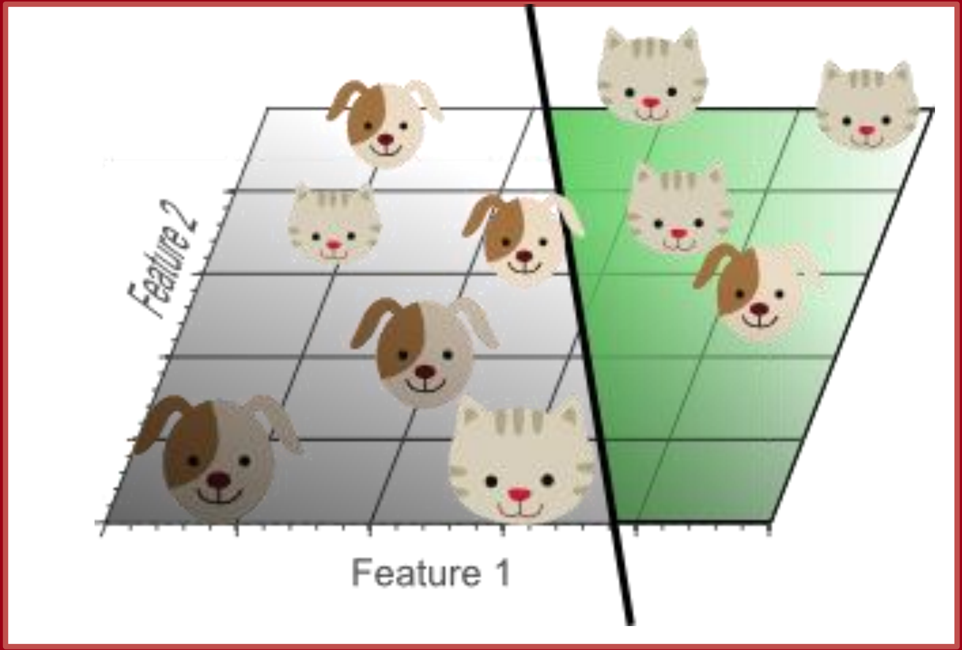
DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Linear Predictors

Machine Learning 2023-24
UML Book Chapter 9
Slides P. Zanuttigh
Some material from F. Vandin, N. Ailon, S. Shwartz, J. Janecek

Linear Predictors

The design of a ML strategy requires 2 main steps:

- Select an hypothesis class $\mathcal{H}$
- Select an algorithm to find the predictor (i.e. to find $ERM_{\mathcal{H}}(S)$)

For linear models:

Hypotheses Classes

- Halfspaces (binary classification)
- Linear Regression (regression)
- Logistic Regression (classification modeled as a regression problem)

Algorithms

- Linear Programming (for halfspaces, not part of the course)
- Perceptron (for halfspaces)
- Least Squares (for regression)

Affine Function Model

Class of *Affine Functions*:

$$L_d = \{h_{\boldsymbol{w},b}, \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

where

$$h_{\boldsymbol{w},b} = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = \left(\sum_{i=1}^{d} w_i x_i \right) + b$$

Each member of L_d is a function $\boldsymbol{x} \to \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b, \;\; \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}$

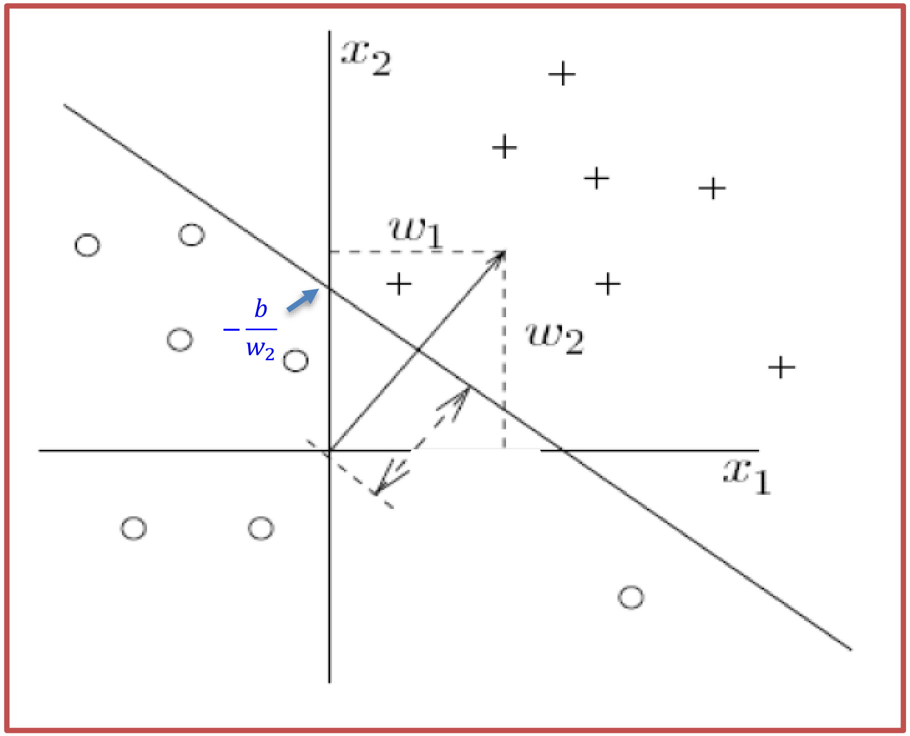
- It is a linear function followed by a sum
- Two parameters: *b* (scalar, called bias) and ***w*** (vector)
- Dimensionality of parameters space: *d+1*

$$\boldsymbol{x} \rightarrow \boxed{L_d} \xrightarrow{z} \boxed{\phi} \rightarrow y$$

Hypothesis class: $\phi \circ L_d \quad \phi: \mathbb{R} \to \mathcal{Y}$

- Binary classification $\mathcal{Y} = \{-1,1\} \to \phi(z) = sign(z)$
- Regression $\mathcal{Y} = \mathbb{R} \to \phi(z) = z$

Geometric Interpretation (2D)



$$x_2 = -\frac{w_1}{w_2}x_1 - \frac{b}{w_2}$$

- The bias is proportional to the offset of the line from the origin
- The weights determine the slope of the line
- The weight vector is perpendicular to the line

Homogeneous Linear Functions

Homogeneous coordinates:

- Idea: incorporate **b** into ***w*** as an extra dimension/coordinate
- Add an extra dimension to ***w***: $\boldsymbol{w} \rightarrow \boldsymbol{w}' = \langle b, w_1, w_2, \dots, w_d \rangle$
- Add an extra element to each vector ***x***: $\boldsymbol{x} \rightarrow \boldsymbol{x}' = \langle 1, x_1, x_2, \dots, x_d \rangle$

Homogeneous linear function:

Rewrite *affine functions*: $L_d = \{h_{\boldsymbol{w},b}, \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$ using homogeneous coord.

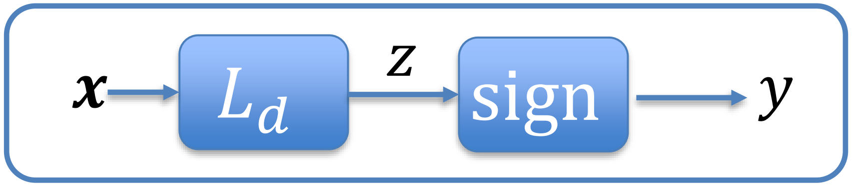
$$h_{\boldsymbol{w},b} = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = \left(\sum_{i=1}^{d} w_i x_i \right) + b = b + w_1 x_1 + \cdots + w_d x_d$$

$$h_{\boldsymbol{w}'} = < \boldsymbol{w}', \boldsymbol{x}' > = \left(\sum_{i=1}^{d+1} w'_i x'_i \right) = b * 1 + w_1 x_1 + \cdots + w_d x_d$$

- $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = < \boldsymbol{w}', \boldsymbol{x}' >$, rewrite affine function as a linear model
- Get rid of bias (incorporated in the weights vector)
- The affine function becomes a linear function !

Halfspaces Hypothesis Class

Halfspace hypothesis class



- Input: $\mathcal{X} = \mathbb{R}^d$ (for each sample a vector of features)
  - Using homogeneous coordinates: $\boldsymbol{x} \to \boldsymbol{x}' = (1, x_1, x_2, \dots, x_d) \in \mathbb{R}^{d+1}$
- Output: $\mathcal{Y} = \{-1,1\}$ (binary classification)
- Loss: 0-1 loss

Halfspace Model:

$$HS_d = \mathrm{sign} \circ L_d = \{x \to sign(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b)\,,\ \ \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

Using homogeneous coordinates:

$$HS_d = \{x \to \mathrm{sign}(\langle \boldsymbol{w}', \boldsymbol{x}' \rangle), \quad \boldsymbol{w}' \in \mathbb{R}^{d+1}\}$$

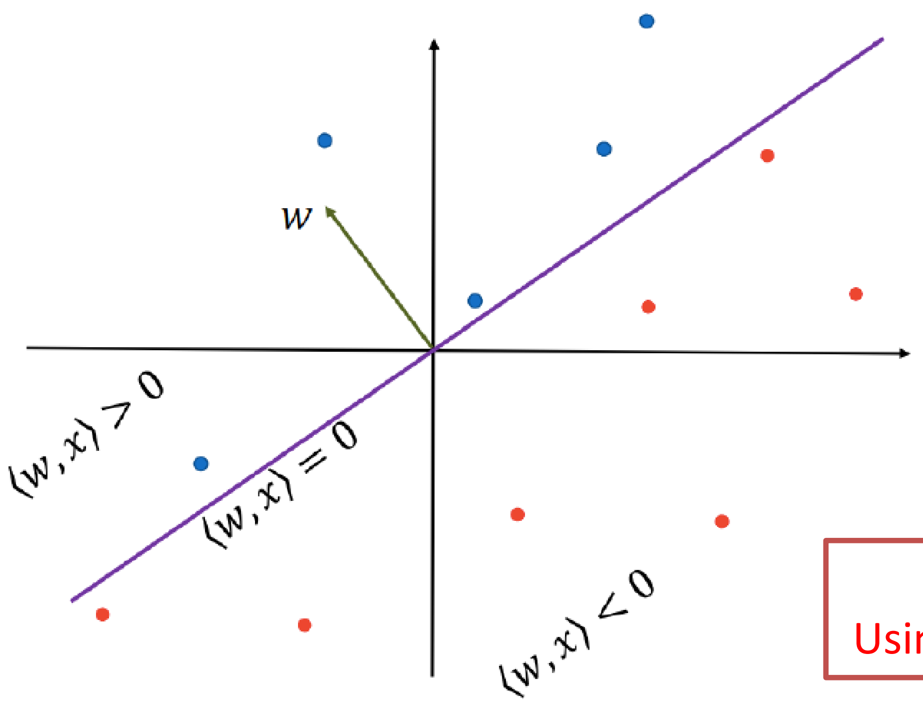
Linear Classification: Halfspace Hypothesis Class

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

- $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{-1,1\}$, 0-1 loss
- Halfspace hypothesis class:

$$HS_d = \text{sign} \circ L_d = \{x \to sign(\langle \boldsymbol{w},\boldsymbol{x}\rangle + b)\}, \quad \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}$$

Example: $\mathcal{X} = \mathbb{R}^2$



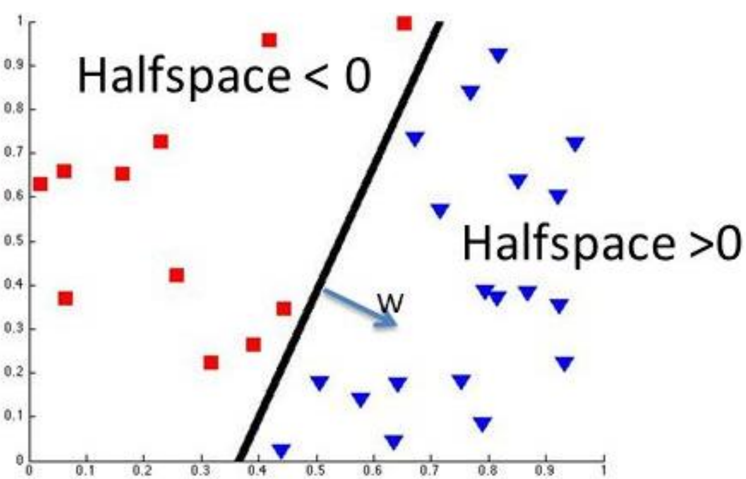
$\boldsymbol{w} \in \mathbb{R}^2, b \in \mathbb{R}$

Using homog. $\boldsymbol{w}' \in \mathbb{R}^3$

Examples: Halfspaces

$\mathcal{X} = \mathbb{R}^2$ *(d=2)*

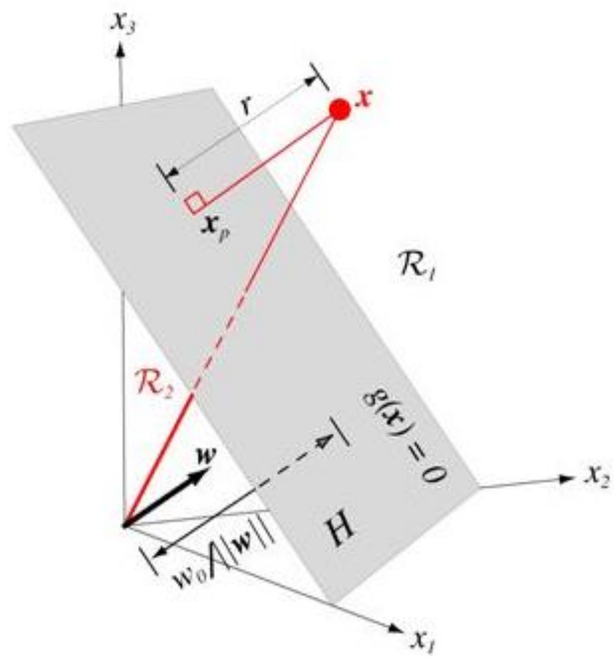
2D space divided by a line



$\boldsymbol{w} \in \mathbb{R}^2, b \in \mathbb{R}$
Using homog. $\boldsymbol{w}' \in \mathbb{R}^3$

$\mathcal{X} = \mathbb{R}^3$ *(d=3)*

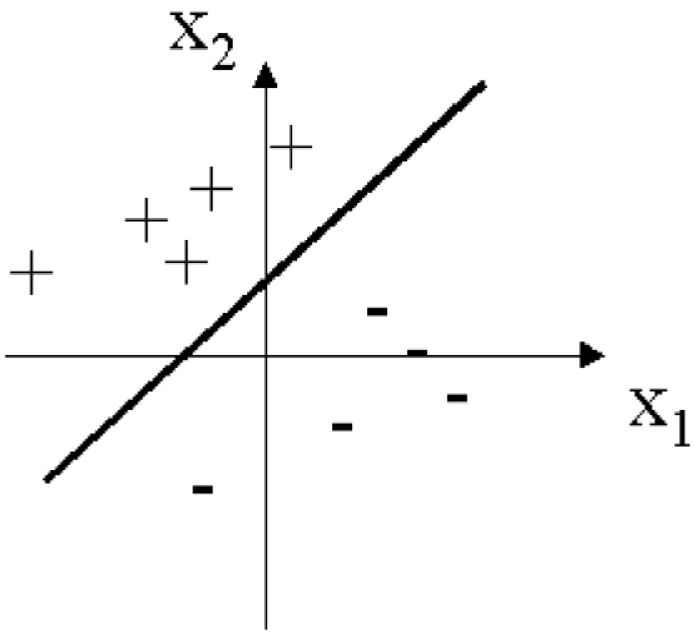
3D space divided by a plane



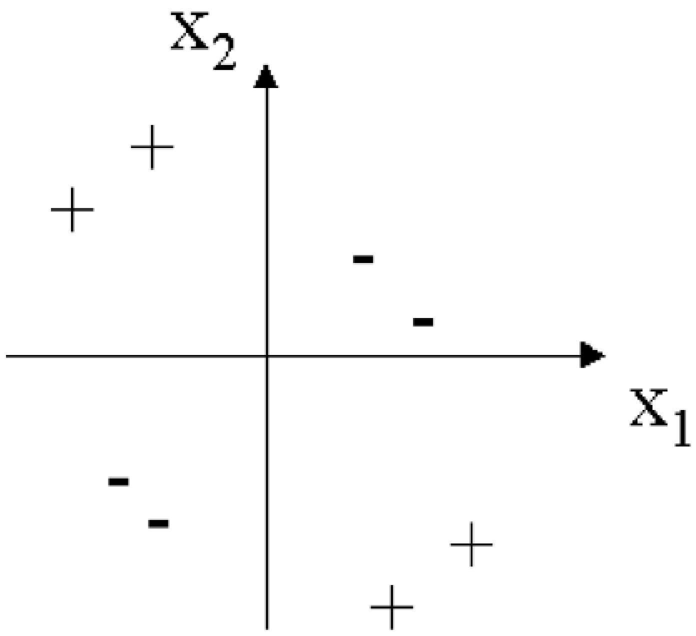
$\boldsymbol{w} \in \mathbb{R}^3, b \in \mathbb{R}$
Using homog. $\boldsymbol{w}' \in \mathbb{R}^4$

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Realizability: Linearly Separable



Linearly Separable



Not Linearly Separable

Linear Programming

Not part of the course

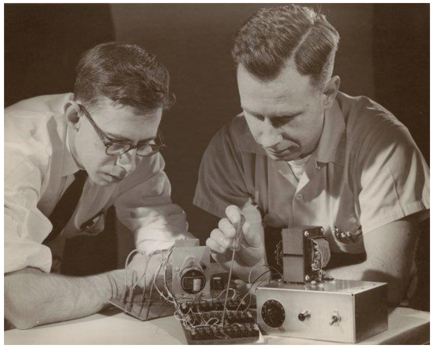
Linear Program (LP): maximize a linear function subject to linear inequalities

$$\text{Target: find} \max_{\boldsymbol{w} \in \mathbb{R}^d} < \boldsymbol{u}, \boldsymbol{w} > \quad \text{subject to } A\boldsymbol{w} \geq \boldsymbol{v}$$

$\boldsymbol{w} \in \mathbb{R}^d$: vector of unknowns, $\boldsymbol{u} \in \mathbb{R}^d$, $A \in \mathbb{R}^{mxd}$, $\boldsymbol{v} \in \mathbb{R}^m$

- Empirical Risk Minimization (ERM) for halfspaces in the *realizable* case can be expressed as a linear program (LP)
  - The ERM predictor is $L_S(h_S) = 0 \rightarrow \mathrm{sign}(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle) = y_i \ \forall i \rightarrow \ y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle > 0 \ \forall i$
  - Need some math to adapt “>” to “$\geq$”
- All solutions satisfying constraints are ok for us (→see SVM later…)
- There exist efficient LP solvers (e.g., simplex algorithm)

Find ERM Halfspace: Perceptron



- Iterative algorithm (introduce by Rosenblatt in 1958)
- Target: find separating hyperplane
- Find vector ***w*** representing separating hyperplane (in homogeneous coordinates)
- At each step focus on a misclassified sample and guide the algorithm to be "*more correct*" on it
- In the realizable case always converge to a (ERM) solution correctly classifying all points
- Simple and fast in most cases (but there exists critical situations)

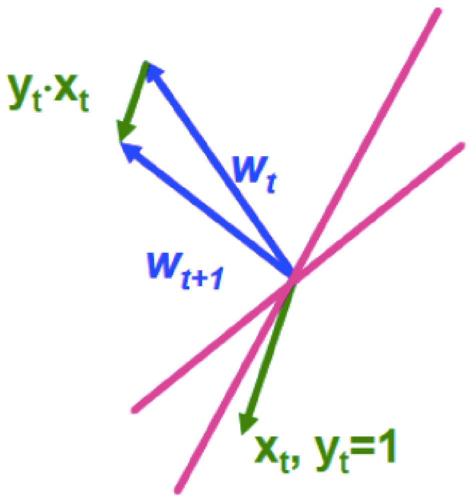
Find ERM Halfspace: Perceptron Algorithm

ERM predictor: $L_S(h_S) = 0$
$\rightarrow$ $\mathrm{sign}(\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle) = y_i \ \forall i$
$\rightarrow$ $y_i \langle \boldsymbol{w}, \boldsymbol{x_i} \rangle > 0 \ \forall i$

Input: training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$
initialize $\mathbf{w}^{(1)} = (0, \ldots, 0)$;
for $t = 1, 2, \ldots$ **do**
  if $\exists i$ s.t. $y_i \langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle \leq 0$ **then** $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + y_i \mathbf{x}_i$;
  else return $\mathbf{w}^{(t)}$;

At each iteration find a misclassified sample and add the sample multiplied by its label

Interpretation of update:



Note that:

$$y_i \langle \mathbf{w}^{(t+1)}, \mathbf{x}_i \rangle = y_i \langle \mathbf{w}^{(t)} + y_i \mathbf{x}_i, \mathbf{x}_i \rangle = y_i \langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle + ||\mathbf{x}_i||^2$$

$\Rightarrow$ update guides $\mathbf{w}$ to be "more correct" on $(\mathbf{x}_i, y_i)$.

$\|\boldsymbol{x}_i\|^2 > 0$ and target is $y_i \langle \boldsymbol{w}^{(t)}, \boldsymbol{x}_i \rangle > 0$: from step *t* to *t+1* "more correct" on i-th sample

Termination? Depends on the realizability assumption!

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Perceptron on Linearly Separable Data

Halfspaces:

realizability assumption corresponds to *linearly separable data*

Theorem:

- $(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_m}, y_m)$ is linearly separable
- $B = \min\{\|\boldsymbol{w}\|: y_i\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle \geq 1 \;\; \forall i = 1,..,m\}$
- $R = max\|\boldsymbol{x}_i\|$

- The perceptron algorithm stops after at most $(RB)^2$ iterations
- When it stops, it holds that $\forall i \in \{1,..,m\}$: $y_i\langle \boldsymbol{w}^{(t)}, \boldsymbol{x_i}\rangle > 0$
- *Notice*: by design the algorithm stops when there are no more wrongly classified samples

Theorem: Demonstration

1. *Define*:
   - $\boldsymbol{w}^*$: vector achieving the *min* in definition of B
     - Recall $B = \min\{\|\boldsymbol{w}\|: y_i\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle \geq 1 \ \ \forall i = 1,..,m\}$
   - T : number of iterations before stopping
     - need to show that $T < (RB)^2$

2. *Consider*: $\frac{\langle \boldsymbol{w}^*, \boldsymbol{w}^{(T+1)}\rangle}{\|\boldsymbol{w}^*\|\|\boldsymbol{w}^{(T+1)}\|}$ → it is smaller than 1 (cosine of angle)

3. We need to demonstrate that:

$$\frac{\sqrt{T}}{RB} = \frac{T}{\sqrt{T}RB} \leq \frac{\langle \boldsymbol{w}^*, \boldsymbol{w}^{(T+1)}\rangle}{\|\boldsymbol{w}^*\|\|\boldsymbol{w}^{(T+1)}\|} \leq 1 \Rightarrow T < (RB)^2$$

4. Proceed in 2 parts:
   a) Numerator: demonstrate that $\langle \boldsymbol{w}^*, \boldsymbol{w}^{(T+1)}\rangle \geq$T
   b) Denominator: demonstrate that $\|\boldsymbol{w}^*\|\|\boldsymbol{w}^{(T+1)}\| \leq \sqrt{T}\,RB$

Theorem: Demonstration (Numerator)

Numerator: demonstrate that $\langle \boldsymbol{w}^*, \boldsymbol{w}^{(T+1)} \rangle \geq T$

- First iteration : $\boldsymbol{w}^{(1)} = (0, .., 0) \rightarrow \langle \boldsymbol{w}^*, \boldsymbol{w}^{(1)} \rangle = 0$
- At each step $\langle \boldsymbol{w}^*, \boldsymbol{w}^{(t+1)} \rangle - \langle \boldsymbol{w}^*, \boldsymbol{w}^{(t)} \rangle \geq 1$ (using perceptron update rule and recalling definition of $\boldsymbol{w}^*$, see *)
- After T iterations: $\langle \boldsymbol{w}^*, \boldsymbol{w}^{(T+1)} \rangle \geq T$ (see *)

$$(*)\ \langle w^*, w^{(T+1)} \rangle = \sum_{t=1}^{T} \left(\langle w^*, w^{(t+1)} \rangle - \langle w^*, w^{(t)} \rangle \right)$$

$$= \sum_{t=1}^{T} \langle w^*, \underbrace{\boxed{w^{(t+1)} - w^{(t)}}}_{\text{Algorithm}} \rangle = \sum_{t=1}^{T} \langle w^*, \underbrace{\boxed{y_i x_i}}_{\text{Assumption on } w^*} \rangle \geq T$$

(perceptron update rule)

(definition of w^* and B)
$\rightarrow \langle \boldsymbol{w}^*, y_i \boldsymbol{x}_i \rangle \geq 1$

Theorem: Demonstration (Denominator)

Denominator: demonstrate that $\|\boldsymbol{w}^*\|\left\|\boldsymbol{w}^{(T+1)}\right\| \le \sqrt{T}\, RB$

a) $\left\|\boldsymbol{w}^{(T+1)}\right\|^2 \le TR^2 \rightarrow \left\|\boldsymbol{w}^{(T+1)}\right\| \le \sqrt{T}R \quad (**)$

b) $\|\boldsymbol{w}^*\| = B$ (by definition)

$$(**)\ \left\|w^{(T+1)}\right\|^2 = \sum_{t=1}^{T}\left(\left\|w^{(t+1)}\right\|^2 - \left\|w^{(t)}\right\|^2\right)$$

$$= \sum_{t=1}^{T}\left(\left\|w^{(t)} + y_i x_i\right\|^2 - \left\|w^{(t)}\right\|^2\right)$$

$$= \sum \left(\underbrace{2y_i\langle w^t, x_i\rangle}_{\le 0 \text{ by algorithm}} + \underbrace{\|x_i\|^2}_{\le R^2 \text{ by algorithm}}\right) \le TR^2$$

Algorithm

the two $\left\|\boldsymbol{w}^{(t)}\right\|^2$ cancel out

(perceptron update condition: select missclassified sample)

(definition of R) $R = max\|\boldsymbol{x}_i\|$

Perceptron: Notes

It is simple to implement !

On separable data:

- ❑ Convergence is guaranteed
- ❑ Convergence depends on *B*, which can be exponential in *d*
  - If the input vectors are not normalized and arranged in some unfavorable ways the running time can be very long
  - A Linear Programming (LP) approach may be better to find ERM solution in some cases
- ❑ Potentially multiple solutions, which one is picked depends on starting values

On non separable data:

- ❑ Run for some time and keep best solution found up to that point (*pocket algorithm*)

Learning Example

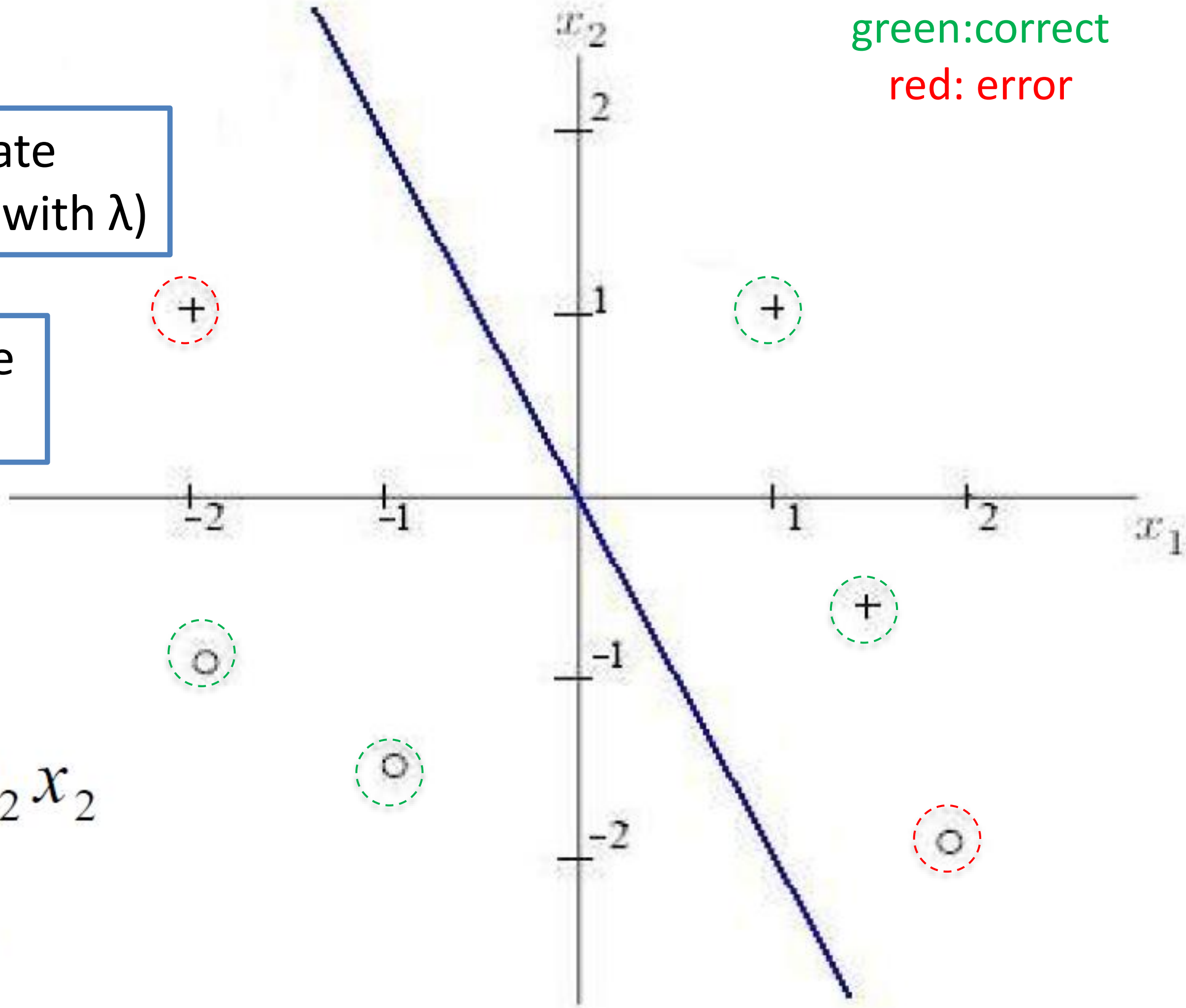
Initial Values:

$\eta = 0.2$

Learning rate
(also denoted with λ)

$$w = \begin{pmatrix} 0 \\ 1 \\ 0.5 \end{pmatrix}$$

Initial line position



green:correct
red: error

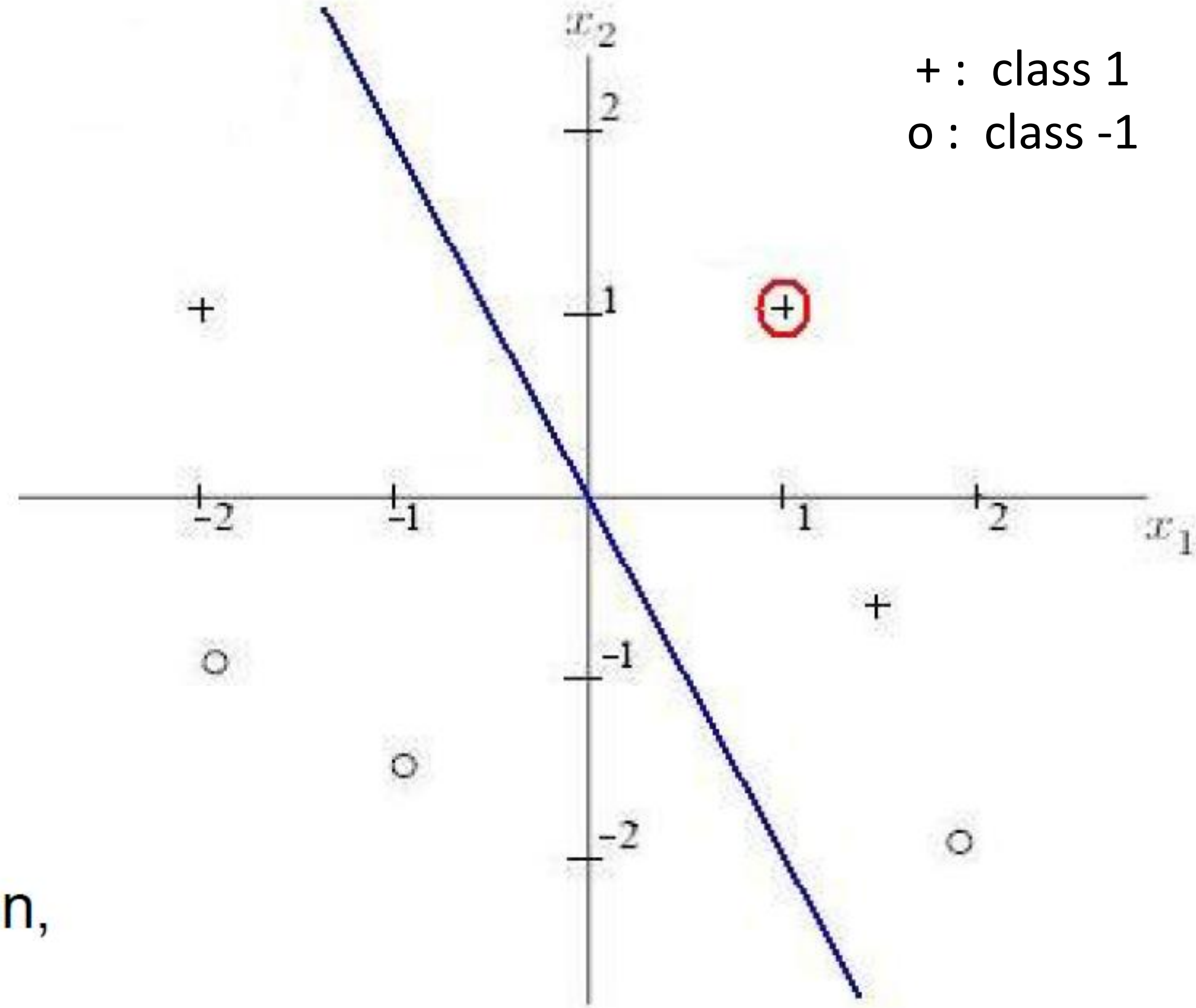
$$\begin{aligned} 0 &= w_0 + w_1 x_1 + w_2 x_2 \\ &= 0 + x_1 + 0.5x_2 \\ \Rightarrow \quad x_2 &= -2x_1 \end{aligned}$$

Perceptron with learning rate: $w^{(t+1)} = w^{(t)} + \lambda y_i x_i$

Learning Example

$\eta = 0.2$

$$w = \begin{pmatrix} 0 \\ 1 \\ 0.5 \end{pmatrix}$$



$x_1 = 1$, $x_2 = 1$
$w^T x > 0$

Correct classification, no action

Learning Example

$\eta = 0.2$

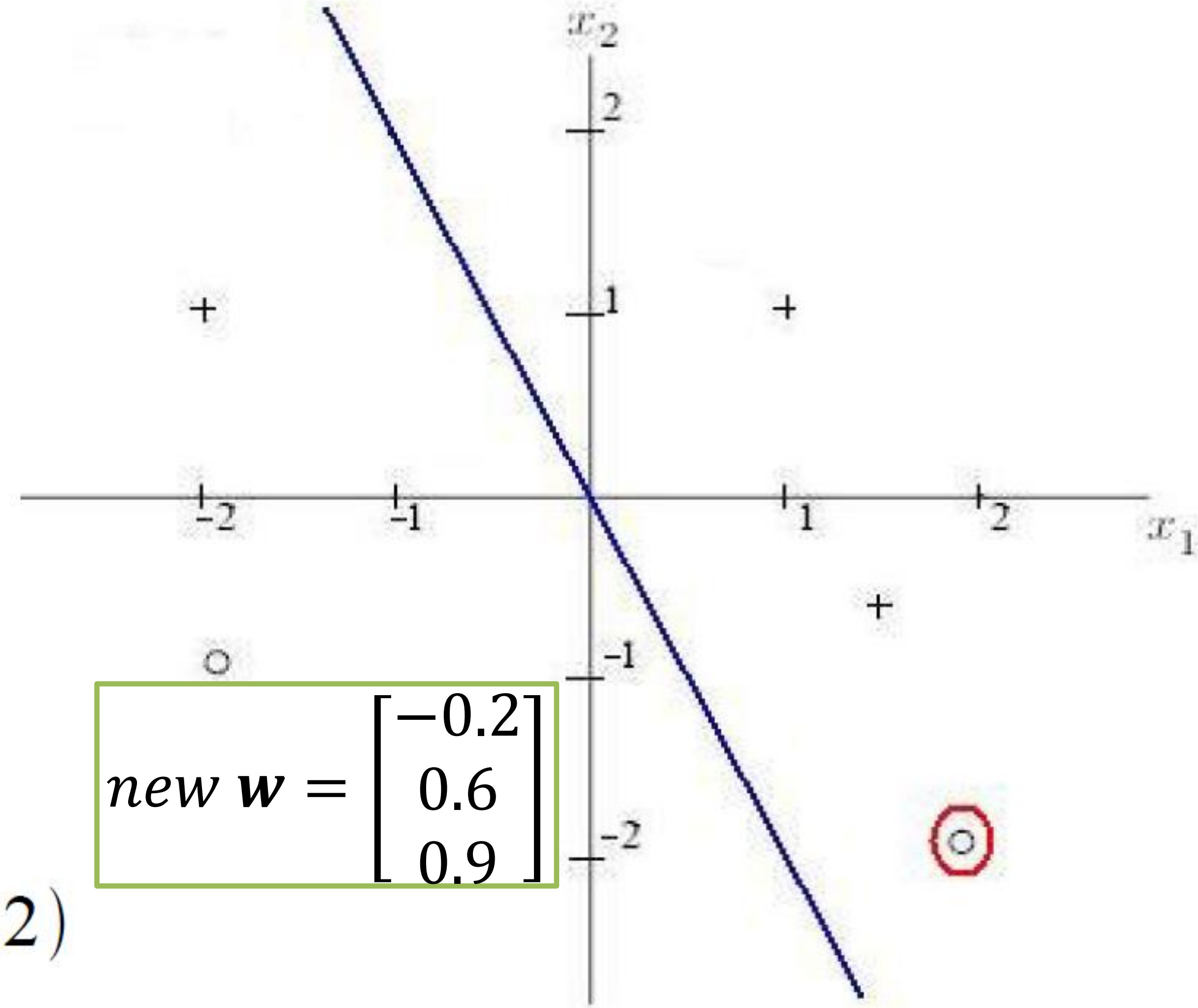
$$w = \begin{pmatrix} 0 \\ 1 \\ 0.5 \end{pmatrix}$$

$x_1 = 2,\ x_2 = -2$

$$w_0 = w_0 - 0.2 * 1$$

$$w_1 = w_1 - 0.2 * 2$$

$$w_2 = w_2 - 0.2 * (-2)$$



$$new\ \boldsymbol{w} = \begin{bmatrix} -0.2 \\ 0.6 \\ 0.9 \end{bmatrix}$$

Learning Example

$\eta = 0.2$

$$w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}$$

previous w

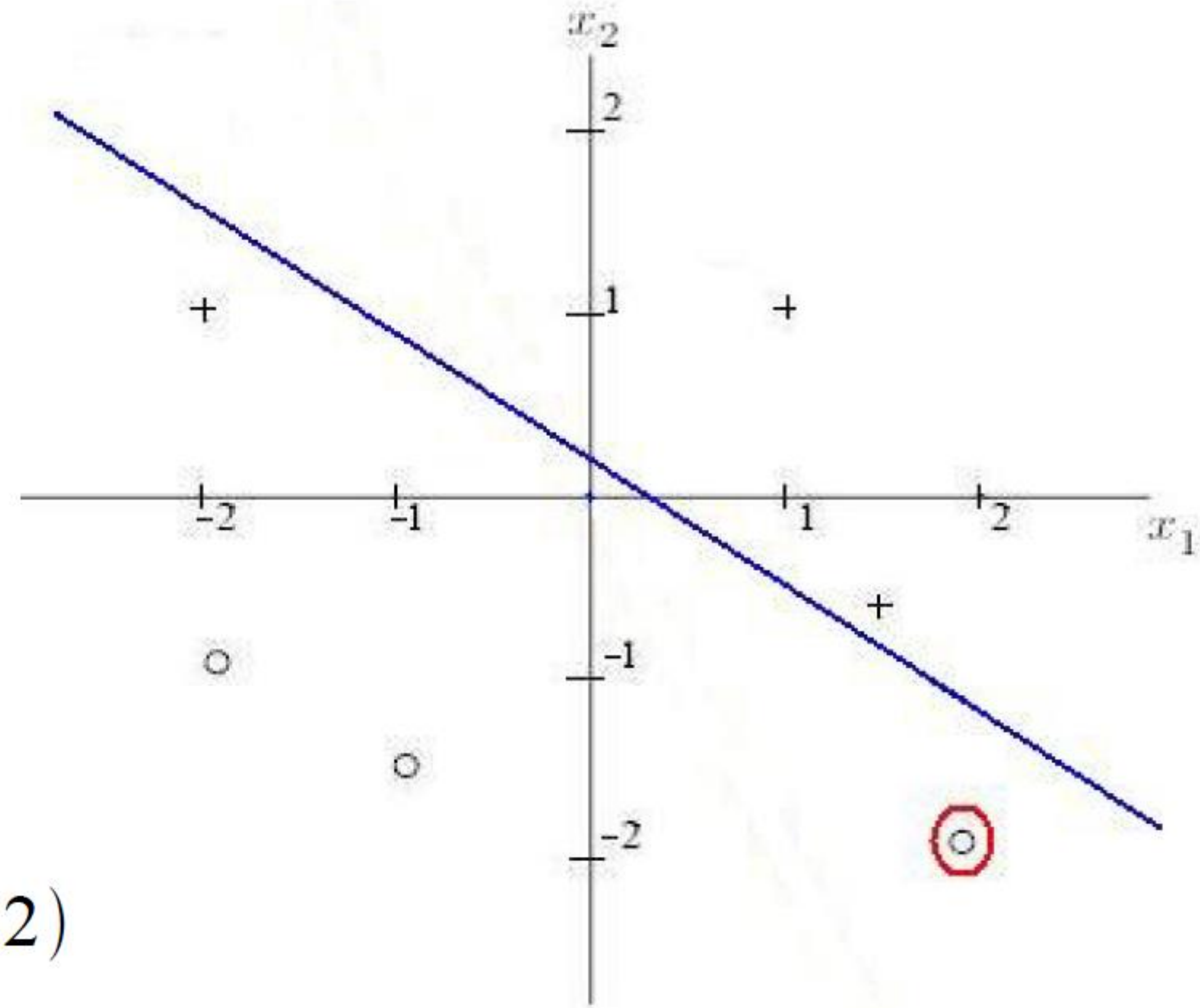
$$w = \begin{pmatrix} 0 \\ 1 \\ 0.5 \end{pmatrix}$$

$x_1 = 2,\ x_2 = -2$

$$w_0 = w_0 - 0.2 * 1$$

$$w_1 = w_1 - 0.2 * 2$$

$$w_2 = w_2 - 0.2 * (-2)$$



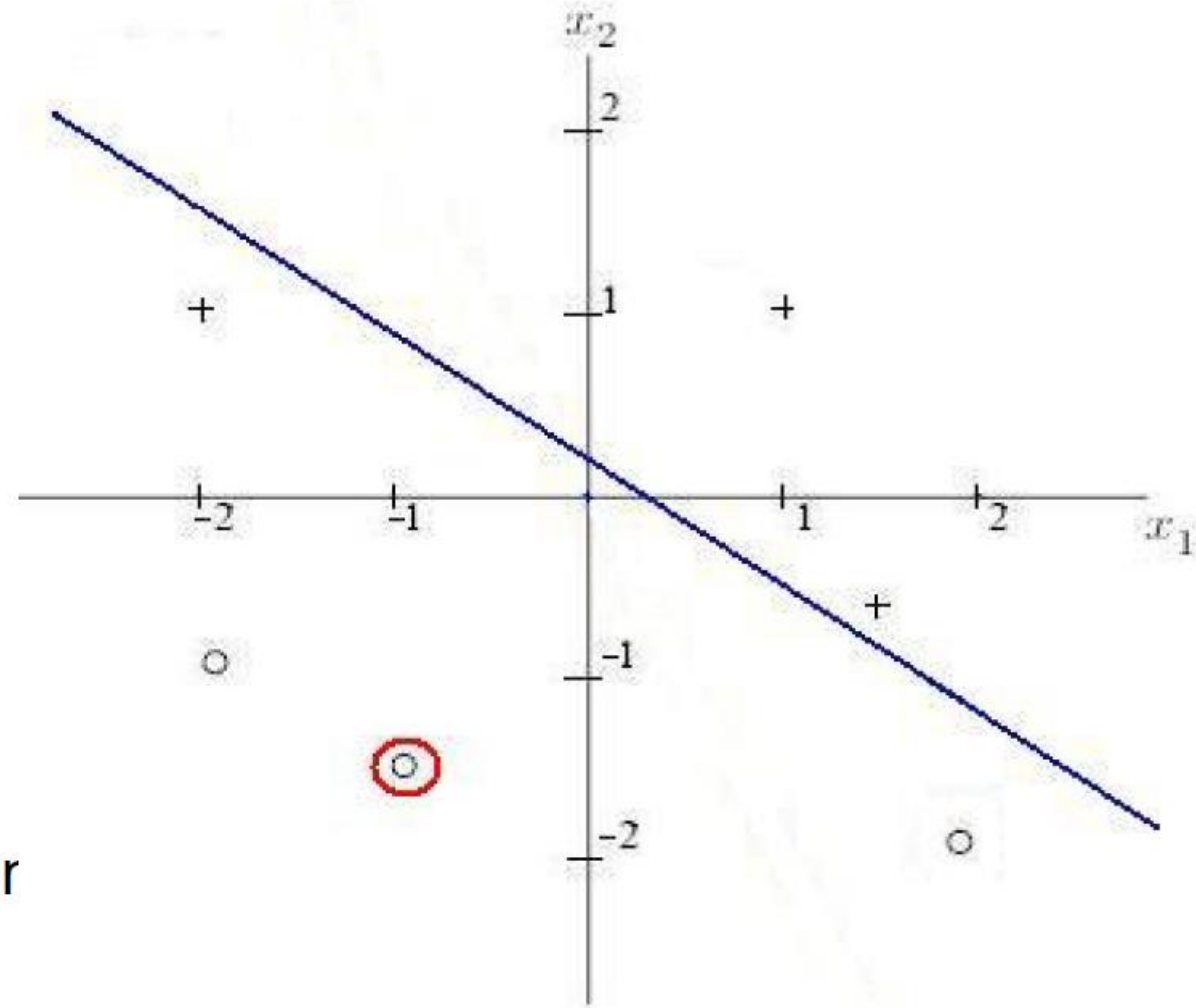
Learning Example

$\eta = 0.2$

$$w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}$$

$x_1 = -1$, $x_2 = -1.5$
$w^T x < 0$

Correct classificatior
no action



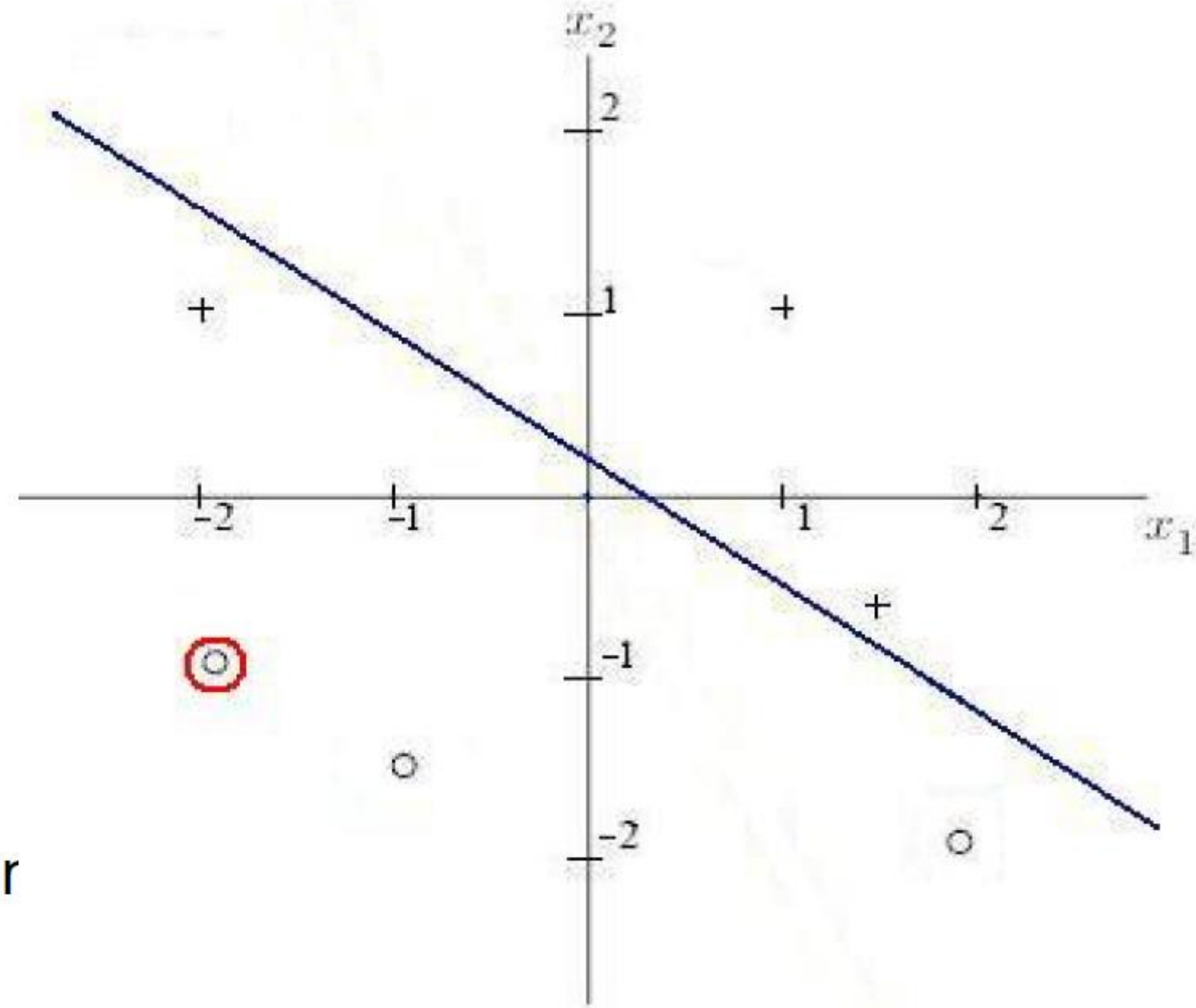
Learning Example

$\eta = 0.2$

$$w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}$$

$x_1 = -2,\ x_2 = -1$
$w^T x < 0$

Correct classificatior
no action



Learning Example

$$\eta = 0.2$$

$$w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}$$

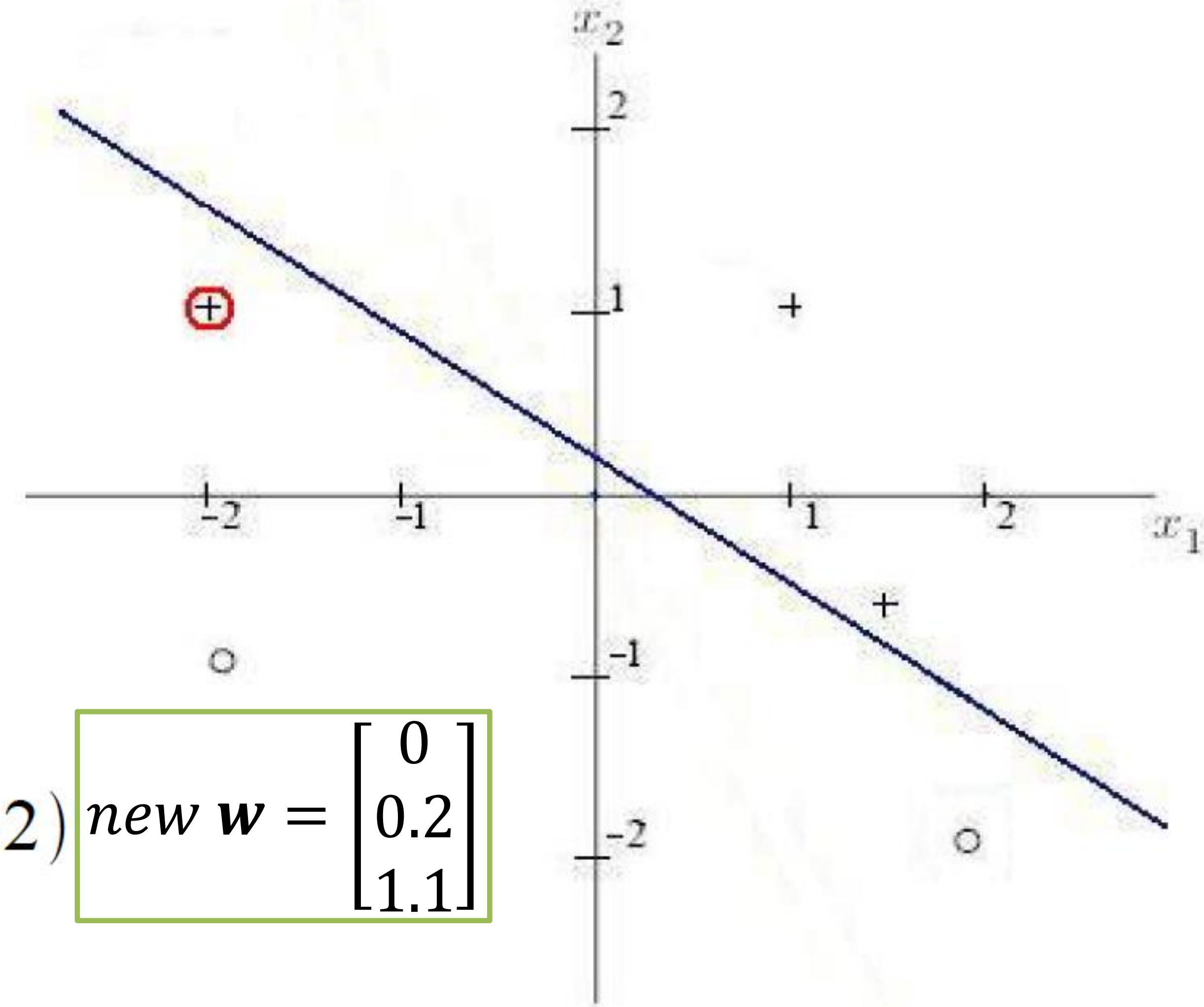
$x_1 = -2,\ x_2 = 1$

$$w_0 = w_0 + 0.2 * 1$$

$$w_1 = w_1 + 0.2 * (-2)$$

$$w_2 = w_2 + 0.2 * 1$$

$$new\ \boldsymbol{w} = \begin{bmatrix} 0 \\ 0.2 \\ 1.1 \end{bmatrix}$$



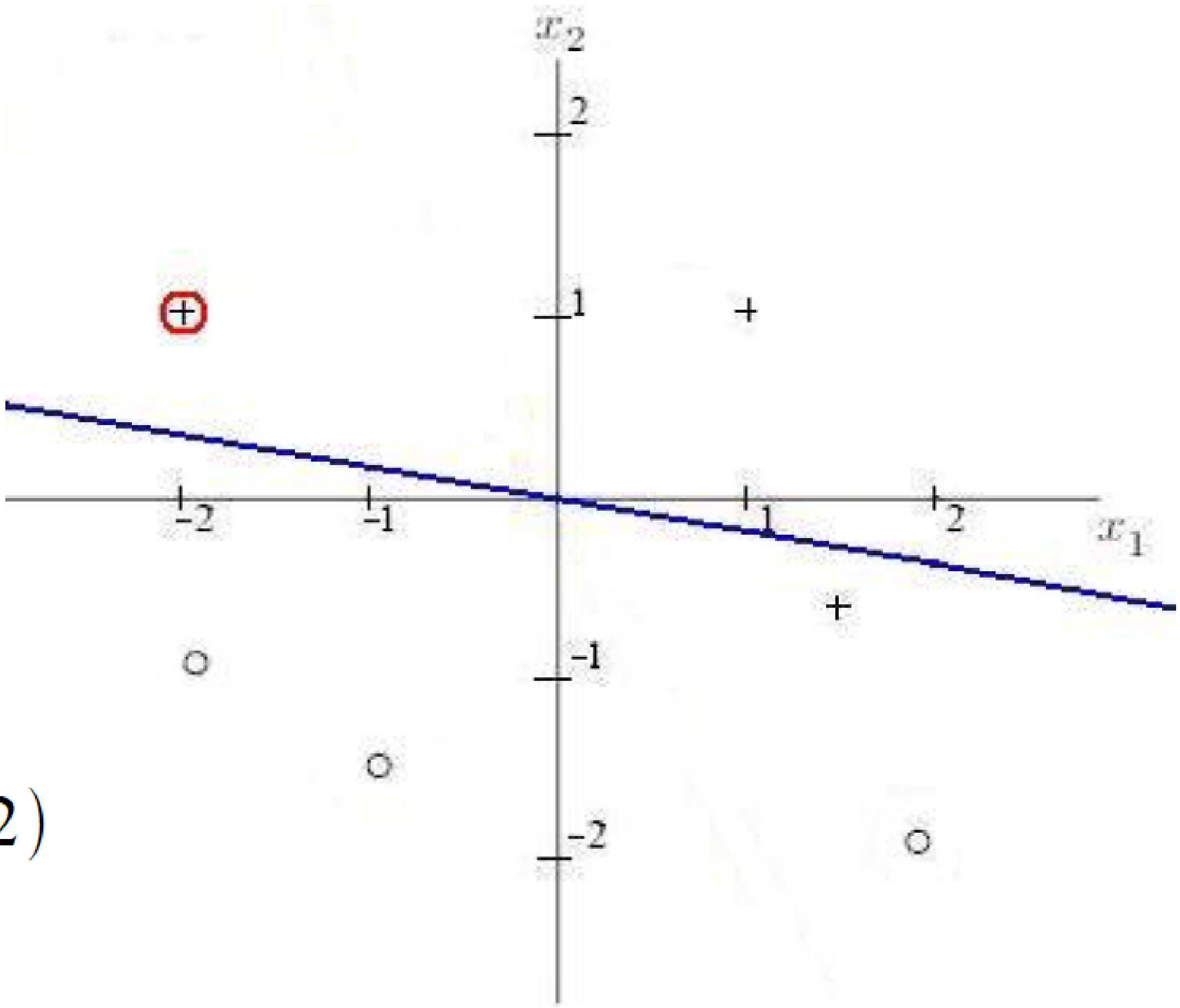
Learning Example

$\eta = 0.2$

$$w = \begin{pmatrix} 0 \\ 0.2 \\ 1.1 \end{pmatrix}$$

previous w

$$\boxed{w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}}$$



$x_1 = -2,\ x_2 = 1$

$$w_0 = w_0 + 0.2 * 1$$

$$w_1 = w_1 + 0.2 * (-2)$$

$$w_2 = w_2 + 0.2 * 1$$

Learning Example

$\eta = 0.2$

$$w = \begin{pmatrix} 0 \\ 0.2 \\ 1.1 \end{pmatrix}$$

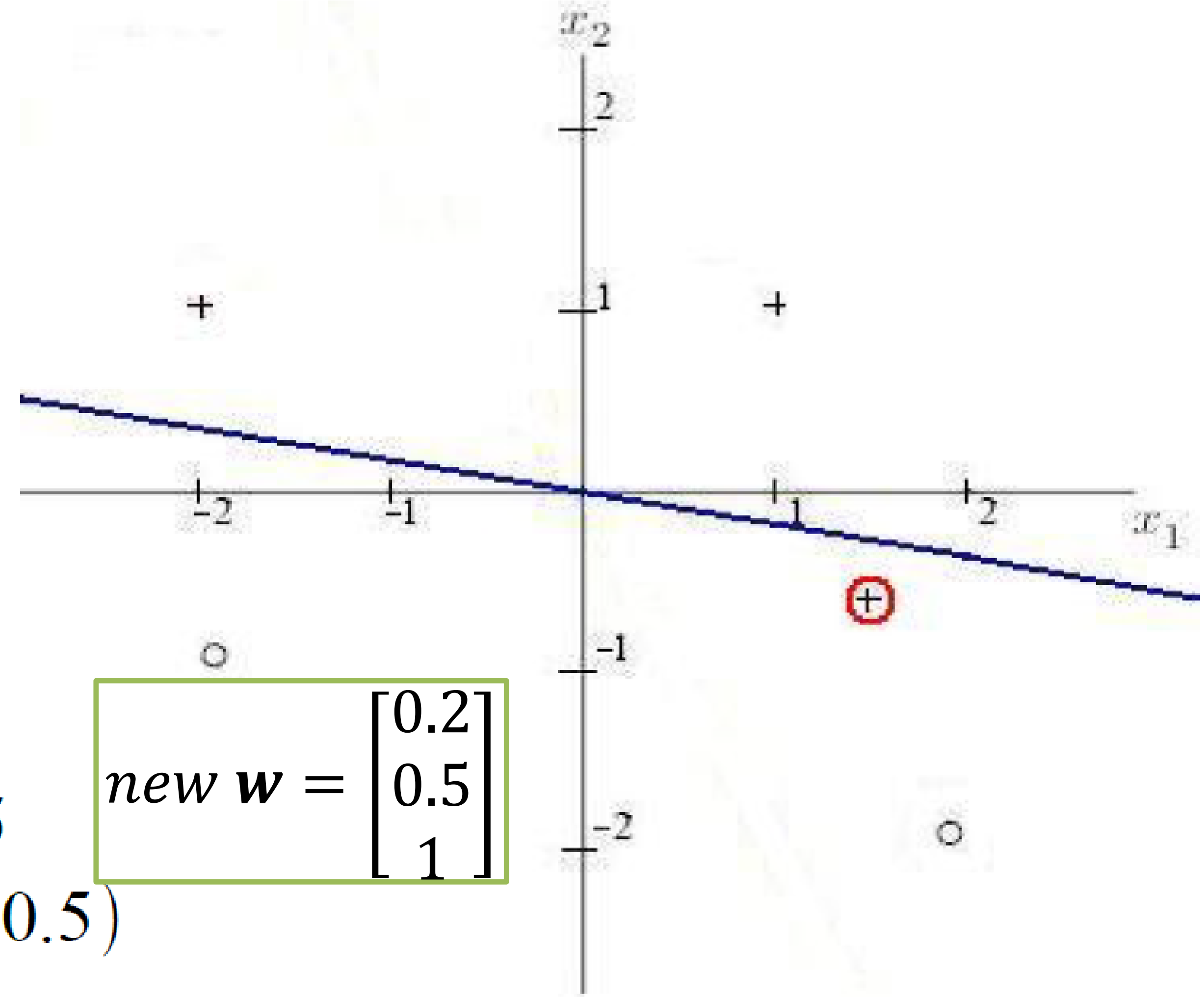
$x_1 = 1.5,\ x_2 = -0.5$

$$w_0 = w_0 + 0.2 * 1$$

$$w_1 = w_1 + 0.2 * 1.5$$

$$w_2 = w_2 + 0.2 * (-0.5)$$

$$new\ \boldsymbol{w} = \begin{bmatrix} 0.2 \\ 0.5 \\ 1 \end{bmatrix}$$



Learning Example

$\eta = 0.2$

$$w = \begin{pmatrix} 0.2 \\ 0.5 \\ 1 \end{pmatrix}$$

previous w

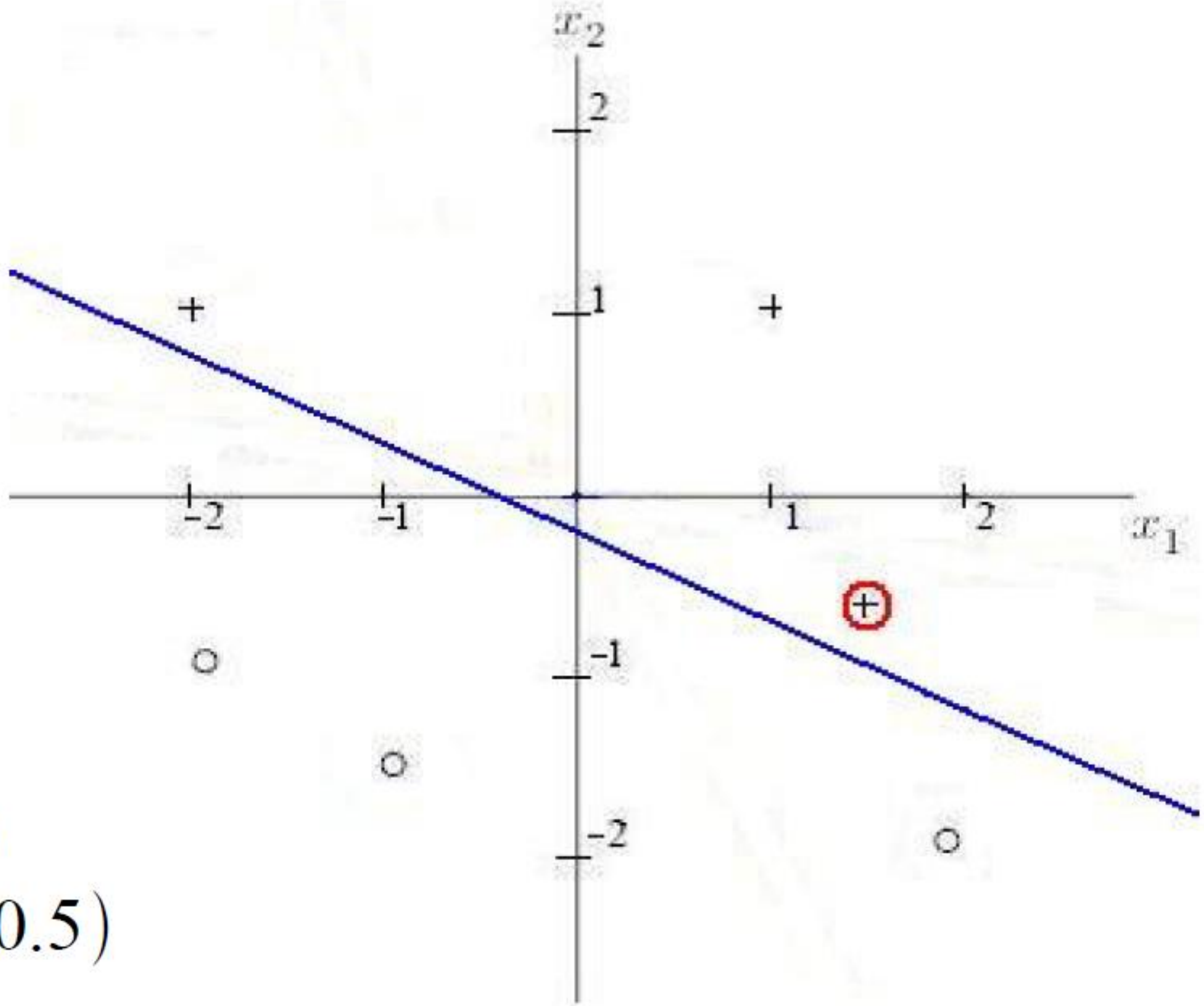
$$\boxed{w = \begin{pmatrix} 0 \\ 0.2 \\ 1.1 \end{pmatrix}}$$

$x_1 = 1.5,\ x_2 = -0.5$

$$w_0 = w_0 + 0.2 * 1$$

$$w_1 = w_1 + 0.2 * 1.5$$

$$w_2 = w_2 + 0.2 * (-0.5)$$



All samples correctly classified →perceptron algorithm stops !

VC Dimension of Halfspaces

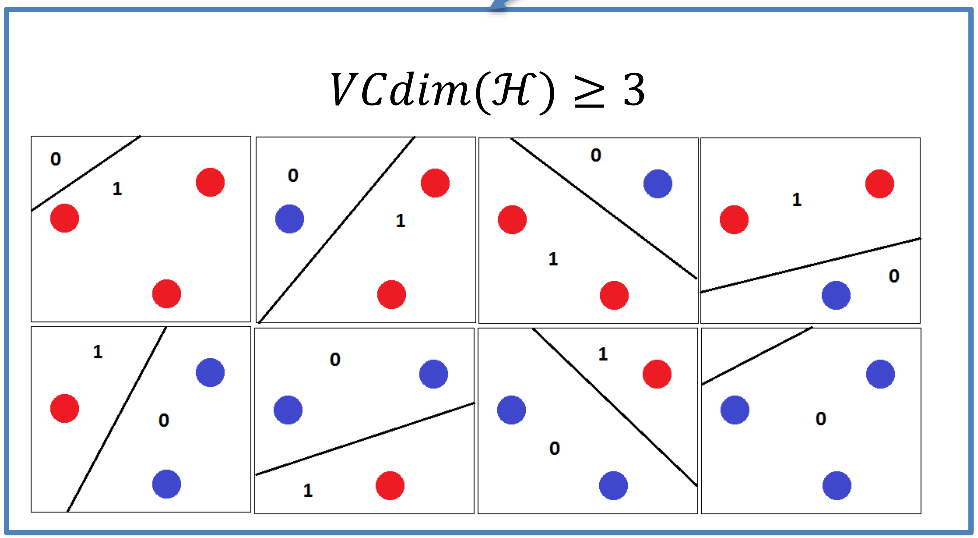
For the halfspace hypothesis class:

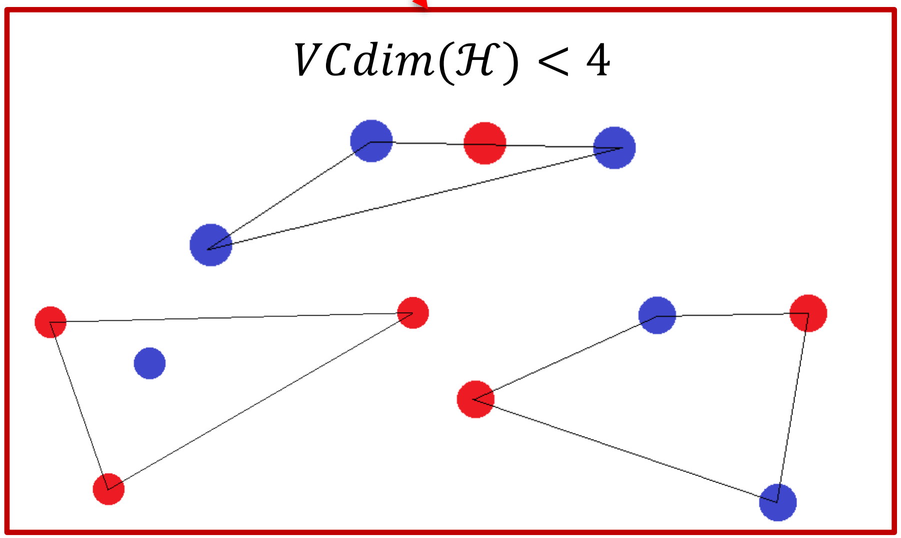
- The VC dimension of the class of homogenous halfspaces in $\mathbb{R}^d$ is *d*
- The VC dimension of the class of non-homogenous halfspaces in $\mathbb{R}^d$ is *d+1*

Example: 2D space

- *i.e., the hyperplane is a line*

  d=2 in non-homogeneous or d+1=3 in homogeneous coord.
- $VCdim(\mathcal{H}) \geq 3$ (see example)
- $VCdim(\mathcal{H}) < 4$ (no set of size 4 can be shattered)





DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

VC Dimension of Halfspaces: Demonstration (1)

The VC dimension of the class of homogenous halfspaces in $\mathbb{R}^d$ is *d*

Demonstration (homogenous case)*:*

$a)\ VCdim\left(HS_d^{hmg}\right) \geq d$

1. Consider the set $\boldsymbol{e}_1, \dots, \boldsymbol{e}_d$ where $\forall i\!: \boldsymbol{e_i} = (0, \dots, 0,1,0, \dots, 0)$
   - i.e., all "0 " except "1" in the i-th coordinate
2. The set is shattered by the homogeneous halfspace: to obtain the labeling $\boldsymbol{e}_1, \dots, \boldsymbol{e}_d$ *set* $\boldsymbol{w} = (y_1, \dots, y_d) \Rightarrow <\boldsymbol{w}, \boldsymbol{e}_i> = y_i \quad \forall i$
   - for each vector only the multiplication with the corresponding label is $\neq$ 0 (only the i-th term remains)

$$<(y_1, \dots, y_{i-1}, y_i, y_{i+1}, \dots, y_d), (0, \dots, 0,1,0, \dots, 0) = y_i$$

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

VC Dimension of Halfspaces: Demonstration (2)

b) $VCdim\left(HS_d^{hmg}\right) < d+1$

1. $\boldsymbol{x}_1, \dots, \boldsymbol{x}_{d+1}$ generic set of *d+1* vectors in $\mathbb{R}^d$
2. They must be linearly dependent:

$$\exists a_1, \dots, a_{d+1} \in \mathbb{R}\ (not\ all\ zero): \sum_{i=1}^{d+1} a_i \boldsymbol{x}_i = 0$$

3. Define $I = \{i: a_i > 0\}$, $J = \{j: a_j < 0\}$: either *I* or *J* are non-empty
4. Assume both non-empty*:* $\sum_{i \in I} a_i \boldsymbol{x_i} = \sum_{j \in J} |a_j| \boldsymbol{x_j}$
5. *By contradiction:* assume that the set is shattered: ∃ a vector ***w*** such that $\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle > 0\ \forall i \in I$ and $\langle \boldsymbol{w}, \boldsymbol{x_j} \rangle < 0\ \forall j \in J$
6. It follows a contradiction :

$$0 < \sum_{i \in I} a_i \langle \boldsymbol{x_i}, \boldsymbol{w} \rangle = \langle \sum_{i \in I} a_i \boldsymbol{x_i}, \boldsymbol{w} \rangle = \langle \sum_{j \in J} |a_j| \boldsymbol{x_j}, \boldsymbol{w} \rangle = \sum_{j \in J} |a_j| \langle \boldsymbol{x_j}, \boldsymbol{w} \rangle < 0$$

7. If I or J are empty just replace one of the two inequalities with "=" but still there is the contradiction!!

Linear Regression

Regression: estimate the relation between some explanatory variables (*features*) and some real valued outcome

- Domain set : $\mathcal{X} \in \mathbb{R}^d$, $\boxed{\text{label set : } \mathcal{Y} = \mathbb{R}}$
- Find $h \in \mathcal{H}_{reg}$: $\mathbb{R}^d \rightarrow \mathbb{R}$ that best approximates the relation between input and output
- Hypothesis class (*linear regression*):

$$\mathcal{H}_{reg} = L_d = \{\boldsymbol{x} \rightarrow <\boldsymbol{w}, \boldsymbol{x}> + b : \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

- Loss function: *squared loss (L2, MSE)* is commonly used but other functions are possible (e.g., *mean absolute error*)

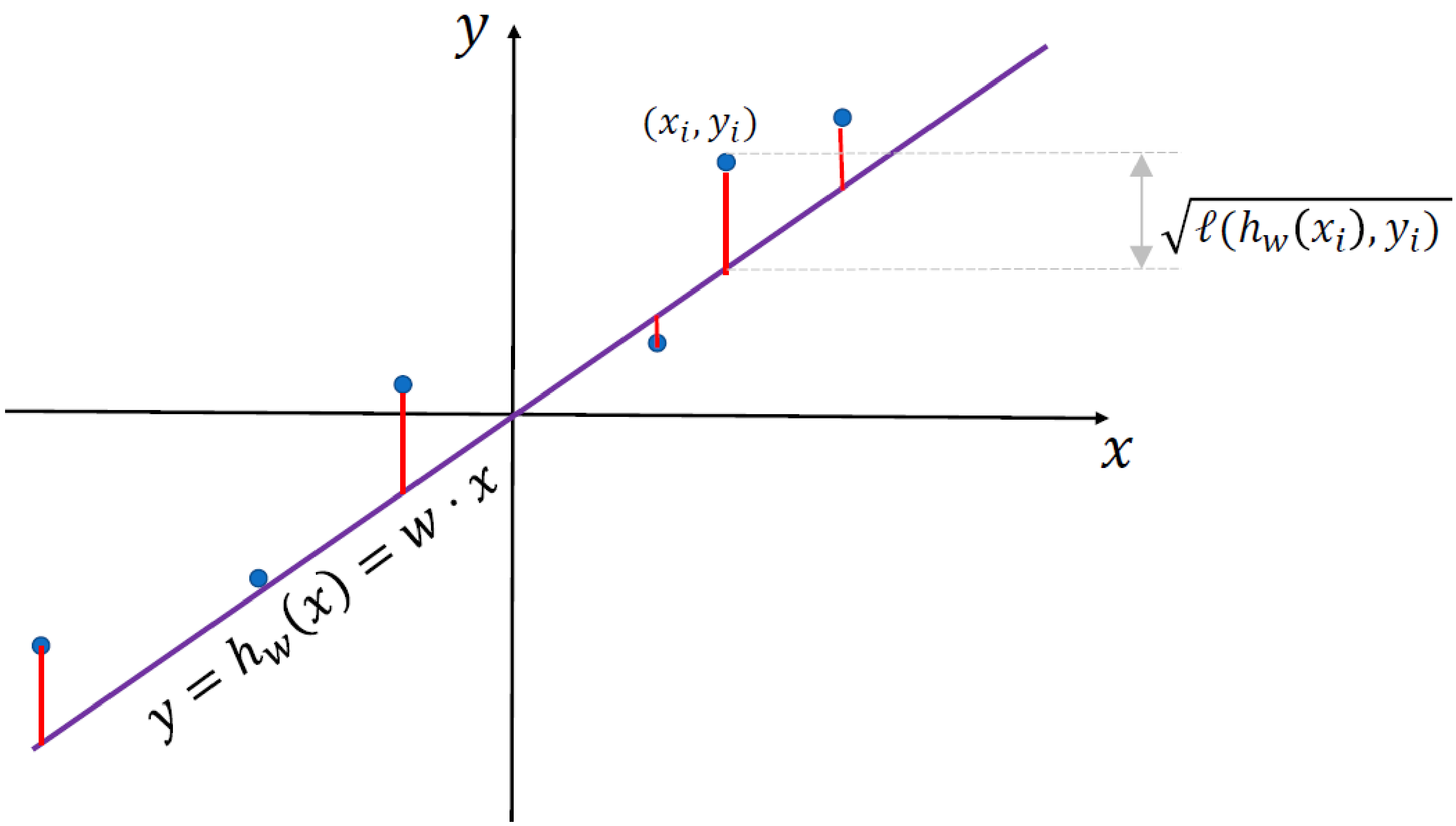
$$\ell(h, (\boldsymbol{x}, y)) \stackrel{\text{def}}{=} (h(\boldsymbol{x}) - y)^2$$

- Empirical Risk function: *Mean Squared Error* on training set

$$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} (h(\boldsymbol{x}_i) - y_i)^2$$

Linear Regression (1D)

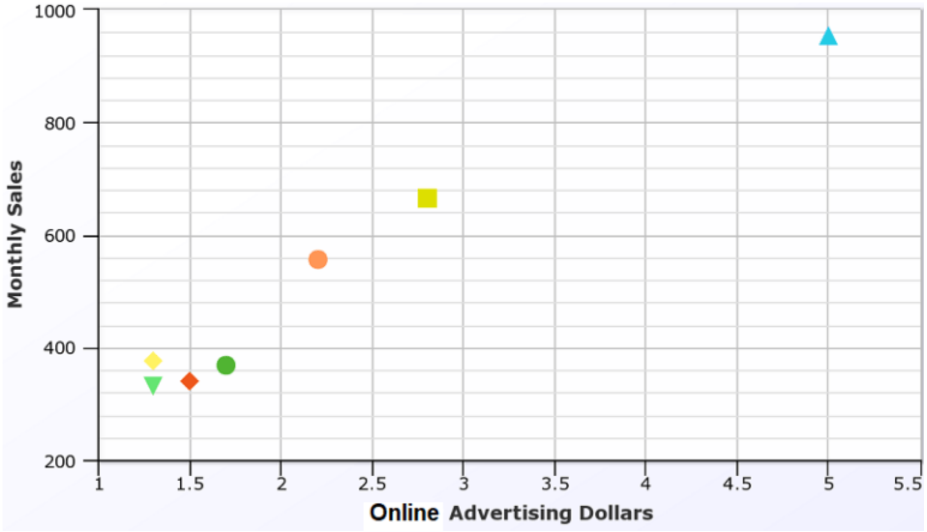
$\mathcal{X} = \mathbb{R}^1 \; \mathcal{Y} = \mathbb{R}$

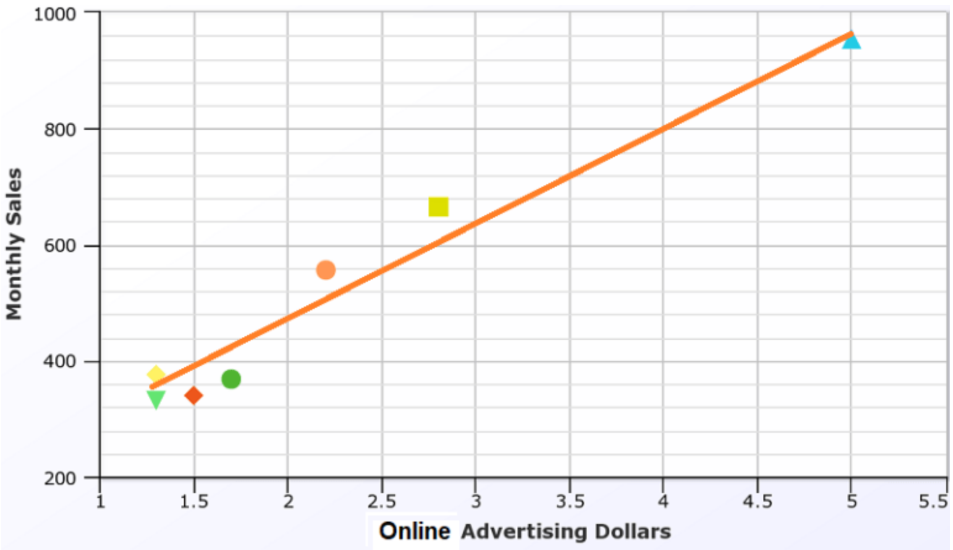


DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Example: Linear Regression (1)

Online Store	*Monthly Sales (in 1000 $)*	*Online Advertising Dollars (1000 $)*
1	368	1.7
2	340	1.5
3	665	2.8
4	954	5.0
5	331	1.3
6	556	2.2
7	376	1.3





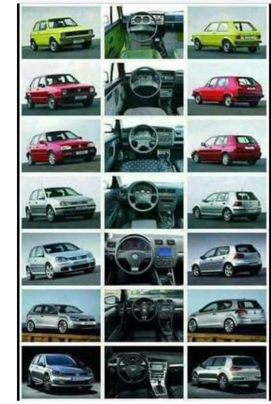
DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

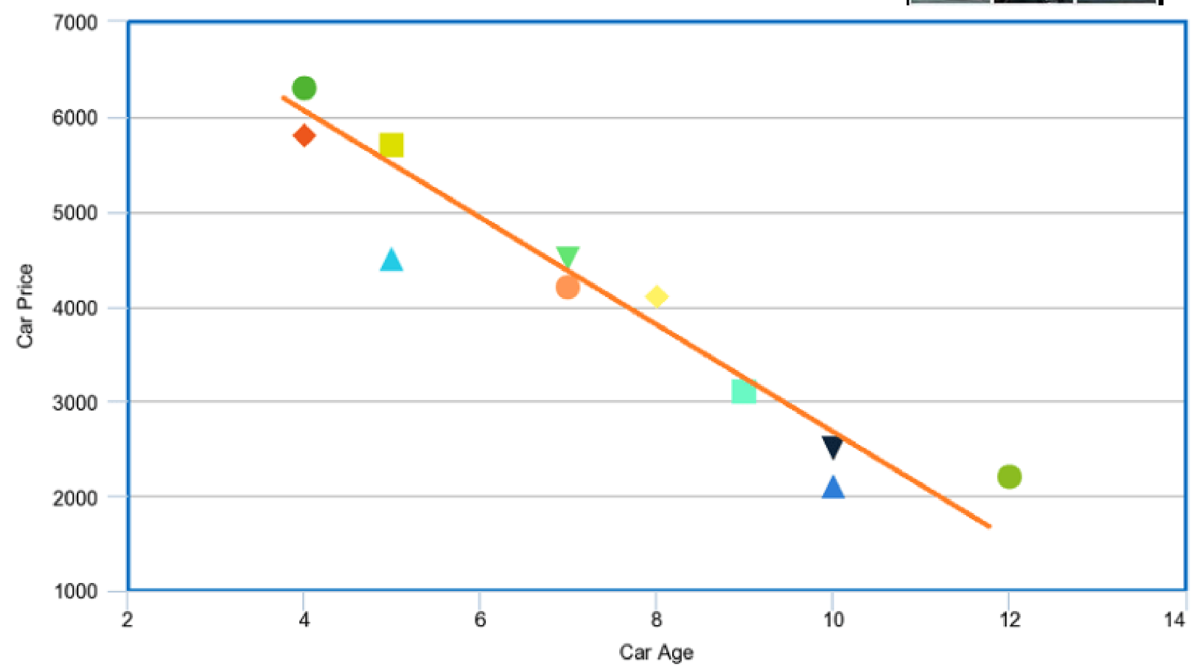
Example: Linear Regression (2)

Car Age (years)	Price (€)
4	6300
4	5800
5	5700
5	4500
7	4500
7	4200
8	4100
9	3100
10	2100
11	2500
12	2200





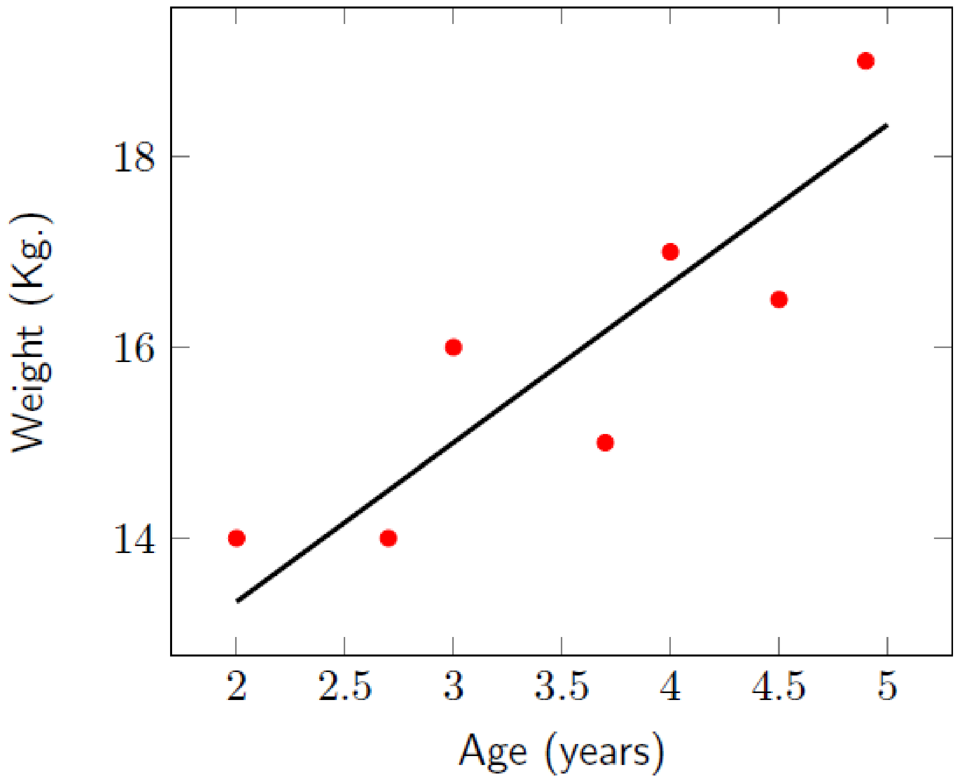





DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Example: Linear Regression (3)

- $\mathcal{X} \subset \mathbb{R}^d$, $\mathcal{Y} \subset \mathbb{R}$, $\mathcal{H} = \{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle : \mathbf{w} \in \mathbb{R}^d\}$
- Example: $d = 1$, predict weight of a child based on his age.



Least Squares

$$\arg\min_{\boldsymbol{w}} L_s(h_{\boldsymbol{w}}) = \arg\min_{\boldsymbol{w}} \frac{1}{m}\sum_{i=1}^{m}(\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle - y_i)^2$$

- The *least squares* algorithm solves the ERM problem for linear regression predictors with the squared loss
- Find the parameters vector that minimize the MSE between the estimated and training values
- To solve the problem: calculate gradient w.r.t vector ***w*** and set to 0

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Least Squares: Solution

$$\underset{\boldsymbol{w}}{\arg\min}\, L_s(h_{\boldsymbol{w}}) = \underset{\boldsymbol{w}}{\arg\min} \frac{1}{m}\sum_{i=1}^{m}(\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle - y_i)^2$$

- Compute gradient w.r.t ***w*** and set to 0

$$\frac{\partial L_s}{\partial \boldsymbol{w}} = \frac{2}{m}\sum_{i=1}^{m}(< \boldsymbol{w}, \boldsymbol{x}_i > -y_i)\boldsymbol{x}_i = 0 \rightarrow \sum_{i=1}^{m}\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle \boldsymbol{x_i} = \sum_{i=1}^{m} y_i \boldsymbol{x}_i$$

- Set

$$A = \left(\sum_{i=1}^{m} \boldsymbol{x}_i \boldsymbol{x}_i^T\right) = \begin{bmatrix} \vdots & & \vdots \\ \boldsymbol{x_1} & \cdots & \boldsymbol{x_m} \\ \vdots & & \vdots \end{bmatrix} \begin{bmatrix} \cdots & \boldsymbol{x_1} & \cdots \\ & \vdots & \\ \cdots & \boldsymbol{x_m} & \cdots \end{bmatrix} \qquad \boldsymbol{b} = \sum_{i=1}^{m} y_i \boldsymbol{x}_i = \begin{bmatrix} \vdots & & \vdots \\ \boldsymbol{x_1} & \cdots & \boldsymbol{x_m} \\ \vdots & & \vdots \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

- The solution is:

$$\sum_{i=1}^{m}\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle \boldsymbol{x_i} = \sum_{i=1}^{m} y_i \boldsymbol{x}_i \rightarrow A\boldsymbol{w} = \boldsymbol{b} \rightarrow \boldsymbol{w} = A^{-1}\boldsymbol{b}$$

- The unknown is ***w***, *A: dxd* matrix, ***b*** and ***w***: *d*-dimensional vectors
- Works also int the case in which A is not invertible, but this requires a special handling (*not part of the course*)

Polynomial Regression

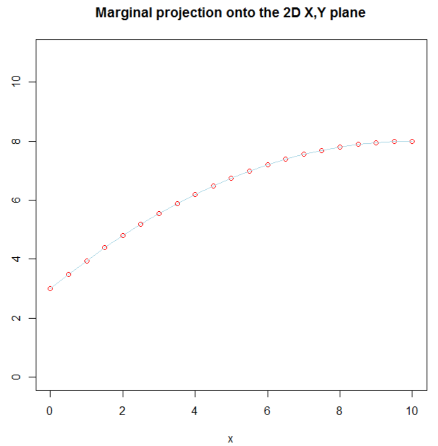
- Polynomial regression: find the one dimensional polynomial of degree *n* that better predicts the data
  - $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$
  - Need to estimate the coefficient vector $\boldsymbol{a}$
  - 1D polynomial pred. deg. *n*: $\mathcal{H}_{poly}^{n} = \{x \to p(x)\}, \mathcal{X} = \mathbb{R}, \mathcal{Y} = \mathbb{R}$
- Reduce the problem to a *n*-dimensional linear regression using the mapping:

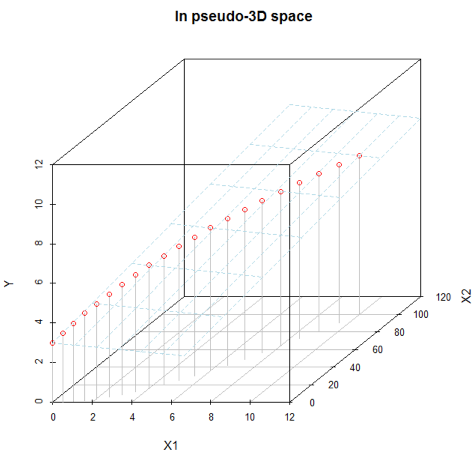
$$\psi: \mathbb{R} \to \mathbb{R}^{n+1} \quad \psi(x) = (1, x, x^2, \dots, x^n)$$

- We obtain:

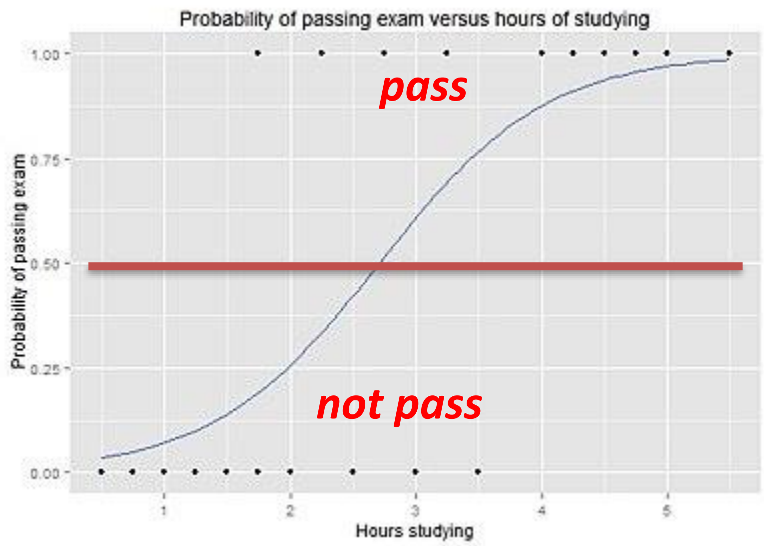
$$< \boldsymbol{a}, \psi(x) > = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

- Find the vector of coefficients $\boldsymbol{a}$ using the Least Square algorithm
- Non-linear relation becomes linear in the higher dimensional space
- Notice that the variables are not independent
- The optimization can become unstable for large *n*



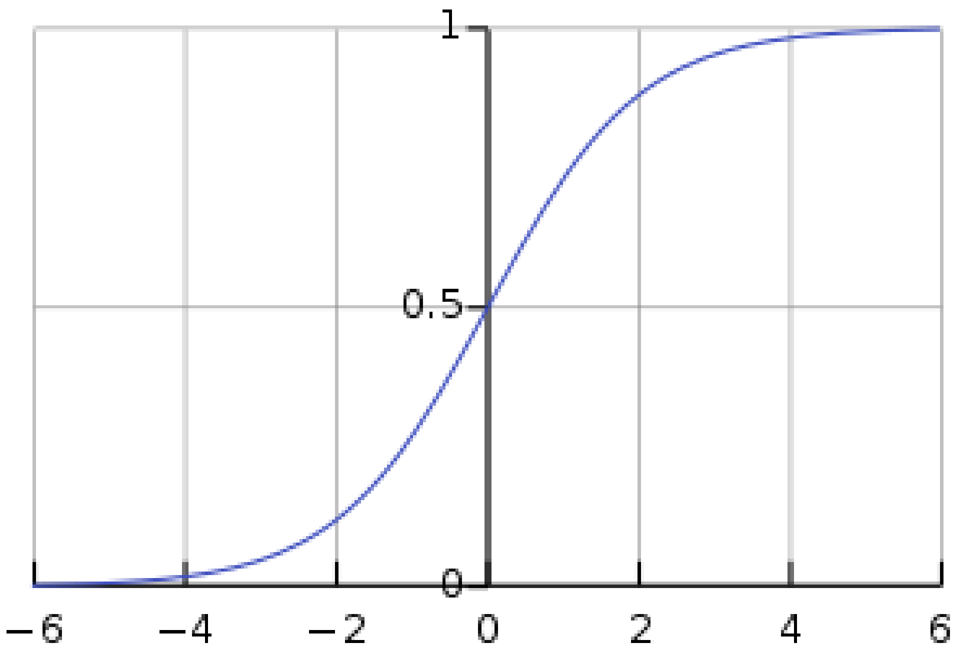


Logistic Regression

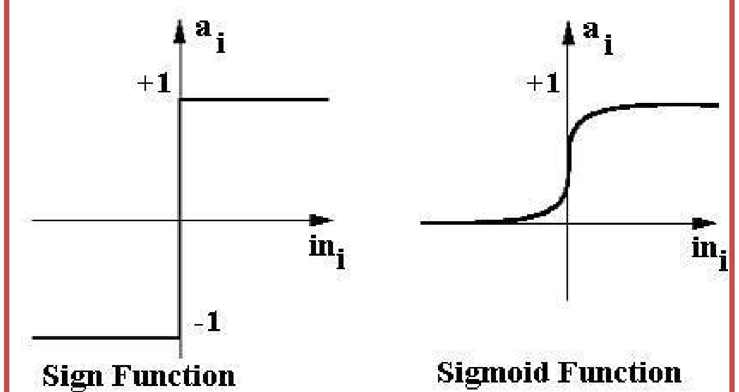


- Reframe a classification problem as a regression one
- Target as in regression:
  - learn a function $h: \mathbb{R}^d \rightarrow [0; 1]$
  - the output of *h* is a real number
- Used for classification:
  - interpret the output of *h* as the probability that the label is 1
  - ➢ *regression-like output for classification !*
- We'll deal with binary classification but the approach can be extended to the multi-class setting
- Hypothesis class $\mathcal{H}: \phi_{sig} \circ L_d$ where $\phi: \mathbb{R} \rightarrow [0,1]$ is the sigmoid function and L_d a linear function

Sigmoid Function



$$\phi_{sig}(z) = \frac{1}{1 + e^{-z}}$$



- Bigger than ½ for positive values and smaller for negative ones
- Tends to 1 for large positive values and to 0 for negative ones
- Can be viewed as a scaled and shifted "soft" sign function

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Loss for Logistic Regression

- Instead of hard choice → use probability of correct label being 0 or 1

$$H_{sig} = \phi_{sig} \circ L_d = \{\boldsymbol{x} \rightarrow \phi_{sig}(< \boldsymbol{w}, \boldsymbol{x} >) : \boldsymbol{w} \in \mathbb{R}^d\}$$

- Hypothesis class : $h_{\boldsymbol{w}}(\boldsymbol{x}) = \frac{1}{1+e^{-\langle \boldsymbol{w}, \boldsymbol{x} \rangle}}$

- Loss function: $\ell\big(h_{\boldsymbol{w}}, (\boldsymbol{x}, y)\big) = \log(1 + e^{-y<\boldsymbol{w},\boldsymbol{x}>})$

- ERM Problem: $argmin_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} log(1 + e^{-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle})$

Logistic Loss Function

Loss function: $\ell\left(h_w, (x, y)\right) = \log(1 + e^{-y<w,x>})$, why ?

Consider $h_w(x) = \frac{1}{1+e^{-\langle w,x \rangle}} \leftrightarrow y \in \{+1, -1\}$

- Case *y=1*: need $h_w(x) \to 1$

$$h_w(x) = \frac{1}{1 + e^{-\langle w,x \rangle}} = \frac{1}{1 + e^{-y\langle w,x \rangle}}$$

  - If denominator small $h_w(x) \to 1$ good case
  - If denominator large $h_w(x) \to 0$ error

- Case *y=-1*: need $h_w(x) \to 0 \Rightarrow 1 - h_w(x) \to 1$

$$1 - h_w(x) = 1 - \frac{1}{1+e^{-\langle w,x \rangle}} = \frac{1+e^{-\langle w,x \rangle}-1}{1+e^{-\langle w,x \rangle}} = \frac{1}{e^{\langle w,x \rangle}+1} = \frac{1}{1+e^{-y\langle w,x \rangle}}$$

- Same as before :
  - If denominator small $1 - h_w(x) \to 1$ good case
  - if denominator large $1 - h_w(x) \to 0$ error

- Loss need to increase with $1 + e^{-y\langle w,x \rangle}$ and log function is monotonic

Maximum Likelihood Estimation (MLE)

Not part of the course

Maximum Likelihood Estimation (MLE) is a statistical approach for finding the parameters that maximize the joint probability of a given dataset assuming a specific parametric probability function

- MLE essentially assumes a generative model for the data
- MLE solution is equivalent to ERM solution for logistic regression

MLE approach:

1. Given training set $S = ((\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_m}, y_m))$, assume each $(\boldsymbol{x_i}, y_i)$ is i.i.d. from some probability distribution (that is characterized by some parameters)
2. Consider $P[S|\theta]$ (likelihood of data given parameters)
3. log likelihood: $L(S;\theta) = \log(P[S|\theta])$
   - *log*: monotonic → same maximum, but simpler to differentiate
4. *Maximum Likelihood Estimator (MLE)*: $\hat{\theta} = argmax_\theta L(S;\theta)$

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

MLE and Logistic Regression

Not part of the course

MLE solution is equivalent to ERM solution for logistic regression

Logistic Regression:

1. Assume training set $S = ((\boldsymbol{x_1}, y_1), \dots, (\boldsymbol{x_m}, y_m))$
2. $P[y_i = 1] = h_{\boldsymbol{w}}(\boldsymbol{x_i}) = \frac{1}{1+e^{-\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle}} = \frac{1}{1+e^{-y_i \langle \boldsymbol{w}, \boldsymbol{x_i} \rangle}}$ (since $y_i = 1$)
3. $P[y_i = -1] = 1 - h_{\boldsymbol{w}}(\boldsymbol{x_i}) = \frac{1}{1+e^{\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle}} = \frac{1}{1+e^{-y_i \langle \boldsymbol{w}, \boldsymbol{x_i} \rangle}}$ (first equality recall logistic loss, 2nd since $y_i = -1$)
4. Likelihood of training set (joint probability $P[S|\boldsymbol{w}]$) : $\prod_{i=1}^{m} \left(\frac{1}{1+e^{-y_i \langle \boldsymbol{w}, \boldsymbol{x_i} \rangle}} \right)$
5. Log likelihood : $\log(P[S|\boldsymbol{w}]) = \log \prod_{i=1}^{m} \left(\frac{1}{1+e^{-y_i \langle \boldsymbol{w}, \boldsymbol{x_i} \rangle}} \right) = -\sum_{i=1}^{m} \log(1 + e^{-y_i \langle \boldsymbol{w}, \boldsymbol{x_i} \rangle})$
   *→corresponds to (-1)*logistic loss*

Maximum Likelihood Estimator:

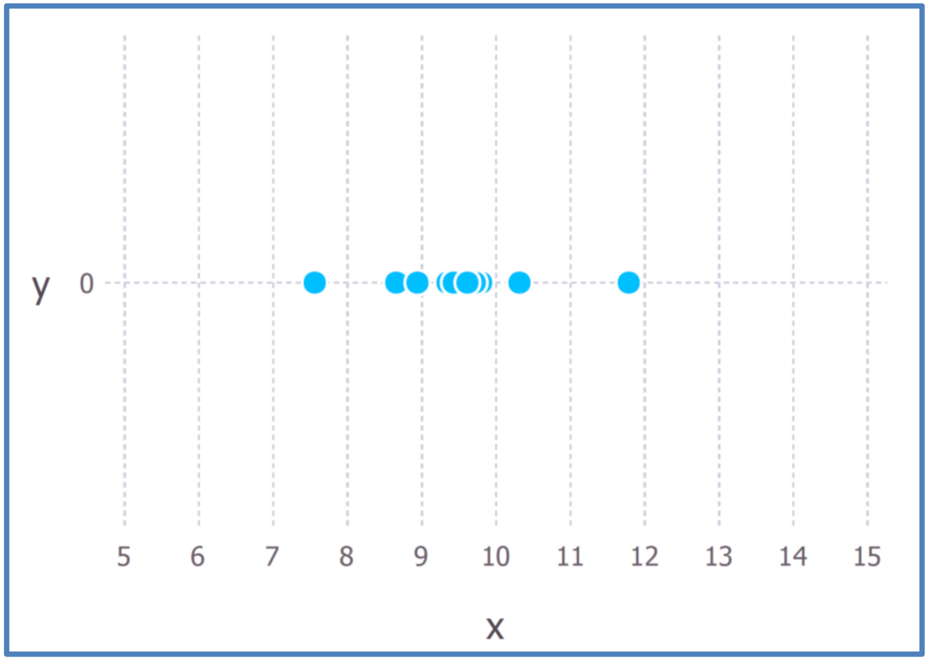
$$argmax_{\boldsymbol{w}} L(S; \boldsymbol{w}) = argmax_{\boldsymbol{w}} \log(P[S|\boldsymbol{w}]) = argmin_{\boldsymbol{w}} \sum_{i=1}^{m} log(1 + e^{-y_i \langle \boldsymbol{w}, \boldsymbol{x_i} \rangle})$$

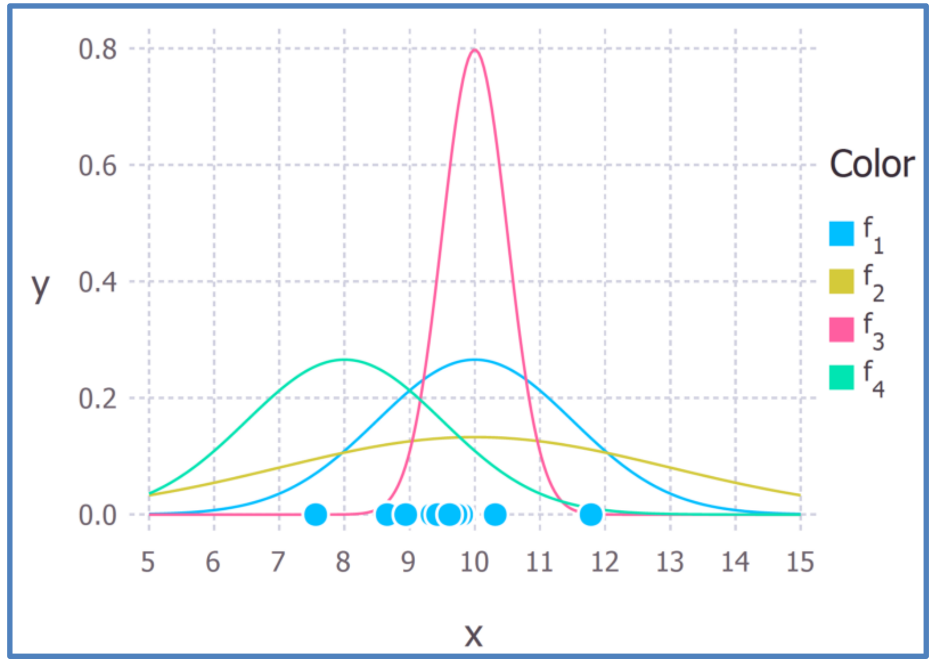
Recall: argmax(-x)=argmin(x) They have the same target !!!

Not part of the course

DELL'INFORMAZIONE

Example: MLE for Gaussian PDF (1)





f1 ~ N (10, 2.25)   f2 ~ N (10, 9),
f3 ~ N (10, 0.25)   f4 ~ N (8, 2.25)

Assume that the data is produced by a Gaussian distribution

$$P(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

find μ, σ maximizing the joint probability of data

Not part of the course

Example: MLE for Gaussian PDF (2)

Joint probability

$$P(9, 9.5, 11; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(9-\mu)^2}{2\sigma^2}\right) \times \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(9.5-\mu)^2}{2\sigma^2}\right) \times \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(11-\mu)^2}{2\sigma^2}\right)$$

Assume 3 samples: 9, 9.5, 11

log

Log likelihood

$$\ln(P(x; \mu, \sigma)) = \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{(9-\mu)^2}{2\sigma^2} + \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{(9.5-\mu)^2}{2\sigma^2} + \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{(11-\mu)^2}{2\sigma^2}$$

Recall log(ab)=log(a)+log(b)

Differentiate w.r.t μ and set to 0

Get optimal mean

$$\frac{\partial \ln(P(x; \mu, \sigma))}{\partial \mu} = \frac{1}{\sigma^2}[9 + 9.5 + 11 - 3\mu] = 0$$

The same can be done for σ

set derivative to 0

$$\mu = \frac{9 + 9.5 + 11}{3} = 9.833$$