



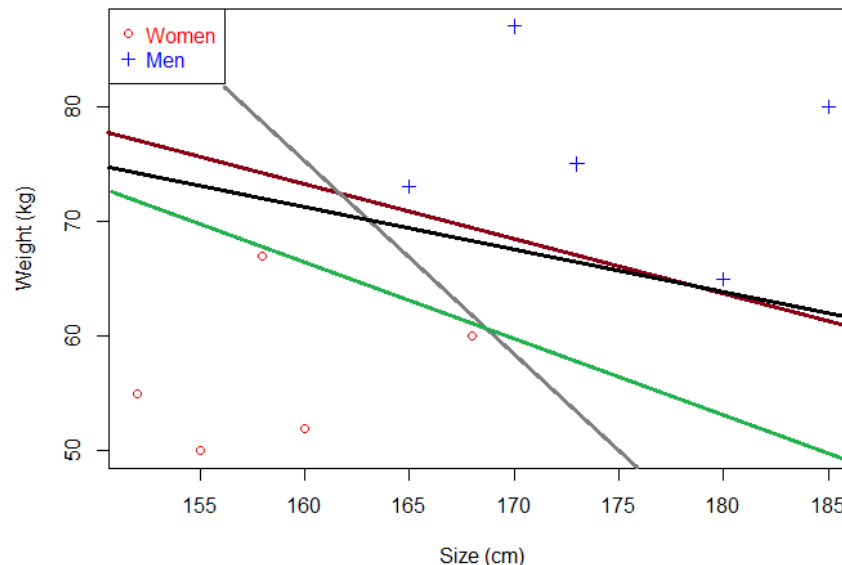
# Support Vector Machine

Department of Computer Languages and Systems

# Introduction

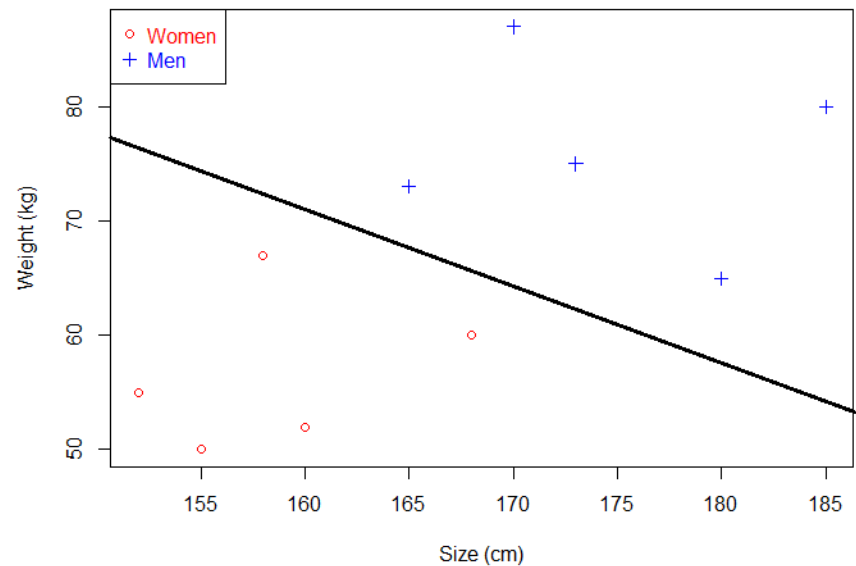
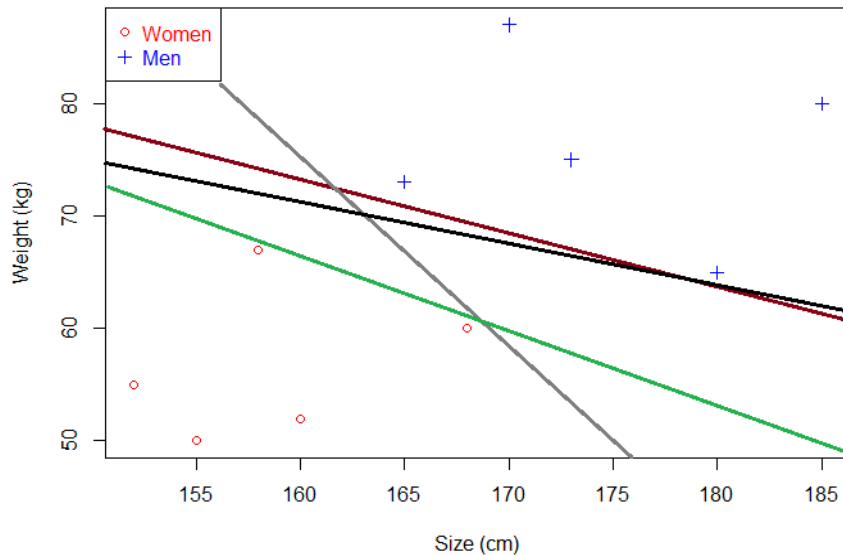
A support vector machine (SVM) tries to find a **hyperplane** that best separates the two classes ...

but there are many (infinite) separating hyperplanes, which one does it select?



# Introduction

We will try to select the hyperplane that maximizes the distance (margin) to the closest samples of each class

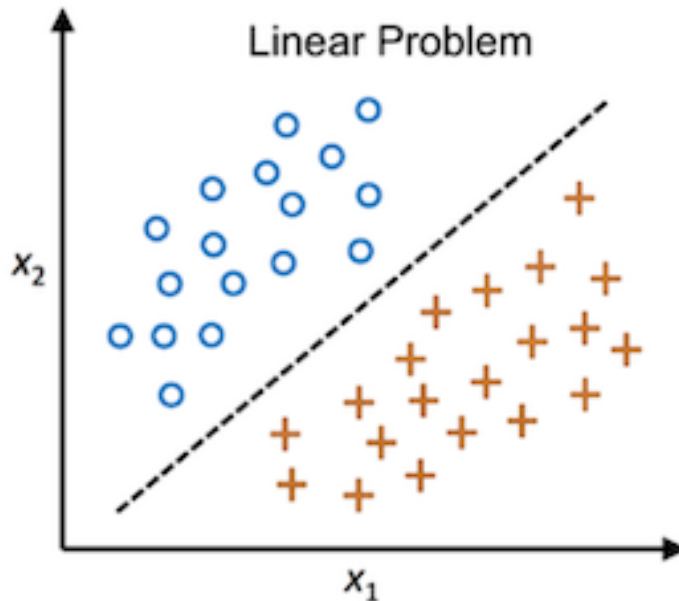


The optimal separation hyperplane is equidistant from the closest example of each class

# Types of SVM algorithms

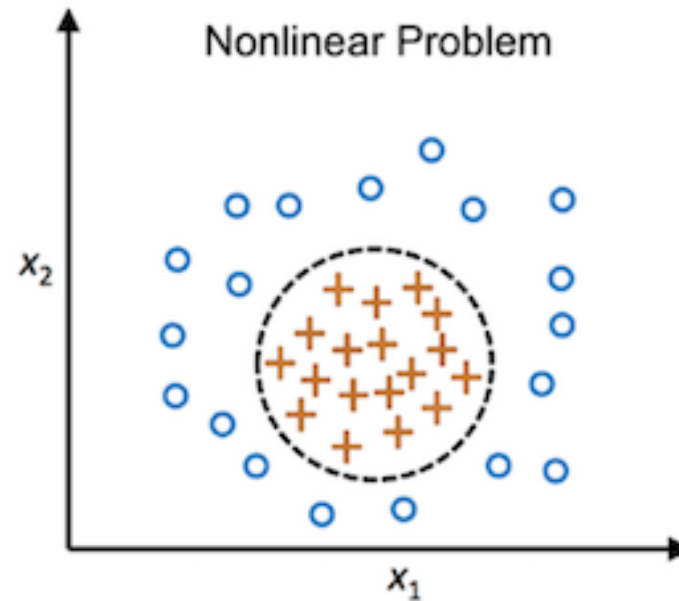
## Linear SVM

When the data is perfectly linearly separable



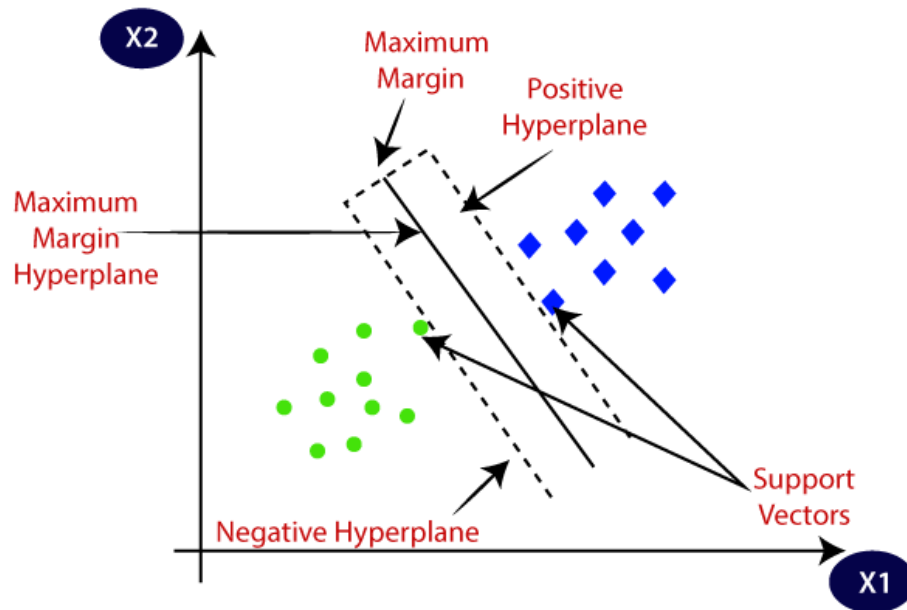
## Non-linear SVM

When the data is not linearly separable, we have to use some kernel



# What is a margin?

it is the distance between the hyperplane and the samples closest to the hyperplane (**support vectors**).



the optimal hyperplane  
will be the one with the  
largest margin!



maximum margin  
=  
minimum loss

# How to find the biggest margin?

Given a data set  $\mathcal{D}$  with  $n$   $d$ -dimensional vectors  $\mathbf{x}_i$ , each associated with a value  $y_i$  to indicate if  $\mathbf{x}_i$  belongs to class -1 or to class +1

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^d, y_i \in \{-1, +1\}\}_{i=1}^n$$

## Steps:

1. Select two hyperplanes that separate the data with no points between them
2. Maximize their distance (the margin)

The region bounded by the two hyperplanes will be the largest possible margin!

# Step 1: Select two hyperplanes ...

Given a hyperplane  $H_0$  separating the data set and satisfying:

$$H_0: \mathbf{w}^T \mathbf{x} + b = 0$$

- we can select two others hyperplanes  $H_1$  and  $H_2$  that also separate the data and have the following equations:

$$H_1: \mathbf{w}^T \mathbf{x} + b = \delta \quad \text{and} \quad H_2: \mathbf{w}^T \mathbf{x} + b = -\delta$$

- to simplify the problem, we can set  $\delta = 1$ :

$$H_1: \mathbf{w}^T \mathbf{x} + b = 1 \quad \text{and} \quad H_2: \mathbf{w}^T \mathbf{x} + b = -1$$

where  $\mathbf{w}$  is the weight vector (to define the orientation of the hyperplane)  
and  $b$  is the bias

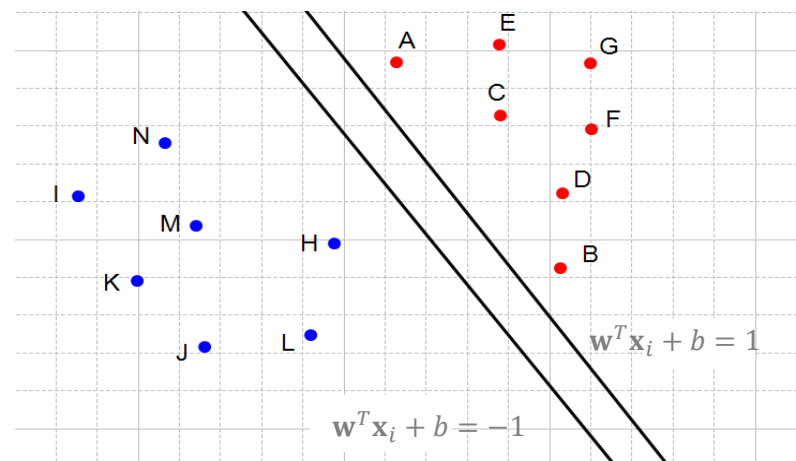
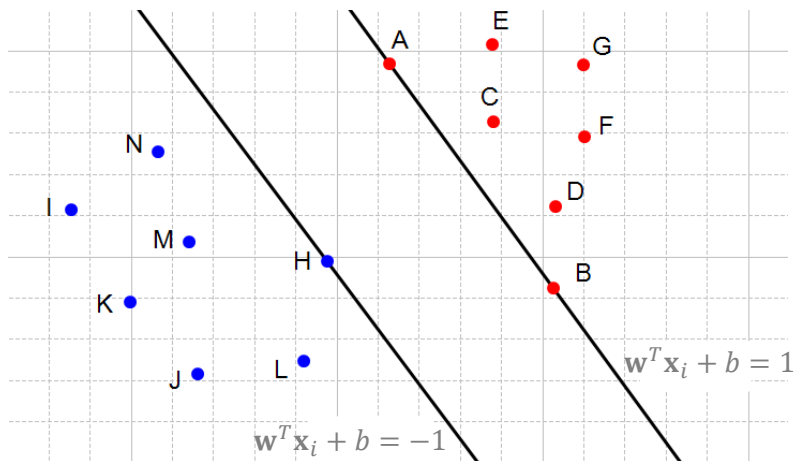
# Step 1 (cont.)

... but not any hyperplane is valid, we will only select those that meet the following two constraints for all  $\mathbf{x}_i$ :

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \text{ for } y_i = 1$$

or

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1 \text{ for } y_i = -1$$



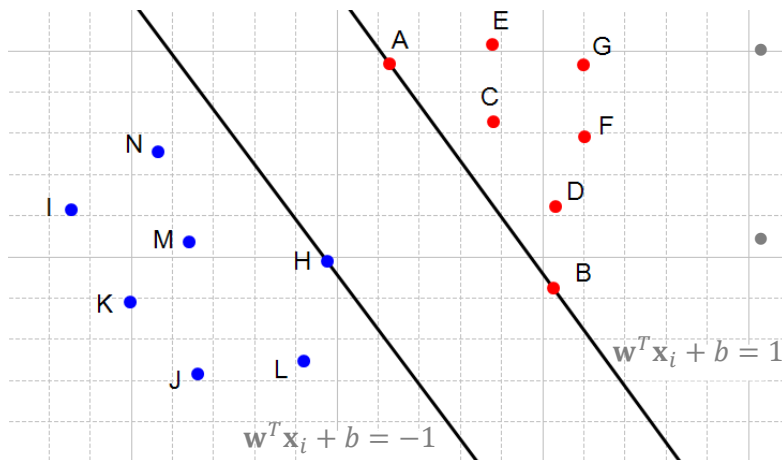


# Step 1 (cont.)

**Understanding the constraints:** we need to verify the points do not violate the constraints

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \text{ for } y_i = 1$$

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1 \text{ for } y_i = -1$$



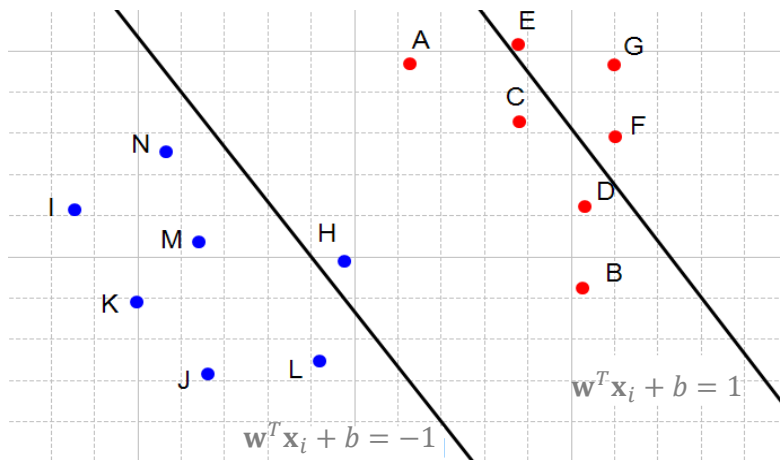
- when  $\mathbf{x}_i = A$ , we see that the point is on the hyperplane, so  $\mathbf{w}^T \mathbf{x}_i + b = 1$  and the constraint is respected
- when  $\mathbf{x}_i = C$ , we see that the point is above the hyperplane, so  $\mathbf{w}^T \mathbf{x}_i + b > 1$  and the constraint is respected

# Step 1 (cont.)

**Understanding the constraints:** we need to verify the points do not violate the constraints

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \text{ for } y_i = 1$$

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1 \text{ for } y_i = -1$$



when a constraint is not satisfied  
(i.e. there are points between the  
two hyperplanes) means that we  
cannot select these two hyperplanes

## Step 1 (cont.)

Previous equations can be combined into a single constraint:

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

## Step 2: Maximize the margin

Recall that the shortest distance between a point  $\mathbf{x}_i$  and a hyperplane  $\mathbf{w}^T \mathbf{x}_i + b = 0$  can be calculated as:

$$d_i = \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|}$$

where  $\|\mathbf{w}\|$  represents the Euclidean norm of the vector  $\mathbf{w}$ .

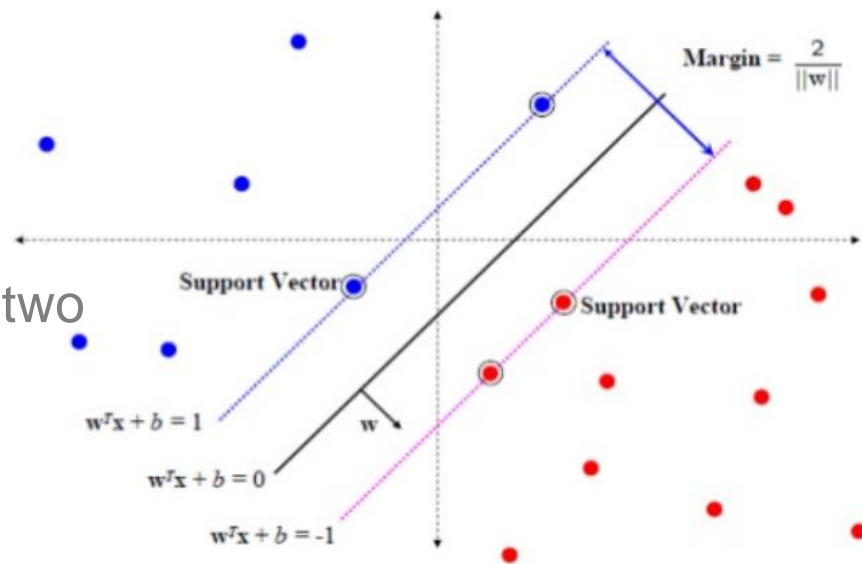
## Step 2 (cont.)

The distance of the closest points (i.e., support vectors) to the optimal hyperplane will be

$$\frac{|\pm 1|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

and therefore the margin (the closest distance between the two classes) will be:

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



## Step 2 (cont.)

We want to maximize the margin  $\rho$ :

$$\max_{\mathbf{w}, b} \frac{2}{\|\mathbf{w}\|}$$

which is equivalent to minimizing the following objective function

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \quad \forall i = 1, \dots, n$$

## Step 2 (cont.)

Note that the objective function is quadratic and the constraints are linear for the parameters  $\mathbf{w}$  and  $b$ , so we have a **quadratic optimization problem**:

- we can construct a dual problem where a **Lagrange multiplier  $\lambda_i$**  is associated with each constraint in the primal problem → **it reduces the problem of  $m$  variables and  $k$  constraints to another dual of  $(m + k)$  variables without constraints**

## Step 2 (cont.)

First, an unconstrained optimization problem is constructed using the Lagrangian function:

$$\min_{\mathbf{w}, b} L_p = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \lambda_i (y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1) \quad (1)$$

subject to  $\lambda_i \geq 0, \forall i$

Solving this optimization problem is still computationally expensive because it involves a large number of parameters:  $\mathbf{w}$ ,  $b$ ,  $\lambda_i$



## Step 2 (cont.)

Second, we can apply the Karush-Kuhn-Tucker conditions (the property of derivatives at min = 0)::

$$\frac{\partial L_p}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^n \lambda_i y_i \mathbf{x}_i = 0$$

$$\frac{\partial L_p}{\partial b} = \sum_{i=1}^n \lambda_i y_i = 0$$

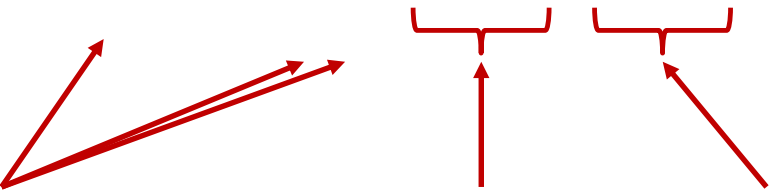
So,

$$\mathbf{w} = \sum_{i=1}^n \lambda_i y_i \mathbf{x}_i \quad (2) \quad \sum_{i=1}^n \lambda_i y_i = 0 \quad (3)$$

## Step 2 (cont.)

The **dual problem** can be obtained substituting (2) and (3) in (1) to express the Lagrangian function only in terms of the Lagrange multipliers:

$$\max_{\lambda_i} L_D(\lambda_i) = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j (\mathbf{x}_i^T \cdot \mathbf{x}_j) \quad (4)$$



sums over all training examples      scalars      dot products

# Prediction

After finding the parameters, a new sample  $\mathbf{u}$  can be classified looking at the sign of:

$$f(\mathbf{u}) = \text{sign}(\mathbf{w} \cdot \mathbf{u} + b) = \text{sign} \left( \sum_{i=1}^n \lambda_i y_i \mathbf{x}_i \cdot \mathbf{u} + b \right)$$

If  $f(\mathbf{u}) = 1$ , then the sample is classified in the positive class; otherwise, it is classified in the negative class

# SVM when outliers

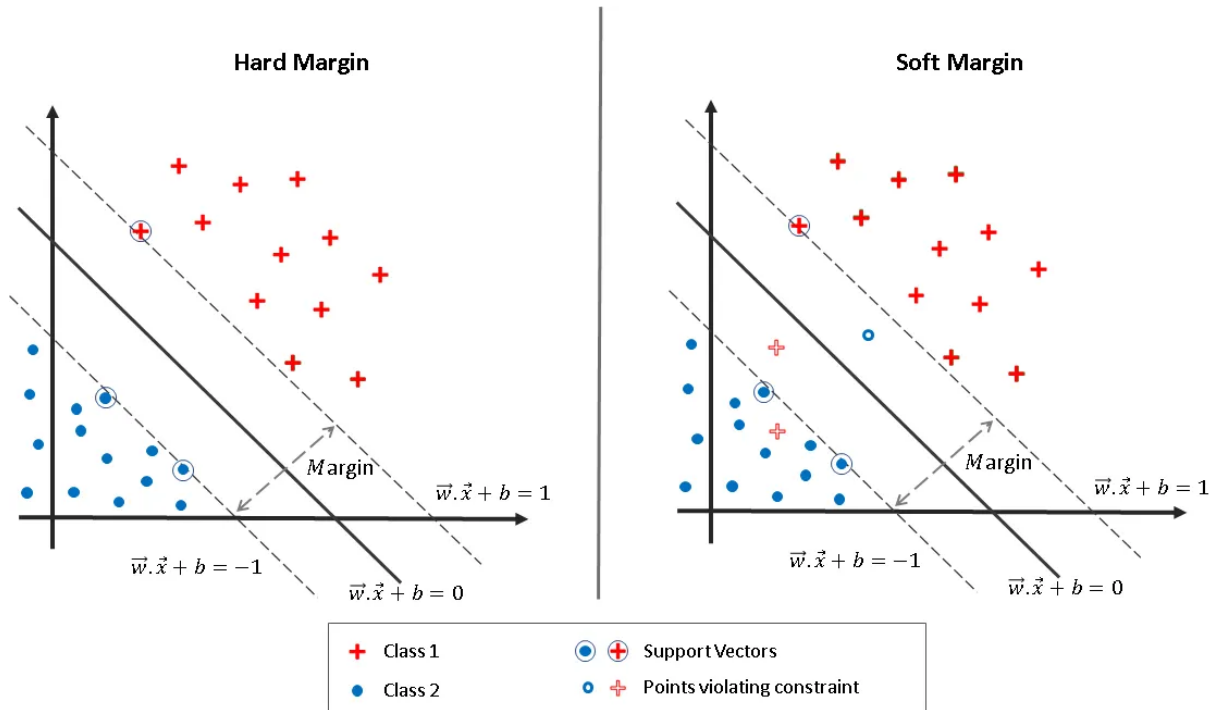
## General situation in real-life applications:

- data is almost linearly separable or non-linearly separable
- there are outliers in the data

## How to tackle the outliers problem?

to modify the optimization function in such a way that it allows few misclassifications (some points are inside or on the wrong side of the margin) by introducing two hyperparameters: a regularization factor  $C$  and a slack variable  $\xi_i$  for each data point  $\mathbf{x}_i \rightarrow$  **SOFT MARGIN SVM**

# Hard margin vs soft margin



- In hard margin SVM, the points in a class must not lie within the two support hyperplanes:

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

# Regularization factor

It tries to balance the trade-off between maximizing the margin and minimizing the misclassifications

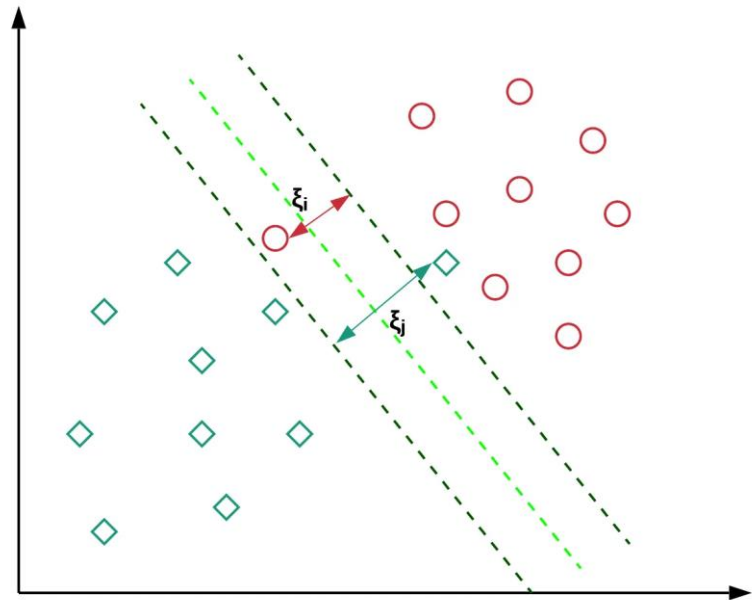
- when  $C$  is small, misclassifications are given less importance and focus is more on maximizing the margin
- when  $C$  is large, the focus is more on avoiding errors at the expense of keeping the margin small

However, not all mistakes are equal. Data points that are far away on the wrong side of the decision boundary should incur more penalty as compared to those that are closer

# Slack variables

A **slack variable**  $\xi_i \geq 0$  ( $i = 1, \dots, n$ ) is the distance of  $\mathbf{x}_i$  from its class's margin if  $\mathbf{x}_i$  is on the wrong side of the margin; otherwise zero (this corresponds to linearly separable samples)

thus the points that are far away from the margin on the wrong side would get more penalty



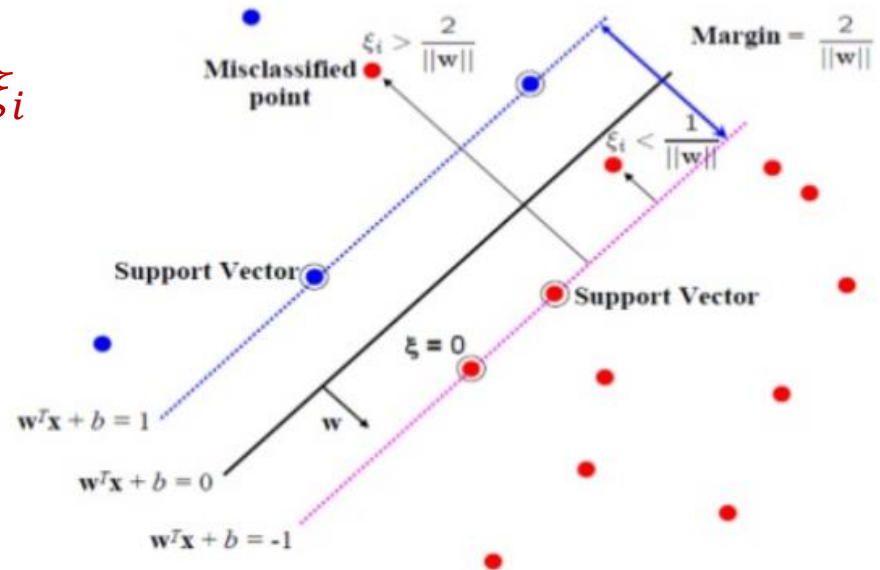
# Soft margin SVM: formulation

Now each data point needs to satisfy the following constraint:

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \text{ for } i = 1, \dots, n$$

and the objective function to minimize will be:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i$$





# Soft margin SVM: formulation

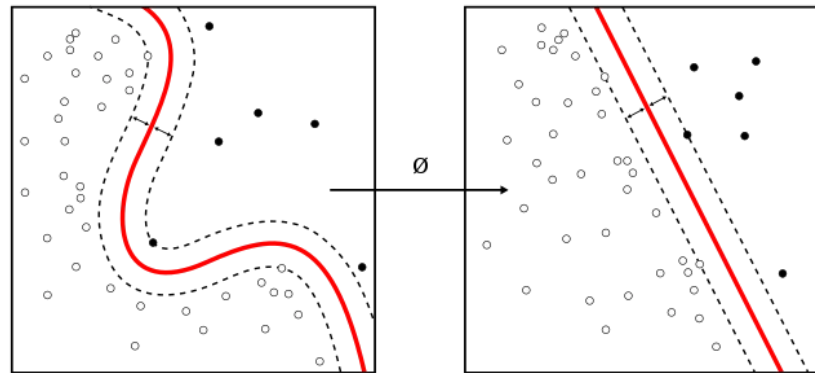
As in the case of linearly separable data, the optimization problem can be transformed into its dual form:

$$L_p = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i + \sum_i \lambda_i (y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i)$$

# Non-linear SVM: the kernel trick

Solution to non-linear problems:

1. Map the data onto another feature space to convert the non-linear data to linear data:  $\mathbf{x}_i \rightarrow \phi(\mathbf{x}_i)$



2. Solve a linear SVM in the new feature space
3. The dual problem is the same, but using a kernel matrix  $K$  with elements  $k(i, j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = (\phi(\mathbf{x}_i), \phi(\mathbf{x}_j))$  in (4)

# Non-linear SVM: the kernel trick (ii)

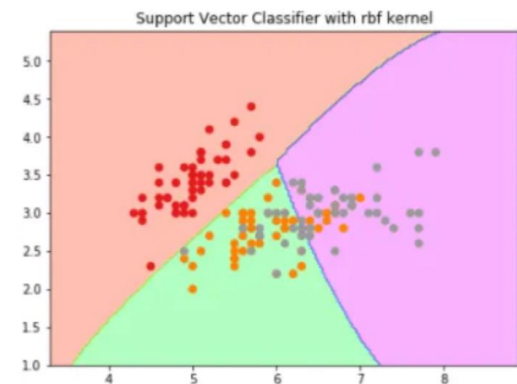
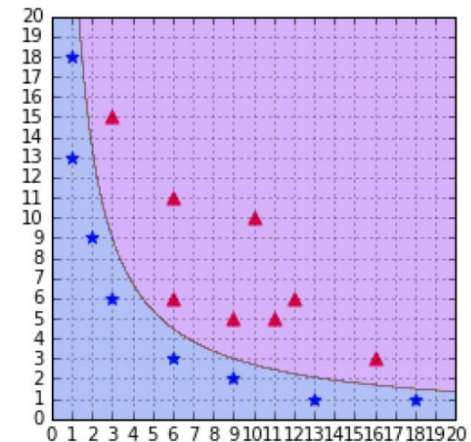
- Polynomial kernel of degree  $d$ :

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \cdot \mathbf{x}_j + 1)^d$$

- RBF kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}$$

where  $\gamma > 0$ , and  $\|\mathbf{x}_i - \mathbf{x}_j\|$ .



# Non-linear SVM: the kernel trick (iii)

- A special case of RBF kernel is  $\gamma = 1/2\sigma^2 \rightarrow$  Gaussian kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$$

where  $\sigma$  is the variance.

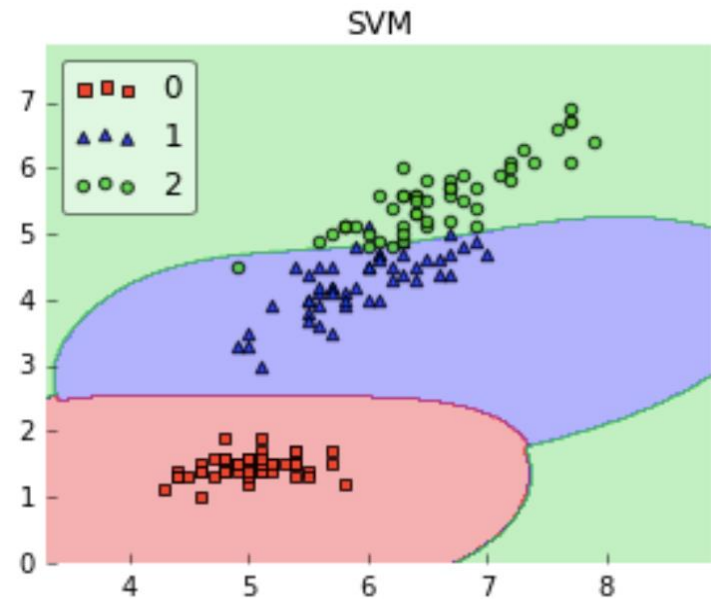
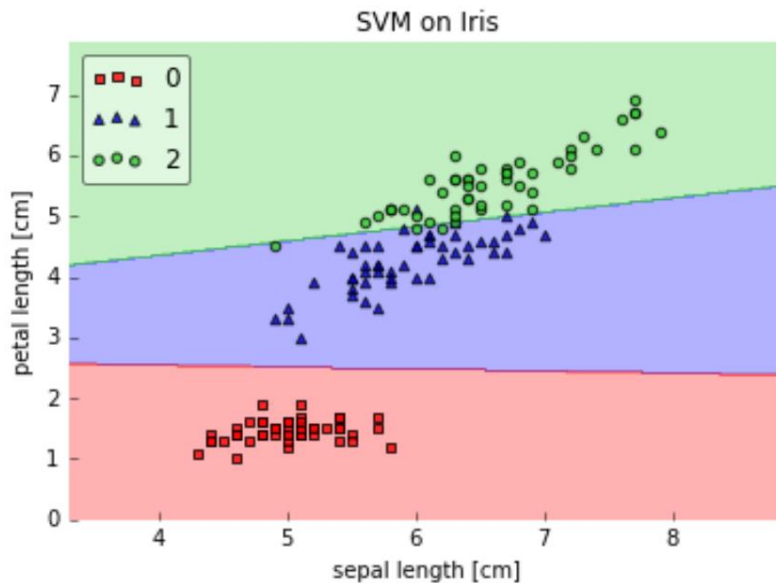
# Non-linear SVM: the kernel trick (iv)

## How to choose the right kernel?

- if the data set is linearly separable, then you must opt for a linear function because it is very easy to use and the complexity is much lower compared to other kernel functions
- so, start with a hypothesis that data is linearly separable and choose a linear function. Then use an RBF kernel function (polynomial kernel is rarely used due to poor efficiency)
- if linear and RBF both give approximately similar results, choose the linear SVM. Otherwise, choose the RBF kernel

# Non-linear SVM: the kernel trick (v)

An example with similar results for both linear and RBF functions



# Non-linear SVM: the kernel trick (vi)

An example where a linear SVM gives poor results, while the RBF kernel makes a correct decision boundary

