Machine Learning

Dr. Mehran Safayani

safayani@iut.ac.ir

safayani.iut.ac.ir

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

https://github.com/safayani/machine_learning_course

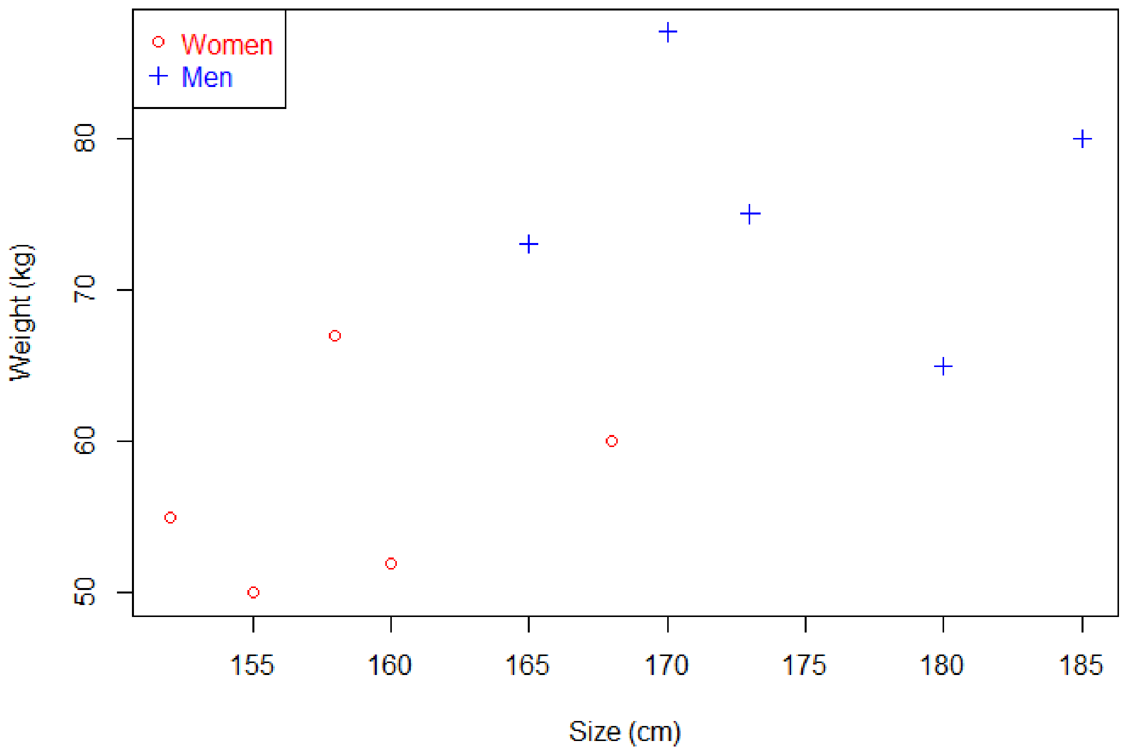
Department of Electrical and computer engineering, Isfahan university of technology, Isfahan, Iran

Machine Learning

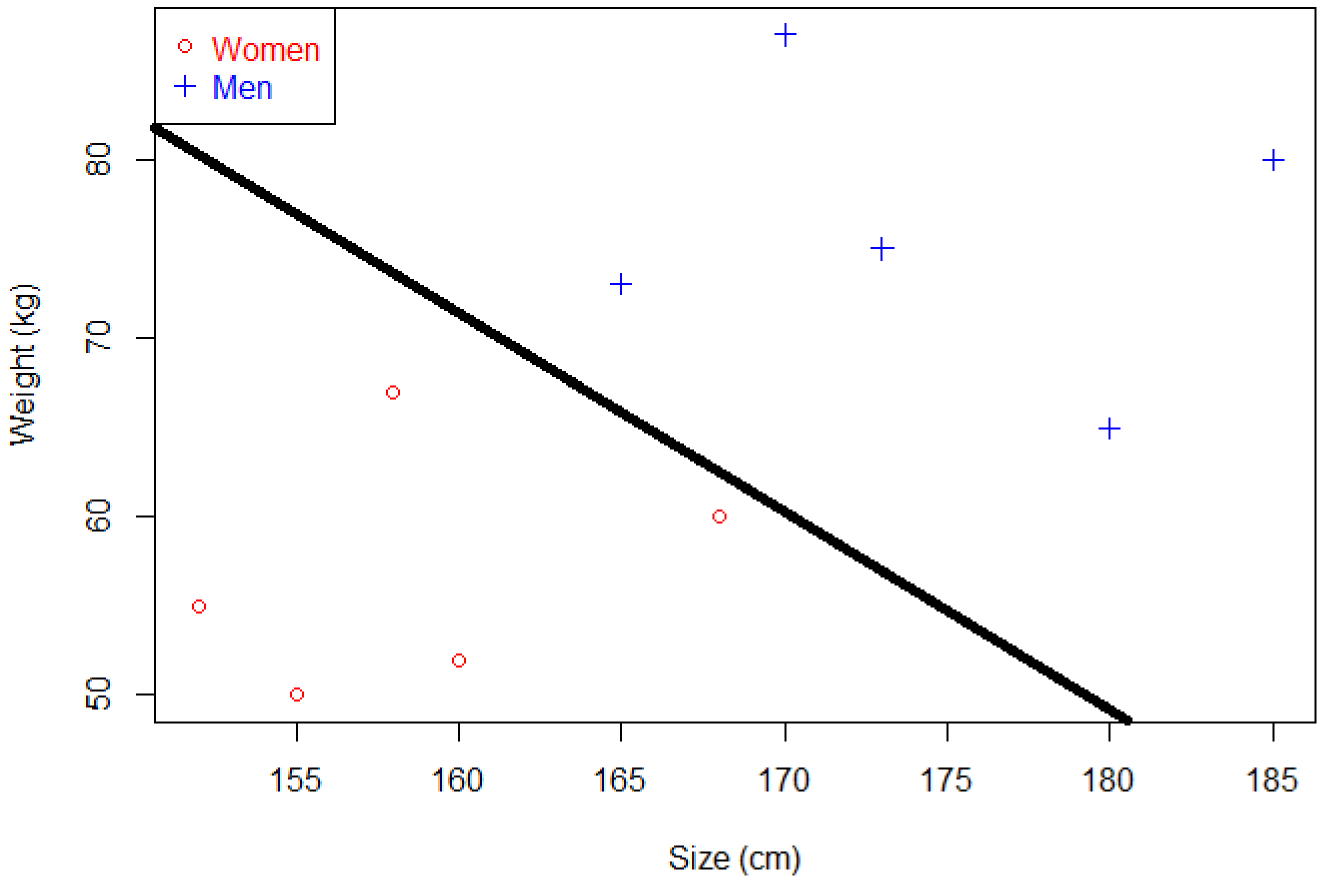
Support Vector Machine (SVM)

Part I: Motivation

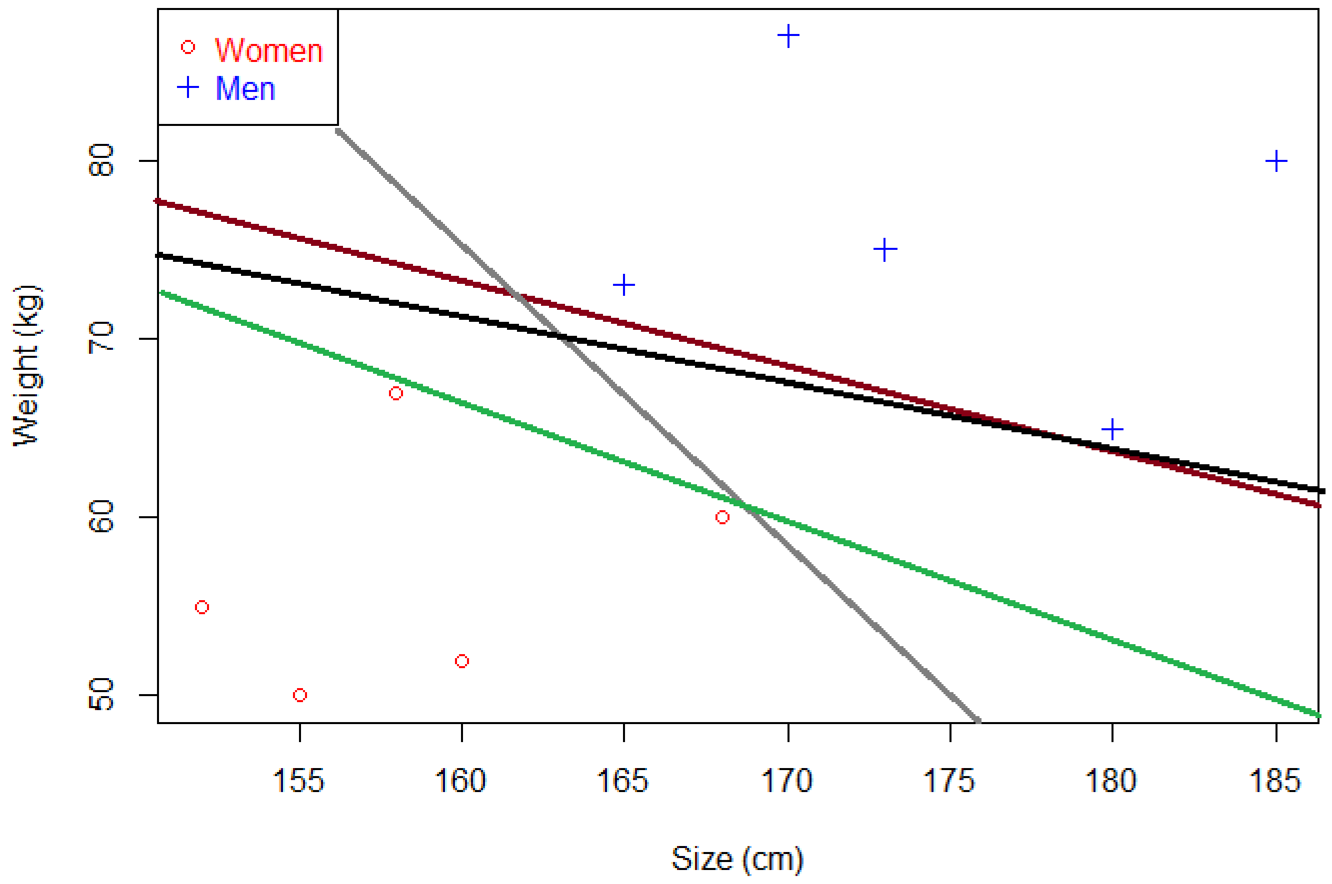
Mehran Safayani


Women
Men
Weight (kg)
50
60
70
80
155
160
165
170
175
180
185
Size (cm)

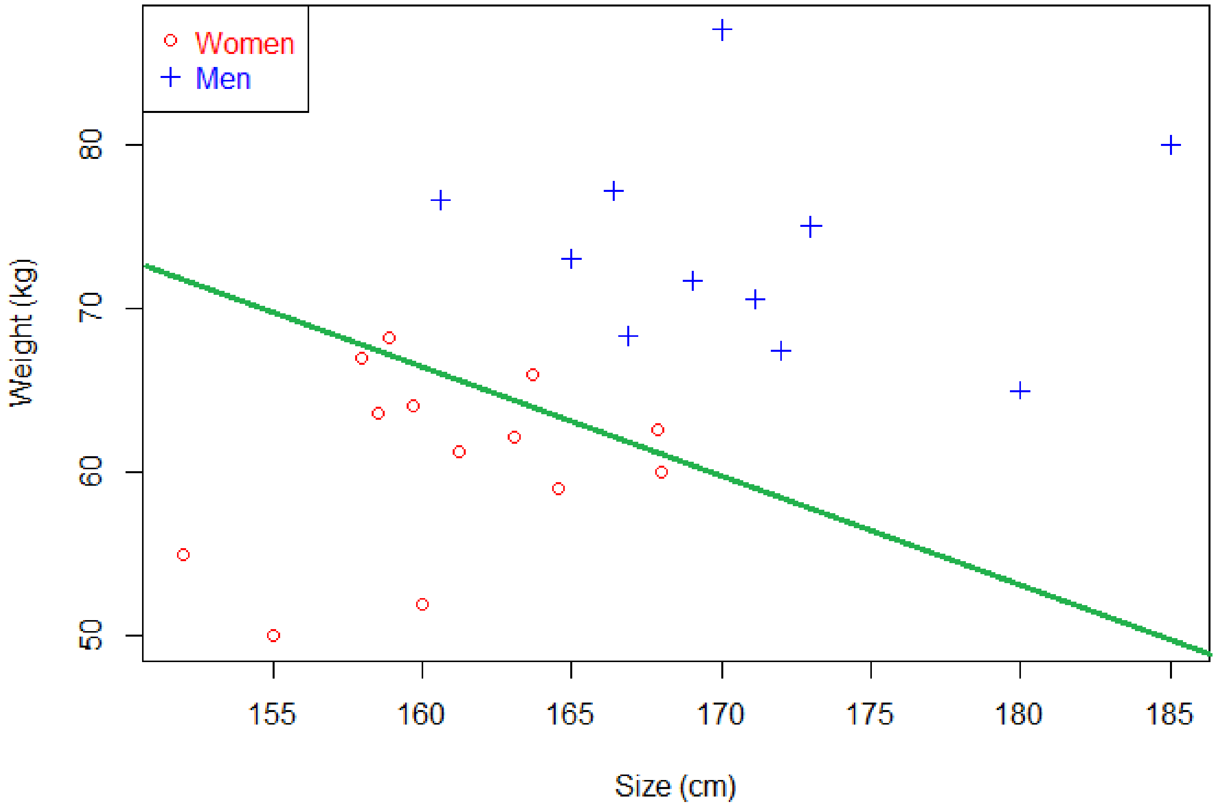
separating hyperplane



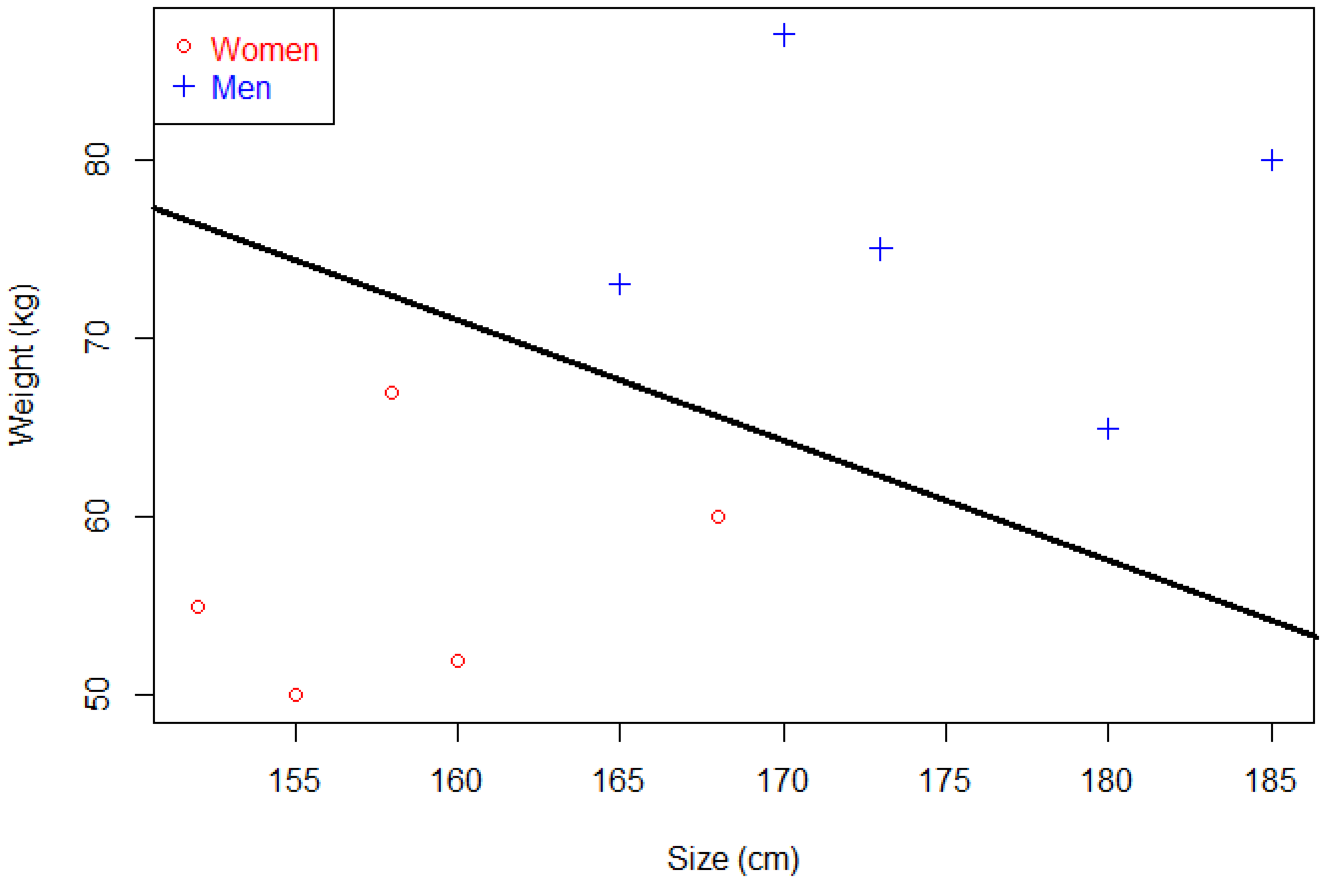
What is the *optimal* separating hyperplane?



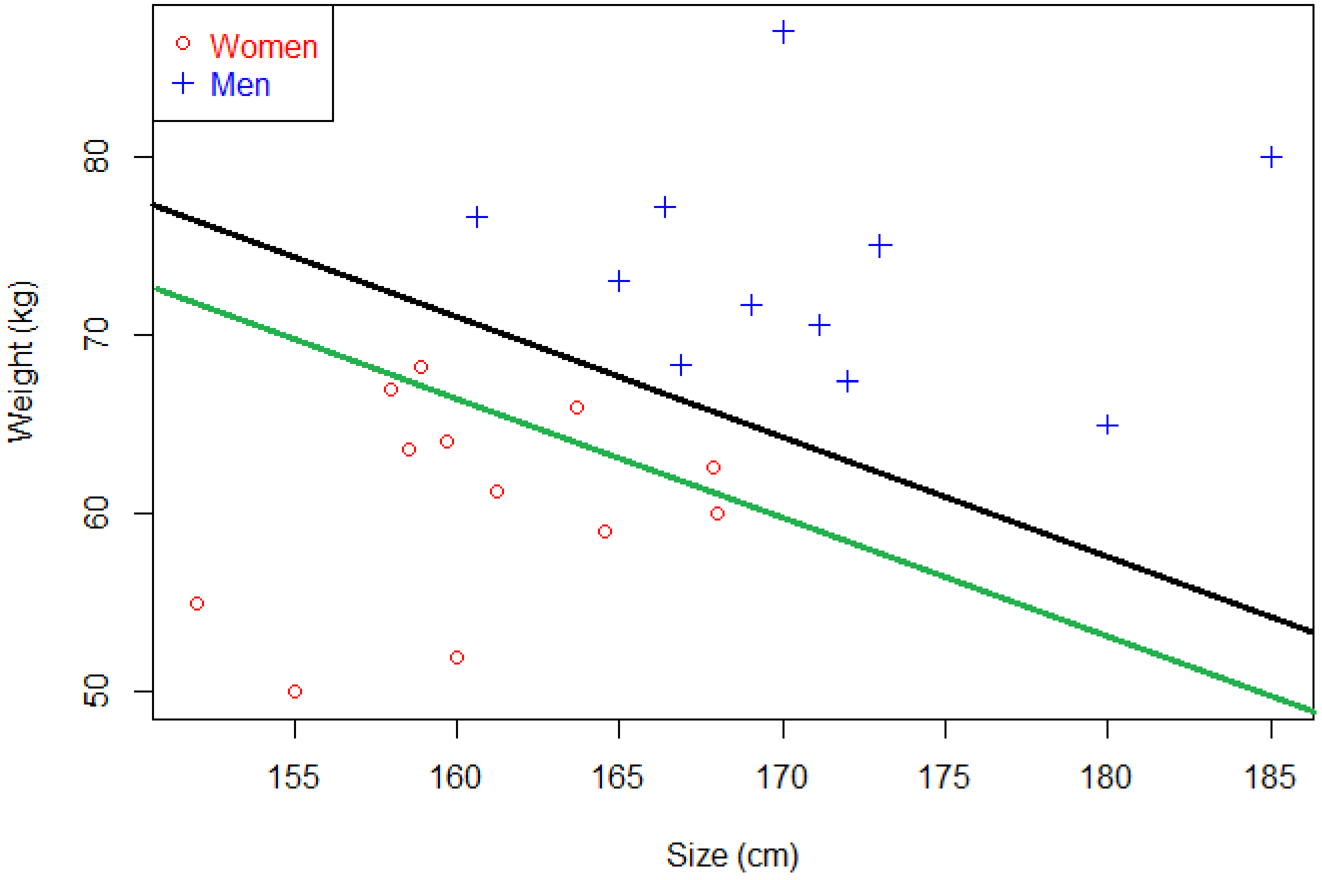
This hyperplane does not generalize well

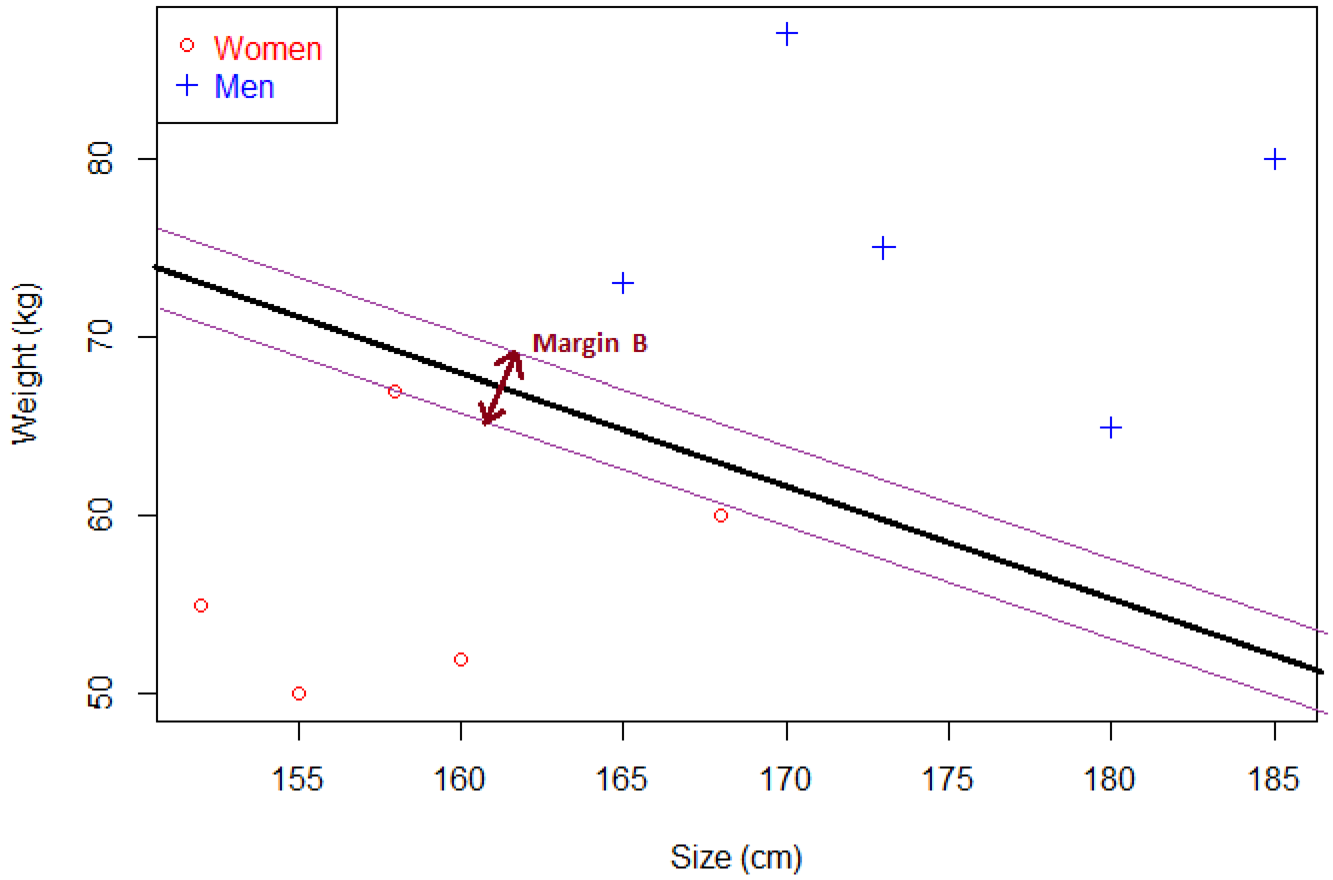


Optimal hyperplane



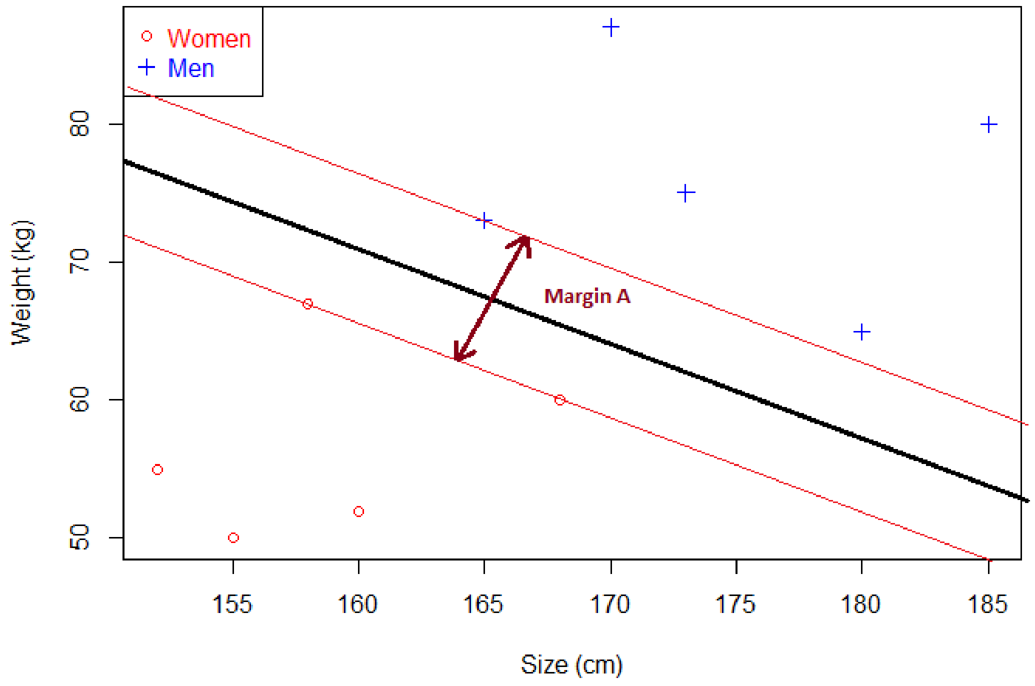
The black hyperplane classifies more accurately than the green one





Women
Men
Margin B
80
70
60
50
Weight (kg)
155
160
165
170
175
180
185
Size (cm)

The margin of our optimal hyperplane



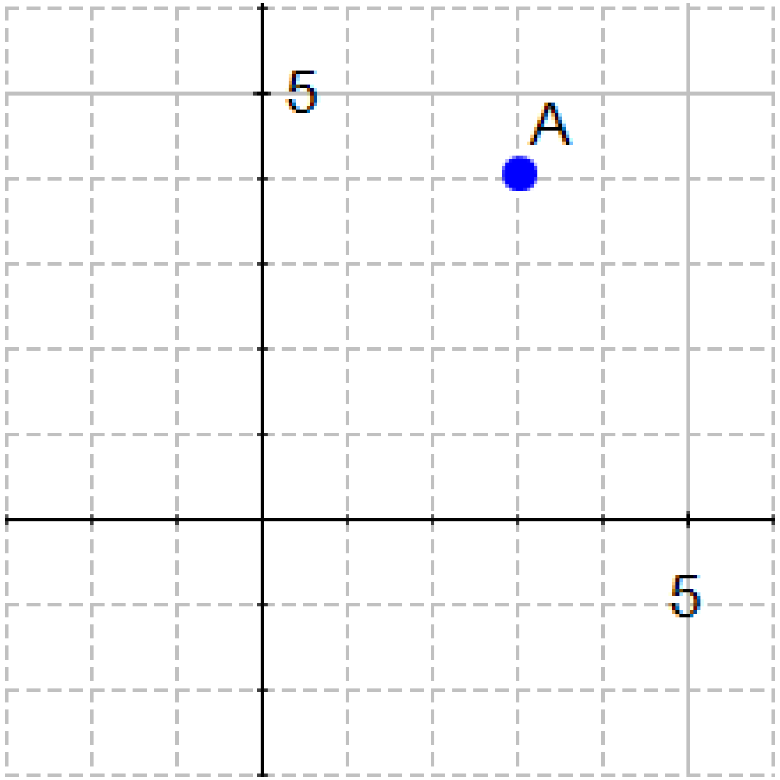
- the optimal hyperplane will be the one with the biggest margin.

Machine Learning

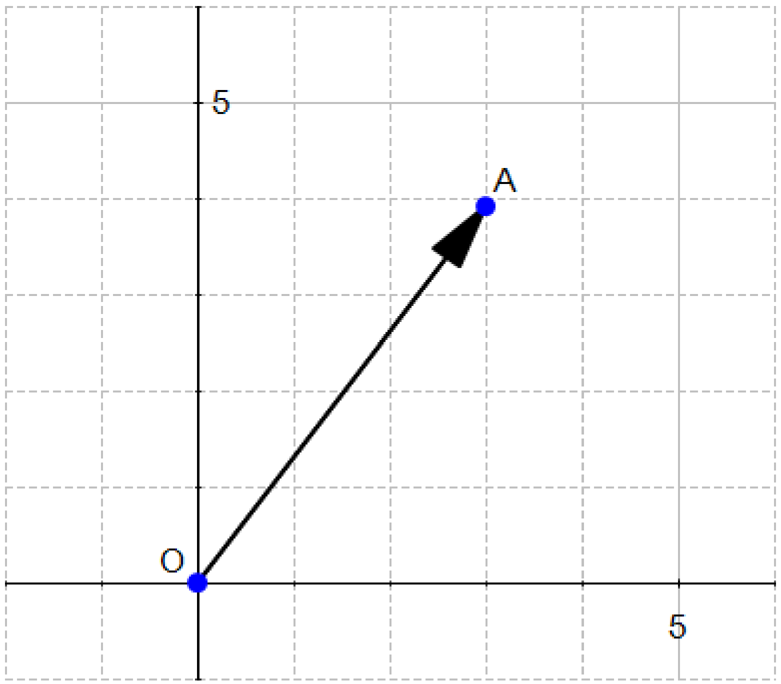
Support Vector Machine (SVM)

Part II: Reminder of linear algebra concepts

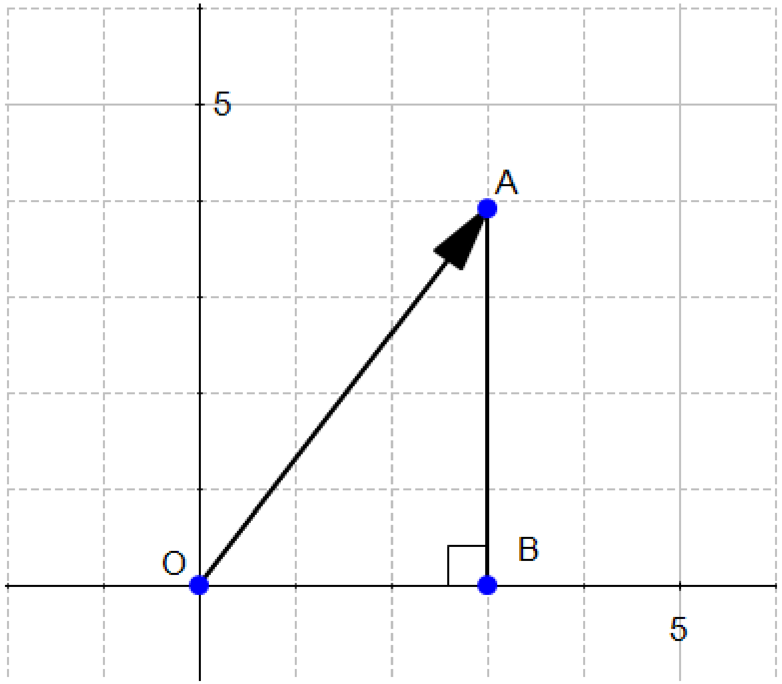
Mehran Safayani


5
A
5

- $\overrightarrow{OA}$



The magnitude



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

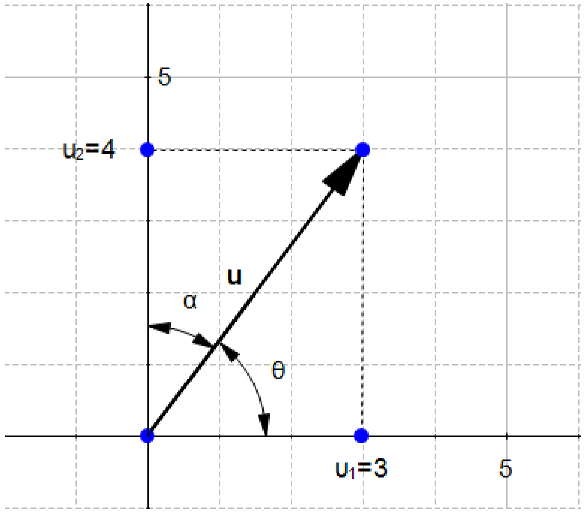
$$OA^2 = 3^2 + 4^2$$

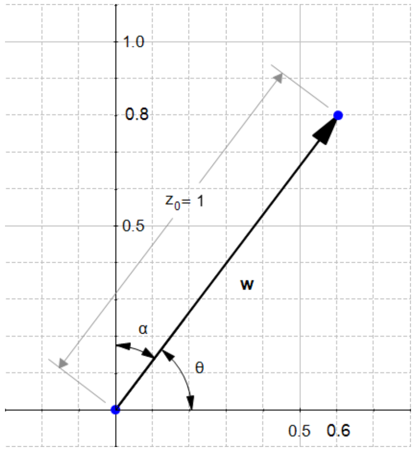
$$OA^2 = 25$$

$$OA = \sqrt{25}$$

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

The direction





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

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

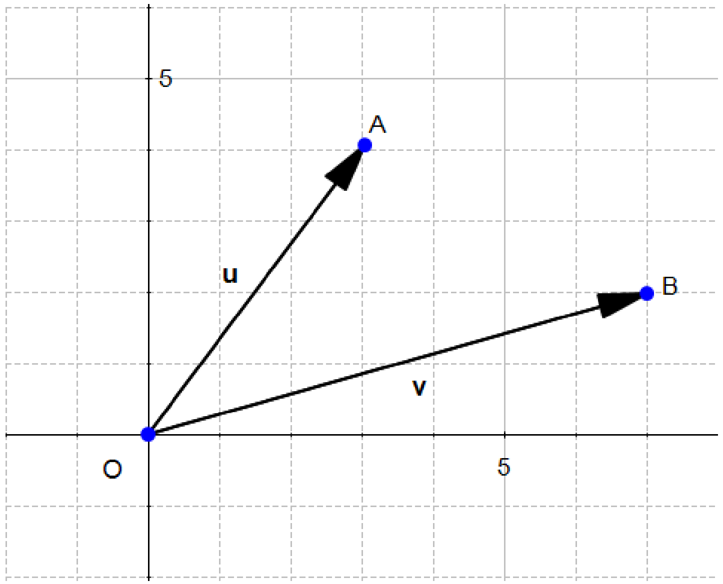
$$cos(\alpha) = \frac{u_2}{\|u\|} = \frac{4}{5} = 0.8$$

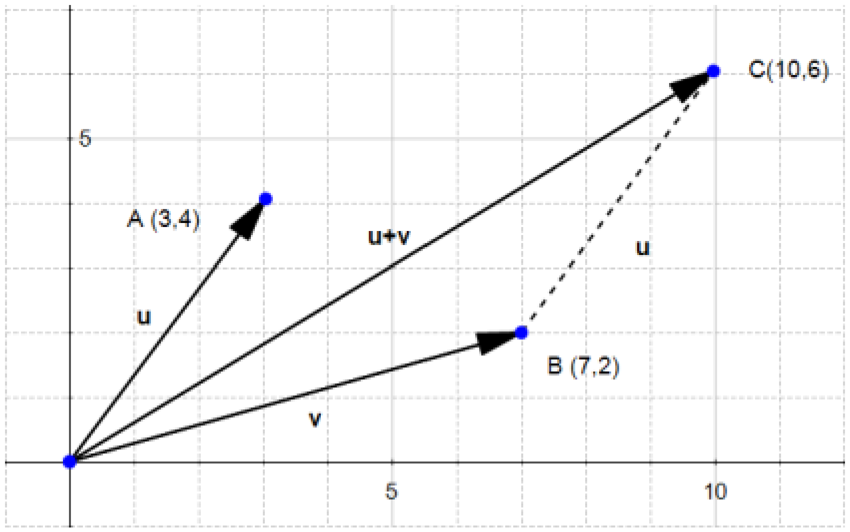
The direction of $\mathbf{u}(3,4)$ is the vector $\mathbf{w}(0.6, 0.8)$

The sum of two vectors

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

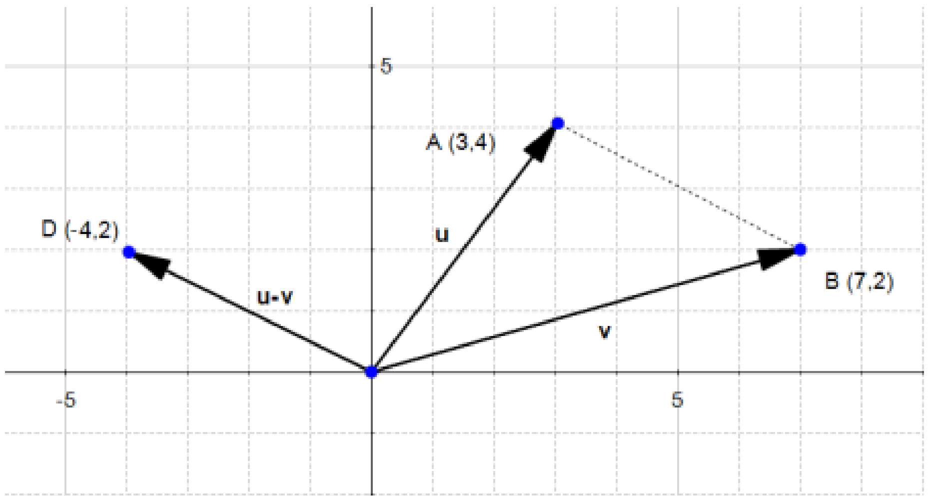
$$\mathbf{u} + \mathbf{v} = (u_1 + v_1, u_2 + v_2)$$

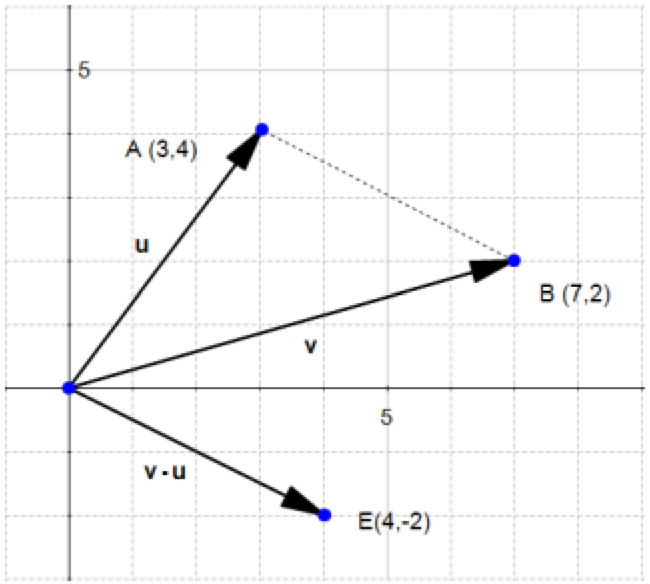




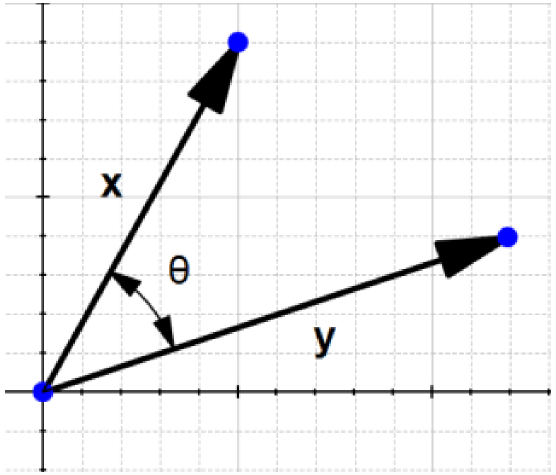
The difference between two vectors

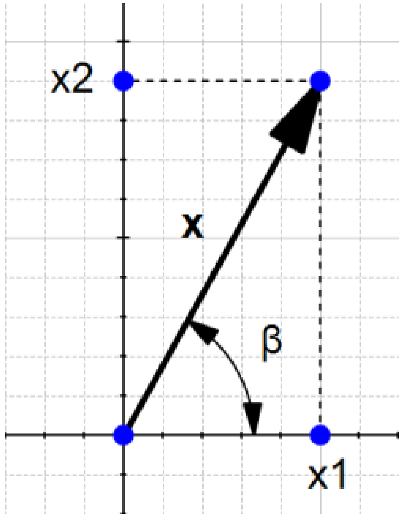
$$\mathbf{u} - \mathbf{v} = (u_1 - v_1, u_2 - v_2)$$

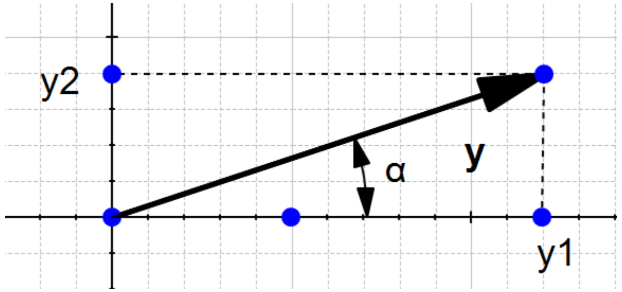


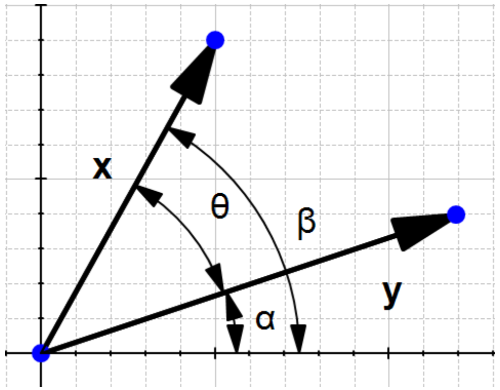


The dot product









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

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

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

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

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

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

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

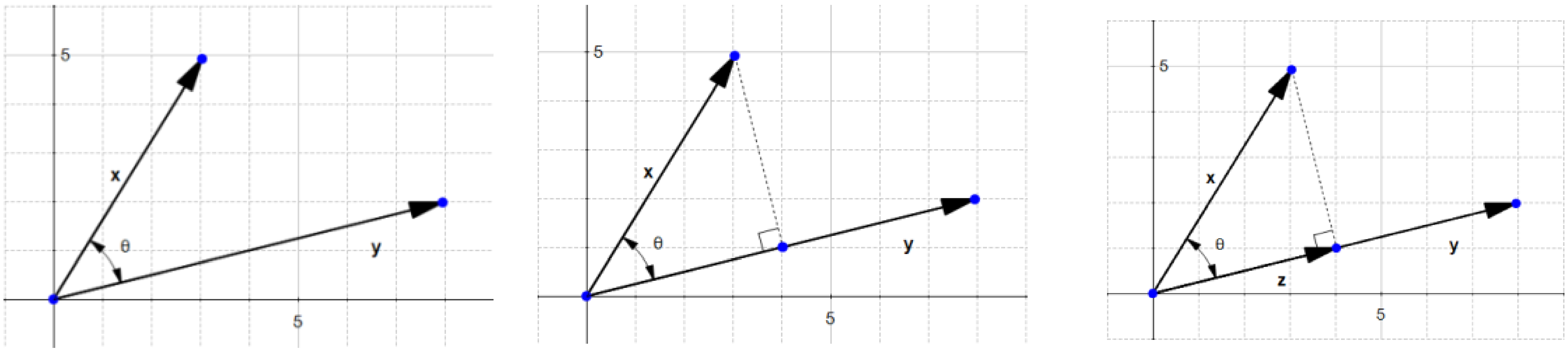
$$cos(\theta) = \frac{x_1y_1 + x_2y_2}{||x||||y||}$$

$$||x||||y||cos(\theta) = x_1y_1 + x_2y_2$$

$$\mathbf{x} \cdot \mathbf{y} = x_1y_1 + x_2y_2 = \sum_{i=1}^{2}(x_iy_i)$$

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

The orthogonal projection of a vector



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

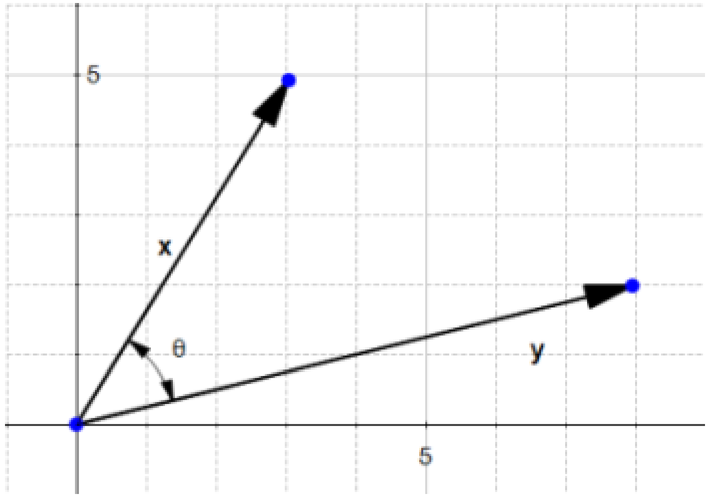
$$||z|| = ||x||cos(\theta)$$

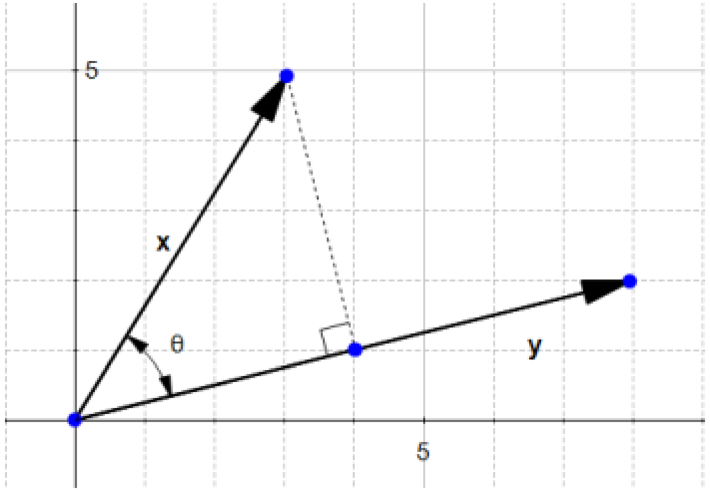
$$cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{||x||||y||}$$

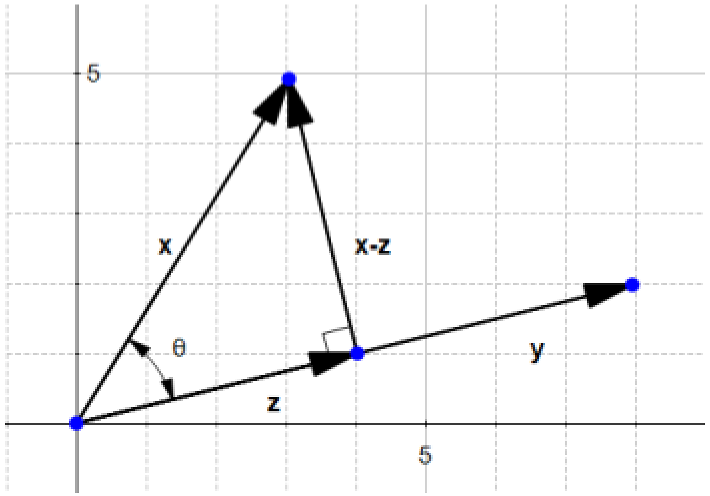
$$||z|| = ||x|| \frac{\mathbf{x} \cdot \mathbf{y}}{||x||||y||}$$

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

The orthogonal projection of a vector







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

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

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

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

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

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

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

$$y - ax - b = 0$$

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

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

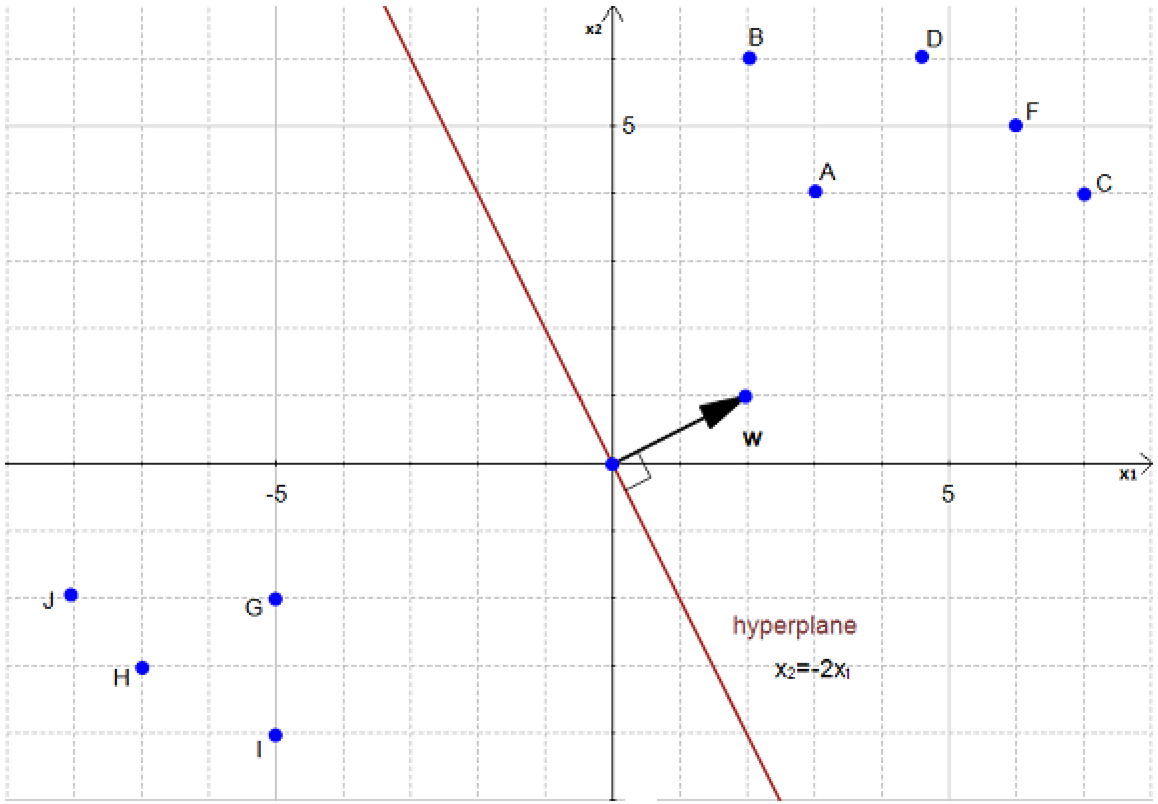
$$\mathbf{w}^T\mathbf{x} = y - ax - b$$

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



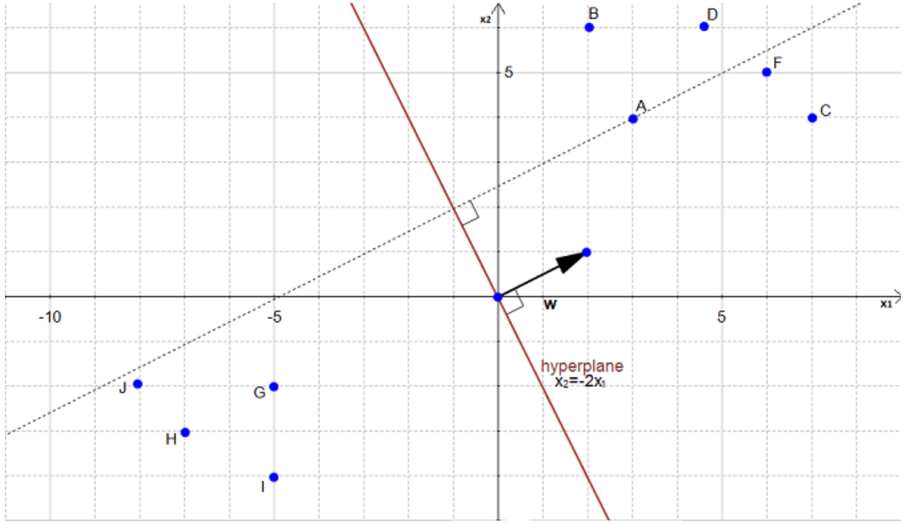
$$w_0 = 0$$

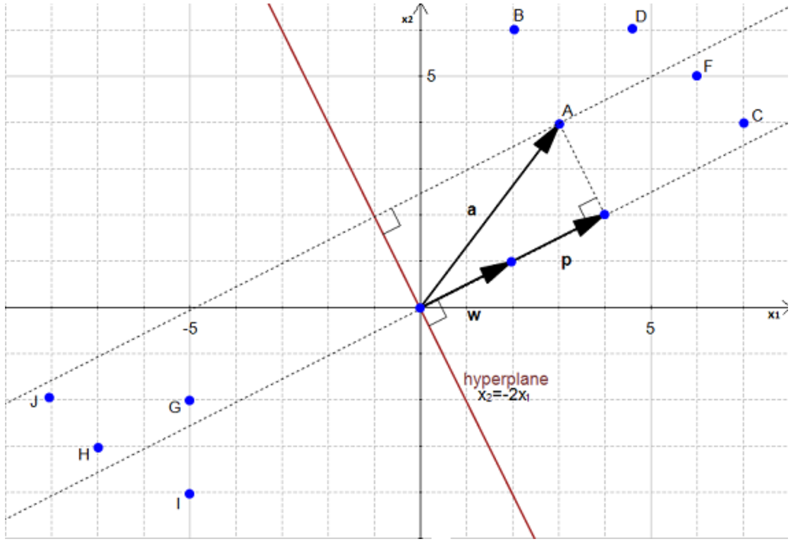
$$x_2 = -2x_1$$

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

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

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





$$\mathbf{w} = (2,1) \qquad \mathbf{a} = (3,4) \qquad \mathbf{u} = \left(\frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}}\right) \qquad \mathbf{p} = (\mathbf{u}\cdot\mathbf{a})\mathbf{u}$$

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

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

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

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

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

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

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

Machine Learning

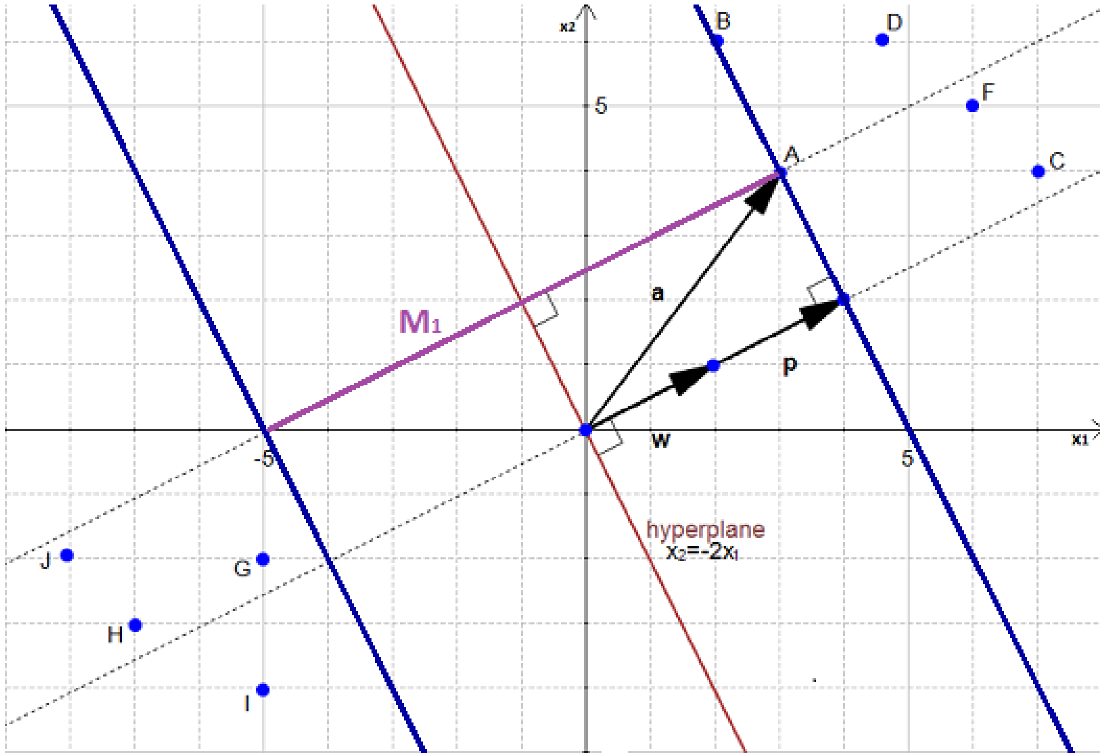
Support Vector Machine (SVM)

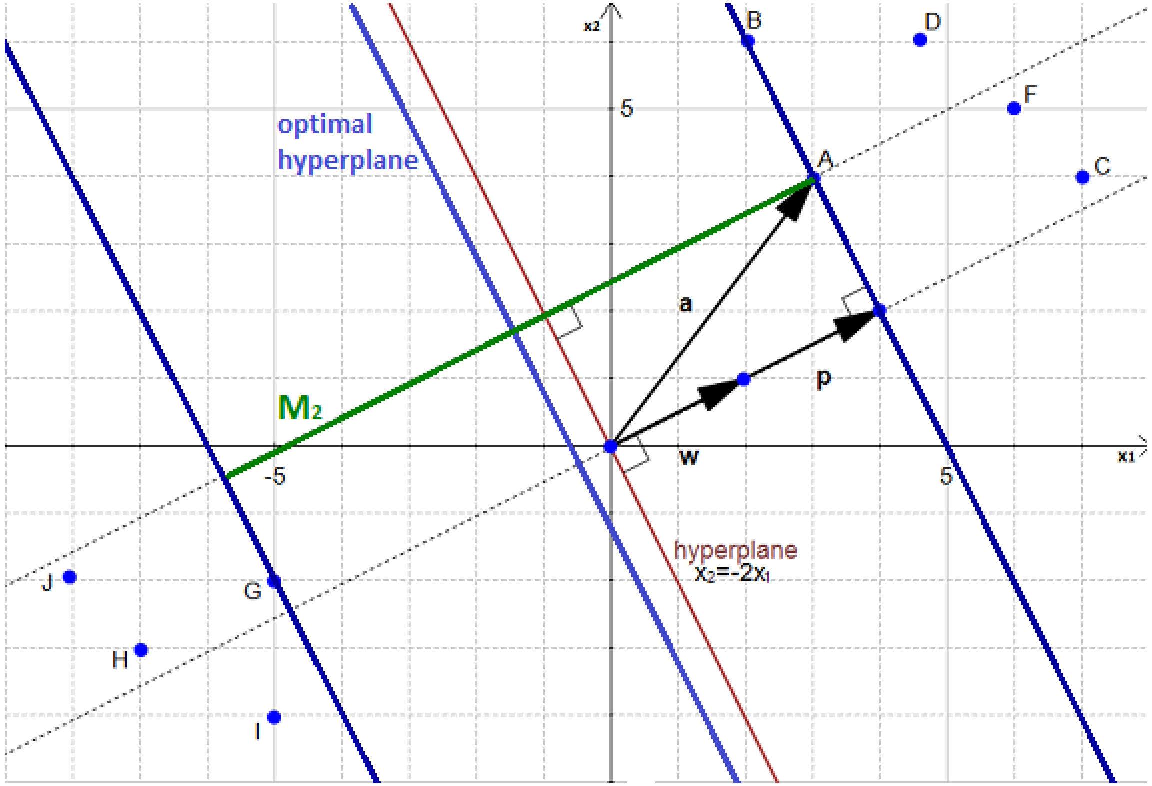
Part III: Linear SVM

Mehran Safayani

the optimal hyperplane

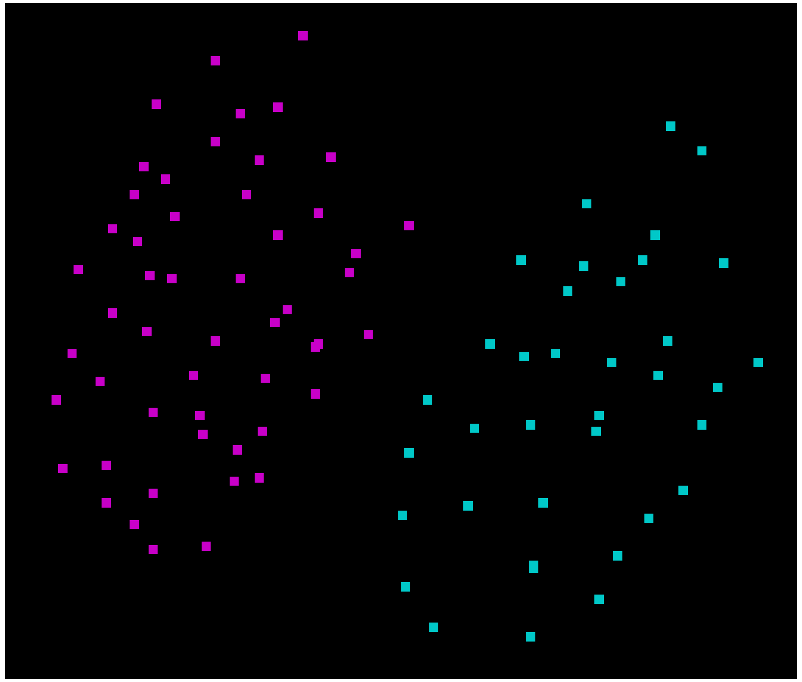
- **How to find the optimal hyperplane ?**



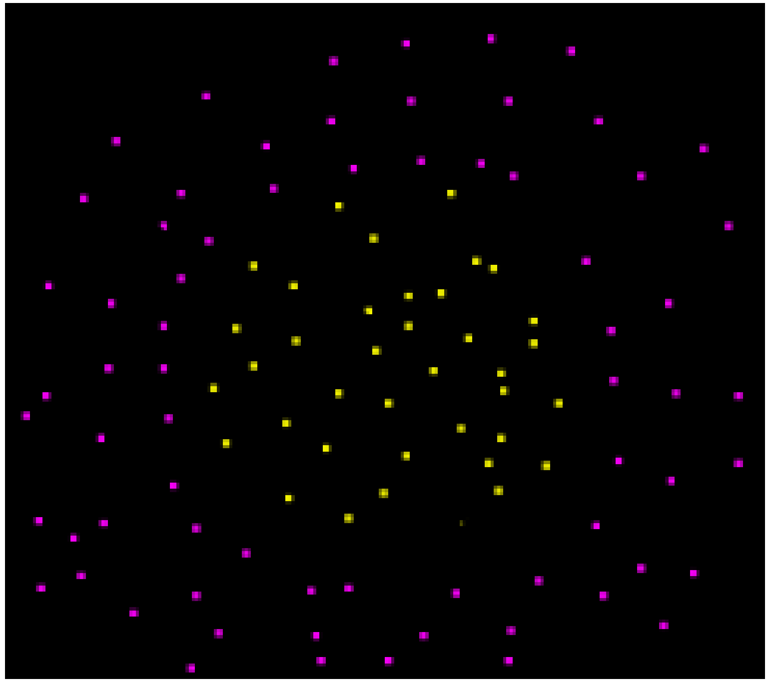


You have a dataset D and you want to classify it

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



Linearly separable data



Non linearly separable data

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

Taking another look at the hyperplane equation

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

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

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

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

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

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

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

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

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

Given a hyperplane H_0 separating the dataset and satisfying:

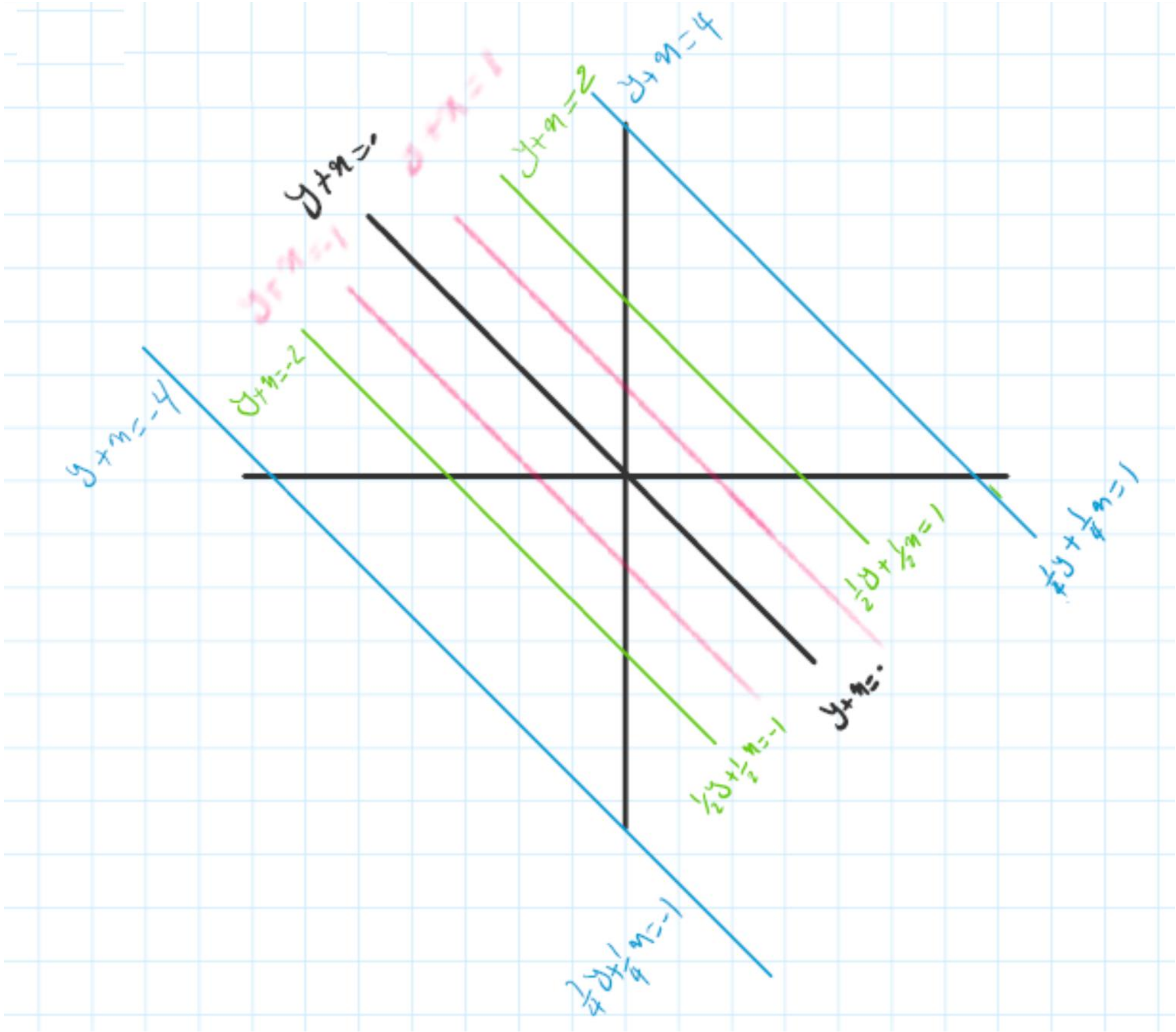
$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

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.



y+x=0
y+x=1
y+x=2
y+x=4
y+x=-1
y+x=-2
y+x=-4
½y+½x=1
¼y+¼x=1
y+x=0
½y+½x=-1
¼y+¼x=-1

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

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

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

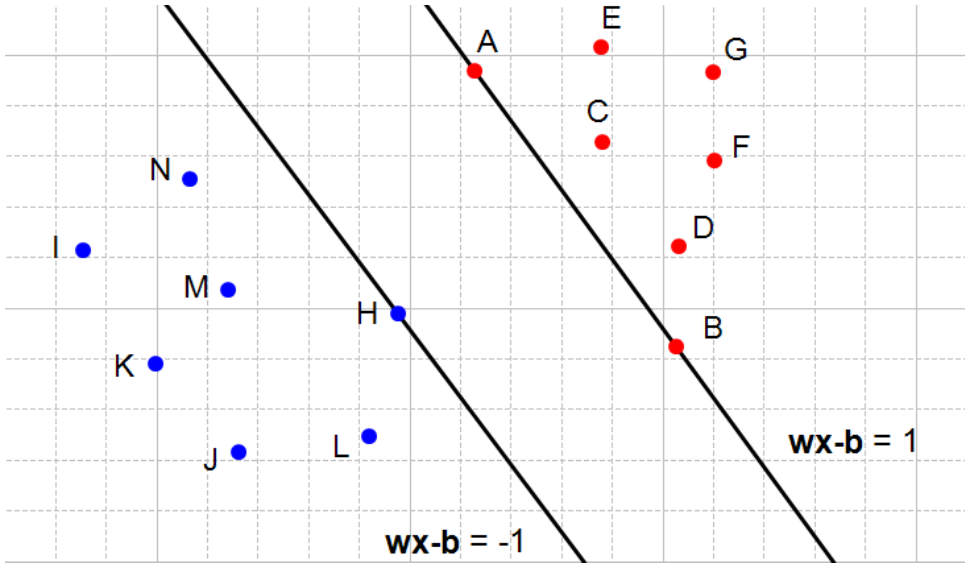
constraints:

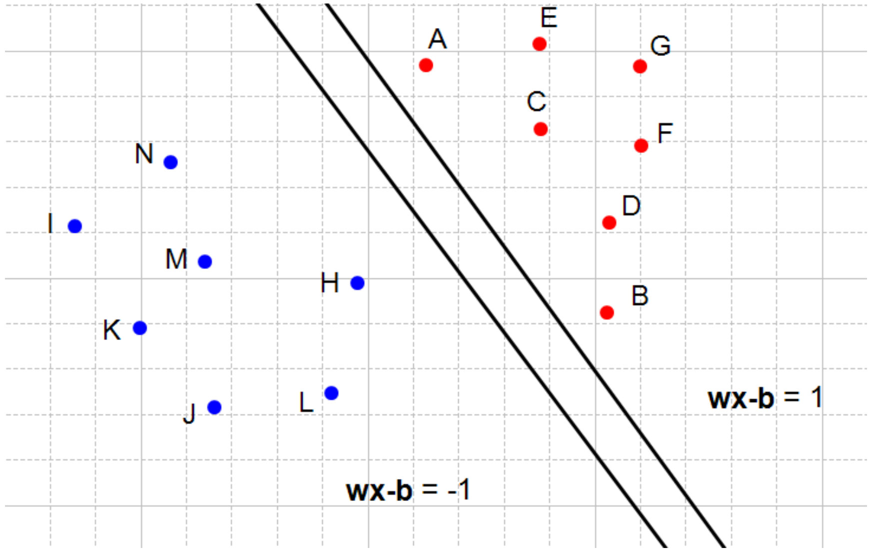
For each vector $\mathbf{x_i}$ either :

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

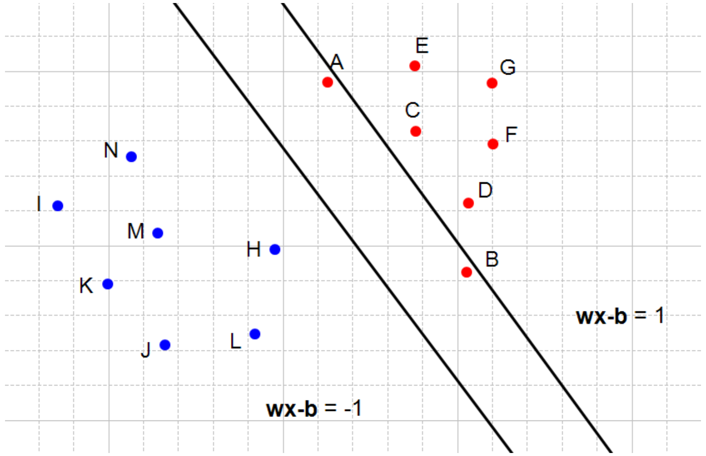
$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1 \text{ for } \mathbf{x_i} \text{ having the class } -1$$

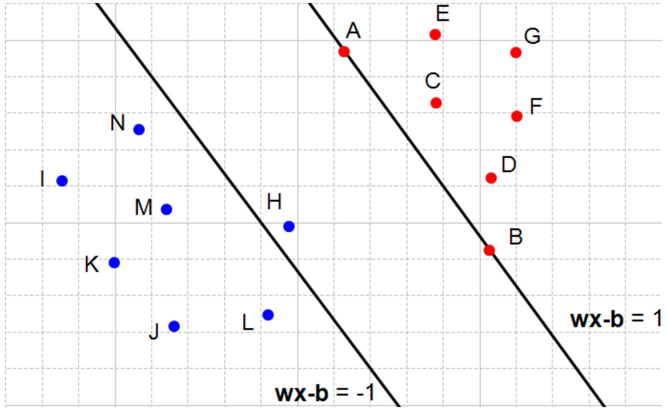
Understanding the constraints

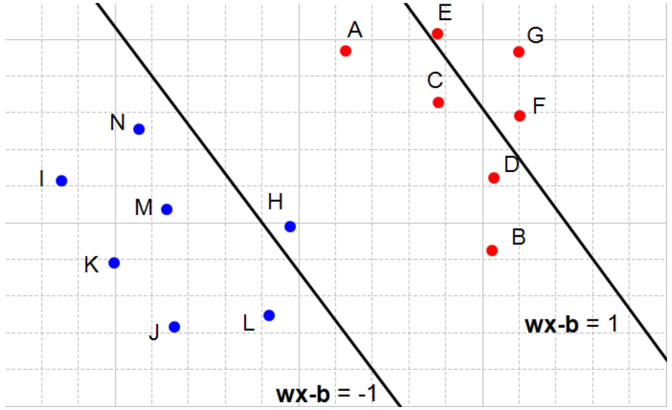




Understanding the constraints







Combining both constraints

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

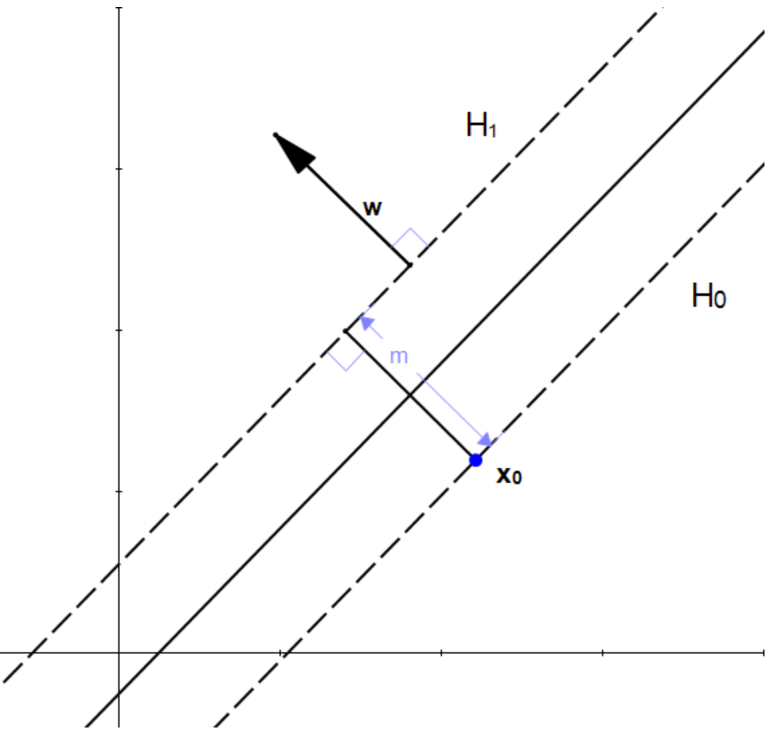
Maximize the distance between the two hyperplanes

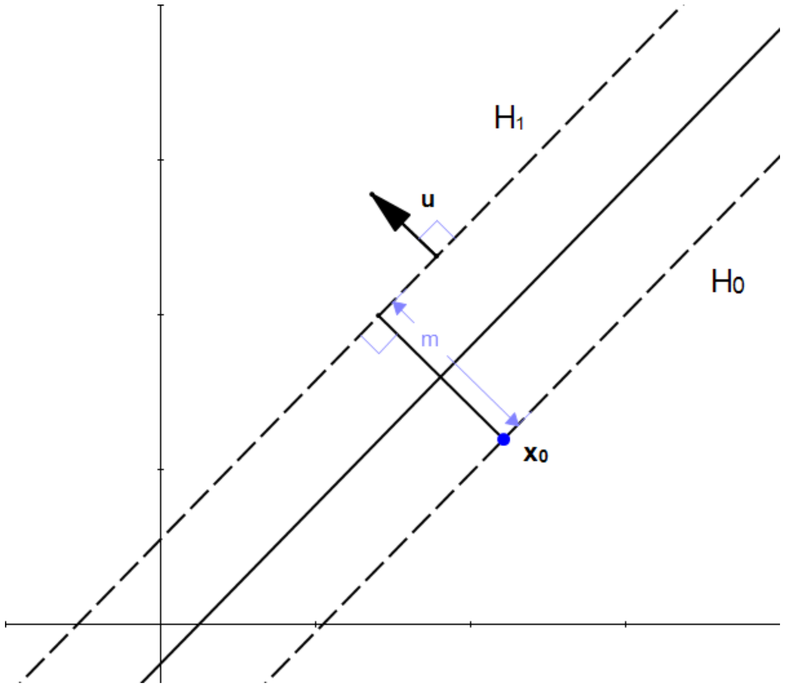
Finding the biggest margin, is the same thing as finding the optimal hyperplane.

- **a) What is the distance between our two hyperplanes ?**

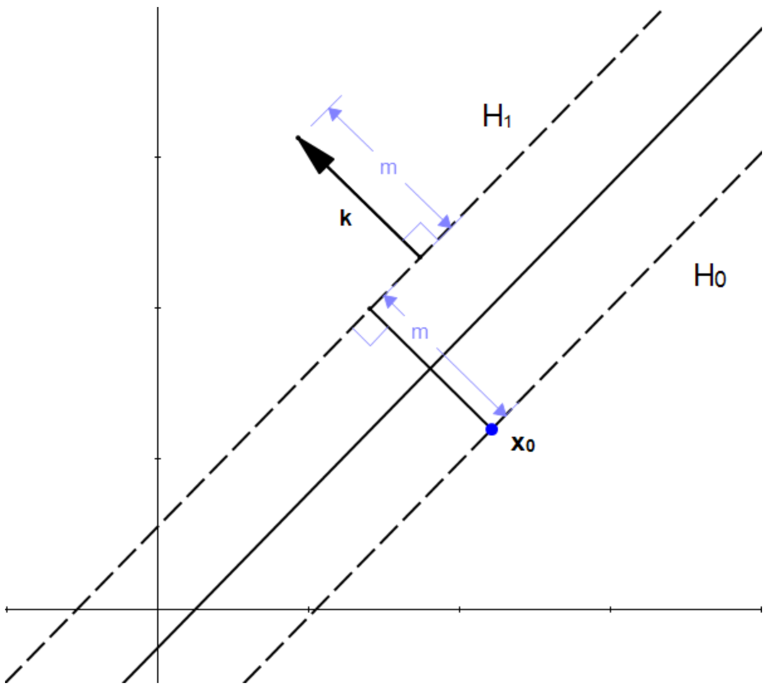
- $\mathcal{H}_0$ be the hyperplane having the equation $\mathbf{w} \cdot \mathbf{x} + b = -1$
- $\mathcal{H}_1$ be the hyperplane having the equation $\mathbf{w} \cdot \mathbf{x} + b = 1$
- $\mathbf{x}_0$ be a point in the hyperplane $\mathcal{H}_0$.

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

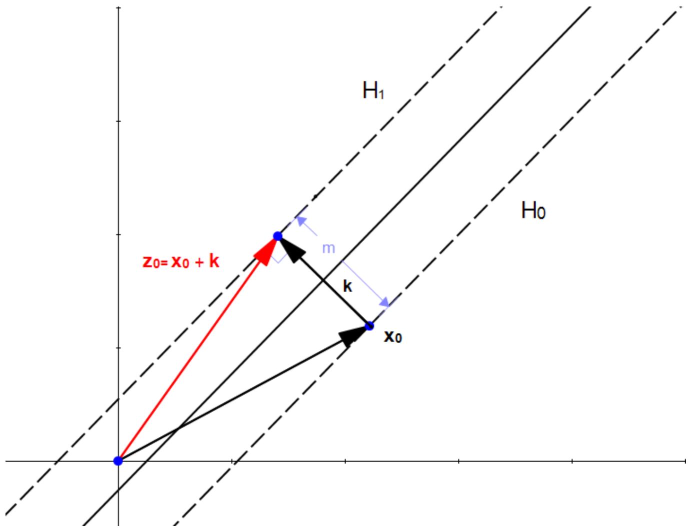




$$\mathbf{u} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$



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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

Maximizing the margin is the same thing as minimizing the norm of w

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

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

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

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

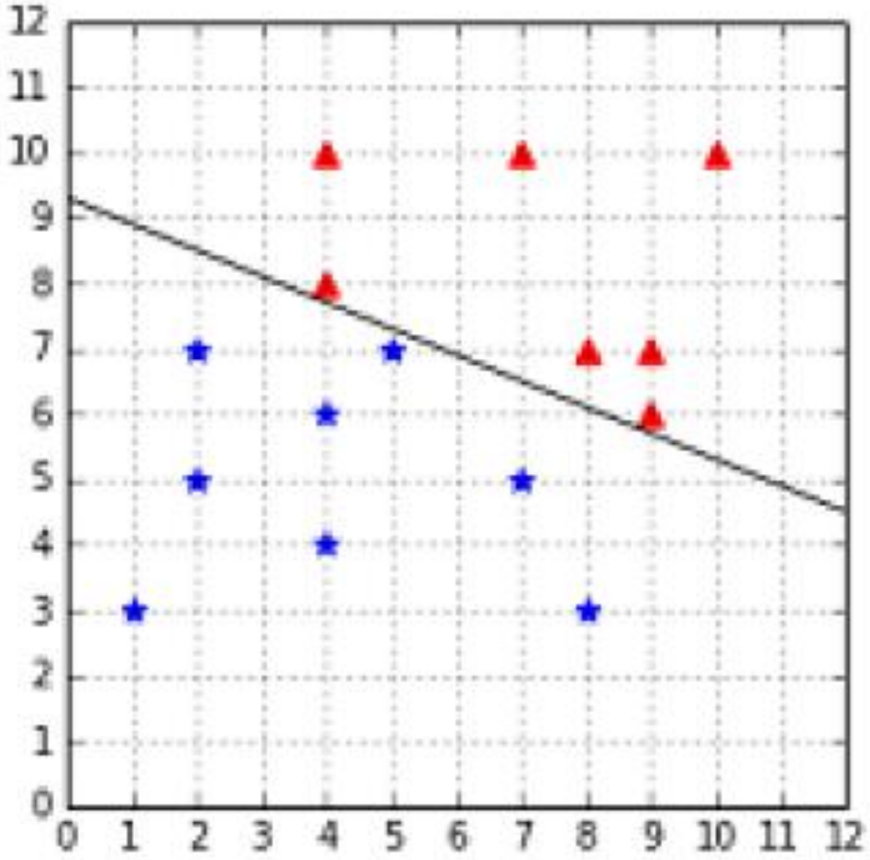
Machine Learning

Support Vector Machine (SVM)

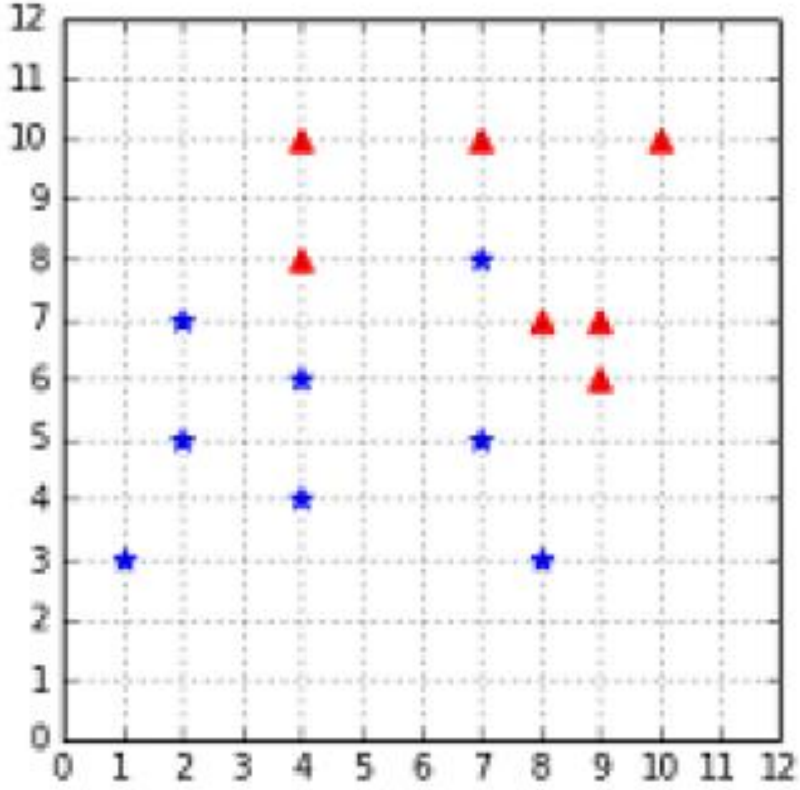
Part IV: Soft margin SVM

Mehran Safayani

- outlier data point at (5, 7)



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



Soft margin

- **Slack variables**

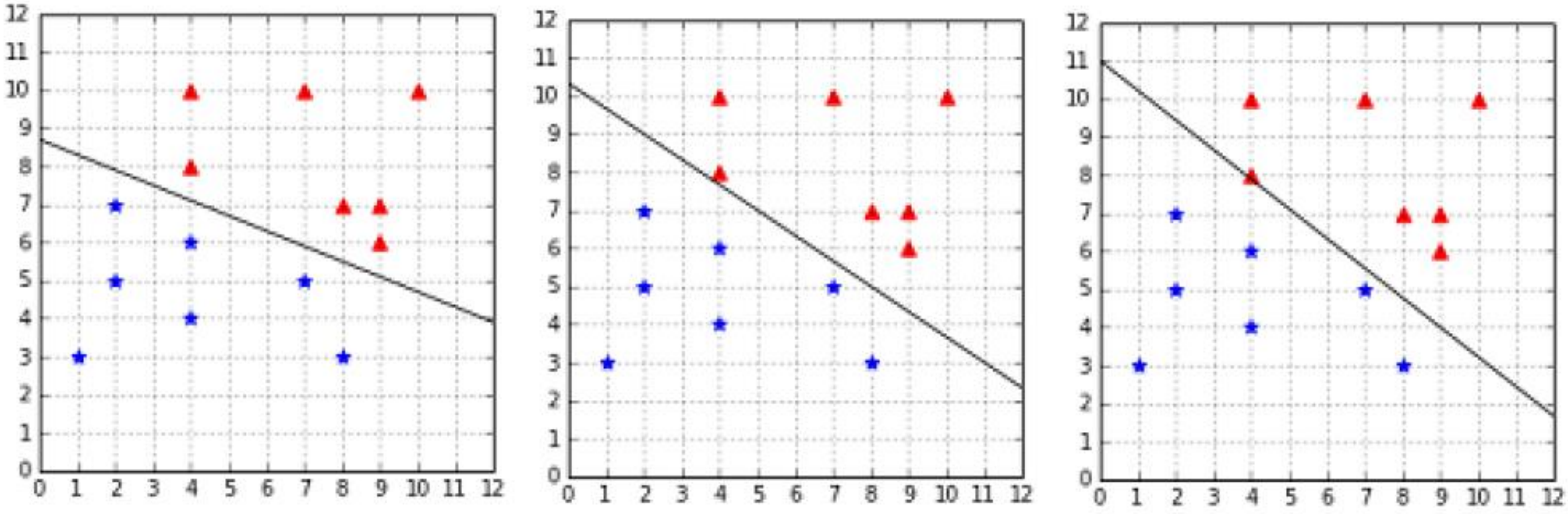
$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$$

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

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

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

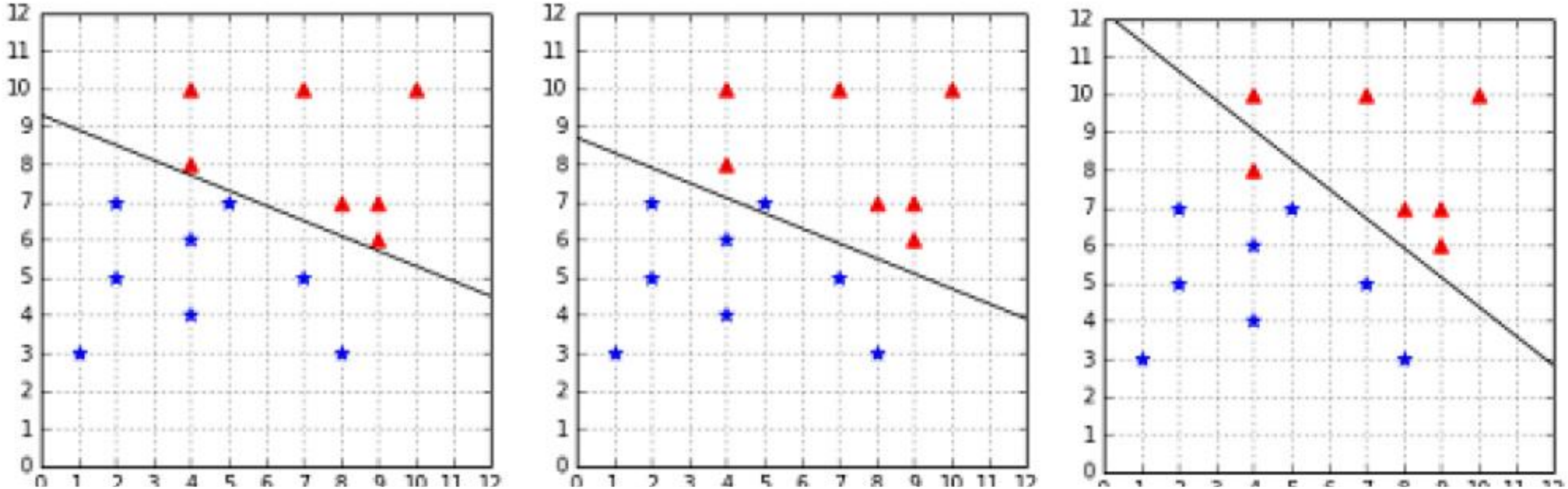
Understanding what C does



Effect of C=+Infinity, 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

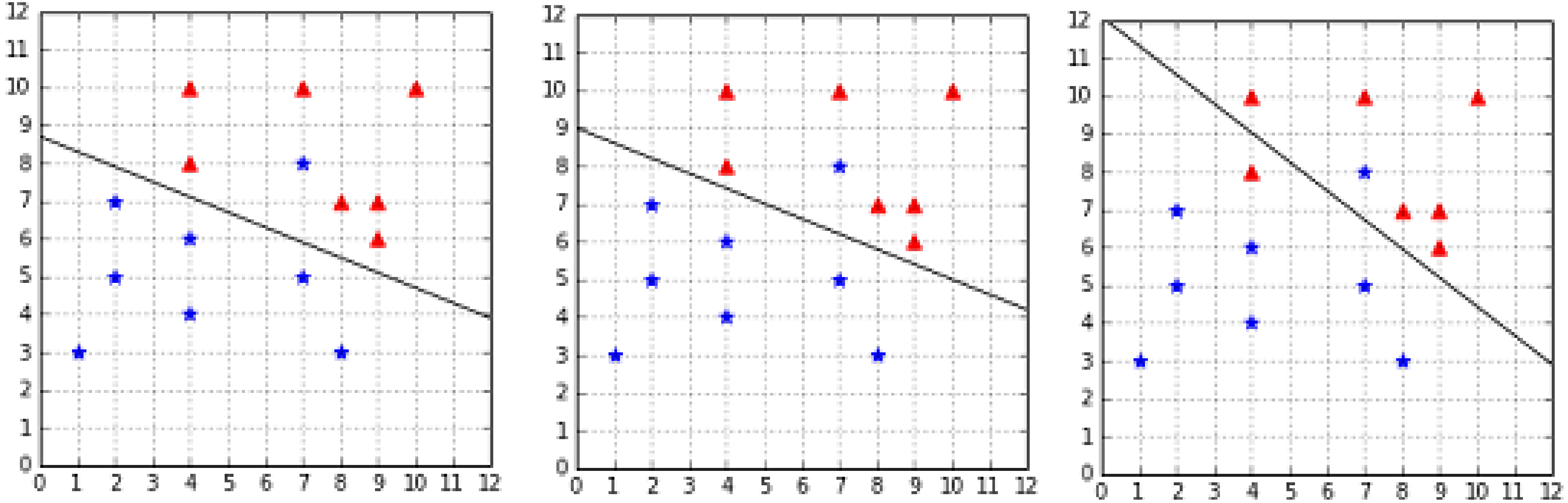
Understanding what C does



Effect of C=+Infinity, 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=infinity 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

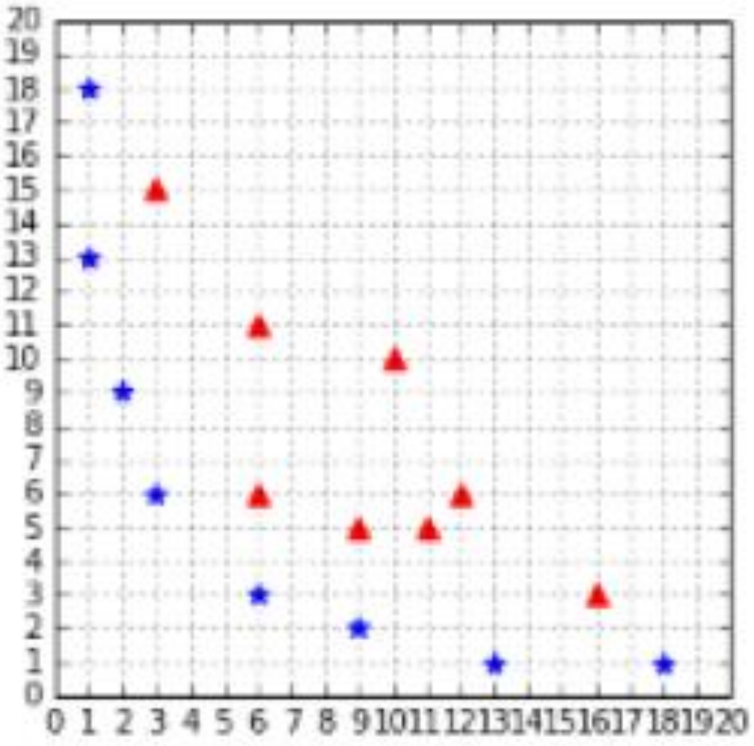
Support Vector Machine (SVM)

Part V: Kernels

Mehran Safayani

Kernels

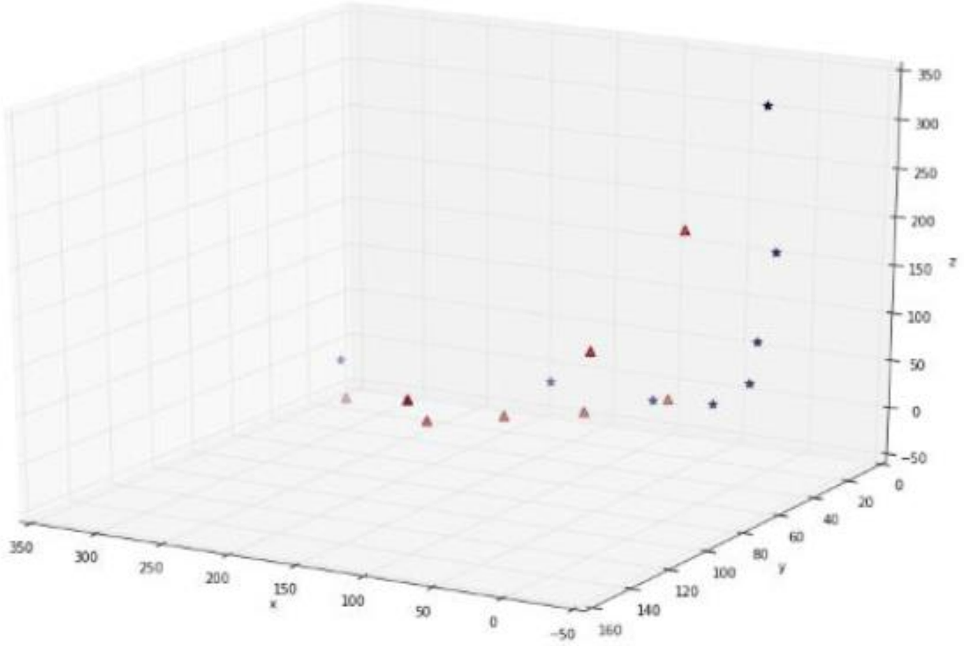
- **Can we classify non-linearly separable data?**

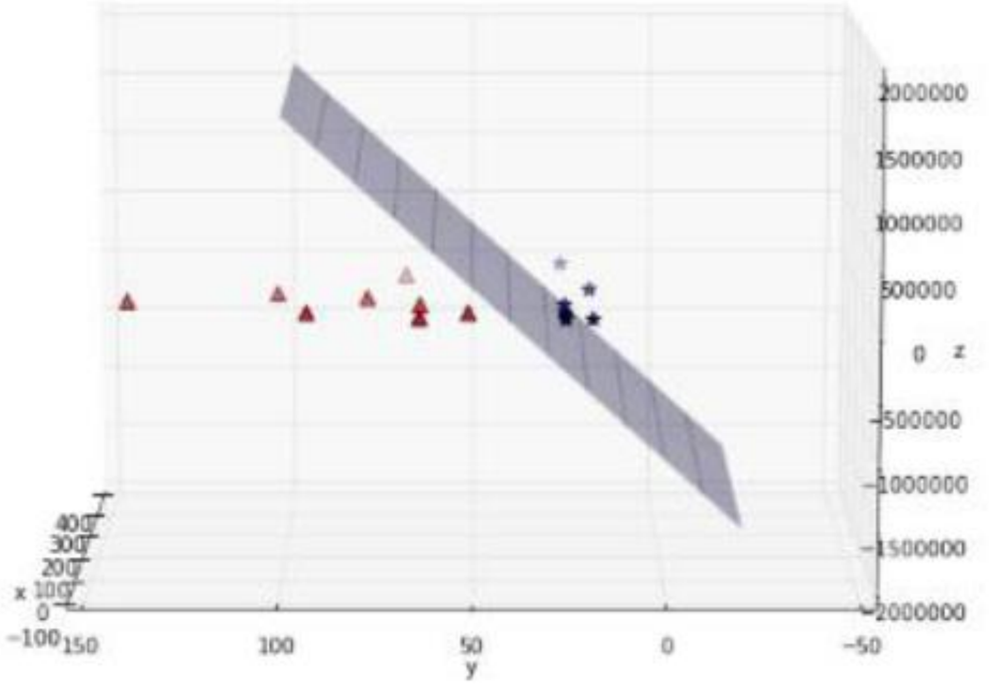


A straight line cannot separate the data

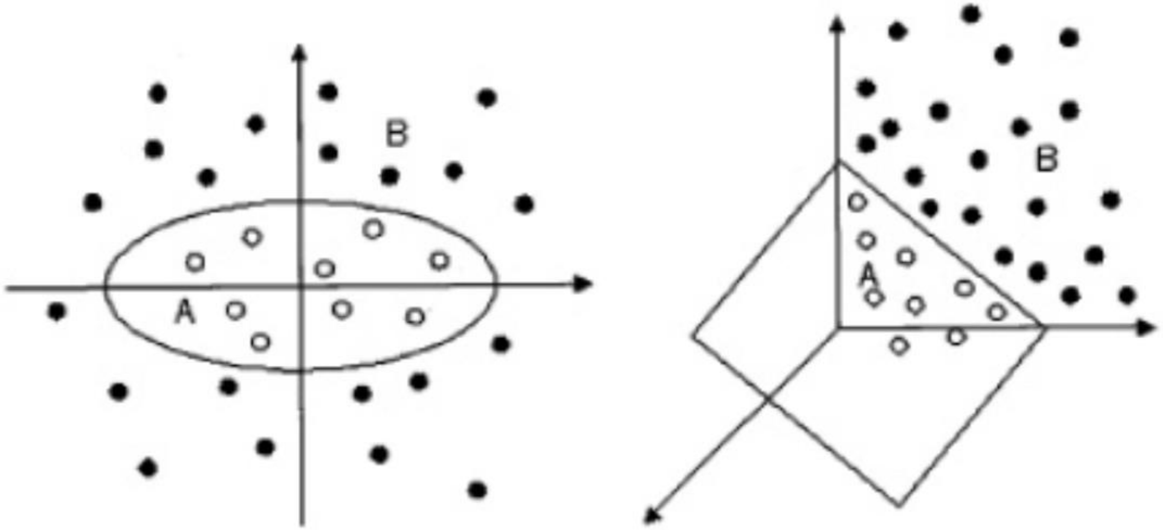
polynomial mapping

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





$$\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$
$$(x_2, x_2) \rightarrow (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$



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

Examples Kernel Trick

$\vec{x} = (x_1, x_2)$
$\vec{z} = (z_1, z_2)$
$K(x, z) = \langle \vec{x} \cdot \vec{z} \rangle^2$

$$K(x_1, x_2) = \langle \phi(x_i) \cdot \phi(x_j) \rangle$$

$$\begin{aligned}
K(x,z) &= \langle \vec{x} \cdot \vec{z} \rangle^2 \\
&= (x_1 z_1 + x_2 z_2)^2 \\
&= (x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2) \\
&= \left\langle (x_1^2, \sqrt{2} x_1 x_2, x_2^2) \cdot (z_1^2, \sqrt{2} z_1 z_2, z_2^2) \right\rangle \\
&= \langle \phi(\vec{x}) \cdot \phi(\vec{z}) \rangle
\end{aligned}$$

mapping function ϕ fused in K
$\rightarrow$ implicit $\phi(\vec{x}) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$

ϕ not necessary any more
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.

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

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

Typical Kernels

Linear kernel

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

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'$$

- **Polynomial Kernel**

$$K(x, z) = (\langle x \cdot z \rangle + \theta)^d, \qquad for \;\; d \geq 0$$

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

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

- (**Sigmoid Kernel**)

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

- **Inverse multi-quadric**

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

Machine Learning

Support Vector Machine (SVM)

Part VI: Kernel SVM

Mehran Safayani

Maximizing the margin is the same thing as minimizing the norm of w

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

Using kernels in SVM

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

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

Using kernels in SVM

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

how to classify

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

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

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

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

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

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

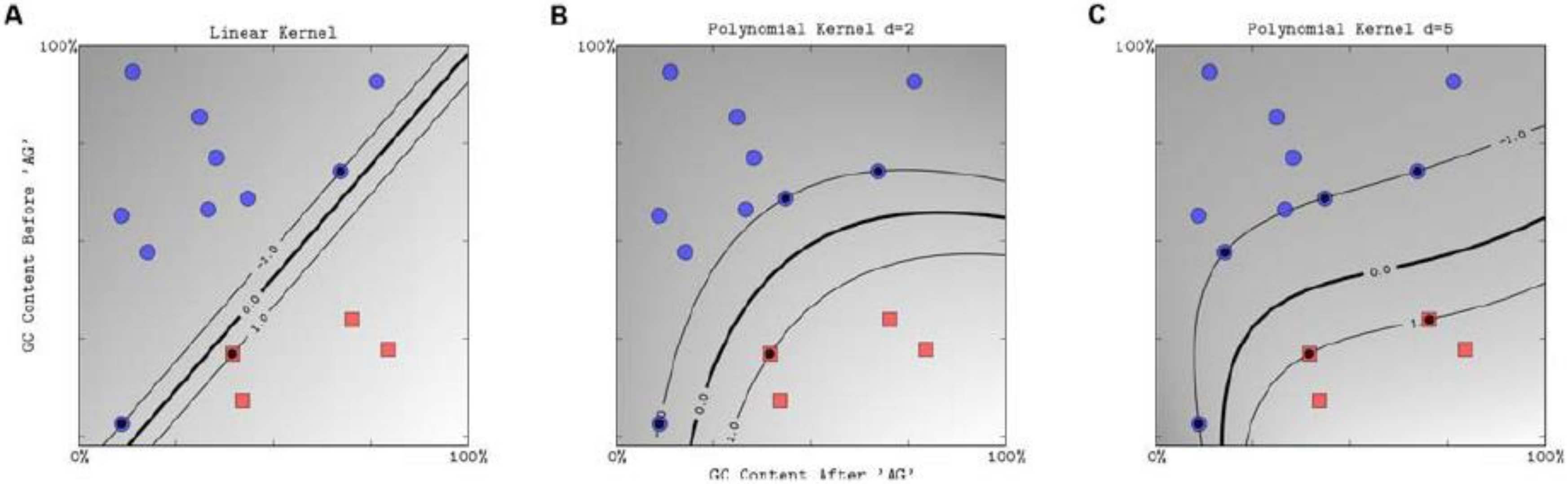


Figure 6. The effect of the degree of a polynomial kernel. The polynomial kernel of degree 1 leads to a linear separation (A). Higher-degree polynomial kernels allow a more flexible decision boundary (B,C). The style follows that of Figure 3.
doi:10.1371/journal.pcbi.1000173.g006

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

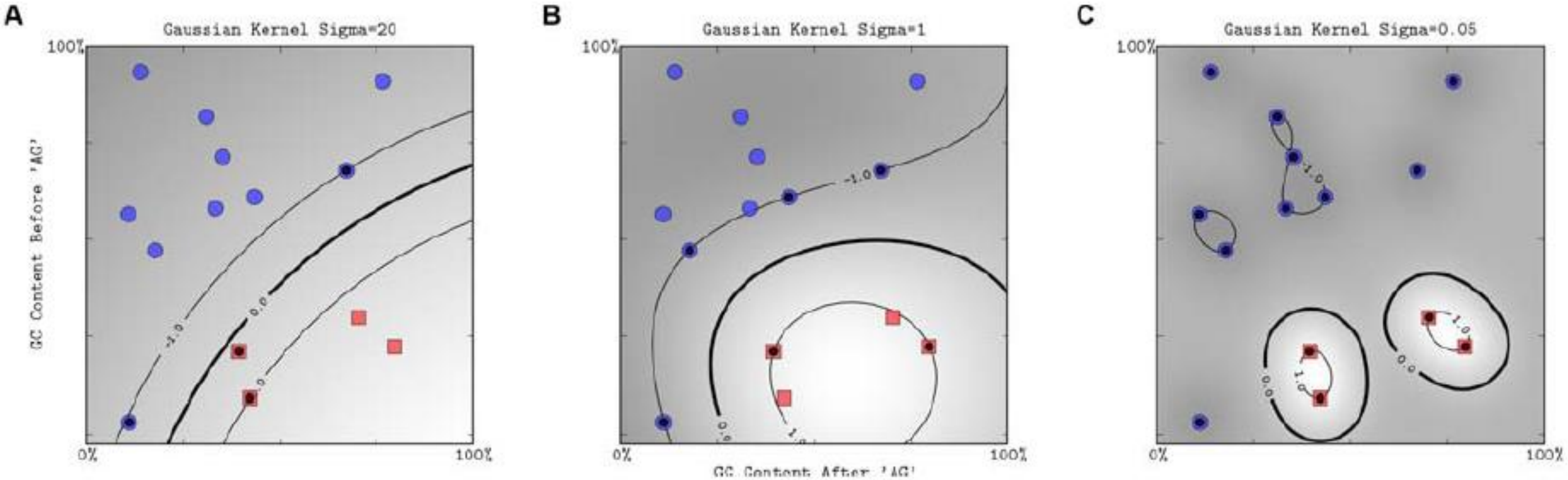


Figure 7. The effect of the width parameter of the Gaussian kernel (σ) for a fixed value of the soft-margin constant. For large values of σ (A), the decision boundary is nearly linear. As σ decreases, the flexibility of the decision boundary increases (B). Small values of σ lead to overfitting (C). The figure style follows that of Figure 3.
doi:10.1371/journal.pcbi.1000173.g007

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

Machine Learning

Support Vector Machine (SVM)

Part VI: Optimization

Mehran Safayani

Affine and convex functions

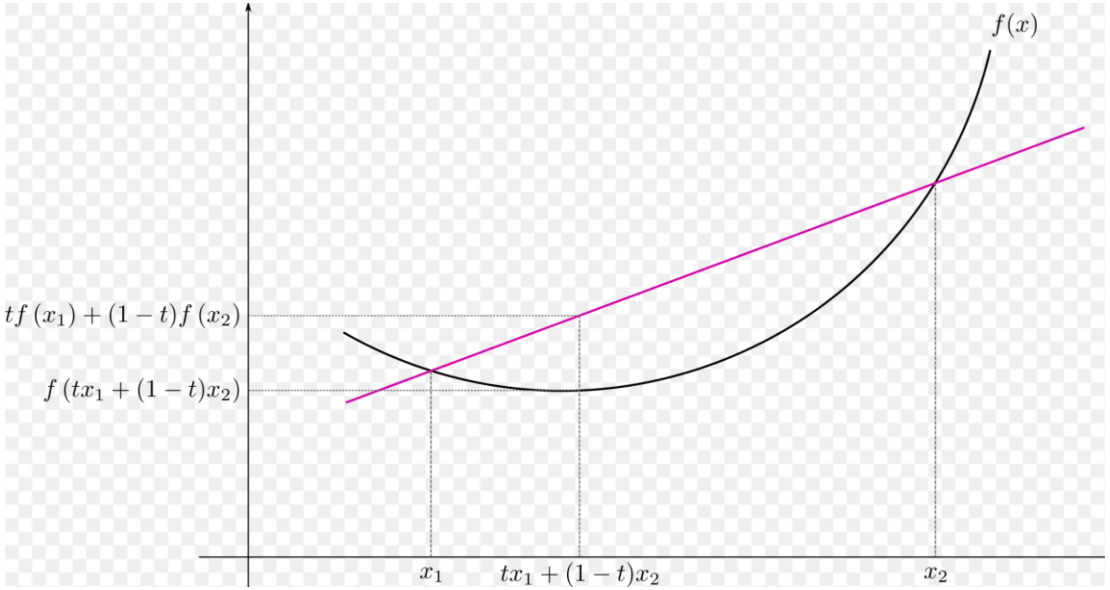
- Affine function

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

- Convex function:

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

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



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

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 α_i's and β_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:

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

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

$$d^* = p^*,$$

Suppose f and the g_i's are convex,[6] and the h_i's are affine.[7] 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.

Given a real valued function $P(x,y) : X \times Y \to \mathbb{R}$ 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^*, α^*, β^* so that w^* is the solution to the primal problem, α^*, β^* are the solution to the dual problem, and moreover $p^* = d^* = \mathcal{L}(w^*, \alpha^*, \beta^*)$. Moreover, w^*, α^* and β^* satisfy the **Karush-Kuhn-Tucker (KKT) conditions**, which are as follows:

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

Moreover, if some w^*, α^*, β^* satisfy the KKT conditions, then it is also a solution to the primal and dual problems.

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

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

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

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

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

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

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

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

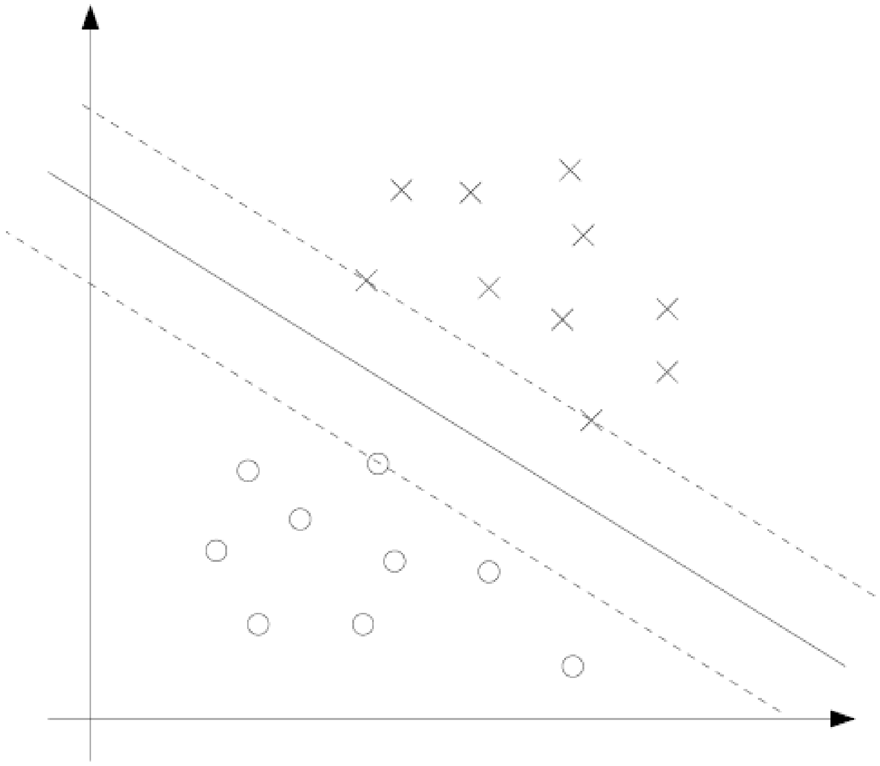
Sequential minimal optimization

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

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

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

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



support vectors

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

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

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

Complementary conditions:

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

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

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

the KKT dual-complementarity conditions

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

SVM as Logistic regression

- Svm loss

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

$$y_n t_n \geqslant 1, \qquad \xi_n = 0,$$

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

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

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

SVM as Logistic regression

- Svm loss

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

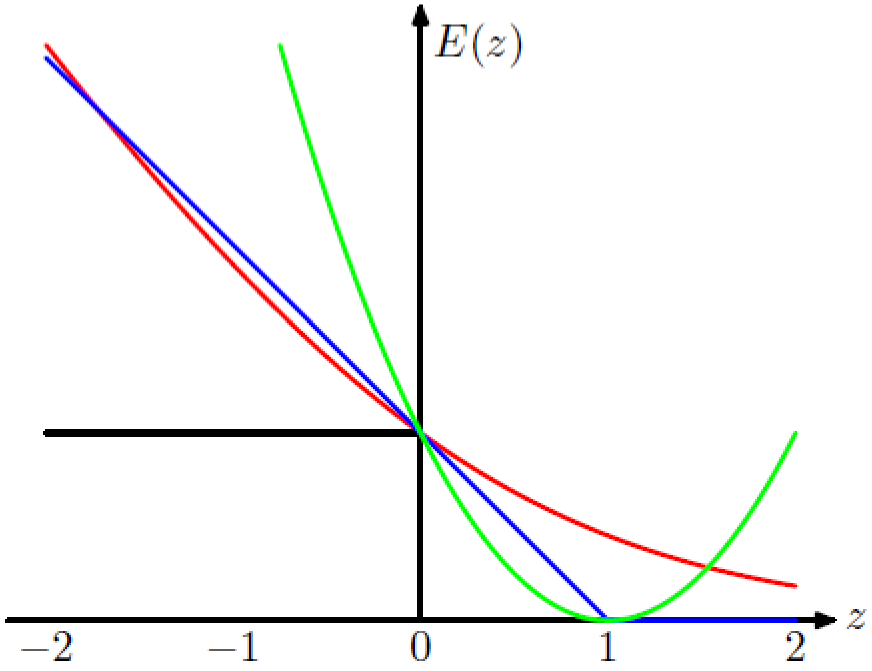
$$E_{\mathrm{SV}}(y_n t_n) = [1 - y_n t_n]_+$$

- Logistic regression loss

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

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

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.



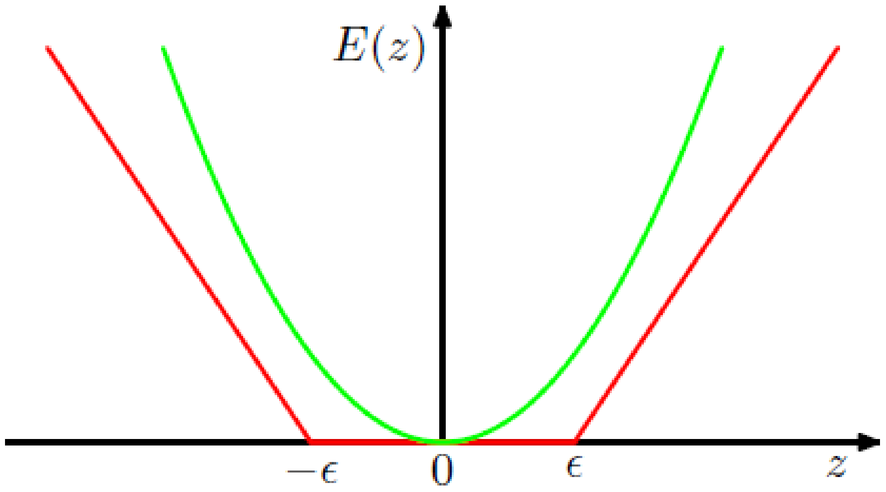
SVM for regression

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

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

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

Figure 7.6 Plot of an ϵ-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>=m ; n=10000; m=10,...,1000

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

If n is small, m is intermediate: n=1,...,1000 ; m=10,..., 10000

Use SVM with Gaussian kernel

If n is small, m is large: n=1,...,1000; m>=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.

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
- 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