



# Machine Learning

## Support Vector Machine (SVM) Part I: Motivation

Dr. Mehran Safayani

safayani@iut.ac.ir

safayani.iut.ac.ir



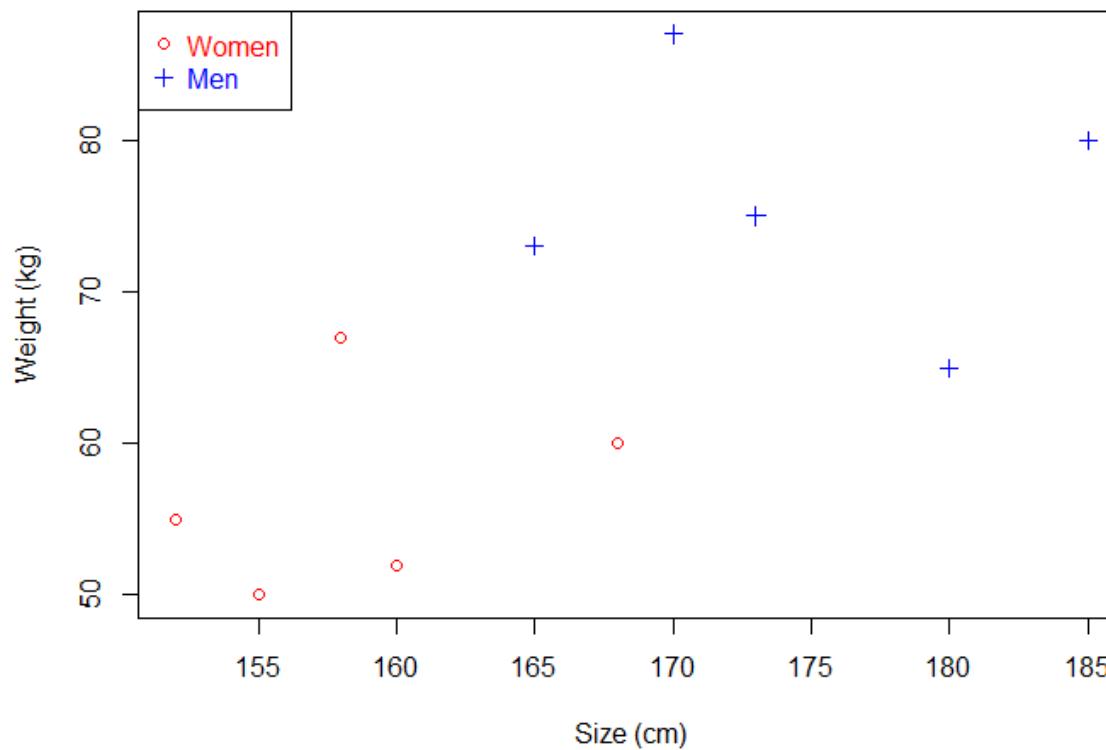
<https://www.aparat.com/mehran.safayani>



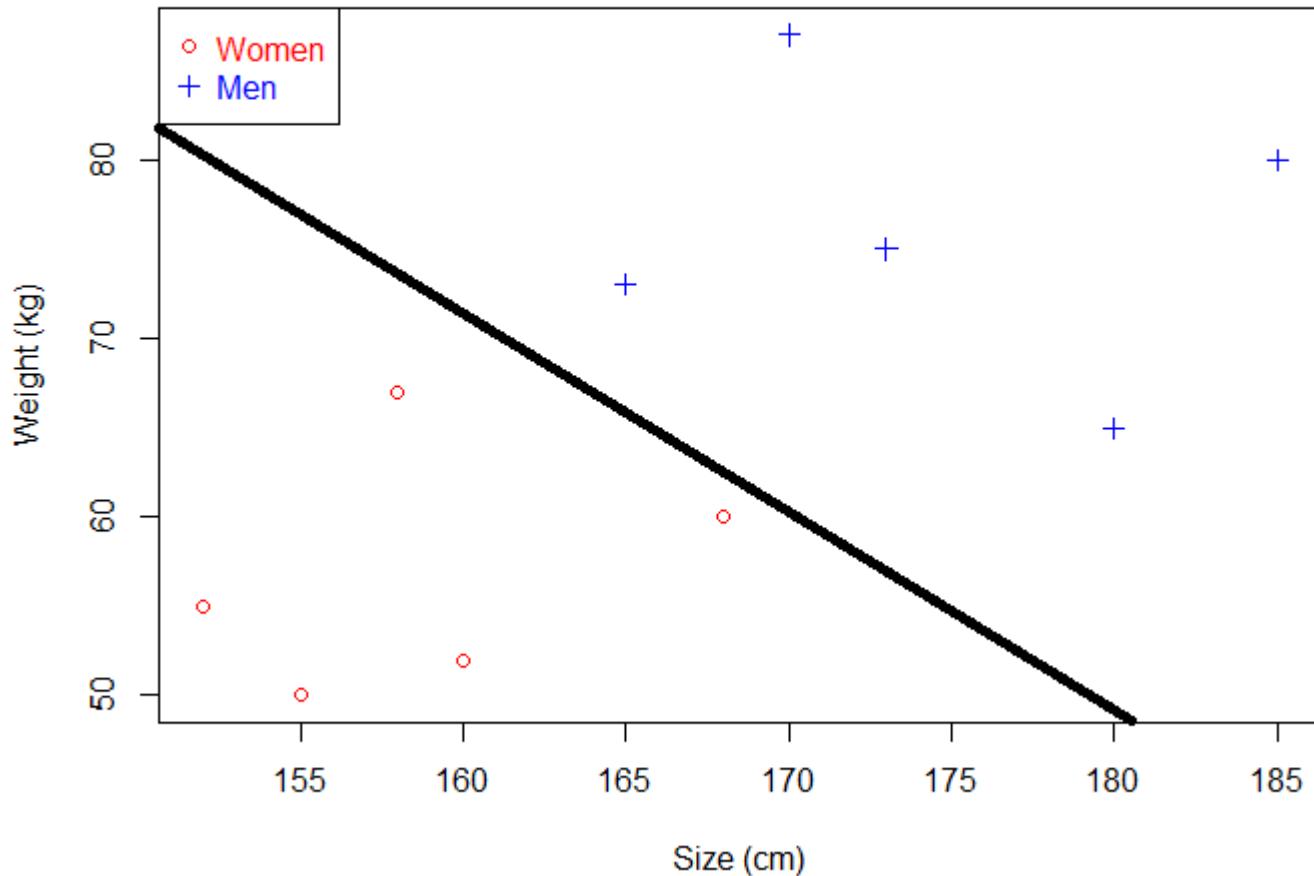
[https://github.com/safayani/machine\\_learning\\_course](https://github.com/safayani/machine_learning_course)



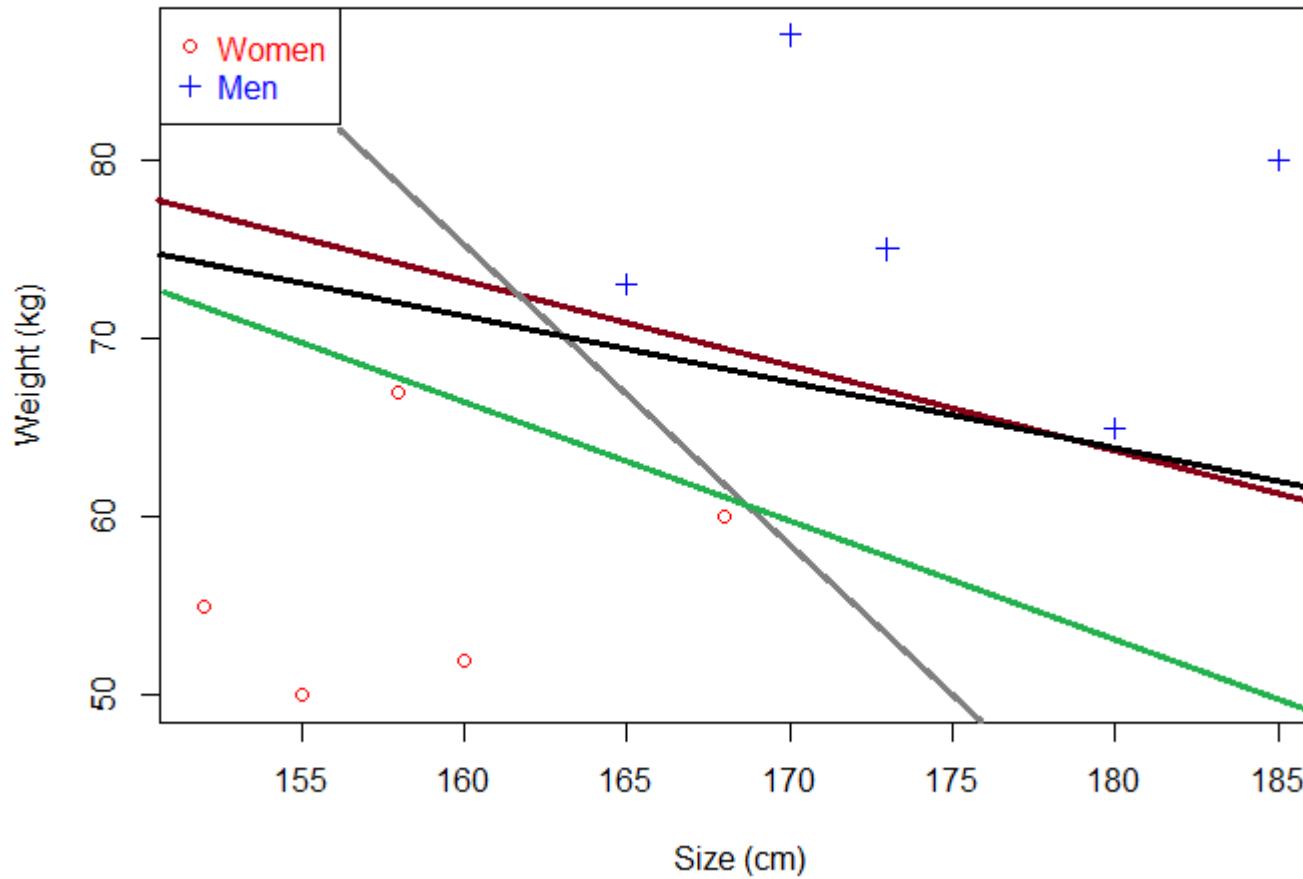
# SVM: Motivation



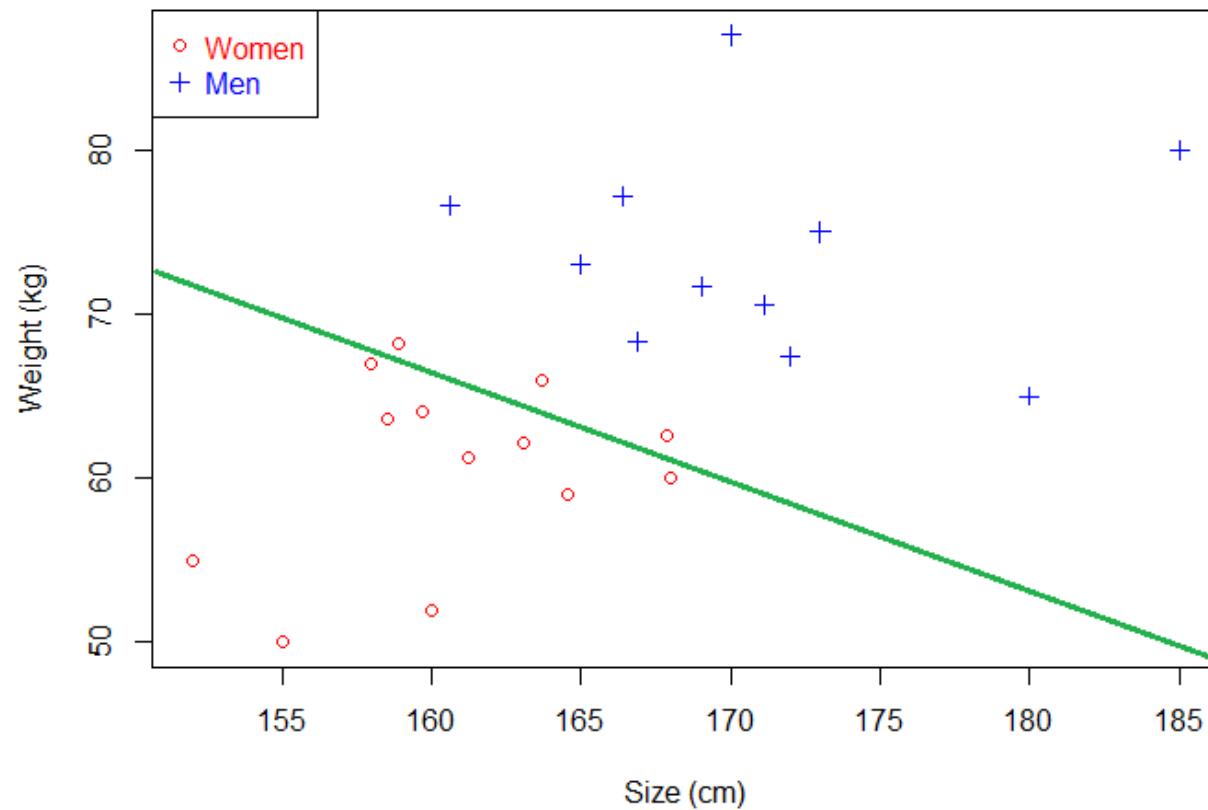
# separating hyperplane



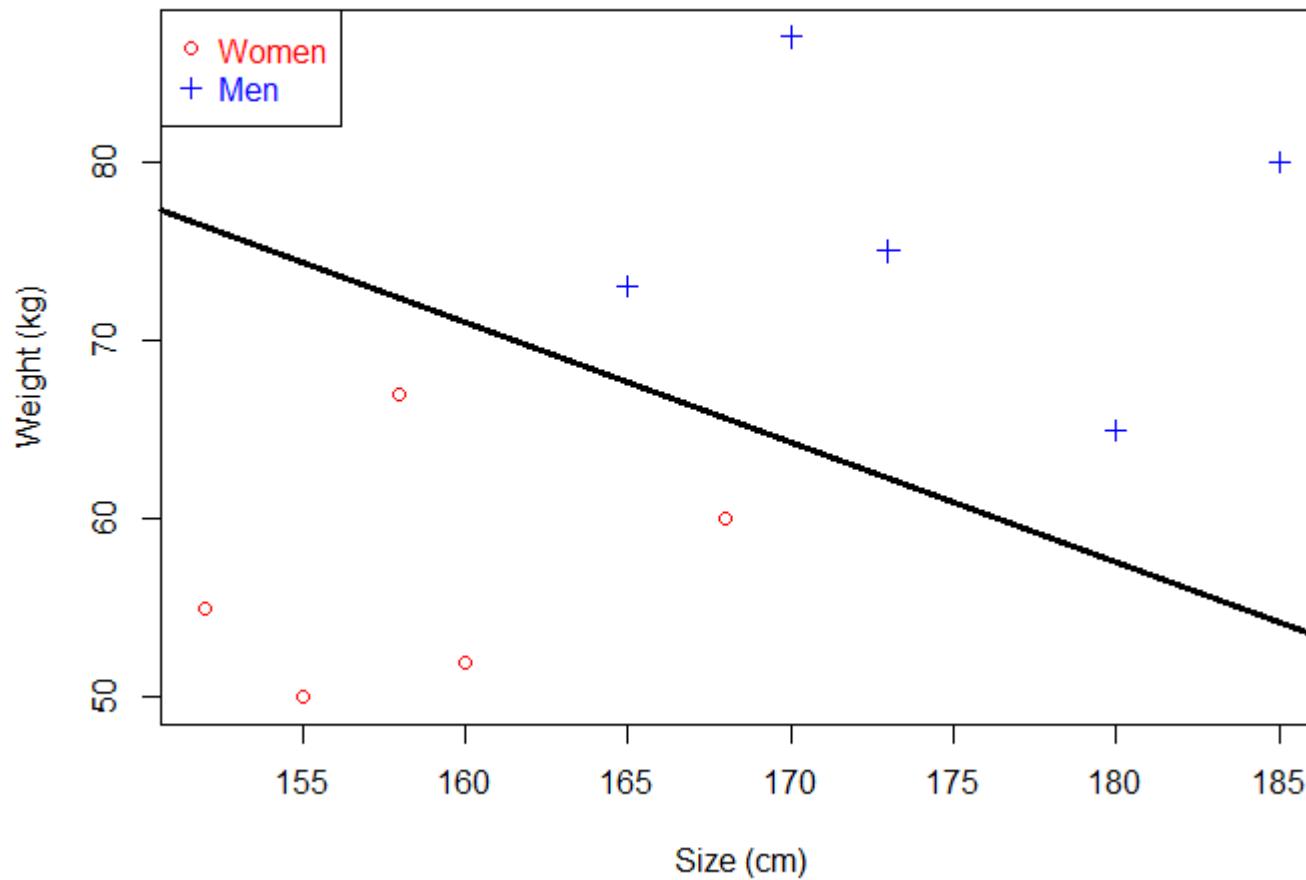
# What is the *optimal* separating hyperplane?



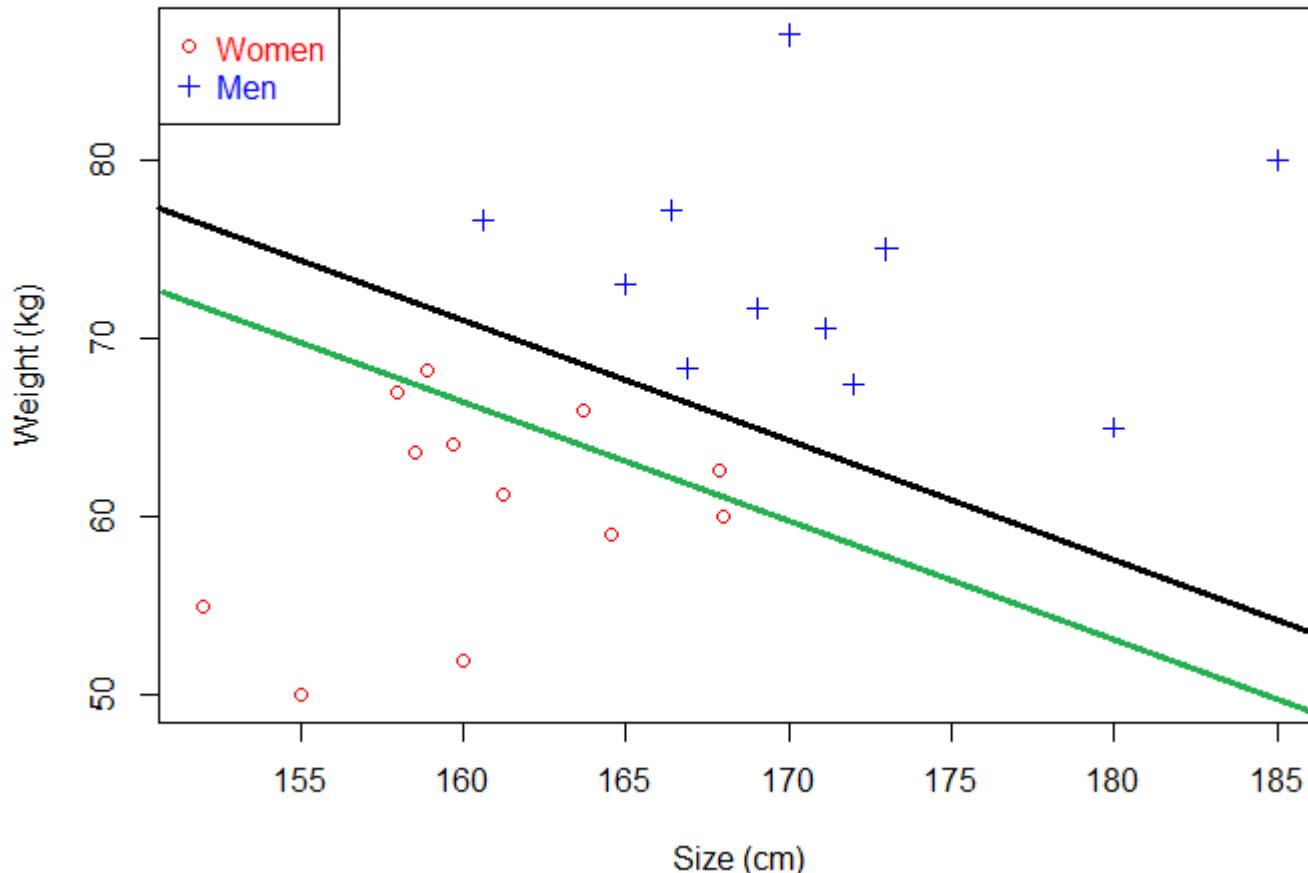
*This hyperplane does not generalize well*

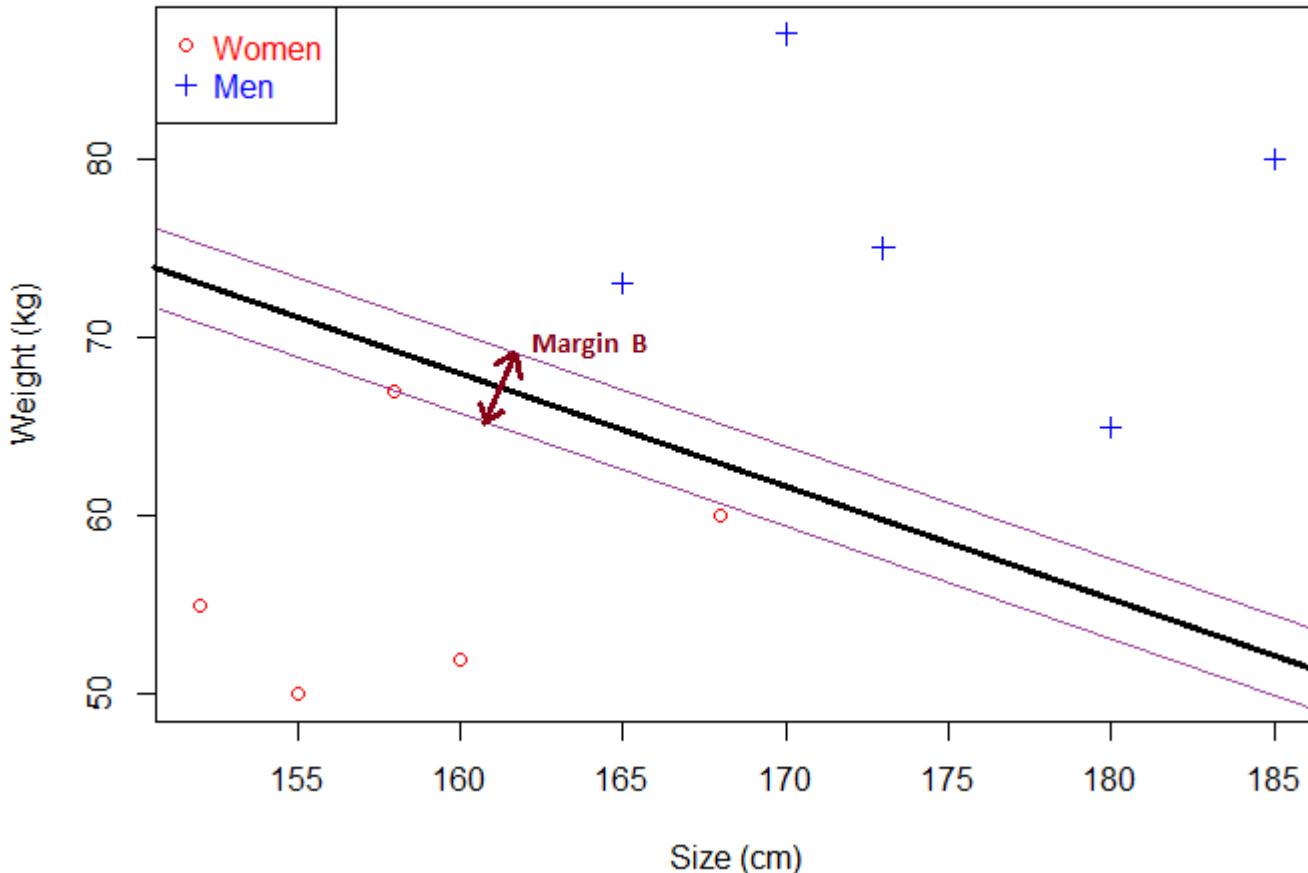


# Optimal hyperplane

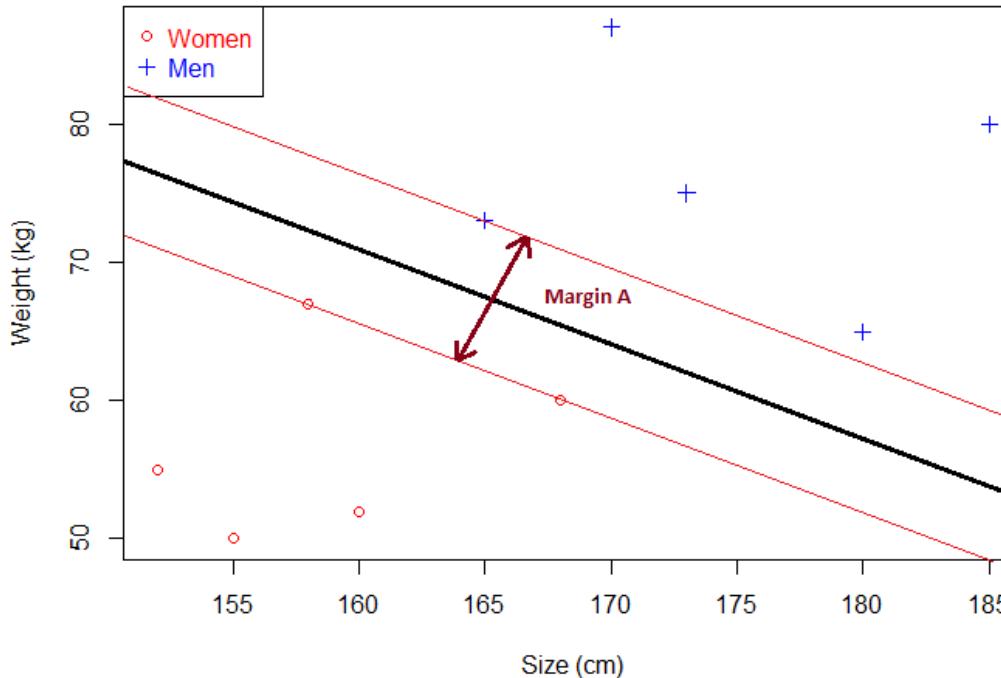


*The black hyperplane classifies more accurately than the green one*





# *The margin of our optimal hyperplane*



- the optimal hyperplane will be the one with the biggest margin.



# Machine Learning

Support Vector Machine (SVM)

Part II: Reminder of linear algebra concepts

Dr. Mehran Safayani

safayani@iut.ac.ir

safayani.iut.ac.ir

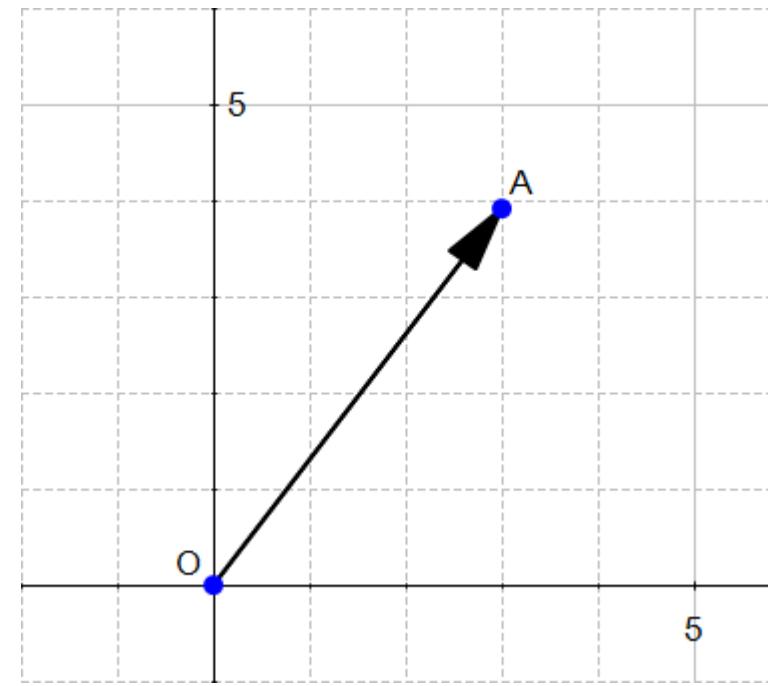
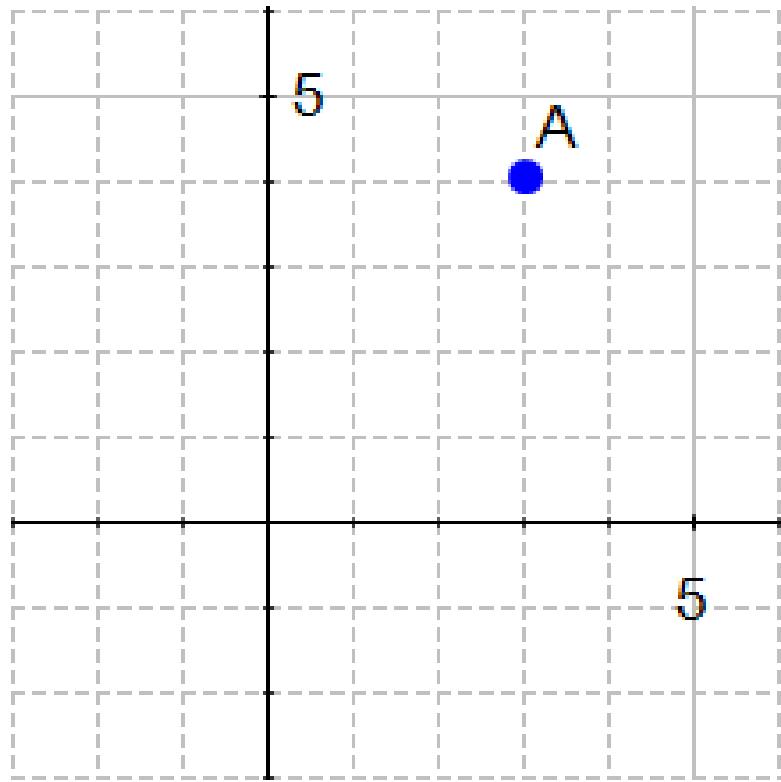


<https://www.aparat.com/mehran.safayani>



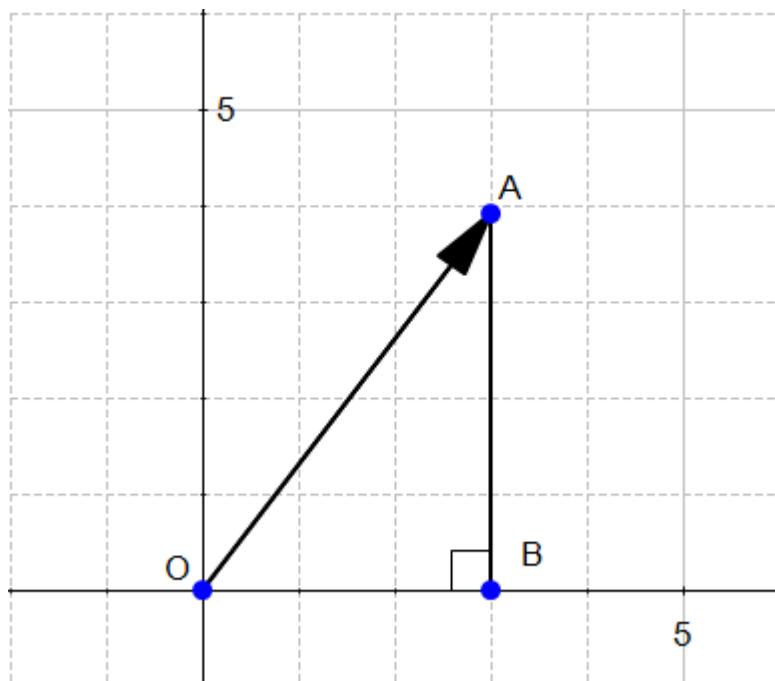
[https://github.com/safayani/machine\\_learning\\_course](https://github.com/safayani/machine_learning_course)





# The magnitude

$$OA^2 = OB^2 + AB^2$$



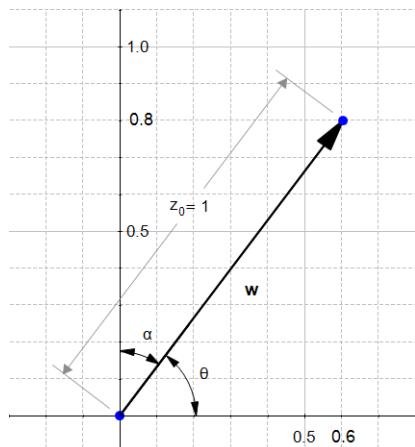
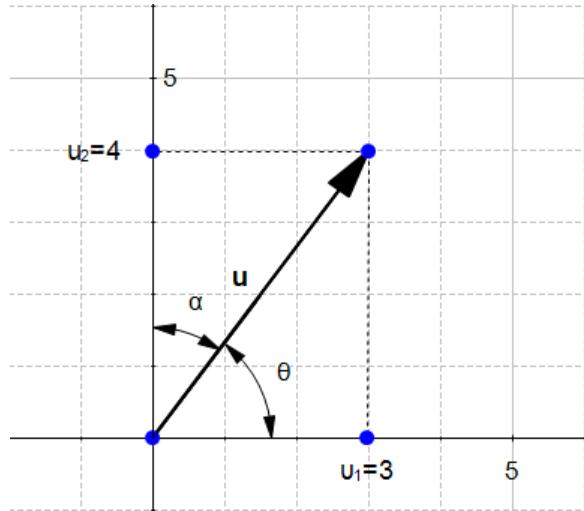
$$OA^2 = 3^2 + 4^2$$

$$OA^2 = 25$$

$$OA = \sqrt{25}$$

$$\|OA\| = OA = 5$$

# The direction



$\vec{p} = t \vec{u} = (tu_1, tu_2)$   
 $\|\vec{p}\| = t \|\vec{u}\|$   
 $w = \left( \frac{u_1}{\|u\|}, \frac{u_2}{\|u\|} \right)$   
 $w = \left( \frac{p_1}{\|\vec{p}\|}, \frac{p_2}{\|\vec{p}\|} \right) = \left( \frac{tu_1}{t\|\vec{u}\|}, \frac{tu_2}{t\|\vec{u}\|} \right) = \left( \frac{u_1}{\|u\|}, \frac{u_2}{\|u\|} \right) = w$

Definition : The **direction** of a vector  $\mathbf{u}(u_1, u_2)$  is the vector  $\mathbf{w}\left(\frac{u_1}{\|u\|}, \frac{u_2}{\|u\|}\right)$

$$\cos(\theta) = \frac{u_1}{\|\mathbf{u}\|} = \frac{3}{5} = 0.6$$

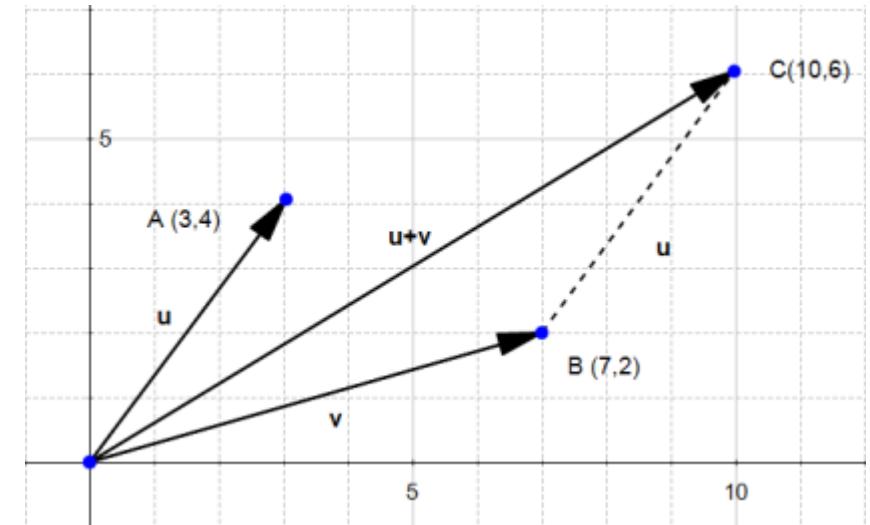
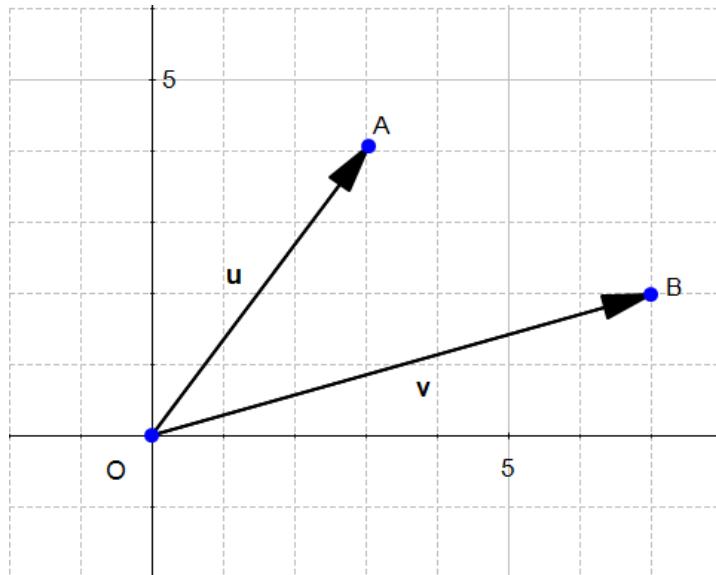
$$\cos(\alpha) = \frac{u_2}{\|\mathbf{u}\|} = \frac{4}{5} = 0.8$$

The direction of  $\mathbf{u}(3, 4)$  is the vector  $\mathbf{w}(0.6, 0.8)$

# The sum of two vectors

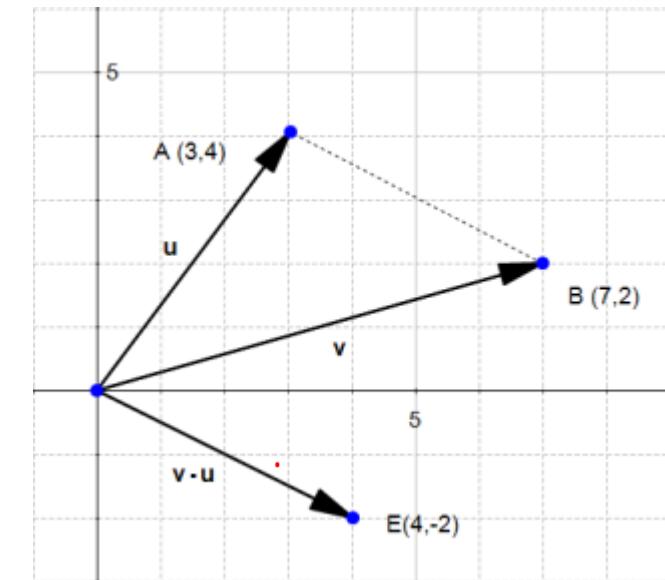
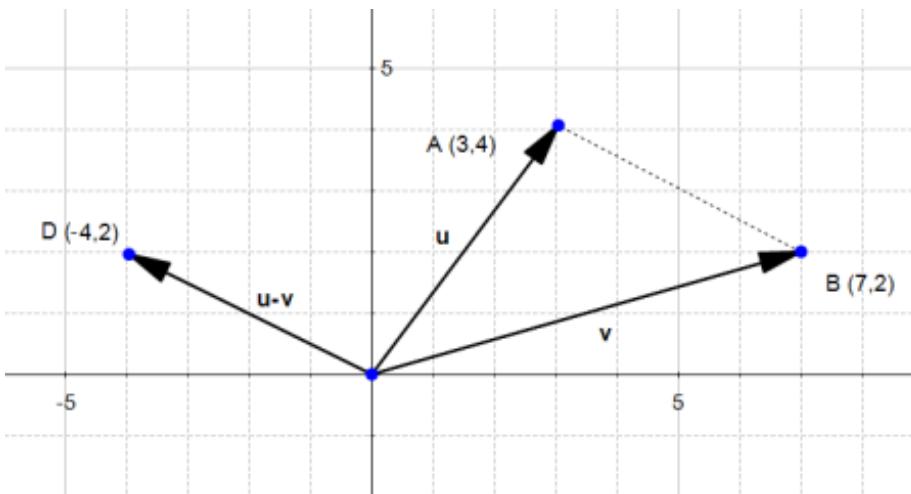
Given two vectors  $\mathbf{u}(u_1, u_2)$  and  $\mathbf{v}(v_1, v_2)$  then :

$$\mathbf{u} + \mathbf{v} = (u_1 + v_1, u_2 + v_2)$$

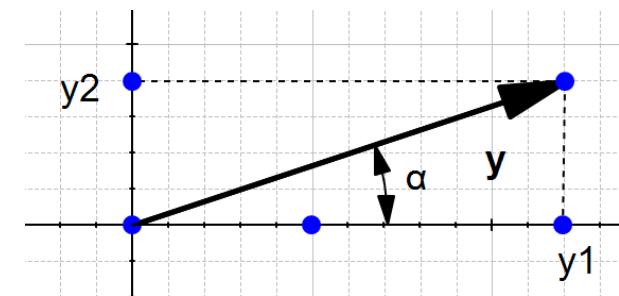
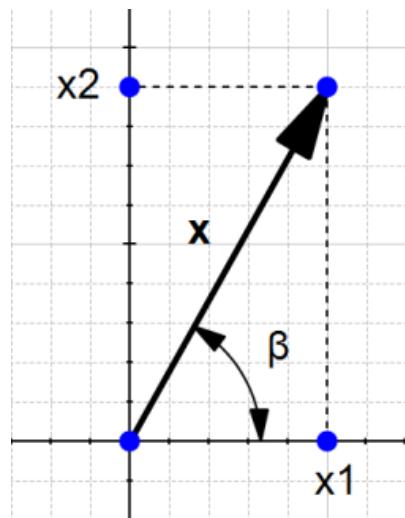
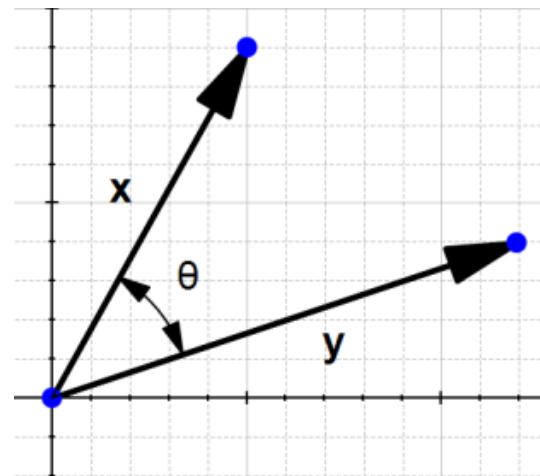


# The difference between two vectors

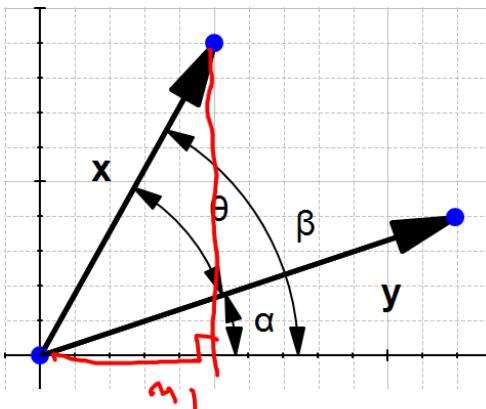
$$\mathbf{u} - \mathbf{v} = (u_1 - v_1, u_2 - v_2)$$



# The dot product



# Dot product



$$\cos(\beta) = \frac{\text{adjacent}}{\text{hypotenuse}} = \frac{x_1}{\|x\|}$$

$$\theta = \beta - \alpha$$

$$\sin(\beta) = \frac{\text{opposite}}{\text{hypotenuse}} = \frac{x_2}{\|x\|}$$

$$\cos(\theta) = \cos(\beta - \alpha) = \cos(\beta)\cos(\alpha) + \sin(\beta)\sin(\alpha)$$

=====

$$\cos(\alpha) = \frac{\text{adjacent}}{\text{hypotenuse}} = \frac{y_1}{\|y\|}$$

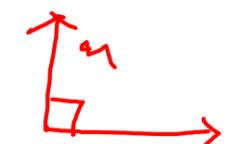
$$\cos(\theta) = \frac{x_1 y_1 + x_2 y_2}{\|x\| \|y\|}$$

$$\sin(\alpha) = \frac{\text{opposite}}{\text{hypotenuse}} = \frac{y_2}{\|y\|}$$

$$\|x\| \|y\| \cos(\theta) = \underline{x_1 y_1 + x_2 y_2}$$

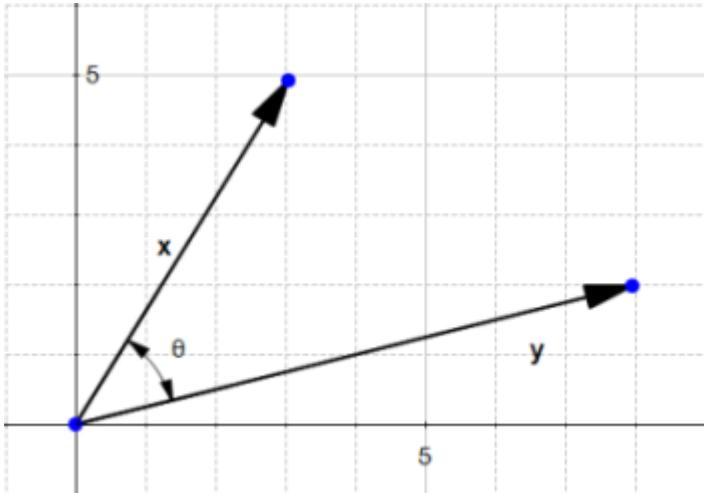
$$\mathbf{x} \cdot \mathbf{y} = x_1 y_1 + x_2 y_2 = \sum_{i=1}^2 (x_i y_i)$$

$$\|x\| \|y\| \cos(\theta) = \mathbf{x} \cdot \mathbf{y}$$



$\cos 90^\circ = 0$   
 $\mathbf{x} \cdot \mathbf{y} = 0$

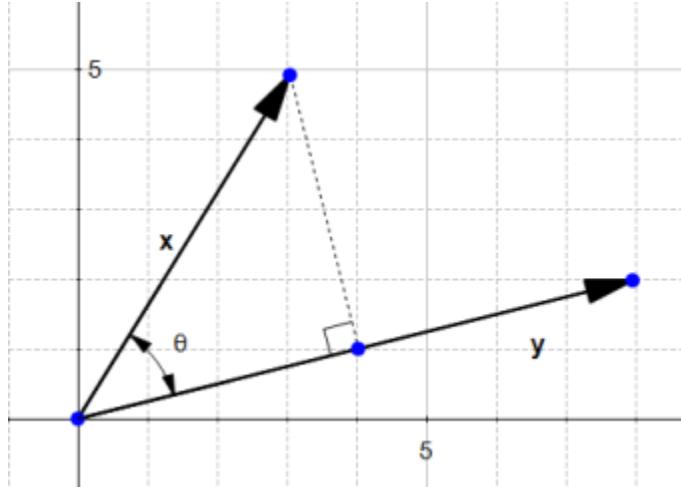
# The orthogonal projection of a vector



$$\cos(\theta) = \frac{\|z\|}{\|x\|}$$

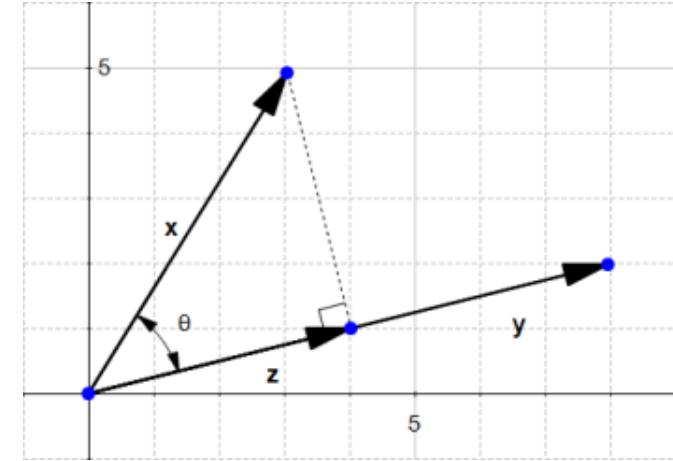
$$\|z\| = \|x\| \cos(\theta)$$

$$\cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

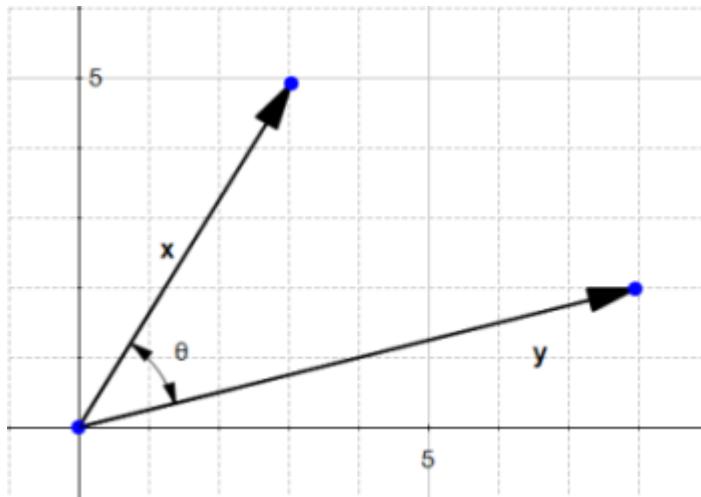


$$\|z\| = \|\mathbf{x}\| \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

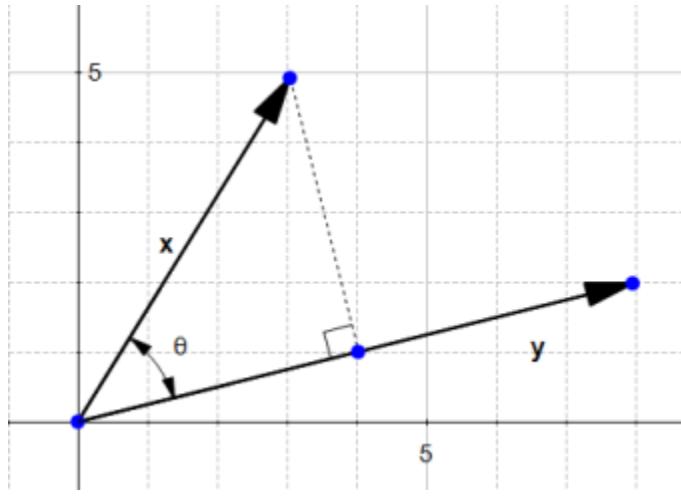
$$\|z\| = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{y}\|}$$



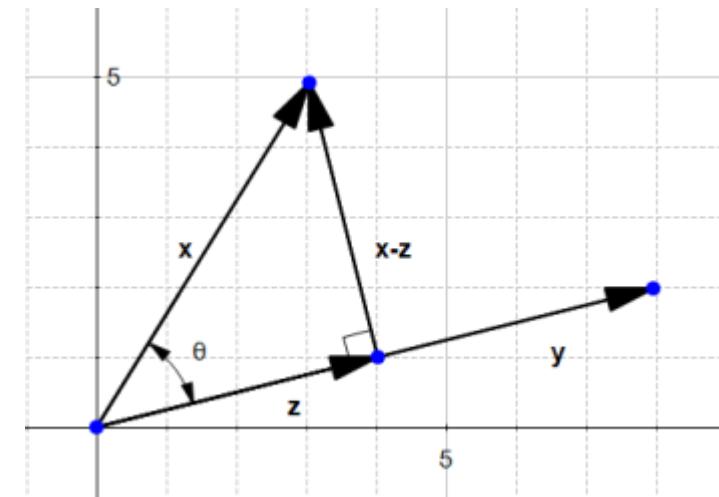
# The orthogonal projection of a vector



$$\|z\| = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{y}\|}$$



$$\mathbf{u} = \frac{\mathbf{z}}{\|z\|}$$



$$\|\mathbf{u}\| = 1$$

If we define the vector  $\mathbf{u}$  as the **direction** of  $\mathbf{y}$  then

$$\underline{\mathbf{u}} = \frac{\mathbf{y}}{\|\mathbf{y}\|}$$

$$\|z\| = \mathbf{u} \cdot \mathbf{x}$$

$$\mathbf{z} = \|z\| \mathbf{u}$$

$$\boxed{\mathbf{z} = (\mathbf{u} \cdot \mathbf{x}) \mathbf{u}}$$

We see that this distance is  $\|x - z\|$

$$\|x - z\| = \sqrt{(3 - 4)^2 + (5 - 1)^2} = \sqrt{17}$$

# Hyperplane

equation of a line is :  $y = ax + b$

equation of an hyperplane is defined by :  $\underline{\mathbf{w}^T \mathbf{x} = 0}$

$$y - ax - b = 0$$

Given two vectors  $\mathbf{w} \begin{pmatrix} -b \\ -a \\ 1 \end{pmatrix}$  and  $\mathbf{x} \begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$

$$\mathbf{w}^T \mathbf{x} = -b \times (1) + (-a) \times x + 1 \times y$$

$$\mathbf{w}^T \mathbf{x} = y - ax - b \quad \text{---}$$

# Hyperplane

Why do we use the hyperplane equation  $\mathbf{w}^T \mathbf{x}$  instead of  $y = ax + b$ ?

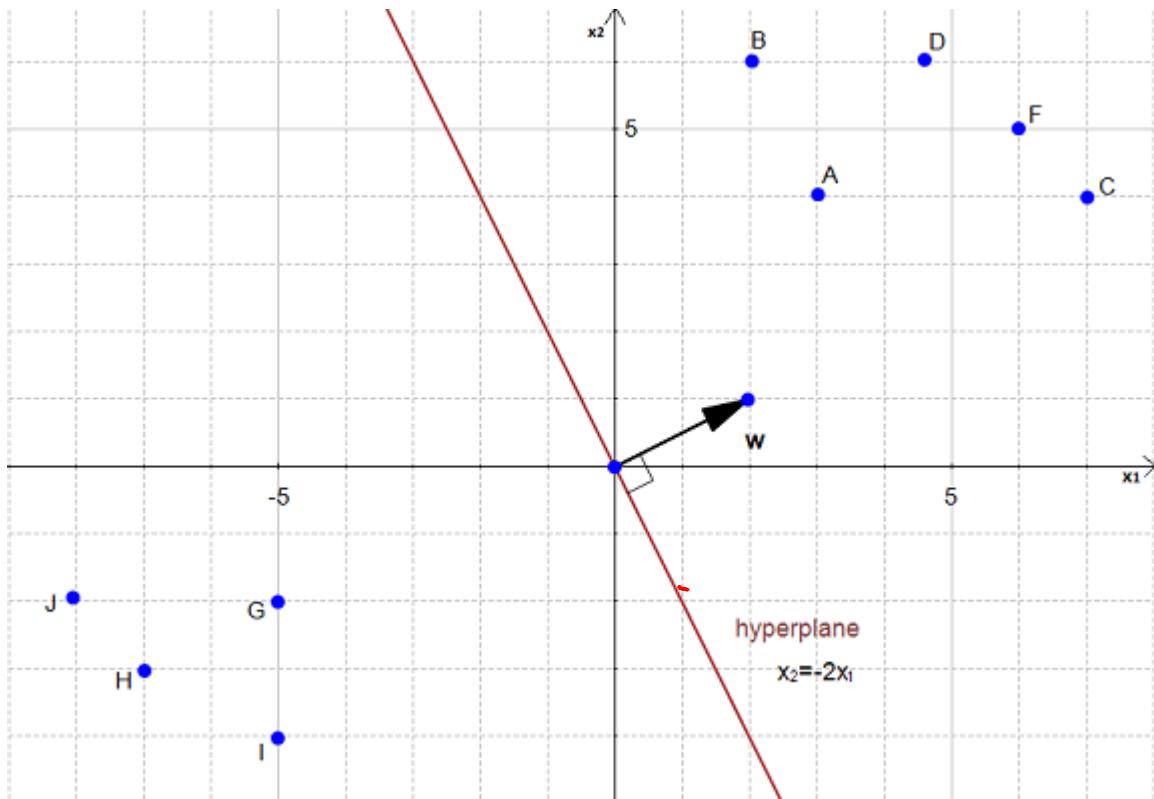
For two reasons:



- it is easier to work in more than two dimensions with this notation,
- the vector  $\mathbf{w}$  will always be normal to the hyperplane



# Compute the distance from a point to the hyperplane



$$w_0 = 0$$

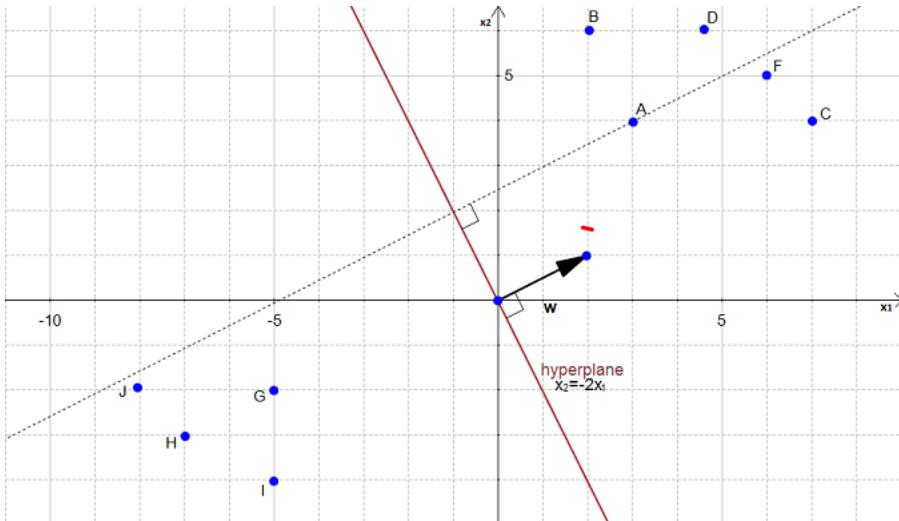
$$x_2 = -2x_1 \quad n_2 + 2x_1 = 0$$

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

$$\mathbf{w} \begin{pmatrix} 2 \\ 1 \end{pmatrix} \text{ and } \mathbf{x} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$2n_1 + n_2 = 0$$

# distance between the point A(3,4) and the hyperplane



$$\mathbf{w} = (2, 1)$$

$$\mathbf{a} = (3, 4)$$

$$\mathbf{u} = \left( \frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}} \right)$$

$$\mathbf{p} = \left( 3 \times \frac{2}{\sqrt{5}} + 4 \times \frac{1}{\sqrt{5}} \right) \mathbf{u}$$

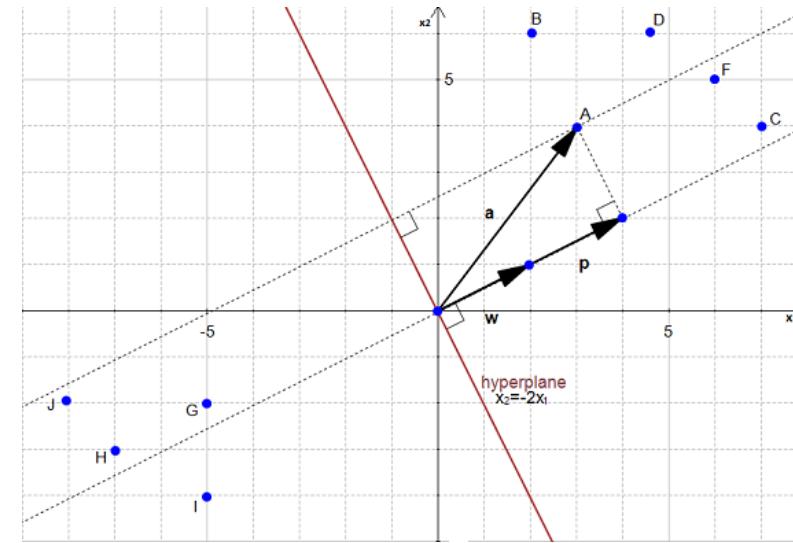
$$\mathbf{p} = \left( \frac{10}{\sqrt{5}} \times \frac{2}{\sqrt{5}}, \frac{10}{\sqrt{5}} \times \frac{1}{\sqrt{5}} \right)$$

$$\mathbf{p} = \left( \frac{6}{\sqrt{5}} + \frac{4}{\sqrt{5}} \right) \mathbf{u}$$

$$\mathbf{p} = \left( \frac{20}{5}, \frac{10}{5} \right)$$

$$\mathbf{p} = \frac{10}{\sqrt{5}} \mathbf{u}$$

$$\mathbf{p} = (4, 2)$$



$$\mathbf{p} = (\underline{\mathbf{u} \cdot \mathbf{a}}) \mathbf{u}$$

$$\|\mathbf{p}\| = \sqrt{4^2 + 2^2} = 2\sqrt{5}$$



# Machine Learning

## Support Vector Machine (SVM)

### Part III: Linear SVM

Dr. Mehran Safayani

safayani@iut.ac.ir

safayani.iut.ac.ir



<https://www.aparat.com/mehran.safayani>

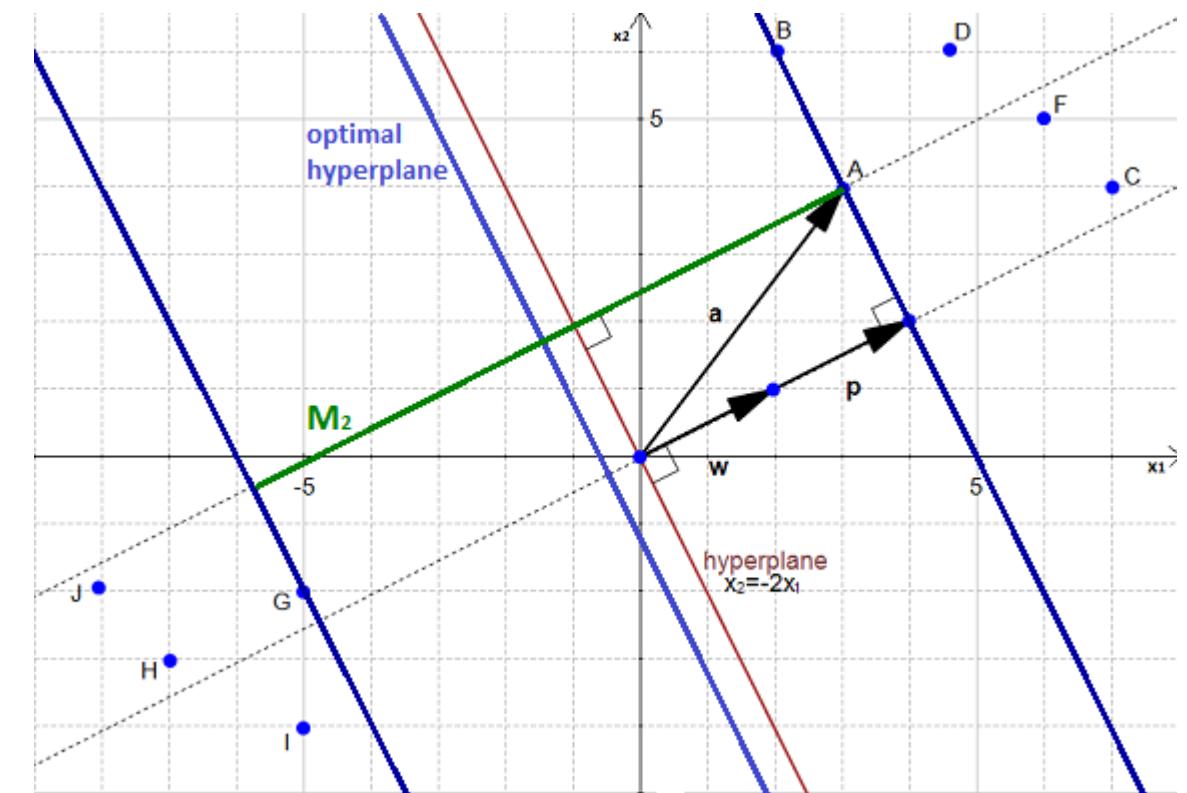
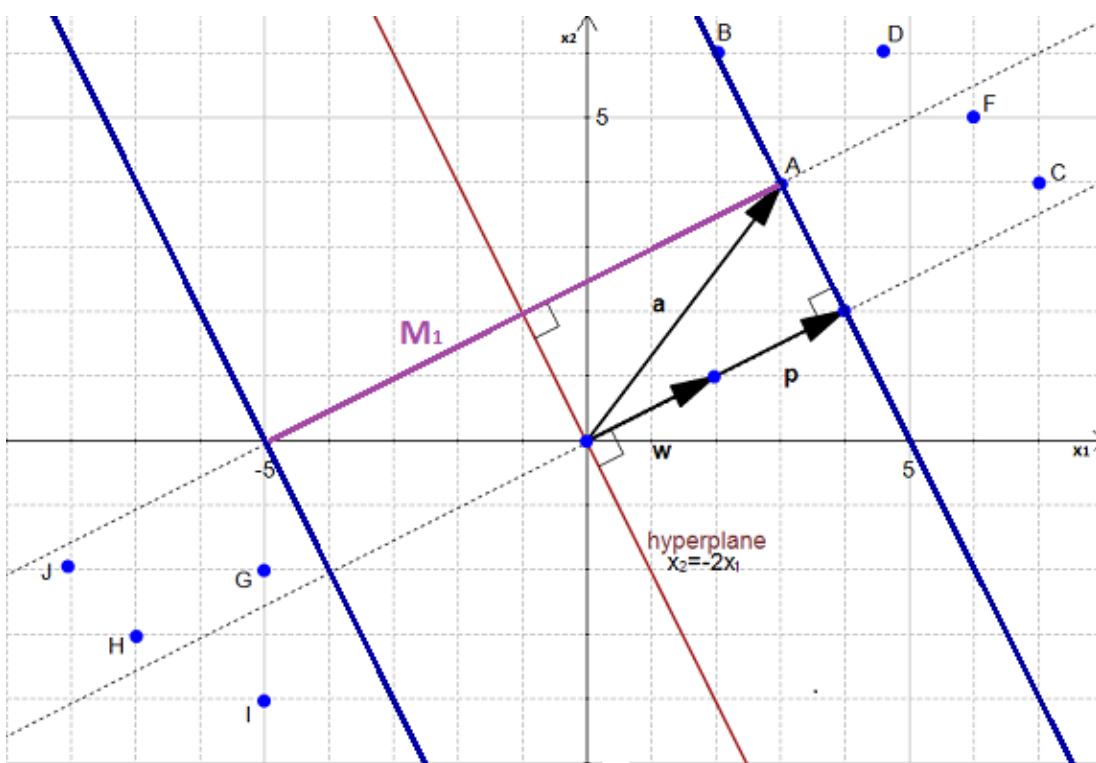


[https://github.com/safayani/machine\\_learning\\_course](https://github.com/safayani/machine_learning_course)



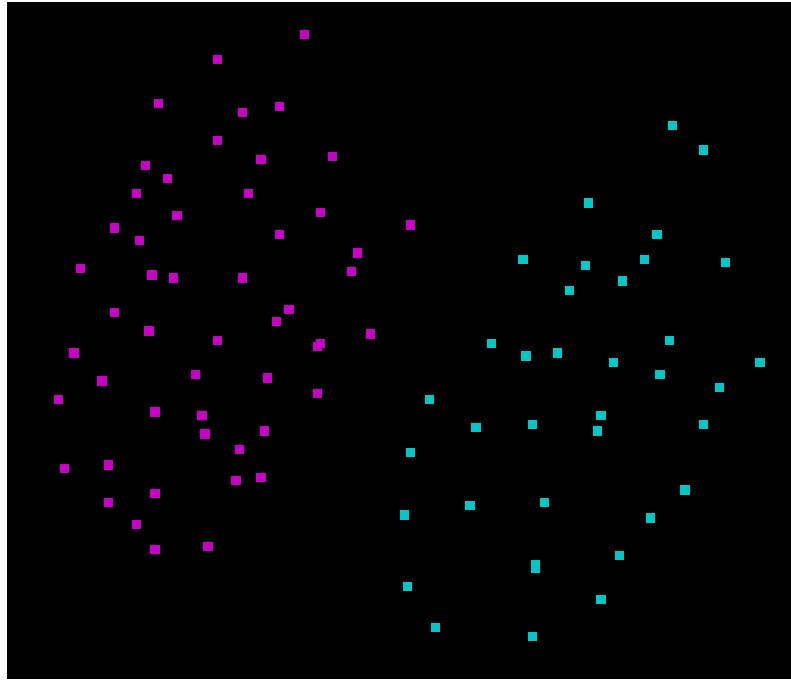
- the optimal hyperplane

- How to find the optimal hyperplane ?

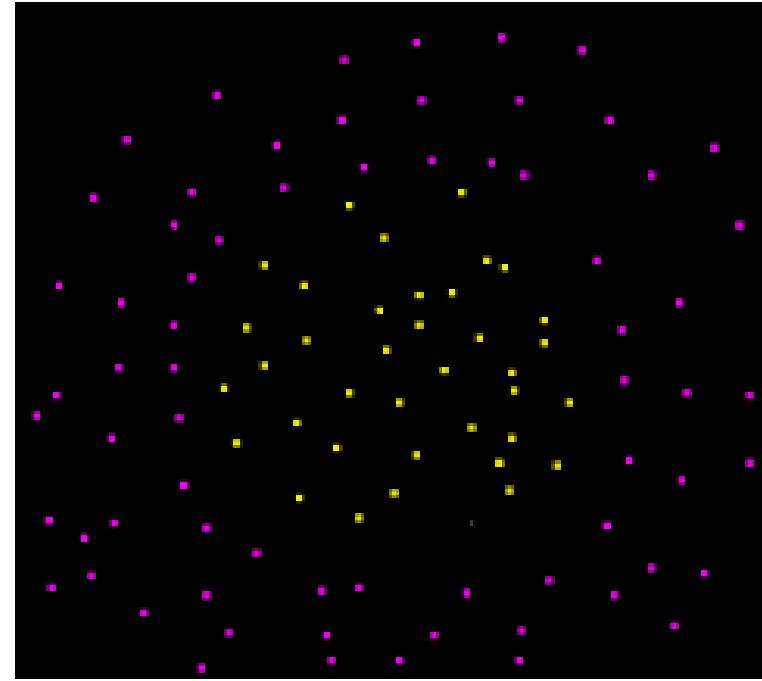


You have a dataset D and you want to classify it

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



Linearly separable data



Non linearly separable data

let's assume that our dataset D IS linearly separable.

# Taking another look at the hyperplane equation

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

$$\mathbf{w}(b, -a, 1) \text{ and } \mathbf{x}(\underline{1}, x, y)$$

$$\mathbf{w} \cdot \mathbf{x} = b \times (1) + (-a) \times x + 1 \times y$$

$$\mathbf{w} \cdot \mathbf{x} = y - ax + b \underset{\textcolor{red}{\approx}}{=} \omega$$

$\mathbf{w}'(-a, 1)$  and  $\mathbf{x}'(x, y)$

$$\mathbf{w}' \cdot \mathbf{x}' = (-a) \times x + 1 \times y$$

$$\mathbf{w}' \cdot \mathbf{x}' = y - ax$$

$$\mathbf{w}' \cdot \mathbf{x}' + \underline{b} = y - ax + b \textcolor{red}{\leq} \sigma$$

$$\mathbf{w}' \cdot \mathbf{x}' + b = \mathbf{w} \cdot \mathbf{x}$$

Given a hyperplane  $H_0$  separating the dataset and satisfying:



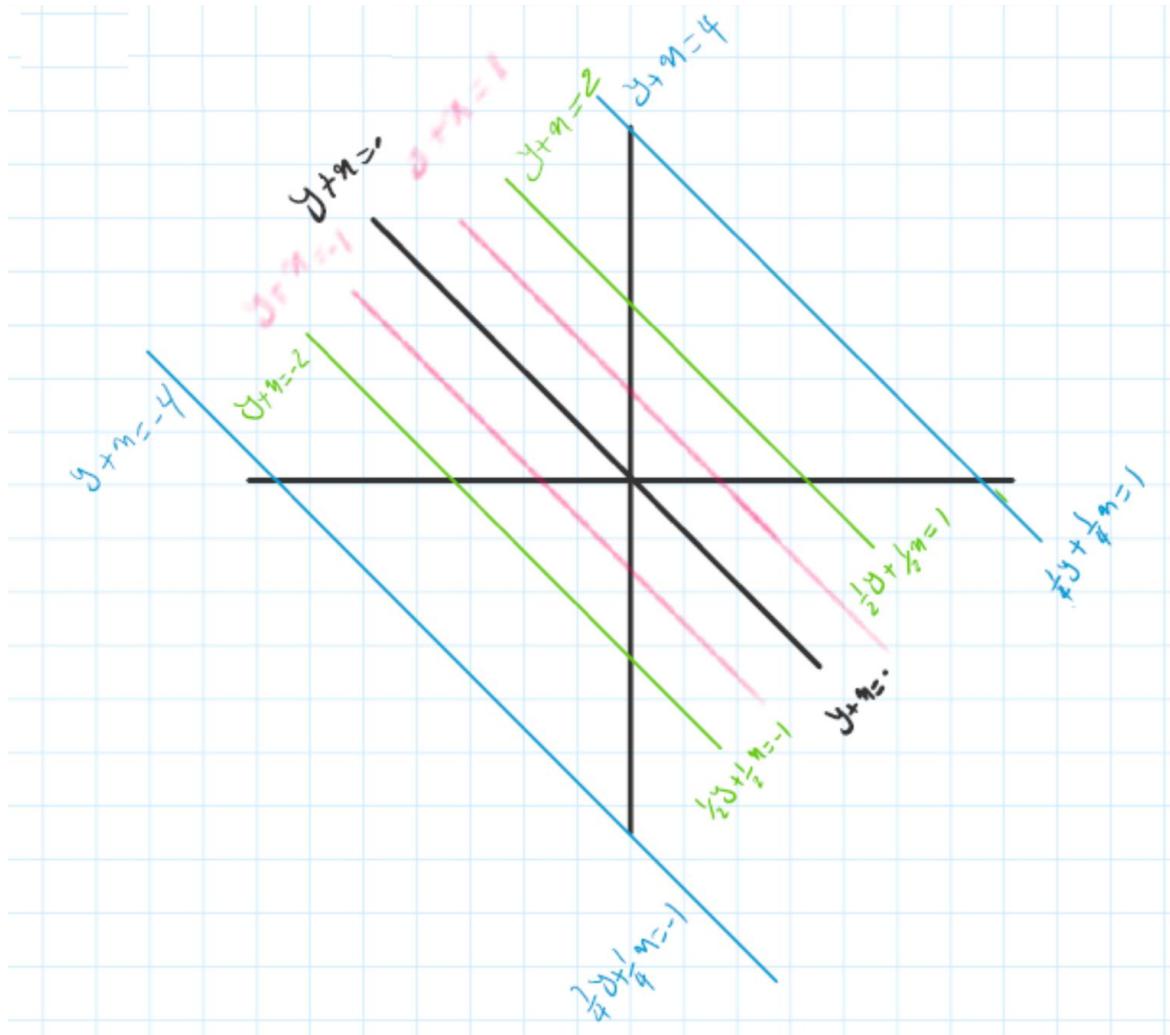
$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

We can select two others hyperplanes  $H_1$  and  $H_2$  which also separate the data and have the following equations :

$$\mathbf{w} \cdot \mathbf{x} + b = \delta$$

$$\mathbf{w} \cdot \mathbf{x} + b = -\delta$$

so that  $H_0$  is equidistant from  $H_1$  and  $H_2$ .



However, here the variable  $\delta$  is not necessary. So we can set  $\delta = 1$  to simplify the problem.

$$w \cdot x + b = 1$$
$$w \cdot x + b = -1$$

Now we want to be sure that they have no points between them.

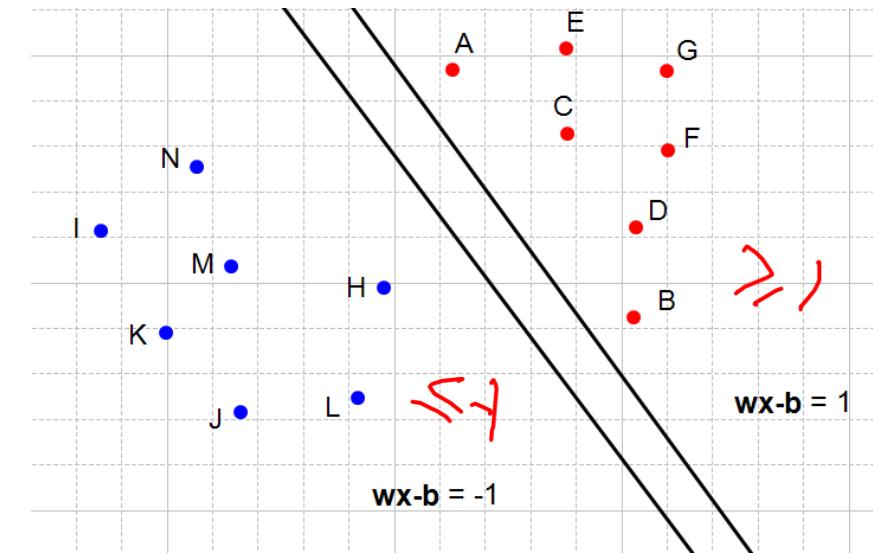
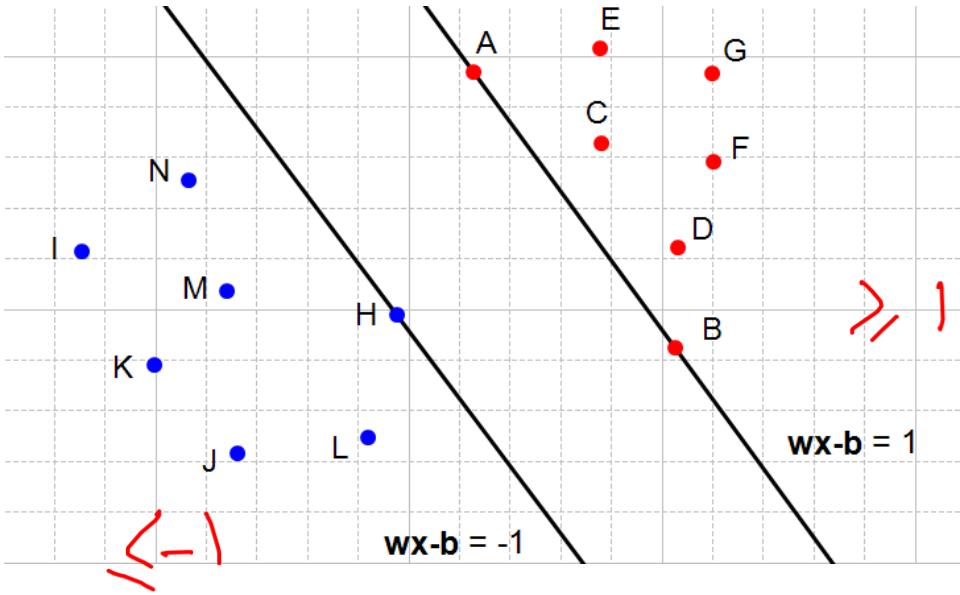
**constraints:**

For each vector  $x_i$  either :

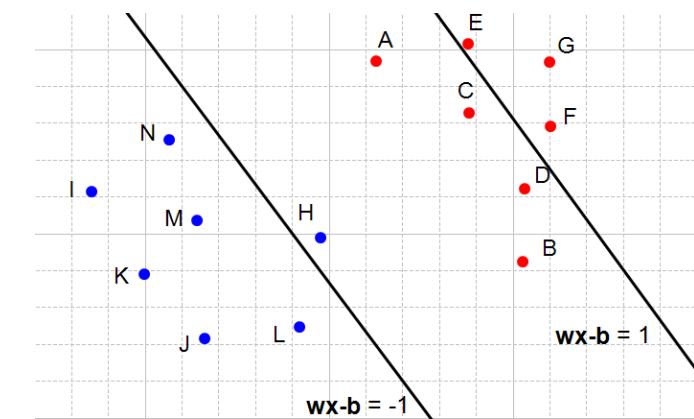
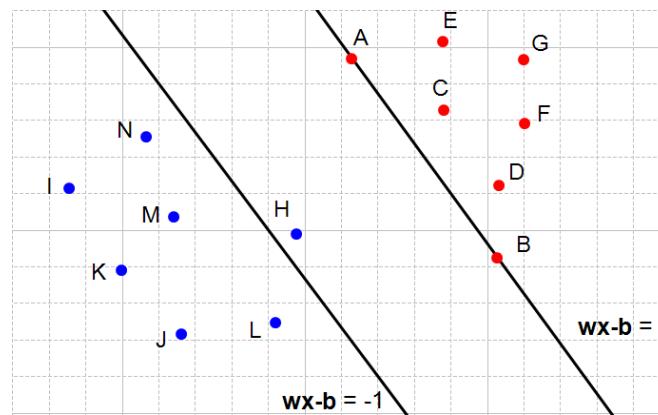
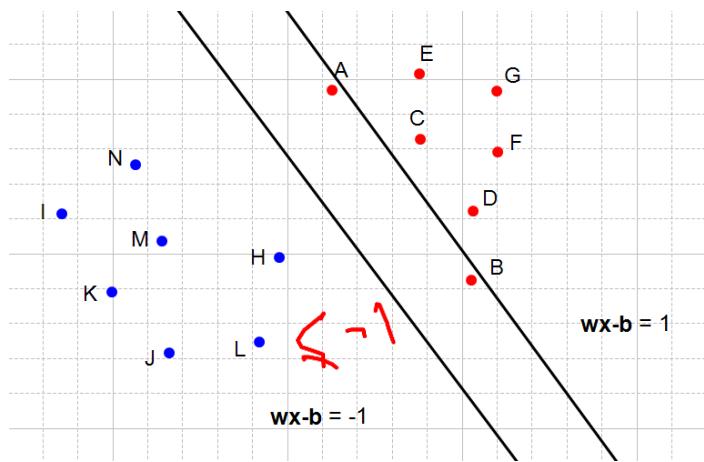
$$w \cdot x_i + b \geq 1 \text{ for } x_i \text{ having the class } 1$$

$$w \cdot x_i + b \leq -1 \text{ for } x_i \text{ having the class } -1$$

# Understanding the constraints



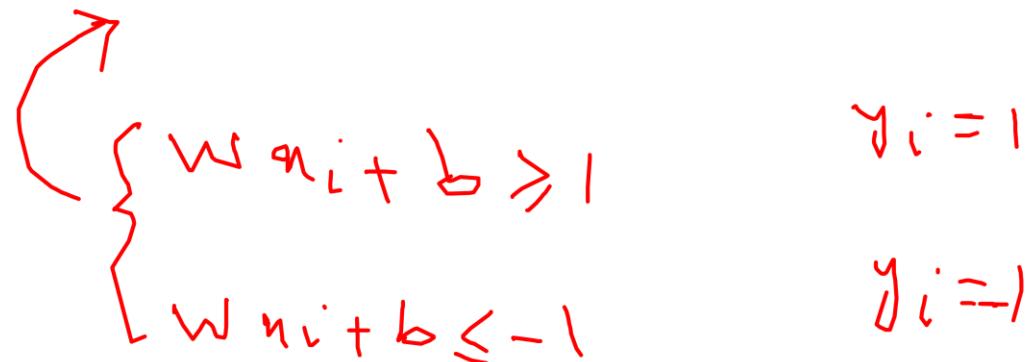
# Understanding the constraints



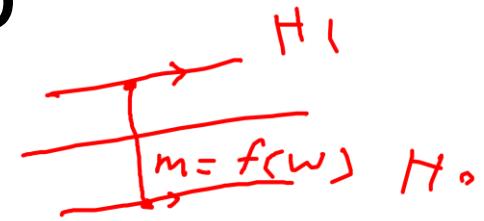
# Combining both constraints

$$-1(w_{ni} + b) \geq 1 \rightarrow w_{ni} + b \leq -1$$

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \text{ for all } 1 \leq i \leq n$$


$$\left\{ \begin{array}{ll} w_{ni} + b > 1 & y_i = 1 \\ w_{ni} + b < -1 & y_i = -1 \end{array} \right.$$

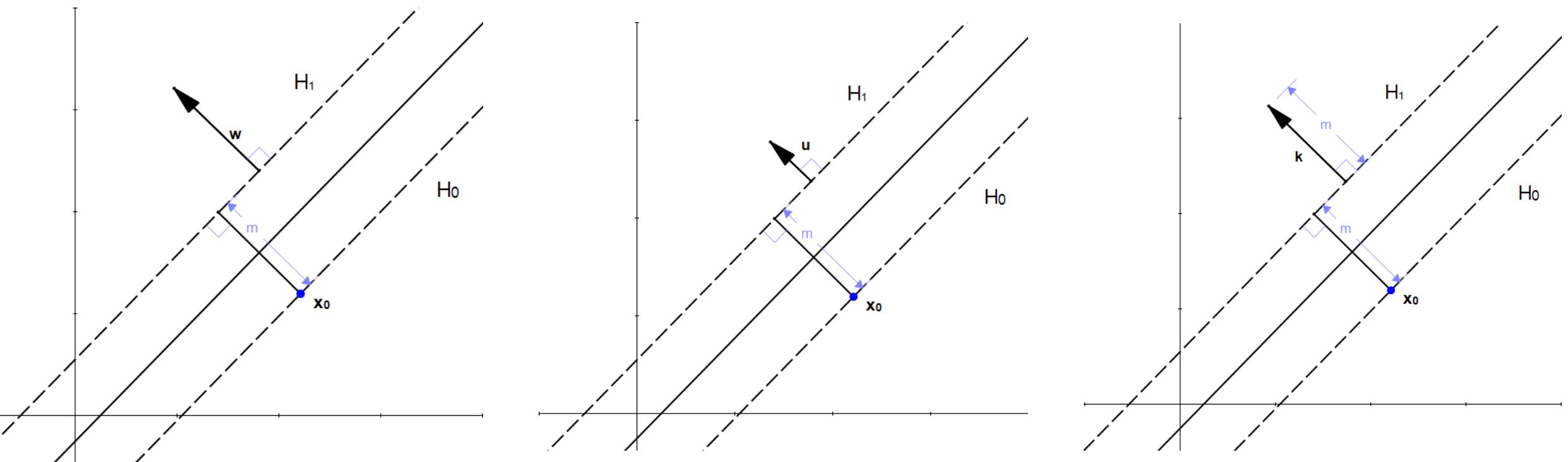
Maximize the distance between the two hyperplanes



Finding the biggest margin, is the same thing as finding the optimal hyperplane.

- a) What is the distance between our two hyperplanes ?
- $\mathcal{H}_0$  be the hyperplane having the equation  $\mathbf{w} \cdot \mathbf{x} + b = -1$
- $\mathcal{H}_1$  be the hyperplane having the equation  $\mathbf{w} \cdot \mathbf{x} + b = 1$
- $\mathbf{x}_0$  be a point in the hyperplane  $\mathcal{H}_0$ .

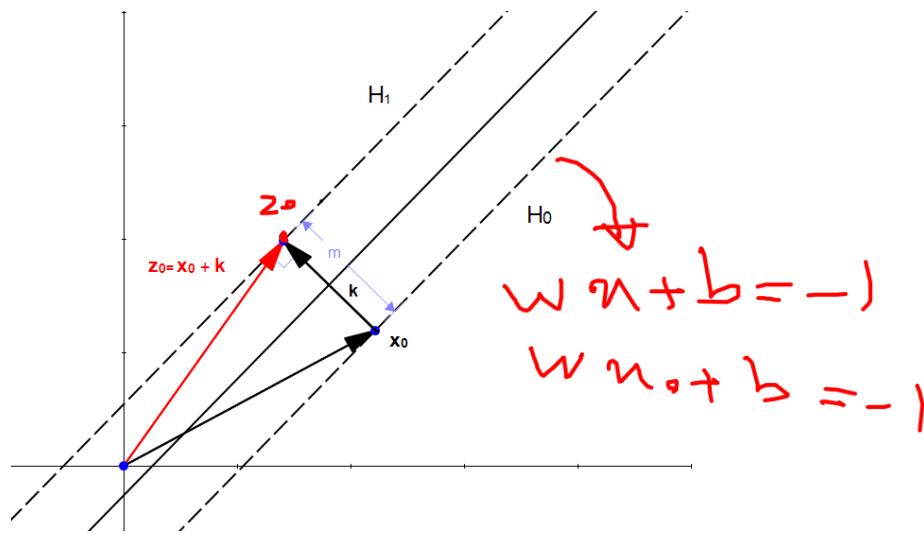
$m$  is the distance between hyperplanes  $\mathcal{H}_0$  and  $\mathcal{H}_1$ .



$$\mathbf{u} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$\|\mathbf{u}\| \approx 1$$

$$\mathbf{k} = m\mathbf{u} = m \frac{\mathbf{w}}{\|\mathbf{w}\|}$$



$$\mathbf{w} \cdot \mathbf{z}_0 + b = 1$$

$$\mathbf{w} \cdot (\mathbf{x}_0 + \mathbf{k}) + b = 1$$

$$\mathbf{w} \cdot \left( \mathbf{x}_0 + m \frac{\mathbf{w}}{\|\mathbf{w}\|} \right) + b = 1$$

$$\mathbf{w} \cdot \mathbf{x}_0 + m \frac{\mathbf{w} \cdot \mathbf{w}}{\|\mathbf{w}\|} + b = 1$$

$$\mathbf{w} \cdot \mathbf{x}_0 + m \frac{\mathbf{w} \cdot \mathbf{w}}{\|\mathbf{w}\|} + b = 1$$

$$\mathbf{w} \cdot \mathbf{x}_0 + m \frac{\|\mathbf{w}\|^2}{\|\mathbf{w}\|} + b = 1$$

$$\mathbf{w} \cdot \mathbf{x}_0 + m\|\mathbf{w}\| + b = 1$$

$$\mathbf{w} \cdot \mathbf{x}_0 + b = 1 - m\|\mathbf{w}\|$$

$$\mathbf{w} \cdot \mathbf{w} = \|\mathbf{w}\|^2$$

$$\mathbf{w} = \langle w_1, w_2 \rangle$$

$$\mathbf{w} \cdot \mathbf{w} = w_1^2 + w_2^2$$

$$\|\mathbf{w}\| = \sqrt{w_1^2 + w_2^2}$$

$$\underline{\mathbf{w} \cdot \mathbf{x}_0 + b} = 1 - m\|\mathbf{w}\|$$

$$\mathbf{w} \cdot \mathbf{x}_0 + b = -1$$

$$-1 = 1 - m\|\mathbf{w}\|$$

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

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

This is it ! We found a way to compute m.

Maximizing the margin is the same thing as  
minimizing the norm of  $\mathbf{w}$

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

Minimize in  $(\mathbf{w}, b)$

$$\|\mathbf{w}\|$$

subject to  $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$

(for any  $i = 1, \dots, n$ )



# Machine Learning

## Support Vector Machine (SVM)

### Part IV: Soft margin SVM

Dr. Mehran Safayani

safayani@iut.ac.ir

safayani.iut.ac.ir



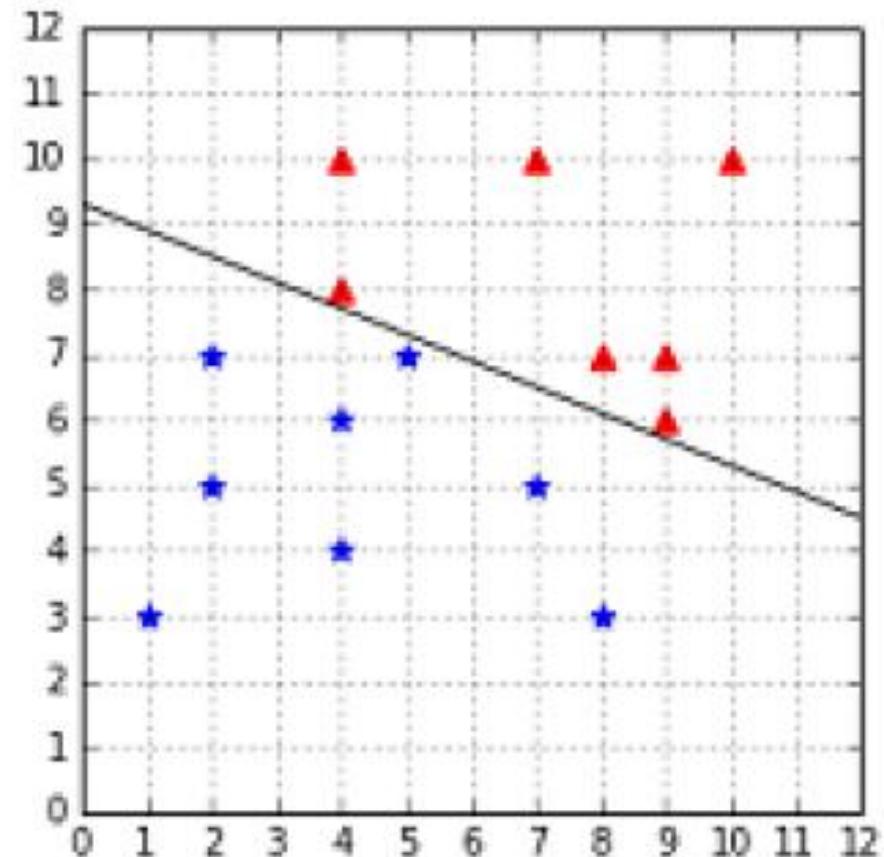
<https://www.aparat.com/mehran.safayani>



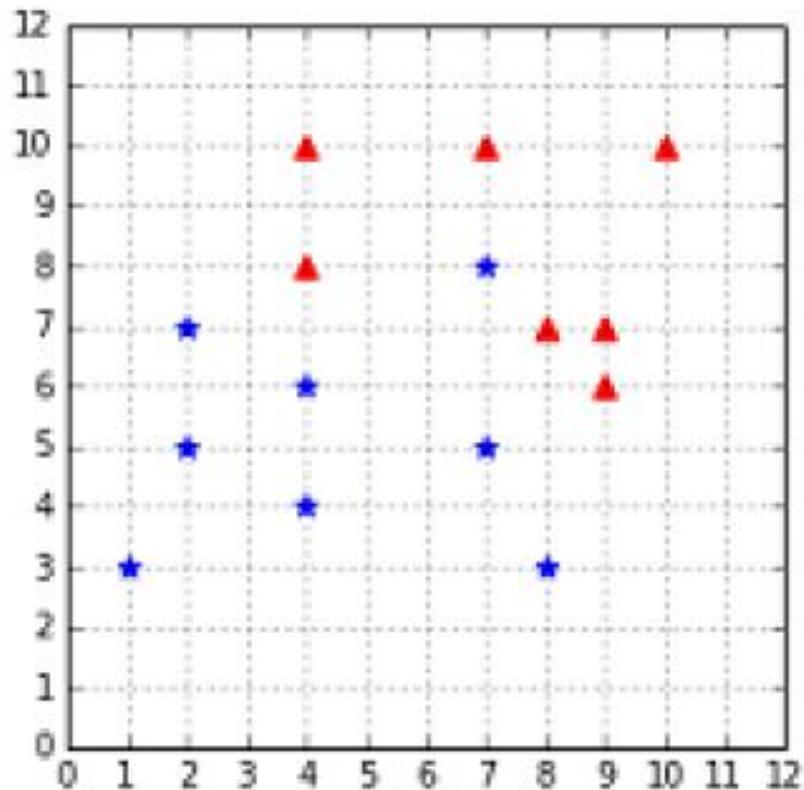
[https://github.com/safayani/machine\\_learning\\_course](https://github.com/safayani/machine_learning_course)



- outlier data point at (5, 7)



- *The outlier at (7, 8) breaks linear separability*



# Soft margin

- Slack variables

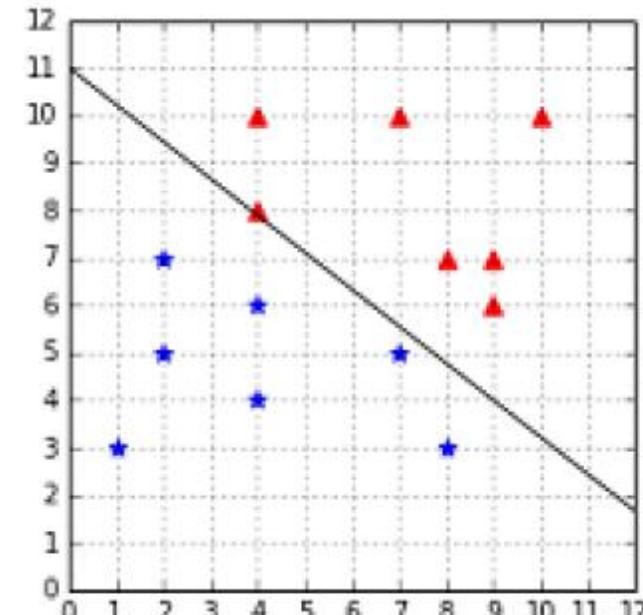
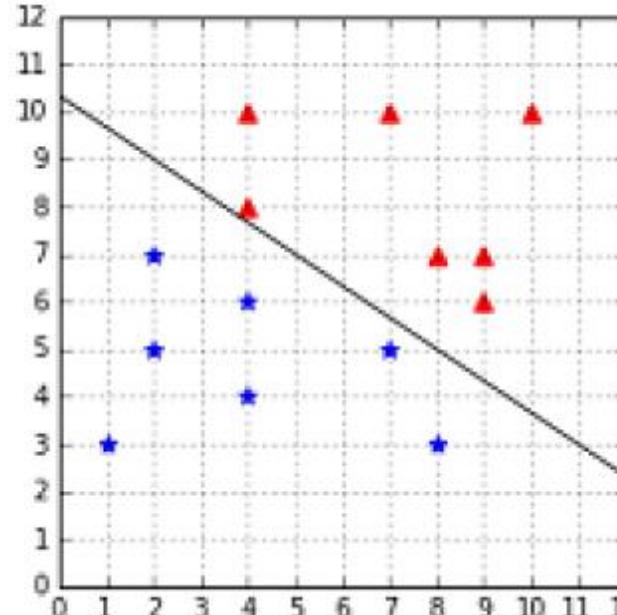
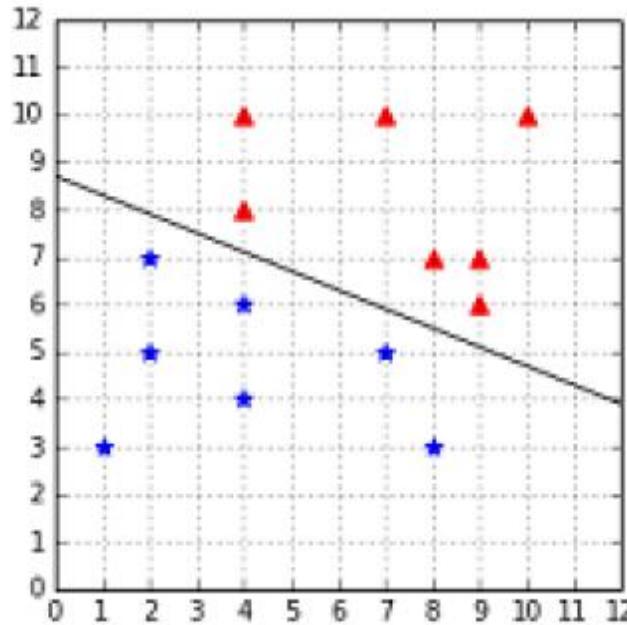
$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$$

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \zeta_i$$

The problem is that we could choose a huge value of  $\zeta_i$  for every example, and all the constraints will be satisfied.

$$\begin{aligned} & \underset{\mathbf{w}, b, \zeta}{\text{minimize}} && \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \zeta_i \\ & \text{subject to} && y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \zeta_i \\ & && \zeta_i \geq 0 \quad \text{for any } i = 1, \dots, m \end{aligned}$$

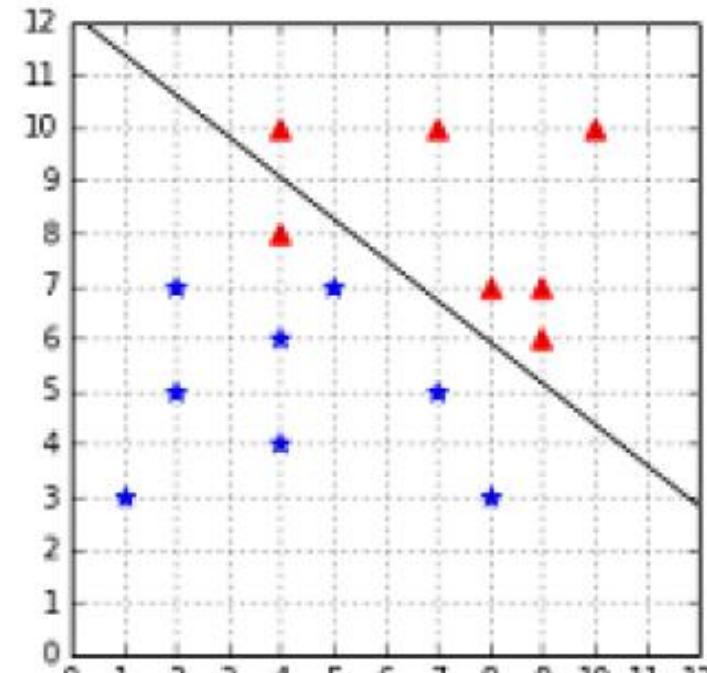
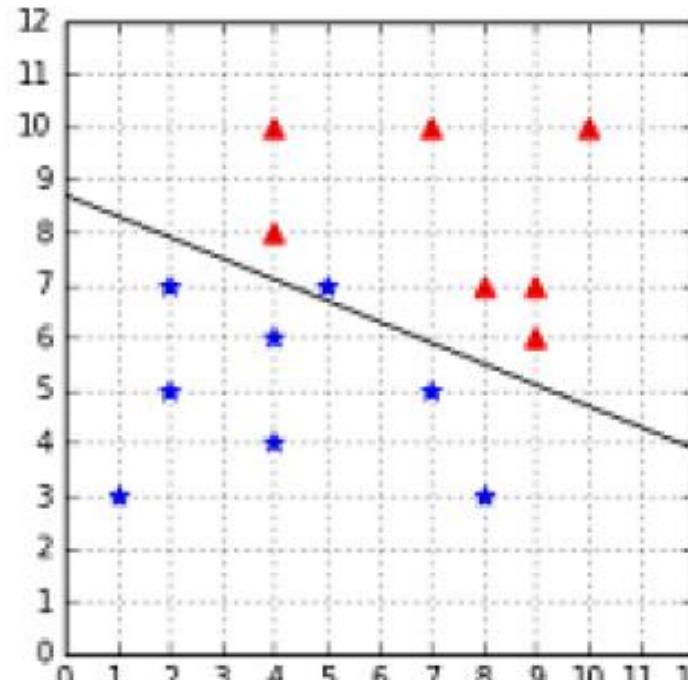
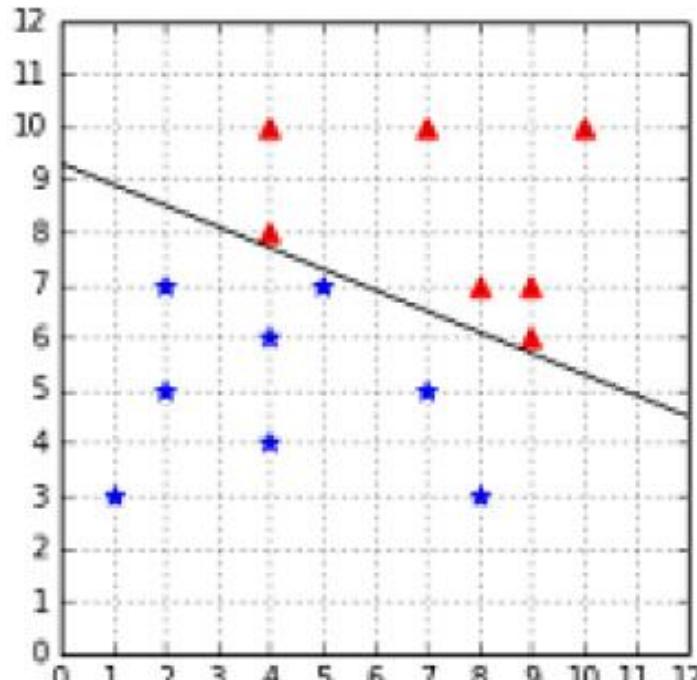
# Understanding what C does



*Effect of  $C=+\infty$ ,  $C=1$ , and  $C=0.01$  on a linearly separable dataset*

- When C is close to zero, there is basically no constraint anymore, and we end up with a hyperplane not classifying anything.
- It seems that when the data is linearly separable, sticking with a big C is the best choice

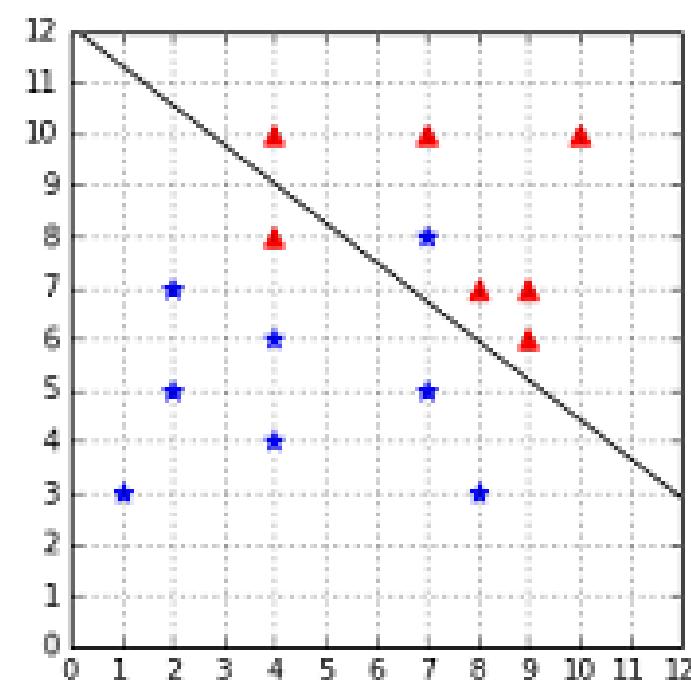
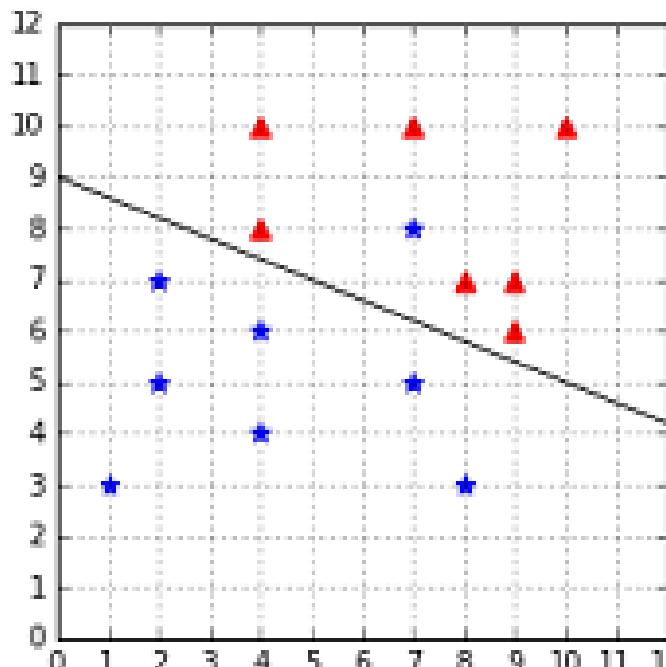
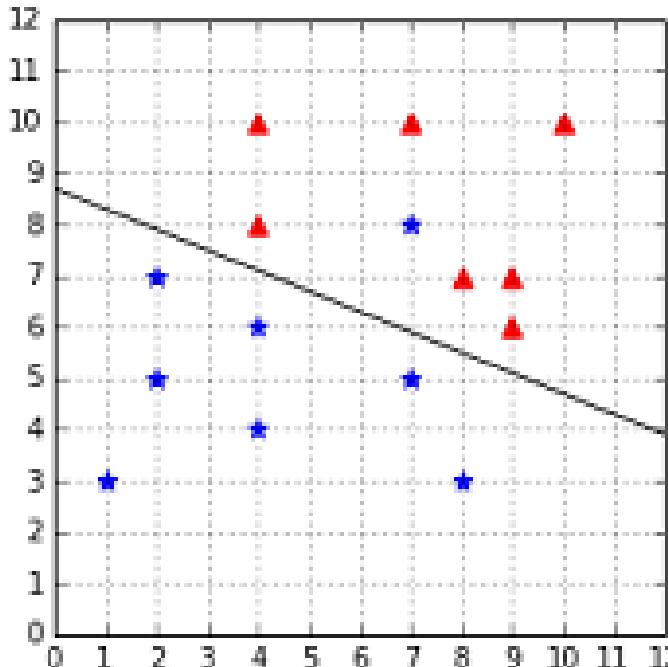
# Understanding what C does



*Effect of  $C=+\infty$ ,  $C=1$ , and  $C=0.01$  on a linearly separable dataset with an outlier*

when we use  $C=1$ , we end up with a hyperplane very close to the one of the hard margin classifier without outlier.

# Understanding what C does



*Effect of  $C=3$ ,  $C=1$ , and  $C=0.01$  on a non-separable dataset with an outlier*

- we cannot use  $C=\infty$  because there is no solution meeting all the hard margin constraints.
- the best hyperplane is achieved with  $C=3$

# Understanding what C does

Rules of thumb:

- A small C will give a wider margin, at the cost of some misclassifications.
- A huge C will give the hard margin classifier and tolerates zero constraint violation.
- The key is to find the value of C such that noisy data does not impact the solution too much.



# Machine Learning

## Support Vector Machine (SVM)

### Part V: Kernels

Dr. Mehran Safayani

safayani@iut.ac.ir

safayani.iut.ac.ir



<https://www.aparat.com/mehran.safayani>

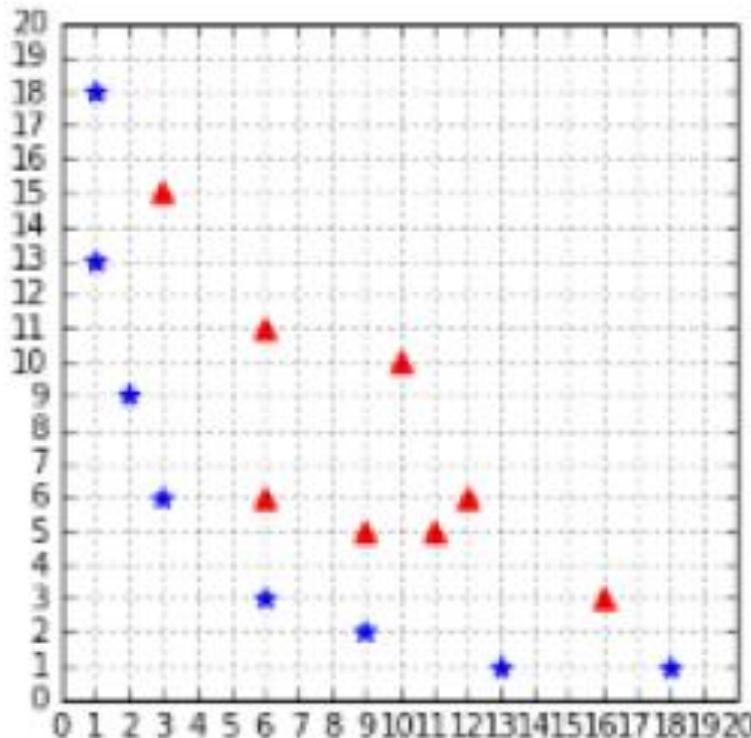


[https://github.com/safayani/machine\\_learning\\_course](https://github.com/safayani/machine_learning_course)



# Kernels

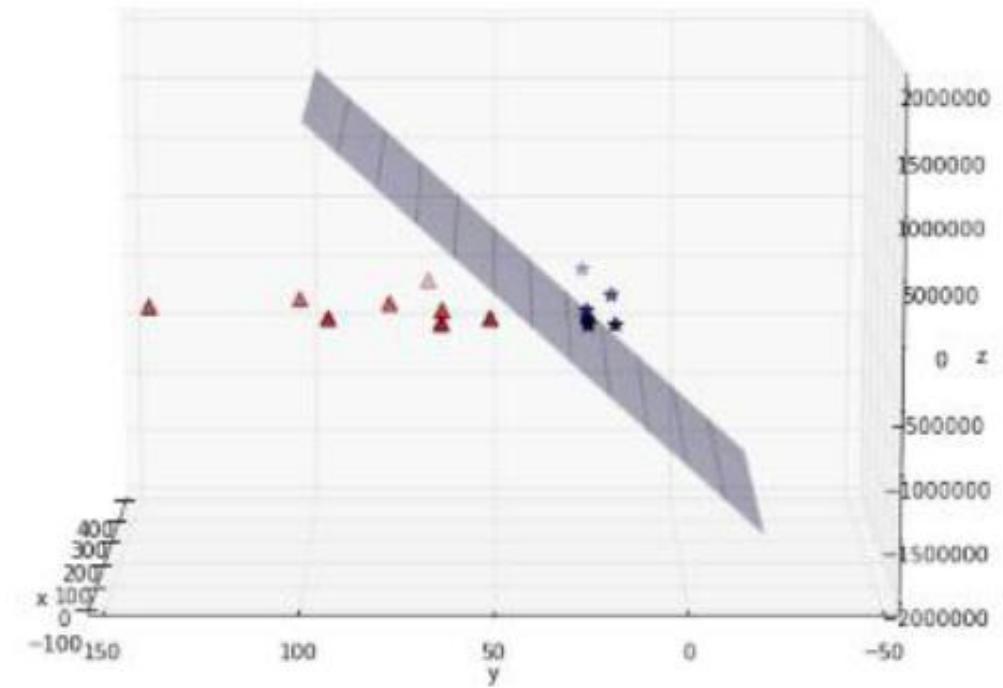
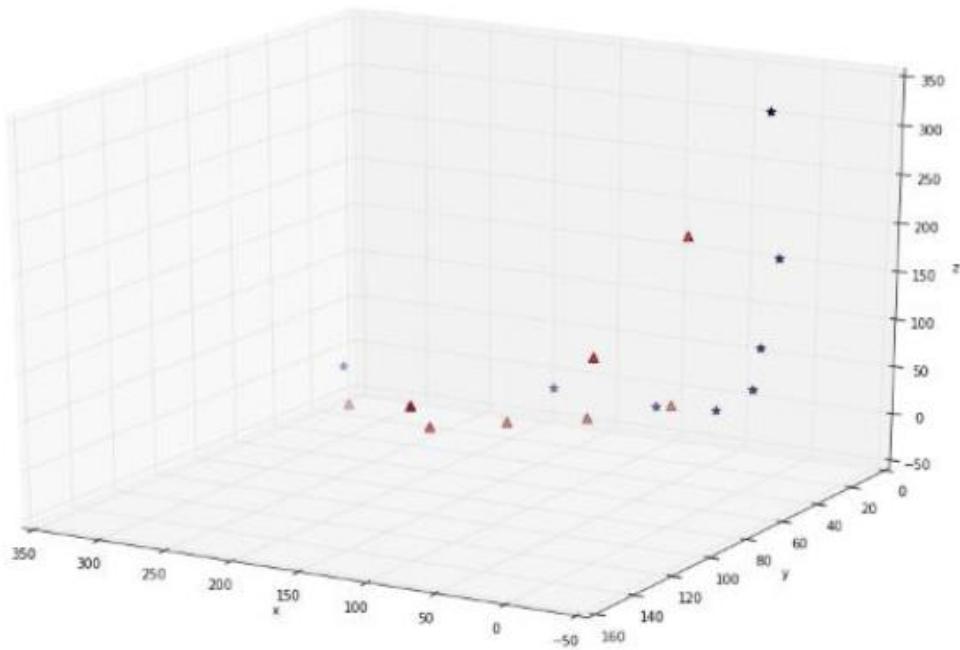
- Can we classify non-linearly separable data?



*A straight line cannot separate the data*

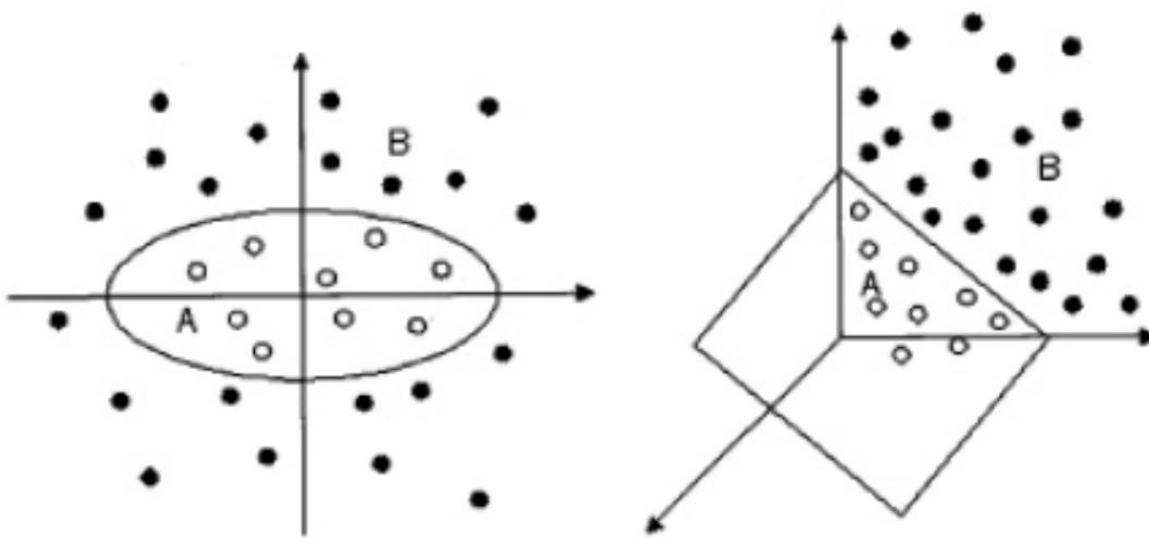
# polynomial mapping

$$\phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$



$$\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$



# Basic recipe

1. Transform every two-dimensional vector into a three-dimensional vector using the `transform` method (quadratic mapping)
2. Train the SVMs using the 3D dataset.
3. For each new example we wish to predict, transform it using the **transform** method before passing it to the **predict** method

One of the main drawbacks of the previous method is that we must transform every example. If we have millions or billions of examples and that transform method is complex, that can take a huge amount of time. This is when kernels come to the rescue.

# Kernel Trick

- Kernel Function:  $K(\vec{x}_i, \vec{x}_j) = \langle \phi(x_i) \cdot \phi(x_j) \rangle$

$$\vec{x}_i = (n_{i1}, n_{i2}) \quad \phi(\vec{n}_i) = (n_{i1}^2, \sqrt{2} n_{i1} n_{i2}, n_{i2}^2)$$

$$\vec{x}_j = (n_{j1}, n_{j2}) \quad \phi(\vec{n}_j) = (n_{j1}^2, \sqrt{2} n_{j1} n_{j2}, n_{j2}^2)$$

$$K(\vec{n}_i, \vec{n}_j) = \underbrace{n_{i1}^2 \times n_{j1}^2 + 2 n_{i1} n_{i2} n_{j1} n_{j2} + n_{i2}^2 n_{j2}^2}_{\rightarrow K}$$

# Examples Kernel Trick

$$\vec{x} = (x_1, x_2)$$

$$\vec{z} = (z_1, z_2)$$

$$K(x, z) = \langle \vec{x} \cdot \vec{z} \rangle^2$$

$$K(x_i, x_j) = \langle \phi(x_i) \cdot \phi(x_j) \rangle$$

$$\begin{aligned} K(x, z) &= \underline{\langle \vec{x} \cdot \vec{z} \rangle}^2 \\ &= (x_1 z_1 + x_2 z_2)^2 \Rightarrow \boxed{3} \\ &= (x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2) \\ &= \left\langle (x_1^2, \sqrt{2}x_1 x_2, x_2^2) \cdot (z_1^2, \sqrt{2}z_1 z_2, z_2^2) \right\rangle \\ &= \langle \phi(\vec{x}) \cdot \phi(\vec{z}) \rangle \\ &\quad \text{not necessary any more} \end{aligned}$$

mapping function  $\phi$  fused in  $K$

$$\rightarrow \text{implicit } \phi(\vec{x}) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)$$

possible to operate in any n-dimensional  $FS$   
complexity independent of  $FS$

# Kernel functions

Kernel functions must be continuous, symmetric, and most preferably should have a positive (semi-) definite [Gram matrix](#).

$$G(x_1, \dots, x_n) = \begin{vmatrix} \langle x_1, x_1 \rangle & \langle x_1, x_2 \rangle & \dots & \langle x_1, x_n \rangle \\ \langle x_2, x_1 \rangle & \langle x_2, x_2 \rangle & \dots & \langle x_2, x_n \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle x_n, x_1 \rangle & \langle x_n, x_2 \rangle & \dots & \langle x_n, x_n \rangle \end{vmatrix}.$$

$$x^T G x \geq 0$$

# Typical Kernels

## Linear kernel

This is the simplest kernel. It is simply defined by:

$$\phi(n) = x$$

$$K(x, x') = x \cdot x'$$

- **Polynomial Kernel**

$$K(x, z) = (\langle x \cdot z \rangle + \theta)^d, \quad \text{for } d \geq 0$$

- **Radial Basis Function (Gaussian Kernel)**

$$K(x, z) = e^{-\frac{\|x-z\|^2}{2\sigma^2}} \quad \|x\| := \sqrt{\langle x \cdot x \rangle}$$

$$k(n, n) = 1$$

- **(Sigmoid Kernel)**

$$K(x, z) = \tanh(\eta \langle x \cdot z \rangle + \theta)$$

$$\phi(n) = (n_1^2, \sqrt{2}n_1n_2, n_2^2)$$

- **Inverse multi-quadratic**

$$K(x, z) = \frac{1}{\sqrt{\|x-z\|^2/2\sigma^2 + c^2}}$$

$$\begin{aligned} \phi(n) &=? \\ k(n, n') &\leftarrow \phi(n) \cdot \phi(n') \\ &\quad \xrightarrow{n \xrightarrow{\phi} \phi(n)} \end{aligned}$$

## Kernel SVM:

Maximizing the margin is the same thing as minimizing the norm of  $\mathbf{w}$

$$\begin{aligned} & \underset{\mathbf{w}, b, \zeta}{\text{minimize}} && \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \zeta_i \\ & \text{subject to} && y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \zeta_i \\ & && \zeta_i \geq 0 \quad \text{for any } i = 1, \dots, m \end{aligned}$$

# Using kernels in SVM

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$$

$$b = \frac{1}{S} \sum_{i=1}^S (y_i - \mathbf{w} \cdot \mathbf{x}_i)$$

$$\begin{aligned} & \underset{\alpha}{\text{maximize}} && \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ & \text{subject to} && 0 \leq \alpha_i \leq C, \text{ for any } i = 1, \dots, m \end{aligned}$$

$$\sum_{i=1}^m \alpha_i y_i = 0$$

$$\underbrace{\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)}_{K(\mathbf{x}_i, \mathbf{x}_j)}$$

# Using kernels in SVM

$$\underset{\alpha}{\text{maximize}} \quad \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

subject to  $0 \leq \alpha_i \leq C$ , for any  $i = 1, \dots, m$

$$\sum_{i=1}^m \alpha_i y_i = 0$$

# how to classify

Linear SVM  $\leftarrow$

$$h(\mathbf{x}_i) = \text{sign}(\mathbf{w} \cdot \mathbf{x}_i + b)$$

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$$

$$h(\mathbf{x}_i) = \text{sign}\left( \sum_{j=1}^S \alpha_j y_j (\underbrace{\mathbf{x}_j \cdot \mathbf{x}_i}_{\phi(\mathbf{x}_j) \cdot \phi(\mathbf{x}_i)}) + b \right)$$

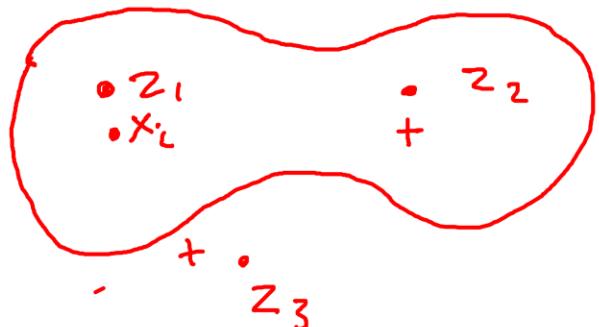
$\underbrace{\phi(\mathbf{x}_j) \cdot \phi(\mathbf{x}_i)}_{K(\mathbf{x}_i, \mathbf{x}_j)}$

K SVM  $\leftarrow$

$$h(\mathbf{x}_i) = \text{sign}\left( \sum_{j=1}^S \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i) + b \right)$$

$$h(\mathbf{x}_i) = \text{sign}\left( \sum_{j=1}^S \underbrace{\alpha_j y_j}_{\theta_j} K(\mathbf{x}_j, \mathbf{x}_i) + b \right)$$

$$\theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + b$$



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

$$K(u, z) = K(z, u)$$

$$K(u, z_1) = f_1$$

$$K(u, z_2) = f_2$$

$$K(u, z_3) = f_3$$

$$\theta_1 = 1 \quad \theta_2 = 1 \quad \theta_3 = -1 \quad b = -0.5$$

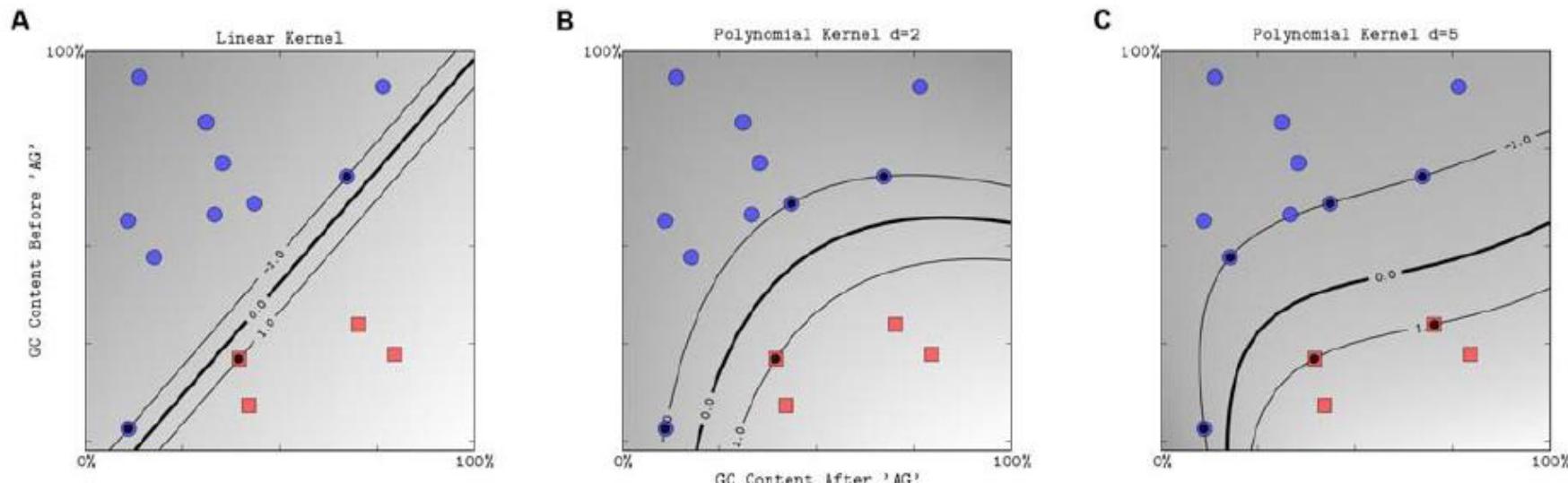
$$f_1 \approx 1 \quad f_2 \approx 0 \quad f_3 \approx 0$$

$$\Rightarrow 1 + 1 \times 0 + (-1) \times 0 - 0.5 > 0$$

$$\begin{aligned} \theta_1 = 0 & \quad \theta_2 = 1 & \theta_3 = 0 & \quad b = 0.5 \\ 1 \times 0 + 1 \times 1 + (-1) \times 0 - 0.5 & = 1 > 0 \end{aligned}$$

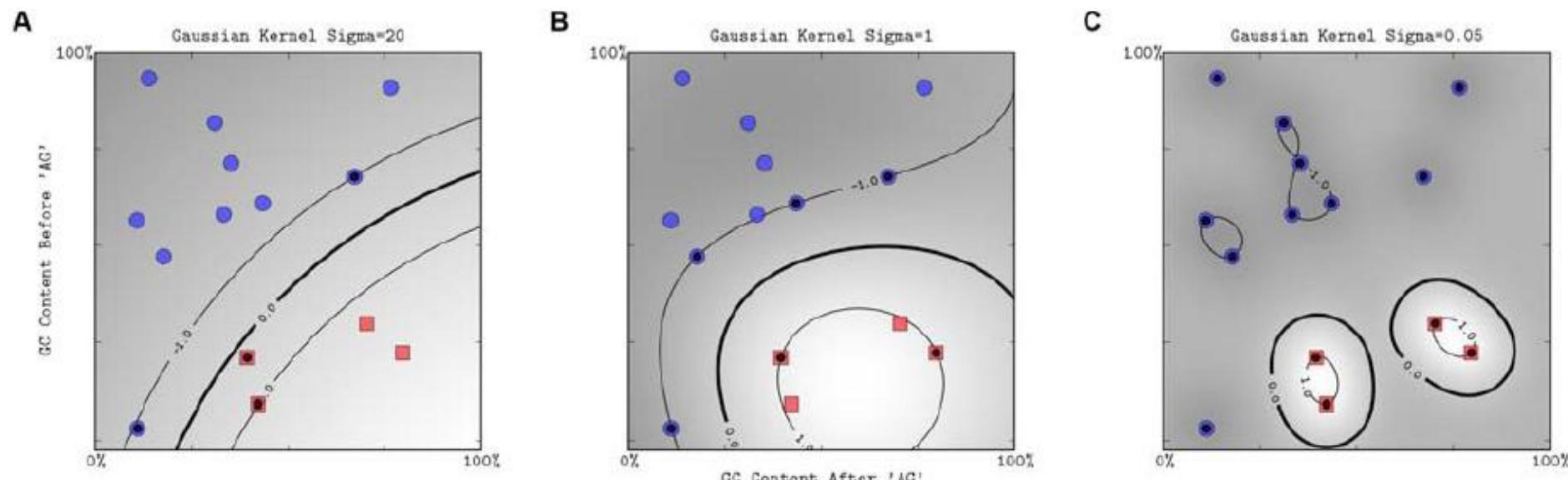
$$f_1 = 0 ; \quad f_2 = 1 ; \quad f_3 = 1$$

$$1 \times 0 + 1 \times 1 + (-1) \times 1 - 0.5 = -1.5 < 0$$



**Figure 6. The effect of the degree of a polynomial kernel.** The polynomial kernel of degree 1 leads to a linear separation (A). Higher-degree polynomial kernels allow a more flexible decision boundary (B,C). The style follows that of Figure 3.  
doi:10.1371/journal.pcbi.1000173.g006

Ben-Hur A, Ong CS, Sonnenburg S, Schölkopf B, Rätsch G (2008) Support Vector Machines and Kernels for Computational Biology. PLoS Comput Biol 4(10): e1000173. <https://doi.org/10.1371/journal.pcbi.1000173>



**Figure 7. The effect of the width parameter of the Gaussian kernel ( $\sigma$ ) for a fixed value of the soft-margin constant.** For large values of  $\sigma$  (A), the decision boundary is nearly linear. As  $\sigma$  decreases, the flexibility of the decision boundary increases (B). Small values of  $\sigma$  lead to overfitting (C). The figure style follows that of Figure 3.

doi:10.1371/journal.pcbi.1000173.g007



# Machine Learning

## Support Vector Machine (SVM)

### Part VI: Optimization

Dr. Mehran Safayani

safayani@iut.ac.ir

safayani.iut.ac.ir



<https://www.aparat.com/mehran.safayani>



[https://github.com/safayani/machine\\_learning\\_course](https://github.com/safayani/machine_learning_course)



# Affine and convex functions

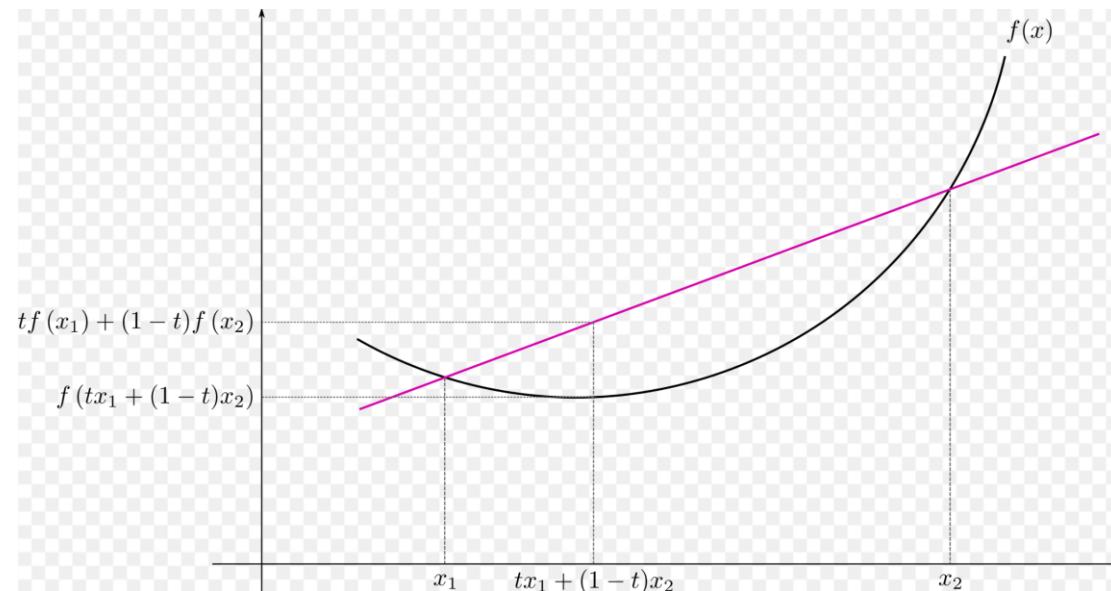
- Affine function

$$\mathbf{y} = f(\mathbf{x}) = A\mathbf{x} + \mathbf{b}.$$

- Convex function:

For all  $0 \leq t \leq 1$  and all  $x_1, x_2 \in X$ :

$$f(tx_1 + (1 - t)x_2) \leq tf(x_1) + (1 - t)f(x_2)$$



$$\begin{aligned}
\min_w \quad & f(w) \\
\text{s.t.} \quad & g_i(w) \leq 0, \quad i = 1, \dots, k \\
& h_i(w) = 0, \quad i = 1, \dots, l.
\end{aligned}$$

## generalized Lagrangian

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^k \alpha_i g_i(w) + \sum_{i=1}^l \beta_i h_i(w).$$

Here, the  $\alpha_i$ 's and  $\beta_i$ 's are the Lagrange multipliers.

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^k \alpha_i g_i(w) + \sum_{i=1}^l \beta_i h_i(w).$$

$$\theta_{\mathcal{P}}(w) = \max_{\alpha, \beta : \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta).$$

Here, the “ $\mathcal{P}$ ” subscript stands for “primal.”

$$\theta_{\mathcal{P}}(w) = \begin{cases} f(w) & \text{if } w \text{ satisfies primal constraints} \\ \infty & \text{otherwise.} \end{cases}$$

$$\min_w \theta_{\mathcal{P}}(w) = \min_w \max_{\alpha, \beta : \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta),$$

Now, let's look at a slightly different problem. We define

$$\theta_{\mathcal{D}}(\alpha, \beta) = \min_w \mathcal{L}(w, \alpha, \beta).$$

We can now pose the **dual** optimization problem:

$$\max_{\alpha, \beta : \alpha_i \geq 0} \theta_{\mathcal{D}}(\alpha, \beta) = \max_{\alpha, \beta : \alpha_i \geq 0} \min_w \mathcal{L}(w, \alpha, \beta).$$

$$d^* = \max_{\alpha, \beta : \alpha_i \geq 0} \min_w \mathcal{L}(w, \alpha, \beta) \leq \min_w \max_{\alpha, \beta : \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta) = p^*.$$

$$\textcolor{brown}{d}^* = p^*,$$

Suppose  $f$  and the  $g_i$ 's are convex,<sup>6</sup> and the  $h_i$ 's are affine.<sup>7</sup> Suppose further that the constraints  $g_i$  are (strictly) feasible; this means that there exists some  $w$  so that  $g_i(w) < 0$  for all  $i$ .

$$\sup_{y'}(x', y) \geq P(x', y') \geq \inf_x P(x, y') \quad \forall x', y'$$

$$\sup_y P(x, y) \geq \inf_x P(x, y') \quad \forall x, \forall y'$$

$$\sup_y P(x', y) \geq \sup_y \inf_x P(x, y) \quad \forall x'$$

Given a real valued function  $P(x, y) : X \times Y \rightarrow \mathbb{R}$  one has

$$\inf_{x \in X} \sup_{y \in Y} P(x, y) \geq \sup_{y \in Y} \inf_{x \in X} P(x, y)$$

To see this pick  $x' \in X$  and  $y' \in Y$ . Clearly  $\sup_{y'} P(x', y) \geq \inf_x P(x, y')$ , and since this is true for all  $y' \in Y$  we have  $\sup_{y'} P(x', y) \geq \sup_{y'} \inf_x P(x, y)$ , by definition of the supremum. Similarly, since this is true for all  $x' \in X$  we have  $\inf_x \sup_{y'} P(x, y) \geq \sup_{y'} \inf_x P(x, y)$  by definition of the infimum.

\*

# KKT conditions

Under our above assumptions, there must exist  $w^*, \alpha^*, \beta^*$  so that  $w^*$  is the solution to the primal problem,  $\alpha^*, \beta^*$  are the solution to the dual problem, and moreover  $p^* = d^* = \mathcal{L}(w^*, \alpha^*, \beta^*)$ . Moreover,  $w^*, \alpha^*$  and  $\beta^*$  satisfy the **Karush-Kuhn-Tucker (KKT) conditions**, which are as follows:

$$\begin{aligned}\frac{\partial}{\partial w_i} \mathcal{L}(w^*, \alpha^*, \beta^*) &= 0, \quad i = 1, \dots, d \\ \frac{\partial}{\partial \beta_i} \mathcal{L}(w^*, \alpha^*, \beta^*) &= 0, \quad i = 1, \dots, l \\ \alpha_i^* g_i(w^*) &= 0, \quad i = 1, \dots, k \\ g_i(w^*) &\leq 0, \quad i = 1, \dots, k \\ \alpha^* &\geq 0, \quad i = 1, \dots, k\end{aligned}$$

Moreover, if some  $w^*, \alpha^*, \beta^*$  satisfy the KKT conditions, then it is also a solution to the primal and dual problems.

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} ||w||^2 \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, \dots, n \end{aligned}$$

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} ||w||^2 - \sum_{i=1}^n \alpha_i [y^{(i)}(w^T x^{(i)} + b) - 1].$$

$$\nabla_w \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^n \alpha_i y^{(i)} x^{(i)} = 0$$

$$w = \sum_{i=1}^n \alpha_i y^{(i)} x^{(i)}.$$

$$\frac{\partial}{\partial b} \mathcal{L}(w, b, \alpha) = \sum_{i=1}^n \alpha_i y^{(i)} = 0.$$

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} - b \sum_{i=1}^n \alpha_i y^{(i)}.$$

■  $\mathcal{L}(w, b, \alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)}.$

$$\max_{\alpha} \quad W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle.$$

s.t.  $\alpha_i \geq 0, \quad i = 1, \dots, n$

$$\sum_{i=1}^n \alpha_i y^{(i)} = 0,$$

Sequential minimal optimization

$$\begin{aligned} w^T x + b &= \left( \sum_{i=1}^n \alpha_i y^{(i)} x^{(i)} \right)^T x + b \\ &= \sum_{i=1}^n \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b. \end{aligned}$$

**Karush-Kuhn-Tucker (KKT) conditions**, which are as follows:

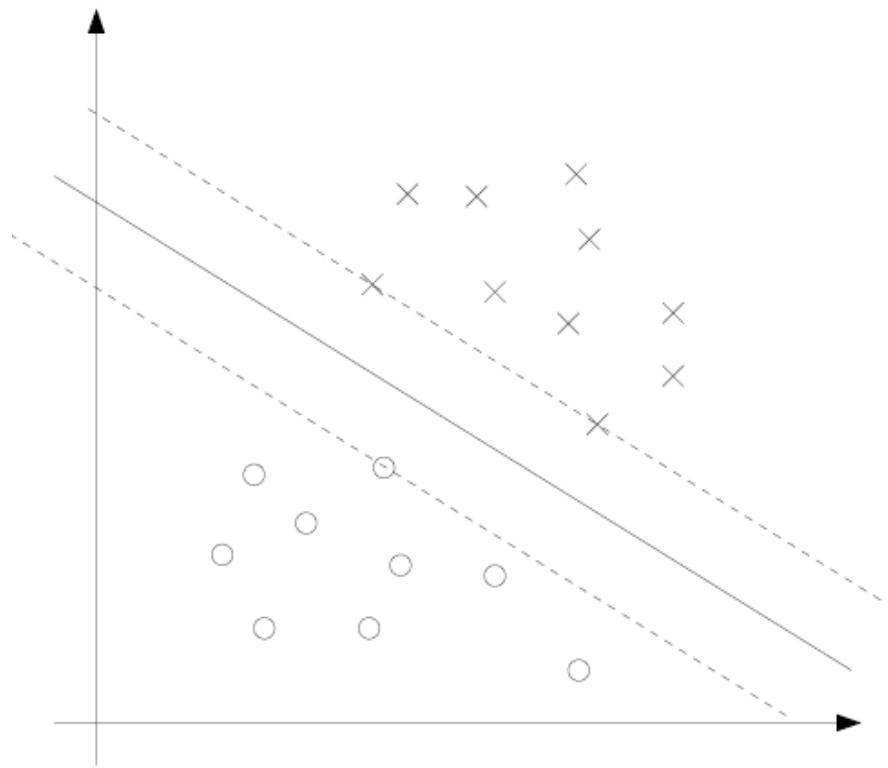
$$\frac{\partial}{\partial w_i} \mathcal{L}(w^*, \alpha^*, \beta^*) = 0, \quad i = 1, \dots, d$$

$$\alpha_i^* g_i(w^*) = 0, \quad i = 1, \dots, k$$

$$g_i(w^*) \leq 0, \quad i = 1, \dots, k$$

$$\alpha^* \geq 0, \quad i = 1, \dots, k$$

$$g_i(w) = -y^{(i)}(w^T x^{(i)} + b) + 1 \leq 0.$$



**support vectors**

$$\begin{aligned}
& \min_{\gamma, w, b} && \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \\
& \text{s.t.} && y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \dots, n \\
& && \xi_i \geq 0, \quad i = 1, \dots, n.
\end{aligned}$$

$$\mathcal{L}(w, b, \xi, \alpha, r) = \frac{1}{2} w^T w + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i [y^{(i)}(x^T w + b) - 1 + \xi_i] - \sum_{i=1}^n r_i \xi_i.$$

$$w = \sum_{i=1}^n \alpha_i y^{(i)} x^{(i)}. \quad \sum_{i=1}^n \alpha_i y^{(i)} = 0. \quad \alpha = C1 - r$$

Complementary conditions:

$$\alpha_i [y^{(i)}(x^T w + b) - 1 + \xi_i] = 0 \quad r_i \xi_i = 0$$

Hence at optimality we have  $w = \sum_{i=1}^n \alpha_i y^{(i)} x^{(i)}$ , and  $\alpha_i$  is nonzero only if  $[y^{(i)}(x^T w + b) - 1 + \xi_i] = 0$ . Such points  $i$  are called support points

- For support point  $i$ , if  $\xi_i = 0$ , then  $x_i$  lies on edge of margin, and  $\alpha_i \in (0, C]$ ;
- For support point  $i$ , if  $\xi_i \neq 0$ , then  $x_i$  lies on wrong side of margin, and  $\alpha_i = C$

$$\begin{aligned}
\max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \\
\text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, n \\
& \sum_{i=1}^n \alpha_i y^{(i)} = 0,
\end{aligned}$$

the KKT dual-complementarity conditions

$$\begin{aligned}
\alpha_i = 0 \quad & \Rightarrow \quad y^{(i)}(w^T x^{(i)} + b) \geq 1 \\
\alpha_i = C \quad & \Rightarrow \quad y^{(i)}(w^T x^{(i)} + b) \leq 1 \\
0 < \alpha_i < C \quad & \Rightarrow \quad y^{(i)}(w^T x^{(i)} + b) = 1.
\end{aligned}$$

# SVM as Logistic regression

- Svm loss

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

$$y_n t_n \geq 1, \quad \xi_n = 0,$$

$$\sum_{n=1}^N E_{\text{SV}}(y_n t_n) + \lambda \|\mathbf{w}\|^2$$

where  $\lambda = (2C)^{-1}$ , and  $E_{\text{SV}}(\cdot)$  is the *hinge* error function defined by

$$E_{\text{SV}}(y_n t_n) = [1 - y_n t_n]_+ \quad \ell(y) = \max(0, 1 - t \cdot y)$$

# SVM as Logistic regression

- Svm loss

$$\sum_{n=1}^N E_{\text{SV}}(y_n t_n) + \lambda \|\mathbf{w}\|^2$$

$$E_{\text{SV}}(y_n t_n) = [1 - y_n t_n]_+$$

- Logistic regression loss

$$\sum_{n=1}^N E_{\text{LR}}(y_n t_n) + \lambda \|\mathbf{w}\|^2.$$

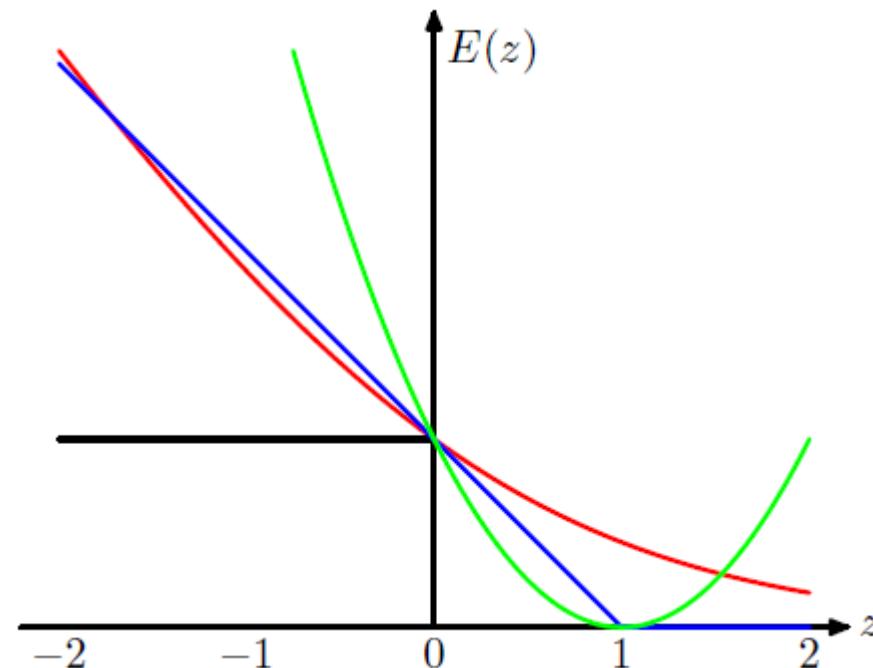
$$E_{\text{LR}}(yt) = \ln(1 + \exp(-yt)).$$

$$\begin{cases} \ln \sigma(t) & y=1 \\ -\ln(1-\sigma(t)) & y=-1 \end{cases}$$

$$-\ln \frac{1}{1+e^{-t}} = \ln(1+e^{-t})$$

$$-\ln\left(1 - \frac{1}{1+e^{-t}}\right) = -\ln\left(\frac{e^{-t}}{1+e^{-t}}\right) =$$
$$=\ln\left(\frac{1+e^{-t}}{e^{-t}}\right) = \ln\left(\frac{e^t+1}{e^t}\right)$$

**Figure 7.5** Plot of the ‘hinge’ error function used in support vector machines, shown in blue, along with the error function for logistic regression, rescaled by a factor of  $1/\ln(2)$  so that it passes through the point  $(0, 1)$ , shown in red. Also shown are the misclassification error in black and the squared error in green.



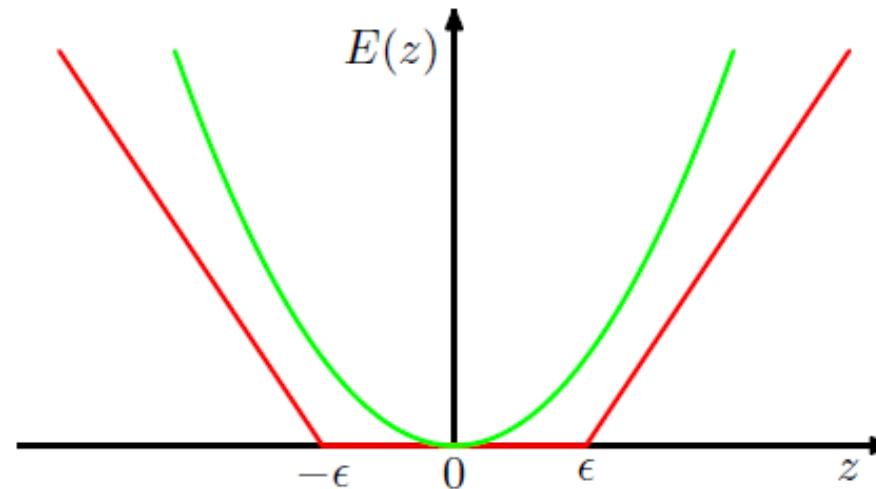
# SVM for regression

$$\frac{1}{2} \sum_{n=1}^N \{y_n - t_n\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2.$$

$$E_\epsilon(y(\mathbf{x}) - t) = \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \epsilon; \\ |y(\mathbf{x}) - t| - \epsilon, & \text{otherwise} \end{cases}$$

$$C \sum_{n=1}^N E_\epsilon(y(\mathbf{x}_n) - t_n) + \frac{1}{2} \|\mathbf{w}\|^2$$

**Figure 7.6** Plot of an  $\epsilon$ -insensitive error function (in red) in which the error increases linearly with distance beyond the insensitive region. Also shown for comparison is the quadratic error function (in green).



## Logistic regression vs. SVMs

$n$  = number of features ( $x \in \mathbb{R}^{n+1}$ ),  $m$  = number of training examples

If  $n$  is large (relative to  $m$ ):  $n \geq m$ ;  $n=10000$ ;  $m=10, \dots, 1000$

Use logistic regression, or SVM without a kernel ("linear kernel")

If  $n$  is small,  $m$  is intermediate:  $n=1, \dots, 1000$ ;  $m=10, \dots, 10000$

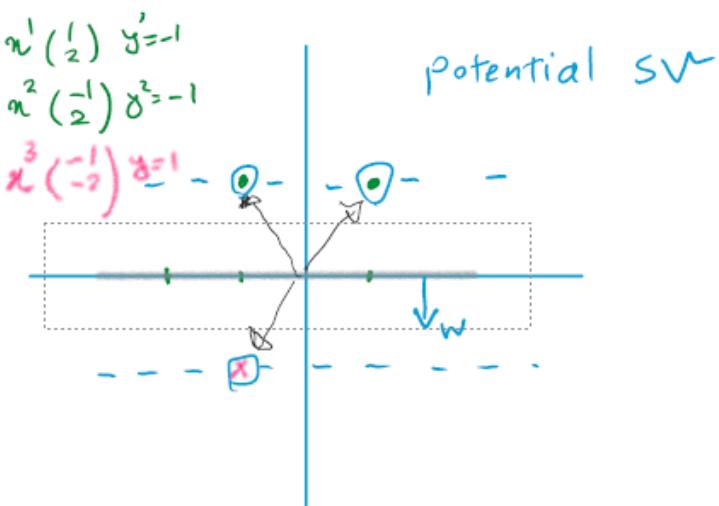
Use SVM with Gaussian kernel

If  $n$  is small,  $m$  is large:  $n=1, \dots, 1000$ ;  $m \geq 50000$

Create/add more features, then use logistic regression or SVM without a kernel

Neural network likely to work well for most of these settings, but may be slower to train.

# SVM Example



$$\begin{aligned}
 L(w, b, \alpha) &= \frac{1}{2} \|w\|_2^2 - \sum_{i=1}^n \alpha_i (y_i (w^T x_i + b) - 1) \\
 &= \frac{1}{2} w^T w - \sum_{i=1}^n \alpha_i y_i w^T x_i - b \left( \sum_{i=1}^n \alpha_i y_i \right) + \sum_{i=1}^n \alpha_i
 \end{aligned}$$

$$① \quad \frac{\partial L}{\partial w} = w - \sum_{i=1}^n \alpha_i y_i x_i \Rightarrow w^* = \sum_{i=1}^n \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^n \alpha_i y_i = 0$$

解

$$① \quad w^* = \sum \alpha_i y_i x_i = -\alpha_1 \begin{pmatrix} 1 \\ 2 \end{pmatrix} - \alpha_2 \begin{pmatrix} -1 \\ 2 \end{pmatrix} + \alpha_3 \begin{pmatrix} -1 \\ -2 \end{pmatrix}$$

$$② \quad -\alpha_1 - \alpha_2 + \alpha_3 = 0$$

Dual

$$\max_{\alpha_1 \dots \alpha_n}$$

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} - b \sum_{i=1}^n \alpha_i y^{(i)}$$

$$\sum \alpha_i y_i = 0 \quad \alpha_i \geq 0 \quad i=1, 2, \dots, n$$

$$\max_{\alpha_1, \alpha_2, \alpha_3} -\frac{1}{2} \sum_{i=1}^3 \sum_{j=1}^3 \alpha_i \alpha_j y_i y_j m_i^T m_j + \sum_{i=1}^3 \alpha_i$$

$$m_1^T m_1 = 5$$

$$m_1^T m_3 = -5$$

$$m_2^T m_3 = -3$$

$$m_1^T m_2 = 3$$

$$m_2^T m_2 = 5$$

$$m_3^T m_3 = 5$$

$$(\max -\frac{1}{2} [5\alpha_1^2 + 6\alpha_1\alpha_2 + 10\alpha_1\alpha_3 + 5\alpha_2^2 + 6\alpha_2\alpha_3 + 5\alpha_3^2] + \alpha_1 + \alpha_2 + \alpha_3)$$

$$\begin{cases} \max_{\alpha_1, \alpha_2, \alpha_3} -\frac{1}{2} [5\alpha_1^2 + 6\alpha_1\alpha_2 + 10\alpha_1\alpha_3 + 5\alpha_2^2 + 6\alpha_2\alpha_3 + 5\alpha_3^2] + \alpha_1 + \alpha_2 + \alpha_3 \\ \quad \alpha_1 - \alpha_2 + \alpha_3 = \phi \end{cases}$$

$$\alpha_3 = \alpha_1 + \alpha_2$$

$$J = \max_{\alpha_1, \alpha_2} -\frac{1}{2} (20\alpha_1^2 + 32\alpha_1\alpha_2 + 16\alpha_2^2) + 2\alpha_1 + 2\alpha_2$$

$$\begin{cases} \frac{\partial J}{\partial \alpha_1} = -20\alpha_1 - 16\alpha_2 + 2 = \phi \Rightarrow \alpha_1 = -\frac{16\alpha_2 + 2}{20} \end{cases}$$

$$\begin{cases} \frac{\partial J}{\partial \alpha_2} = -16\alpha_1 - 16\alpha_2 + 2 = \phi \end{cases}$$

$$\boxed{\alpha_2 = \frac{1}{8}}$$

$$\boxed{\alpha_1 = \phi}$$

$$\boxed{\alpha_3 = \frac{1}{8}}$$

$$w^* = \sum_{i=1}^3 \alpha_i y_i n^i = \frac{1}{8} \begin{bmatrix} -1 \\ 2 \end{bmatrix} + \frac{1}{8} \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{1}{2} \end{bmatrix} \quad m = \frac{2}{\|w\|} = \frac{2}{\sqrt{\frac{1}{2}}} = 4$$

$$b^* =$$

$$y_2(n_2^T w + b^*) = 1 \quad b^* = -1 - \begin{bmatrix} -1 \\ 2 \end{bmatrix} \begin{bmatrix} 0 \\ -\frac{1}{2} \end{bmatrix}$$

$$n_2^T w + b = -1$$

$$b^* = -1 - n_n^T w$$

# References and further readings

- Alexandre Kowalczyk, SVM Tutorial, 11 may 2020, <https://www.svm-tutorial.com/>
- Alexandre Kowalczyk, “Support vector machines succinctly”, syncfusion, Inc,
- Martin Hafmann, “kernels and the kernel trick”, 11 may 2020, [https://cogsys.uni-bamberg.de/teaching/ss06/hs\\_svm/slides/SVM\\_and\\_Kernels.pdf](https://cogsys.uni-bamberg.de/teaching/ss06/hs_svm/slides/SVM_and_Kernels.pdf)
- Andrew NG., Machine Learning Course, Coursera, slide: Support Vector Machines
- Tengyu Ma and Andrew Ng, CS229 Lecture Notes, 2020, <http://cs229.stanford.edu/notes2020fall/notes2020fall/cs229-notes3.pdf>

- Ryan Tibshirani, Lecture 12: KKT Conditions, convex optimization course, spring 2015, <https://www.stat.cmu.edu/~ryantibs/convexopt-S15/scribes/12-kkt-scribed.pdf>
- Bishop , pattern recognition and machine learning