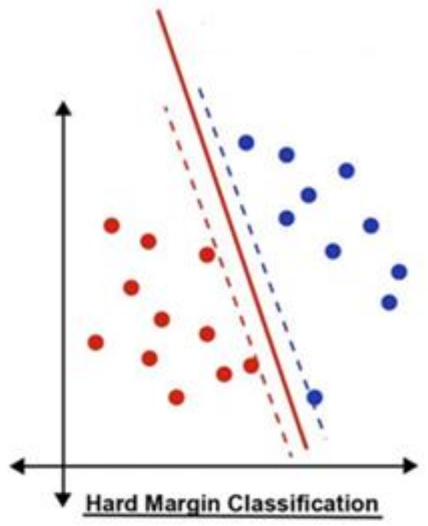
$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \qquad \forall i$$

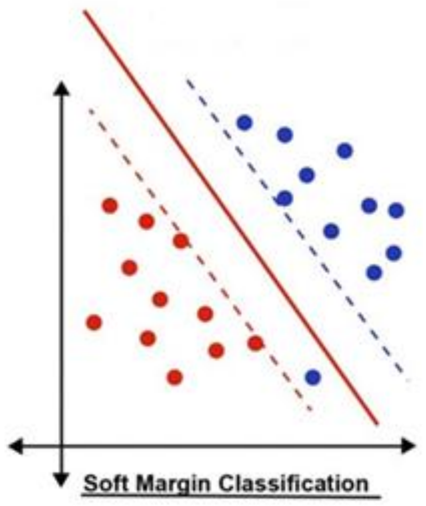
- To see this: when y=-1, we wish (wx+b)<1, when y=1, we wish (wx+b)>1. For support vectors, we wish y(wx+b)=1.
- The decision boundary can be found by solving the following constrained optimization problem

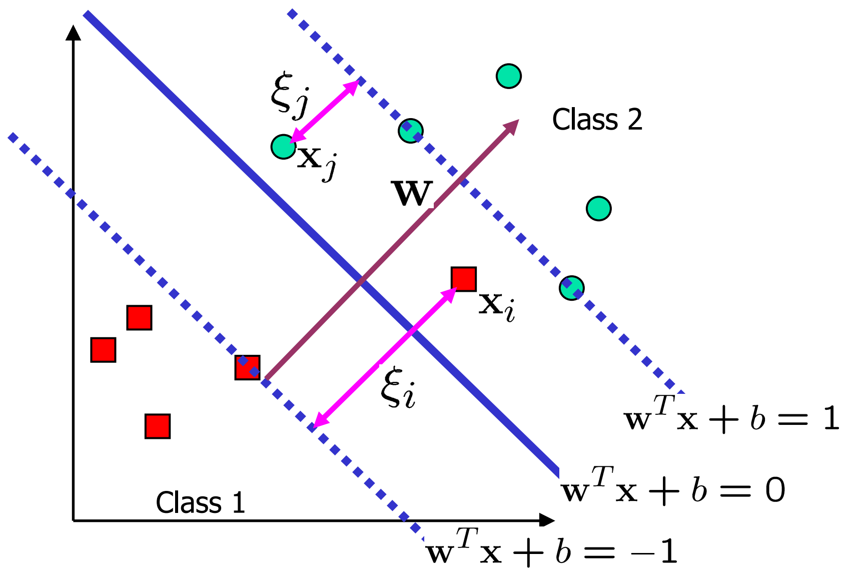
$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

Allowing errors in our solutions








ξ_j
$\mathbf{x}_j$
$\mathbf{w}$
Class 2
$\mathbf{x}_i$
ξ_i
$\mathbf{w}^T\mathbf{x}+b=1$
$\mathbf{w}^T\mathbf{x}+b=0$
Class 1
$\mathbf{w}^T\mathbf{x}+b=-1$

- We allow "error" ξ_i in classification; it is based on the output of the discriminant function $\mathbf{w}^T\mathbf{x}+b$
- ξ_i approximates the number of misclassified samples

Soft Margin Hyperplane

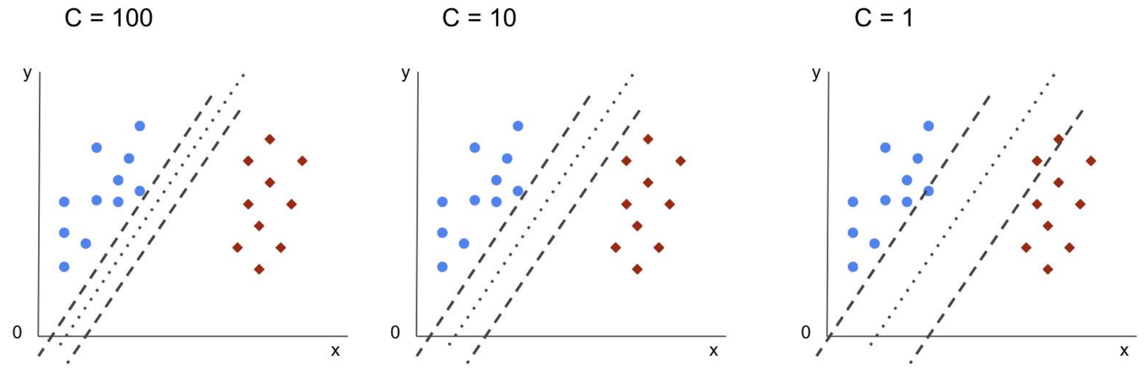
- If we minimize $\sum_i \xi_i$, ξ_i can be computed by

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

  - ξ_i are "slack variables" in optimization
  - Note that ξ_i=0 if there is no error for $\mathbf{x}_i$
  - ξ_i is an upper bound of the number of errors

- We want to minimize
  - *C* : tradeoff parameter between error and margin

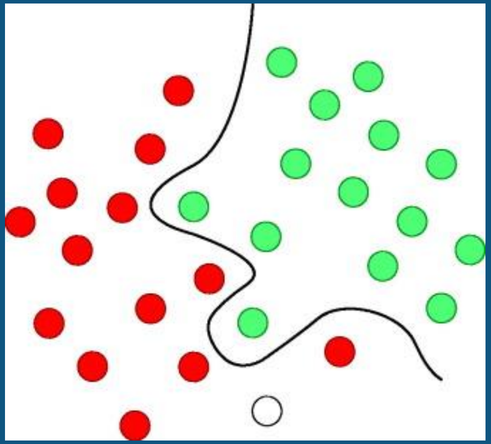
$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

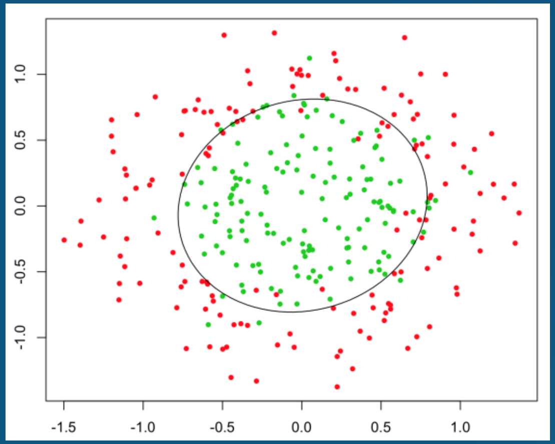


- The optimization problem becomes

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Support Vector machine (SVM)



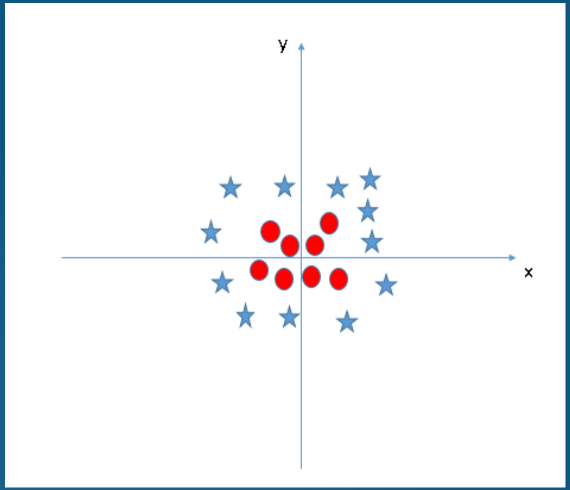


Support Vector Machines:

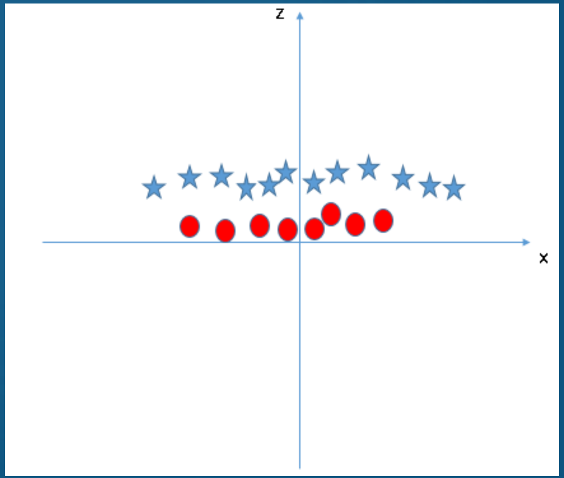
When non-linear boundary is used for the classification purpose, it is called Support Vector Machines (SVMs). It is an extension of the Support Vector Classifier and done by enlarging the feature space in a specific way, using *kernels*.

Find the Hyperplane

In the scenario below, we can’t have linear hyperplane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyperplane.



SVM can solve this problem. It solves this problem by introducing additional feature. Here, we will add a new feature $z=x^2+y^2$.

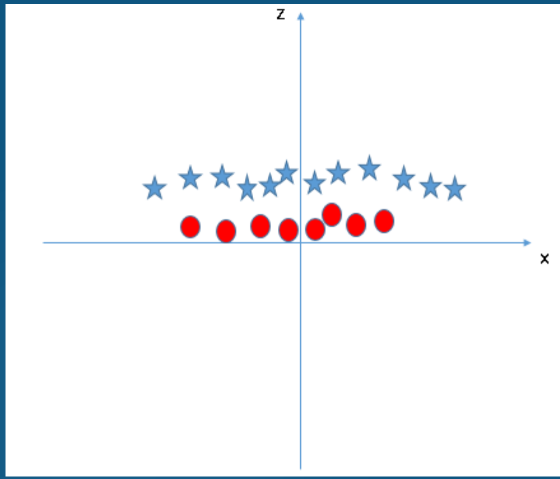


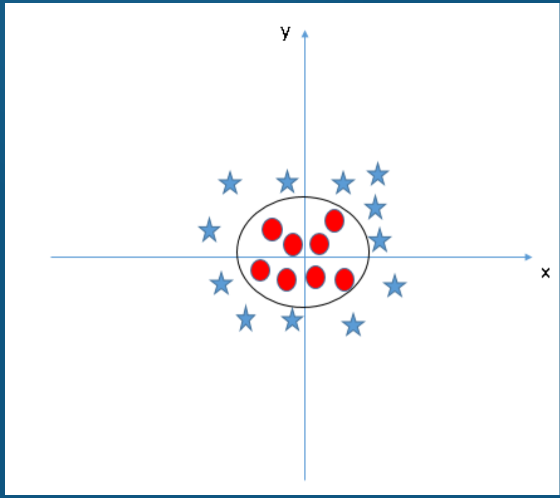
In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

KERNEL TRICK

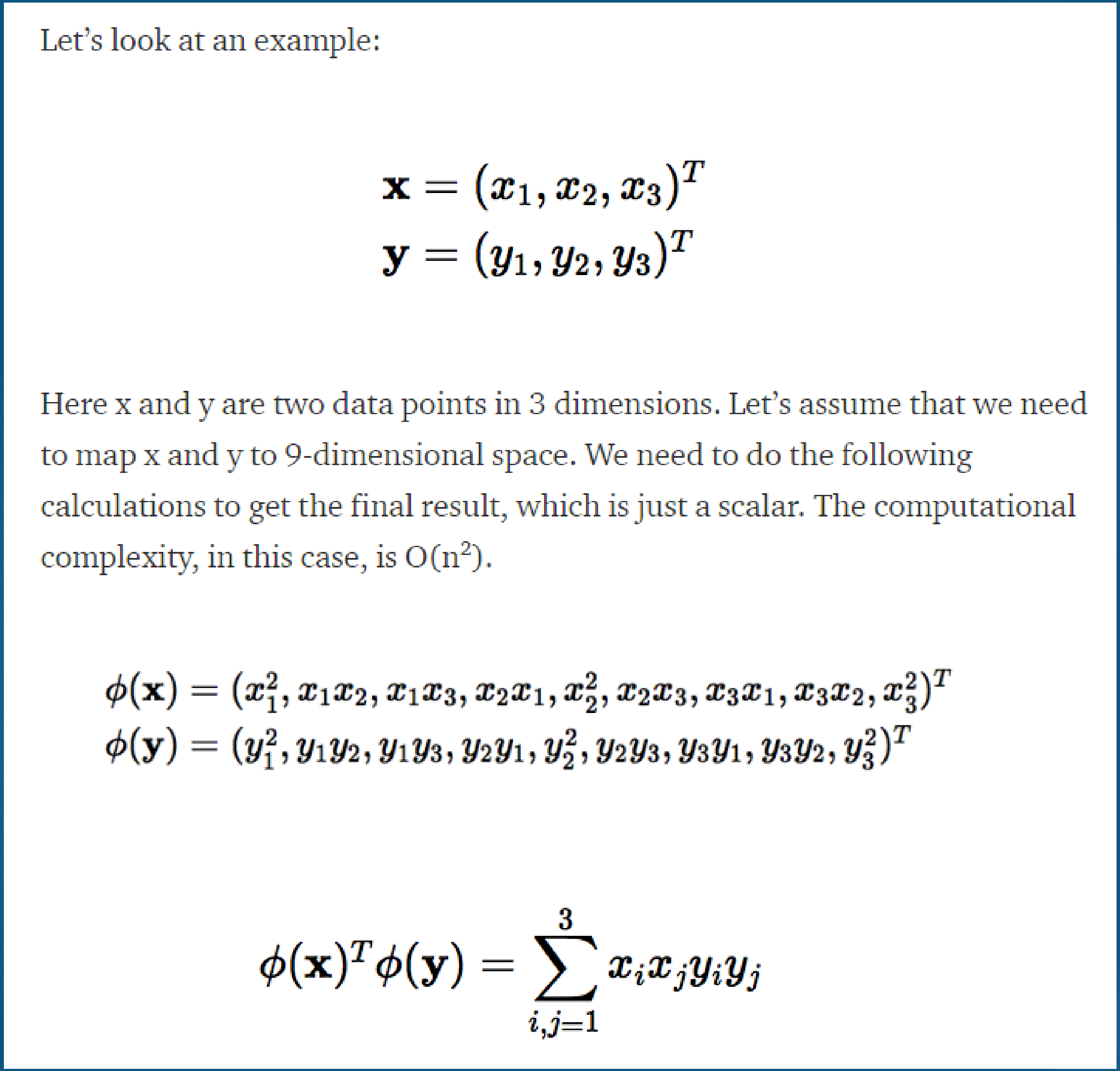
Should we need to add this feature manually (inefficient with much computation) to have a hyperplane?

- No, SVM has a technique called the **kernel trick**. These are functions which takes low dimensional input space and transform it efficiently (trick) to a higher dimensional space i.e. it converts non separable problem to separable problem, these functions are called **kernels**.





KERNEL FUNCTİONS



Let’s look at an example:

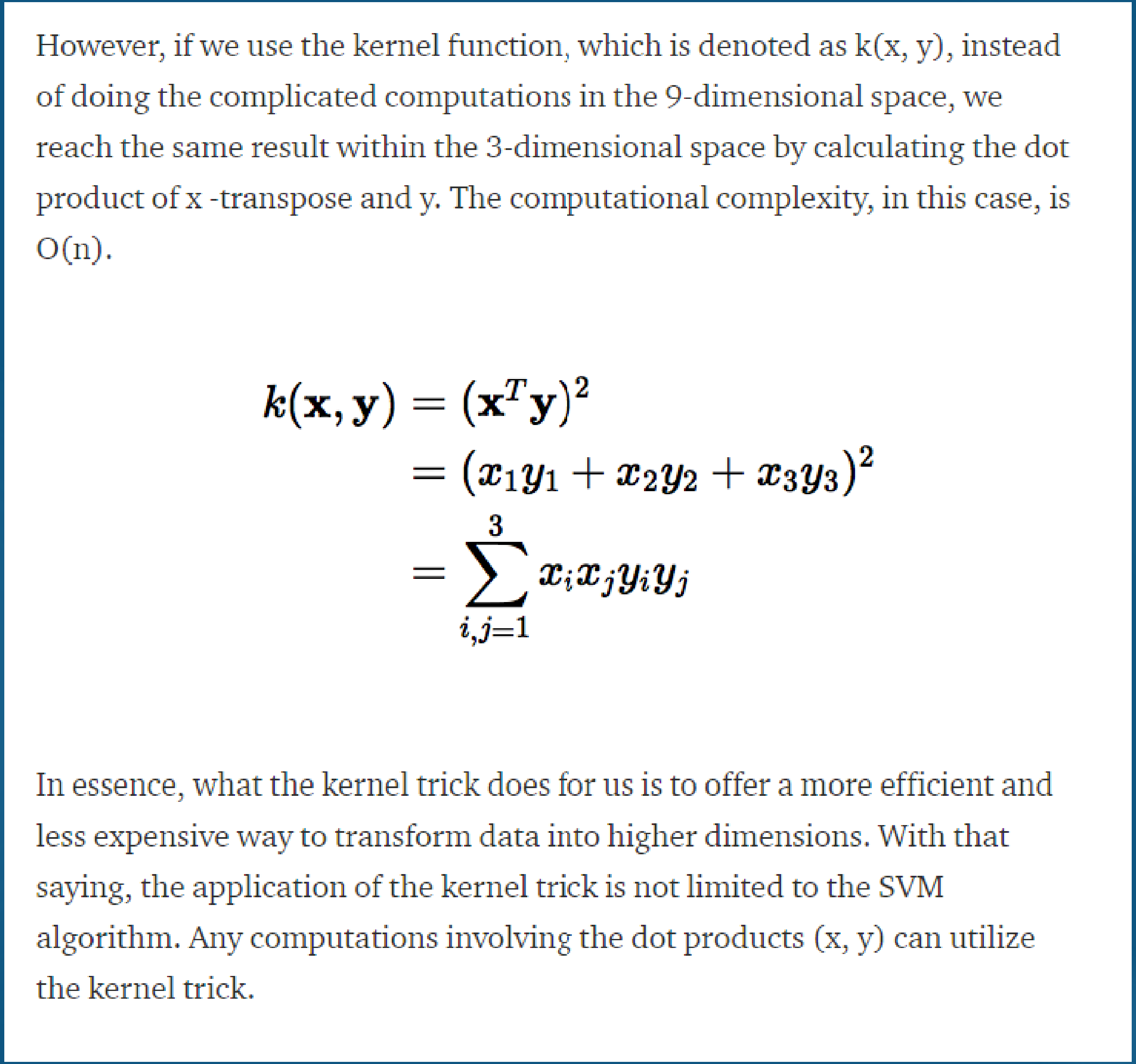
$$\mathbf{x} = (x_1, x_2, x_3)^T$$
$$\mathbf{y} = (y_1, y_2, y_3)^T$$

Here x and y are two data points in 3 dimensions. Let’s assume that we need to map x and y to 9-dimensional space. We need to do the following calculations to get the final result, which is just a scalar. The computational complexity, in this case, is O(n^2).

$$\phi(\mathbf{x}) = (x_1^2, x_1x_2, x_1x_3, x_2x_1, x_2^2, x_2x_3, x_3x_1, x_3x_2, x_3^2)^T$$
$$\phi(\mathbf{y}) = (y_1^2, y_1y_2, y_1y_3, y_2y_1, y_2^2, y_2y_3, y_3y_1, y_3y_2, y_3^2)^T$$

$$\phi(\mathbf{x})^T\phi(\mathbf{y}) = \sum_{i,j=1}^{3} x_ix_jy_iy_j$$

KERNEL FUNCTİONS



However, if we use the kernel function, which is denoted as k(x, y), instead of doing the complicated computations in the 9-dimensional space, we reach the same result within the 3-dimensional space by calculating the dot product of x -transpose and y. The computational complexity, in this case, is O(n).

$$\begin{aligned} k(\mathbf{x},\mathbf{y}) &= (\mathbf{x}^T\mathbf{y})^2 \\ &= (x_1y_1 + x_2y_2 + x_3y_3)^2 \\ &= \sum_{i,j=1}^{3} x_i x_j y_i y_j \end{aligned}$$

In essence, what the kernel trick does for us is to offer a more efficient and less expensive way to transform data into higher dimensions. With that saying, the application of the kernel trick is not limited to the SVM algorithm. Any computations involving the dot products (x, y) can utilize the kernel trick.

EXAMPLES

For example let x and y be defined as **$x = (x1, x2, x3)$** and **$y = (y1, y2, y3)$.**

The mapping to 9 dimensions would be

$$f(x) = (x1x1, x1x2, x1x3, x2x1, x2x2, x2x3, x3x1, x3x2, x3x3)$$

We can define a kernel which would be equivalent to the above equation

$$K(x, y\ ) = dot(x, y) = x.y = (x^T y)^2$$

Let,

$$x = (1, 2, 3)$$
$$y = (4, 5, 6).$$

Then:

$$f(x) = (1, 2, 3, 2, 4, 6, 3, 6, 9)$$
$$f(y) = (16, 20, 24, 20, 25, 30, 24, 30, 36)$$

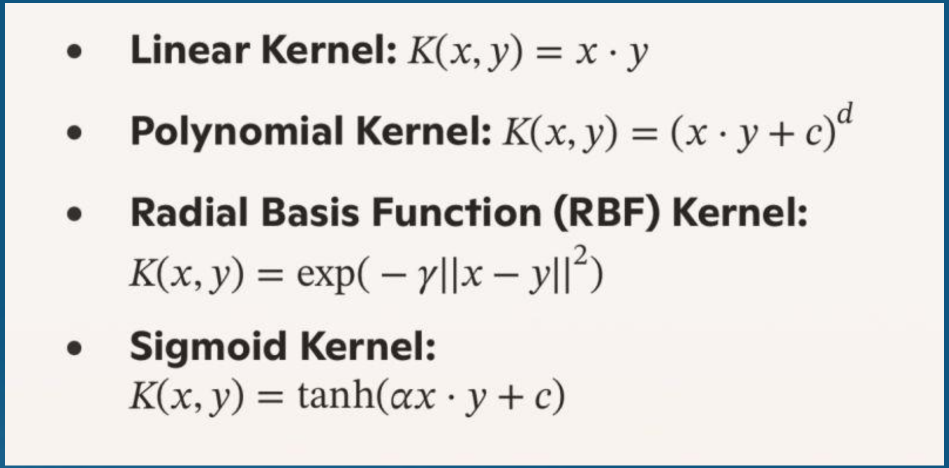
Calculating <f(x), f(y)> , gives us

$$16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324 = 1024$$

Instead of doing so many calculations, if we apply the kernel instead:

$$K(x, y) = (4 + 10 + 18\ )\ \hat{}2 = 32^2 = 1024$$

KERNELS



- **Linear Kernel:** $K(x, y) = x \cdot y$
- **Polynomial Kernel:** $K(x, y) = (x \cdot y + c)^d$
- **Radial Basis Function (RBF) Kernel:** $K(x, y) = \exp(-\gamma ||x - y||^2)$
- **Sigmoid Kernel:** $K(x, y) = \tanh(\alpha x \cdot y + c)$

Polynomial Kernel:

Suppose we have two features X1 and X2 and output variable as Y, so using polynomial kernel we can write it as:

$$\begin{aligned} X1^T . X2 &= \begin{bmatrix} X1 \\ X2 \end{bmatrix} . \begin{bmatrix} X1 & X2 \end{bmatrix} \\ &= \begin{bmatrix} X1^2 & X1 . X2 \\ X1 . X2 & X2^2 \end{bmatrix} \end{aligned}$$

So, we basically need to find X_1^2 , X_2^2 and $X_1.X_2$, and now we can see that 2 dimensions got converted into 5 dimensions.

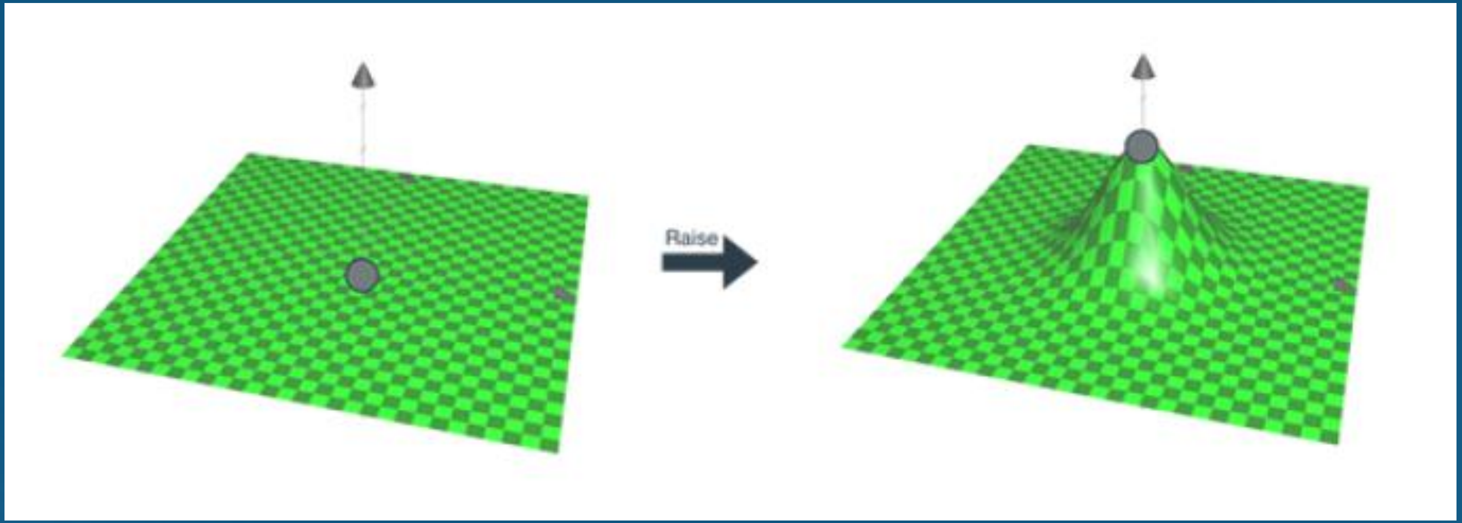
Linear Kernel:

Linear kernel is the dot product of the input samples. The dot product of the two points determines the cosine similarity, The higher the value, the more similar the points are.

Sigmoid Kernel:

It is just taking your input, mapping them to a value of 0 and 1 so that they can be separated by a simple straight line.

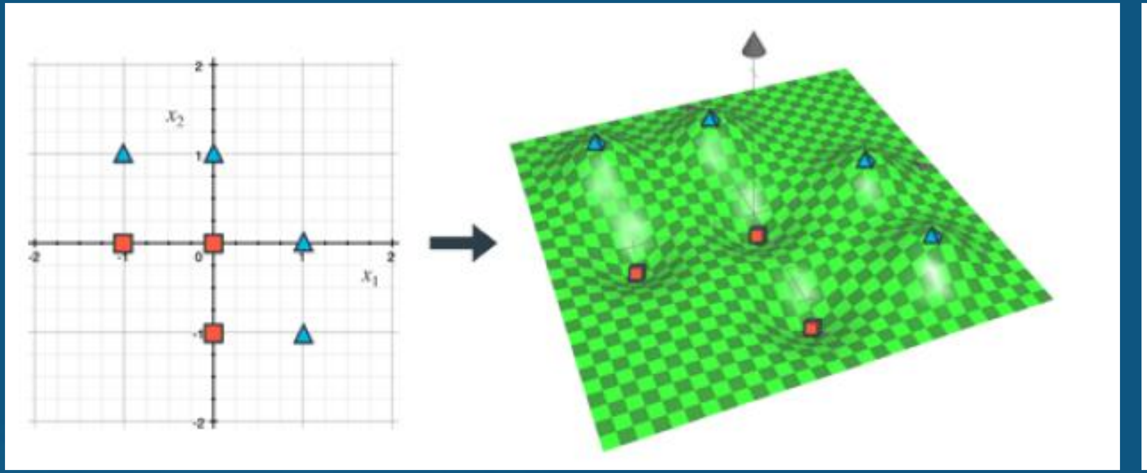
Radial basis Function (RBF) kernel

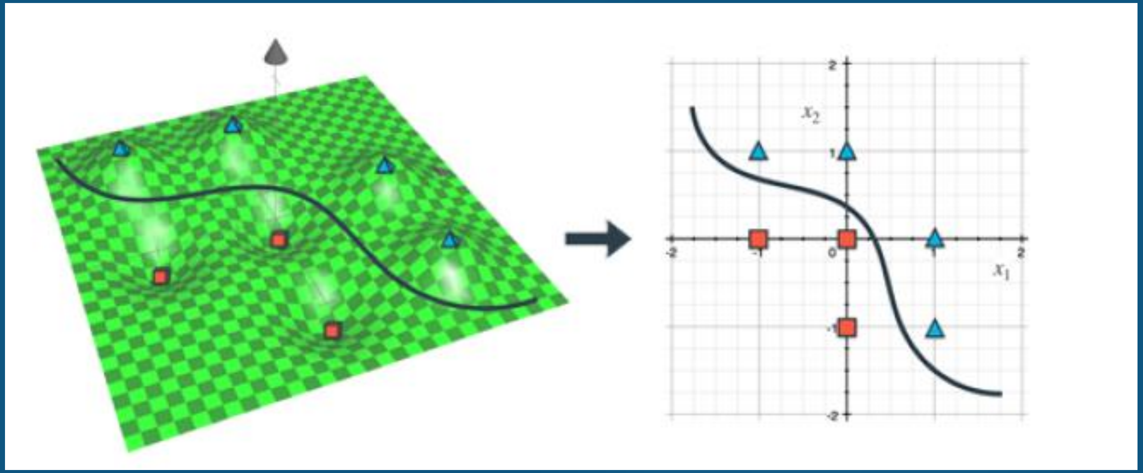


Imagine if you had a point in the plane, and the plane was like a blanket. Then you pinch the blanket at that point, and raise it. This is how a *radial basis function* looks like.

We can raise the blanket at any point we like, and that gives us one different radial basis function. The *radial basis function kernel* (also called rbf kernel) is precisely the set of all radial basis functions for every point in the plane.

Radial basis Function (RBF) kernel

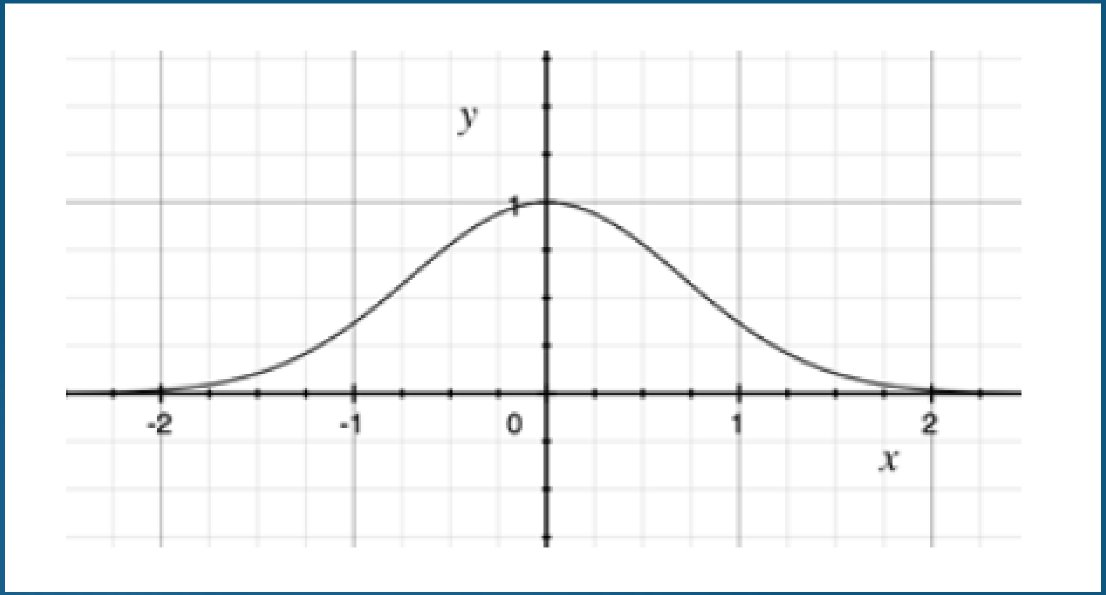




Lift the plane at every triangle, and push it down at every square. Then simply draw a plane at height 0, and intersect it with our surface. This is the same as looking at the curve formed by the points at height 0.

Imagine if there is a landscape with mountains and with the sea. The curve will correspond to the coastline, namely, where the water and the land meet. This gives us the curves (when we project everything back to the plane), and we obtain our desired classifier.

Radial basis Function (RBF) kernel

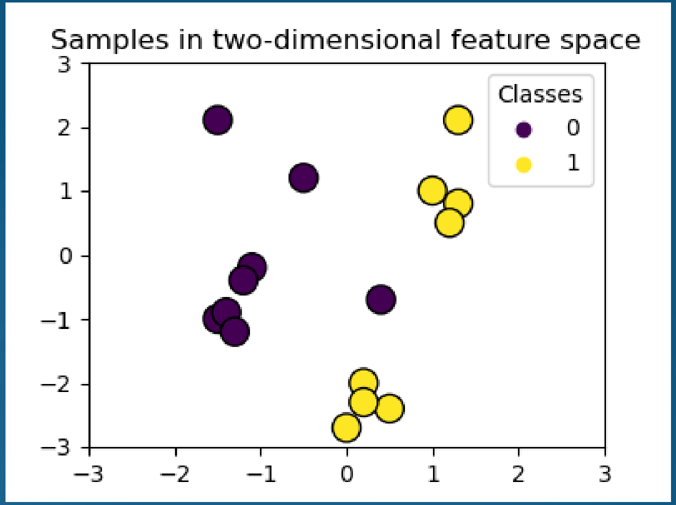


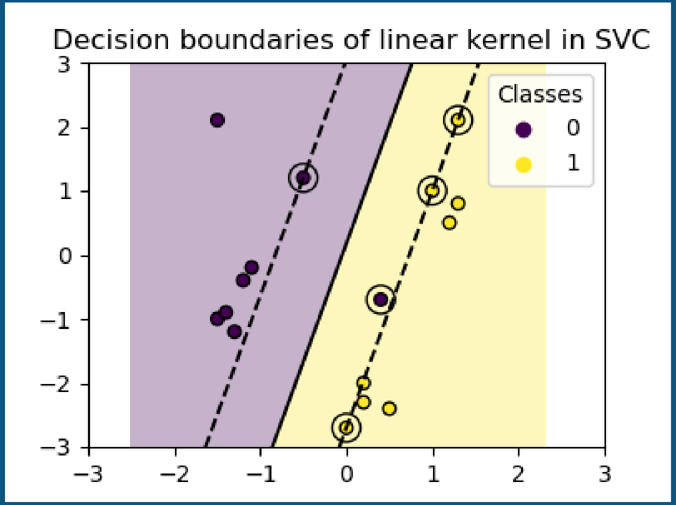
1. One of the simplest radial basis function has the formula: $y = e^{-x^2}$

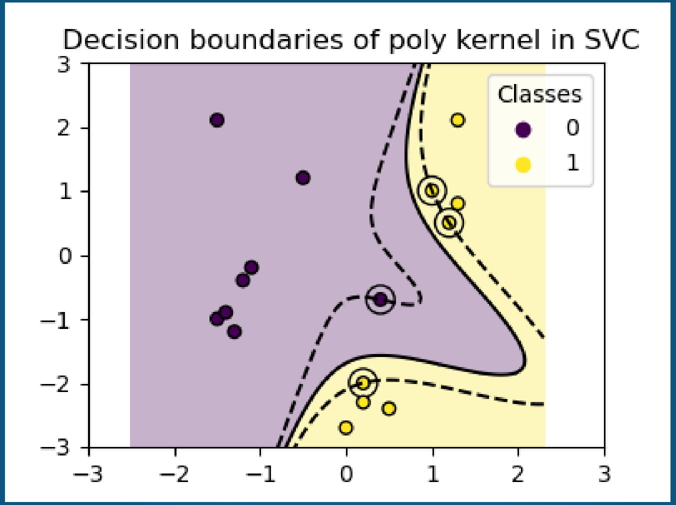
2. Notice that this bump happens at 0. If we wanted it at any different point, say p, we simply translate the formula to $y = e^{-(x-p)2}$

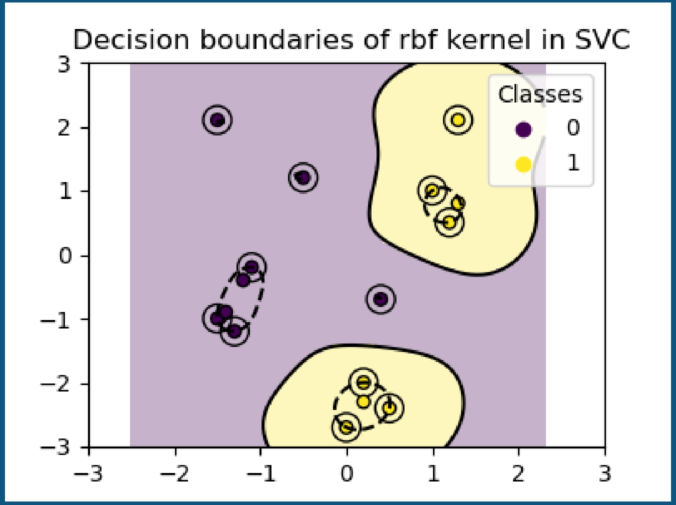
3. Thus, if we want to obtain the radial basis function centered at the point 5: $y = e^{-(x-5)2}$

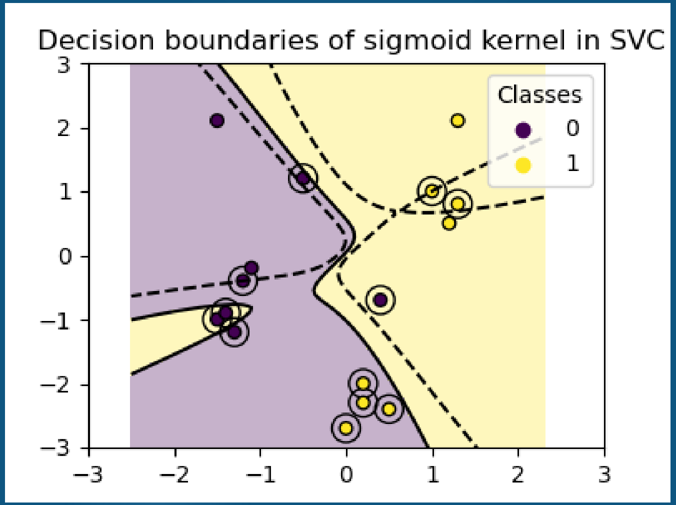
DIFFERENT KERNELS











Thank You