Natural Language Processing and Information Retrieval

Statistical Learning Theory: Linear Classifiers

Alessandro Moschitti
Department of information and communication technology
University of Trento
Email: moschitti@dit.unitn.it

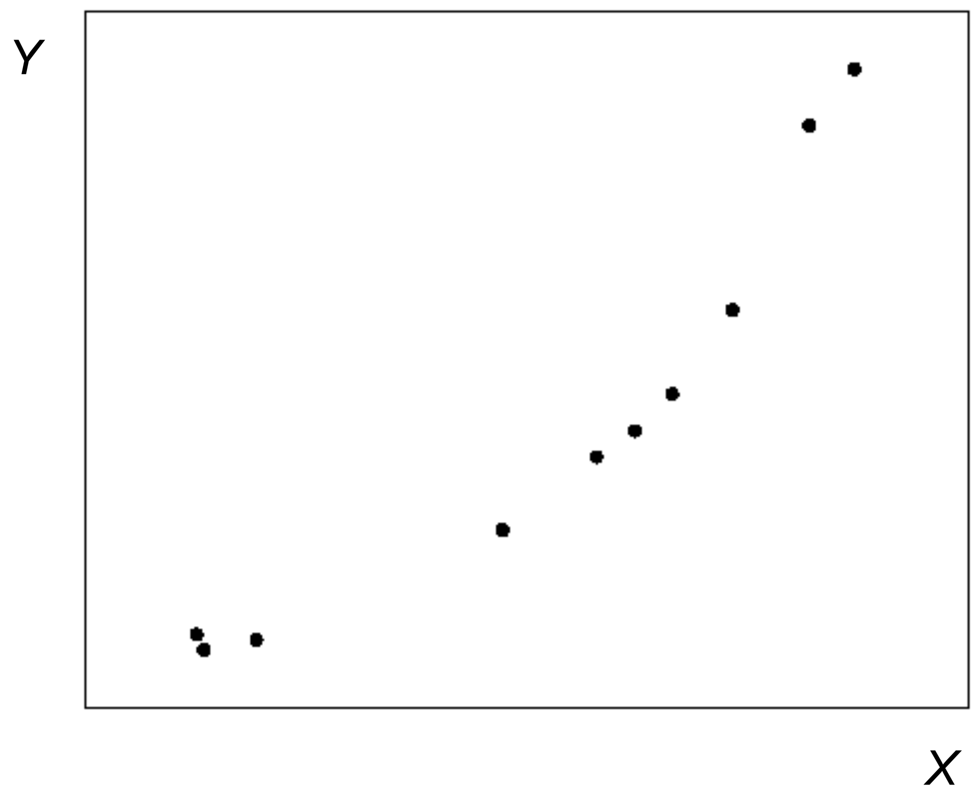
Outline

- Computational Learning theory
  - Introduction to Statistical Learning
  - Perceptron Learning
  - Margins

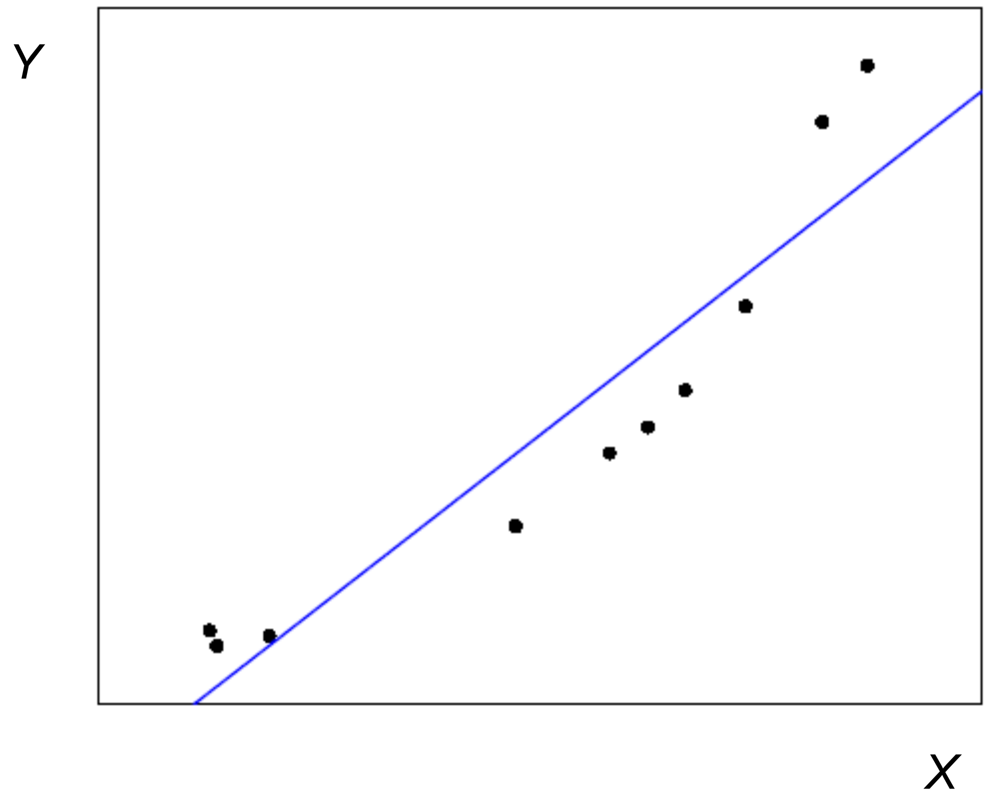
What is Statistical Learning?

- Statistical Methods – Algorithms that learn relations in the data from examples
- Simple relations are expressed by pairs of variables: $\langle x_1, y_1 \rangle$, $\langle x_2, y_2 \rangle, \ldots,$ $\langle x_n, y_n \rangle$
- Learning f such that evaluate y^* given a new value x^*, i.e. $\langle x^*, f(x^*) \rangle = \langle x^*, y^* \rangle$

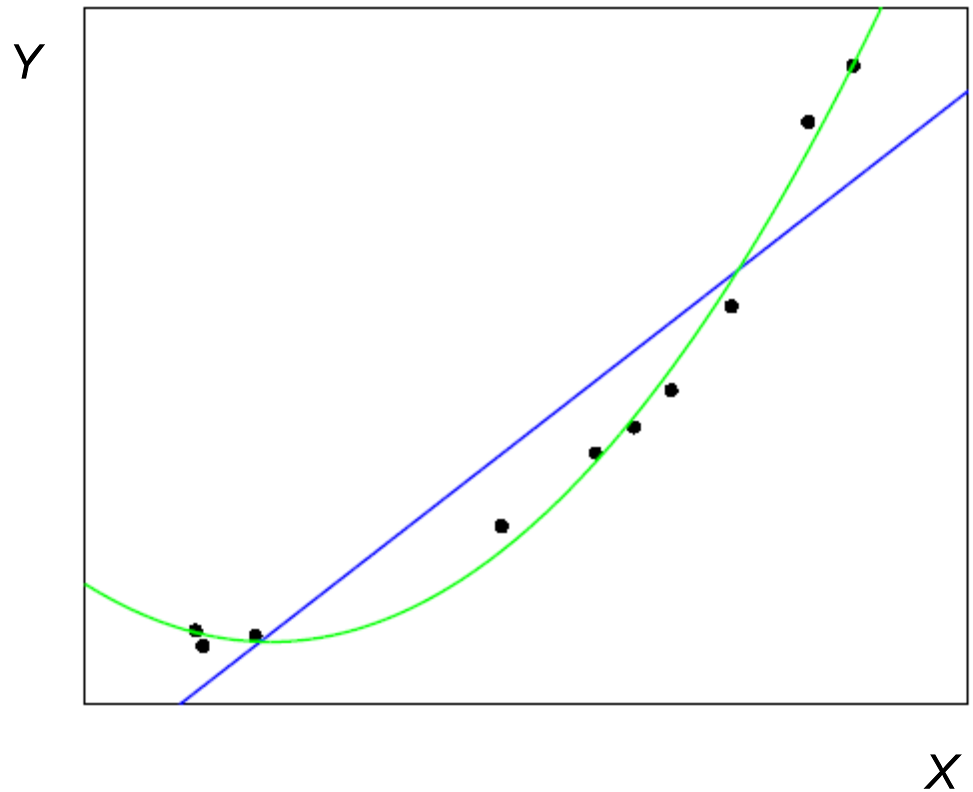
You have already tackled the learning problem



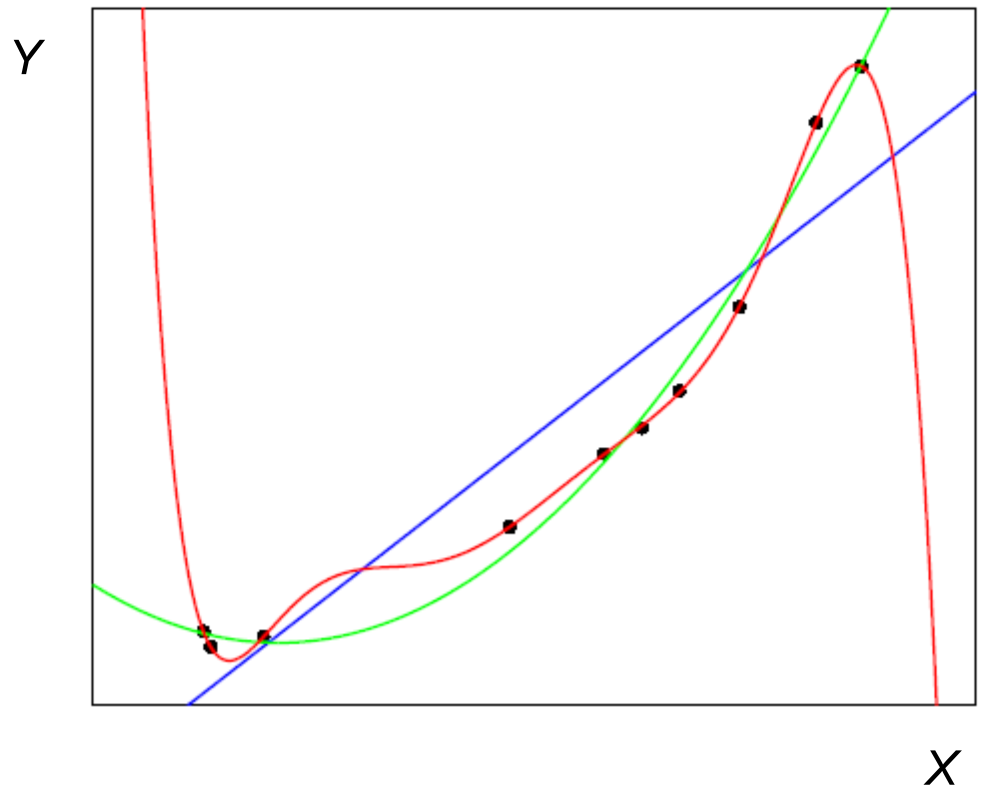
Linear Regression



Degree 2



Degree



Machine Learning Problems

- Overfitting
- How dealing with millions of variables instead of only two?
- How dealing with real world objects instead of real values?

Objectives: defining a well defined statistical framework

- What can we learn and how can we decide if our learning is effective?
- Efficient learning with many parameters
- Trade-off (generalization/and training set error)
- How to represent real world objects

Objectives: defining a well defined statistical framework

- What can we learn and how can we decide if our learning is effective?
- Efficient learning with many parameters
- Trade-off (generalization/and training set error)
- How to represent real world objects

PAC Learning Definition (1)

- Let ***c*** be the function (i.e. a ***concept***) we want to learn
- Let ***h*** be the learned concept and *x* an instance (e.g. a person)
- $error(h) = Prob\ [c(x) < > h(x)]$
- It would be useful if we could find:
- $Pr(error(h) > \varepsilon\ ) < \delta$
- Given a target error ε, the probability to make a larger error is less δ

Definizione di PAC Learning (2)

- This methodology is called Probably Approximately Correct Learning
- The smaller ε and δ are the better the learning is
- Problem:
  - Given ε and δ, determine the size m of the training-set.
  - Such size may be independent of the learning algorithm
- **Let us do it for a simple learning problem**

Lower Bound on training-set size

- Let us reconsider a first general bound:
  - h is bad: $error(h) > \varepsilon$
  - $P(f(x)=h(x))$ for m examples is lower than $(1- \varepsilon)^m$
  - Multiplying by the number of bad hypotheses we calculate the probability of selecting a bad hypothesis
  - $P(bad\ hypothesis) < N \cdot (1- \varepsilon)^m < \delta$
  - $P(bad\ hypothesis) < N \cdot (e^{-\varepsilon})^m = N \cdot e^{-\varepsilon m} < \delta$

$\Rightarrow m > (1/\varepsilon)\ (ln(1/\delta) + ln(N))$

This is a general lower bound

Example

- Suppose we want to learn a boolean function in n variable
- The maximum number of different functions are 2^{2^n}

$$\Rightarrow m > (1/\varepsilon)\,(ln(1/\delta)+ln(2^{2^n}))=$$

$$= (1/\varepsilon)\,(ln(1/\delta)+2^n ln(2))$$

Some Numbers

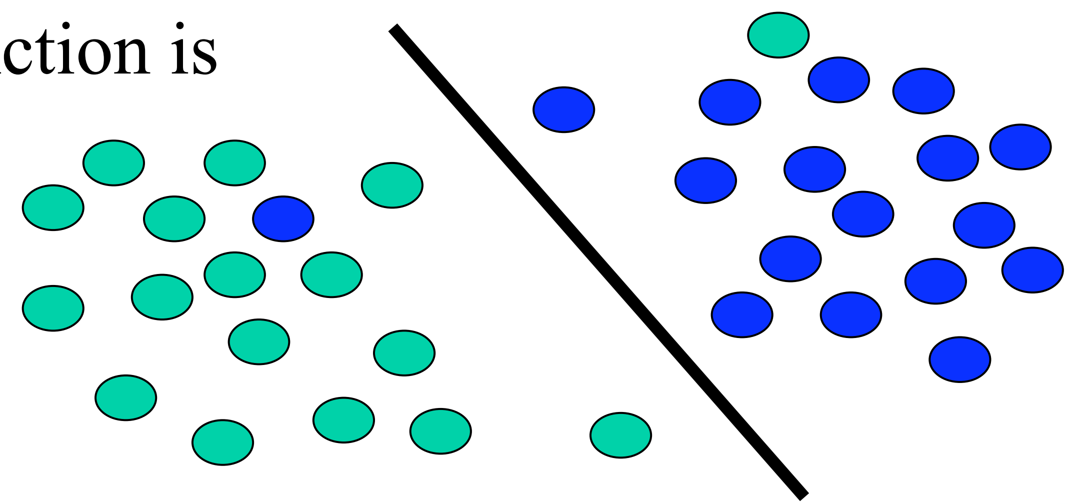
n	epsilon	delta	m
5	0.1	0.1	245
5	0.1	0.01	268
5	0.01	0.1	2450
5	0.01	0.01	2680
10	0.1	0.1	7123
10	0.1	0.01	7146
10	0.01	0.1	71230
10	0.01	0.01	71460

Linear Classifier (1)

- The equation of a hyperplane is

$$f(\vec{x}) = \vec{x} \cdot \vec{w} + b = 0, \quad \vec{x}, \vec{w} \in \Re^n, b \in \Re$$

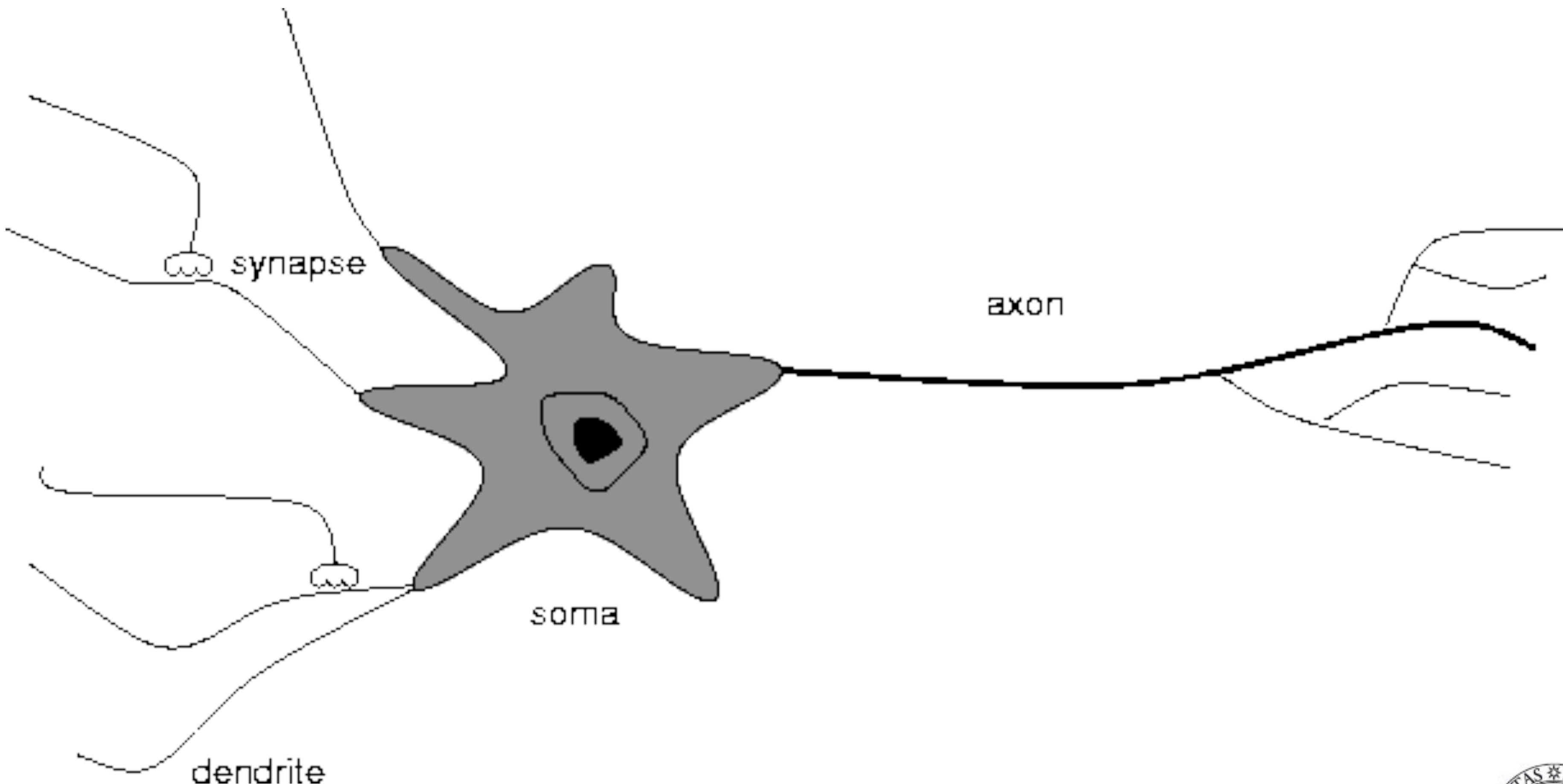
- $\vec{x}$ is the vector representing the classifying example
- $\vec{w}$ is the gradient to the hyperplane
- The classification function is
  $h(x) = \text{sign}(f(x))$



Linear classifiers (2)

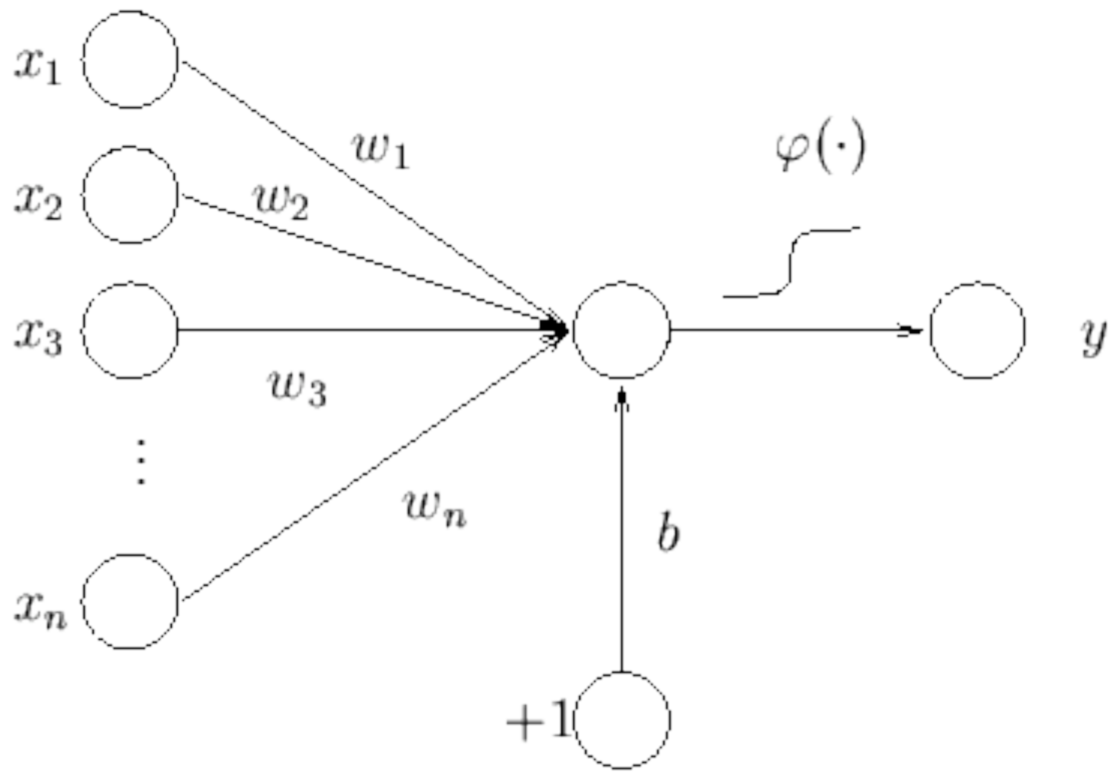
- Linear Functions are the simplest ones from an analytical point of view.
- The basic idea is to select a hypothesis with null error on the training-set.
- To learn a linear function a simple neural network of only one neuron is enough (Perceptron)

An animal neuron



The Perceptron

$$\varphi(\vec{x}) = \mathrm{sgn}\left(\sum_{i=1..n} w_i \times x_i + b \right)$$



Useful Concepts

- ***Functional Margin*** of an example with respect to a hyperplane: $\gamma_i = y_i(\vec{w} \cdot \vec{x}_i + b)$
- ***The distribution of functional margins*** of a hyperplane with respect to a training set S is the distribution of the margins of the examples in S *wrt* the hyperplane $(\vec{w}, b)$.
- ***The functional margin of a hyperplane*** is the minimum margin of the distribution

Notations (con’td)

- If we normalize the hyperplane equation, i.e.
$$\left(\frac{\vec{w}}{\|\vec{w}\|}, \frac{b}{\|\vec{w}\|}\right)$$
, we obtain the ***geometric margin***
- The ***geometric margin*** measure the Euclidean distance between the target point and the hyperplane.
- ***The training set Margin*** is the maximum geometric (functional) margin among all hyperplanes which separates the examples in S.
- The hyperplane associated with the above quantity is called ***maximal margin hyperplane***

Basic Concepts

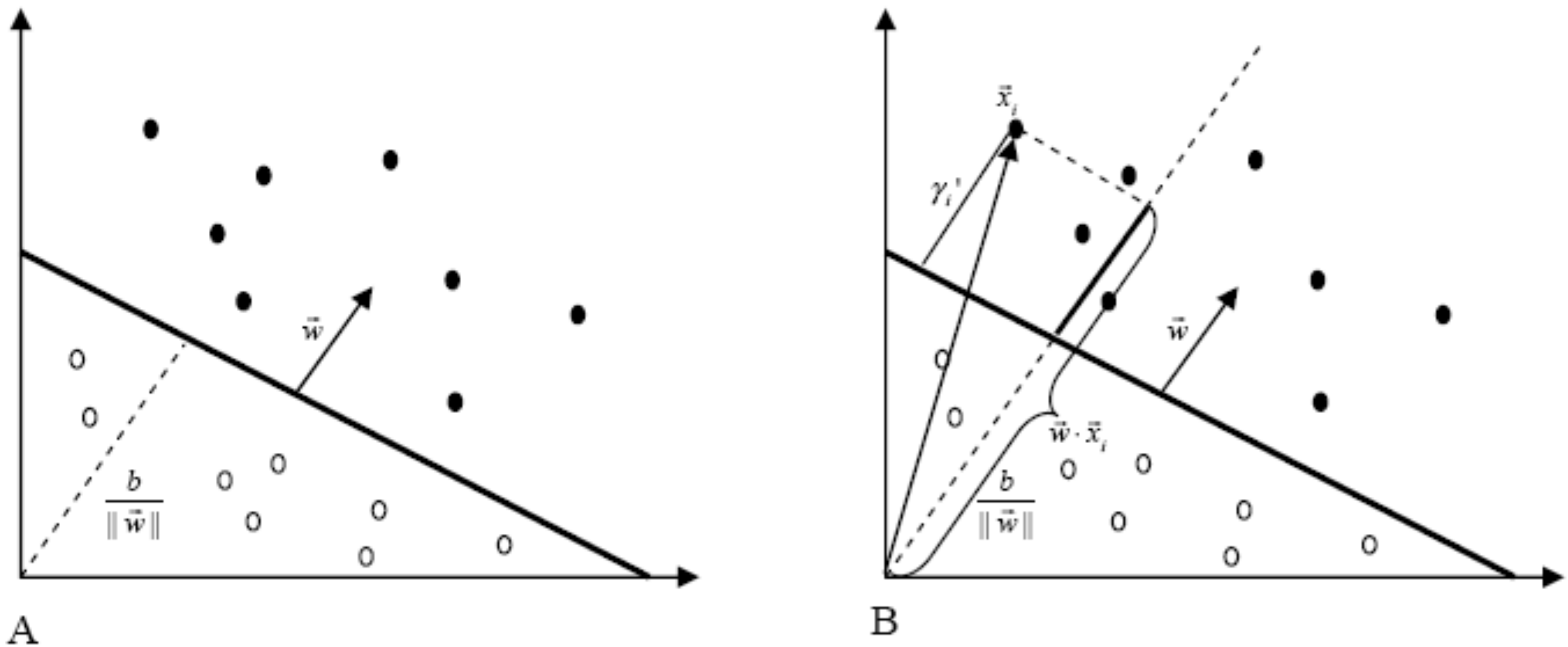
- From $\cos(\vec{x}, \vec{w}) = \dfrac{\vec{x} \cdot \vec{w}}{\|\vec{x}\| \cdot \|\vec{w}\|}$

- It follows that

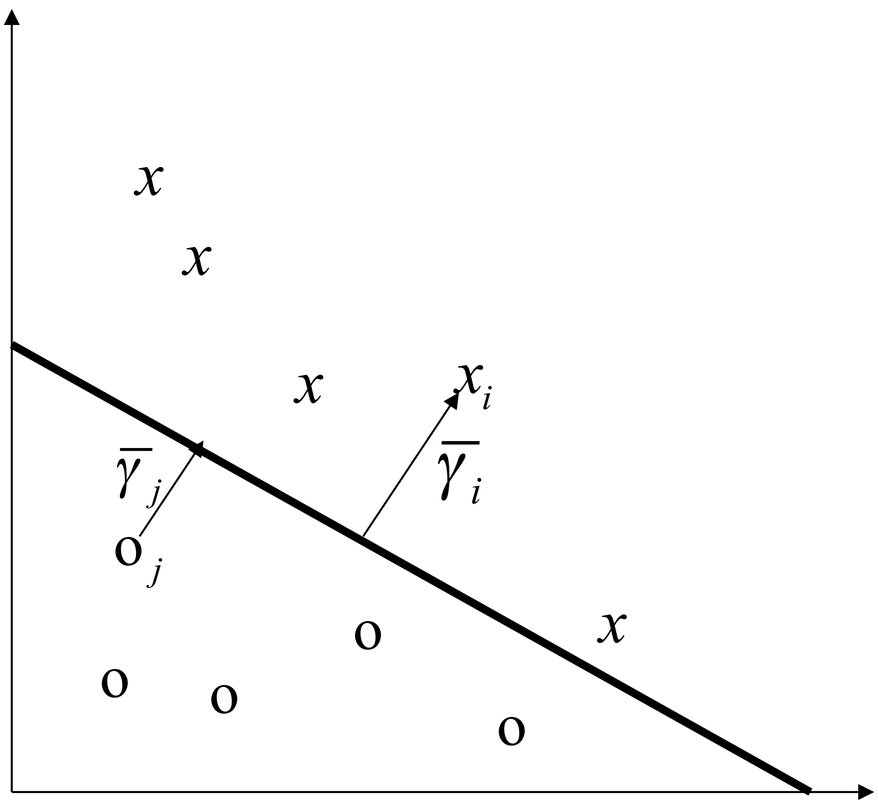
$$\|\vec{x}\| \cos(\vec{x}, \vec{w}) = \frac{\vec{x} \cdot \vec{w}}{\|\vec{w}\|} = \vec{x} \cdot \frac{\vec{w}}{\|\vec{w}\|}$$

- Norm of $\vec{x}$ times the cosine between $\vec{x}$ and $\vec{w}$, i.e. the projection of $\vec{x}$ on $\vec{w}$

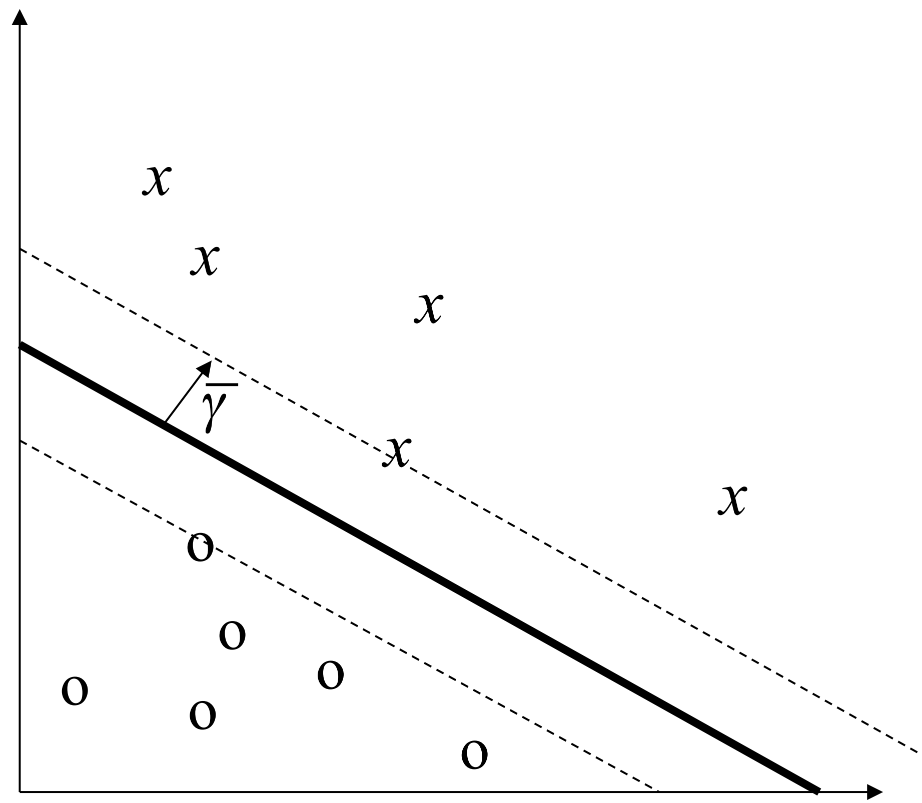
Geometric Margin



Geometric margins of 2 points and hyperplane margin

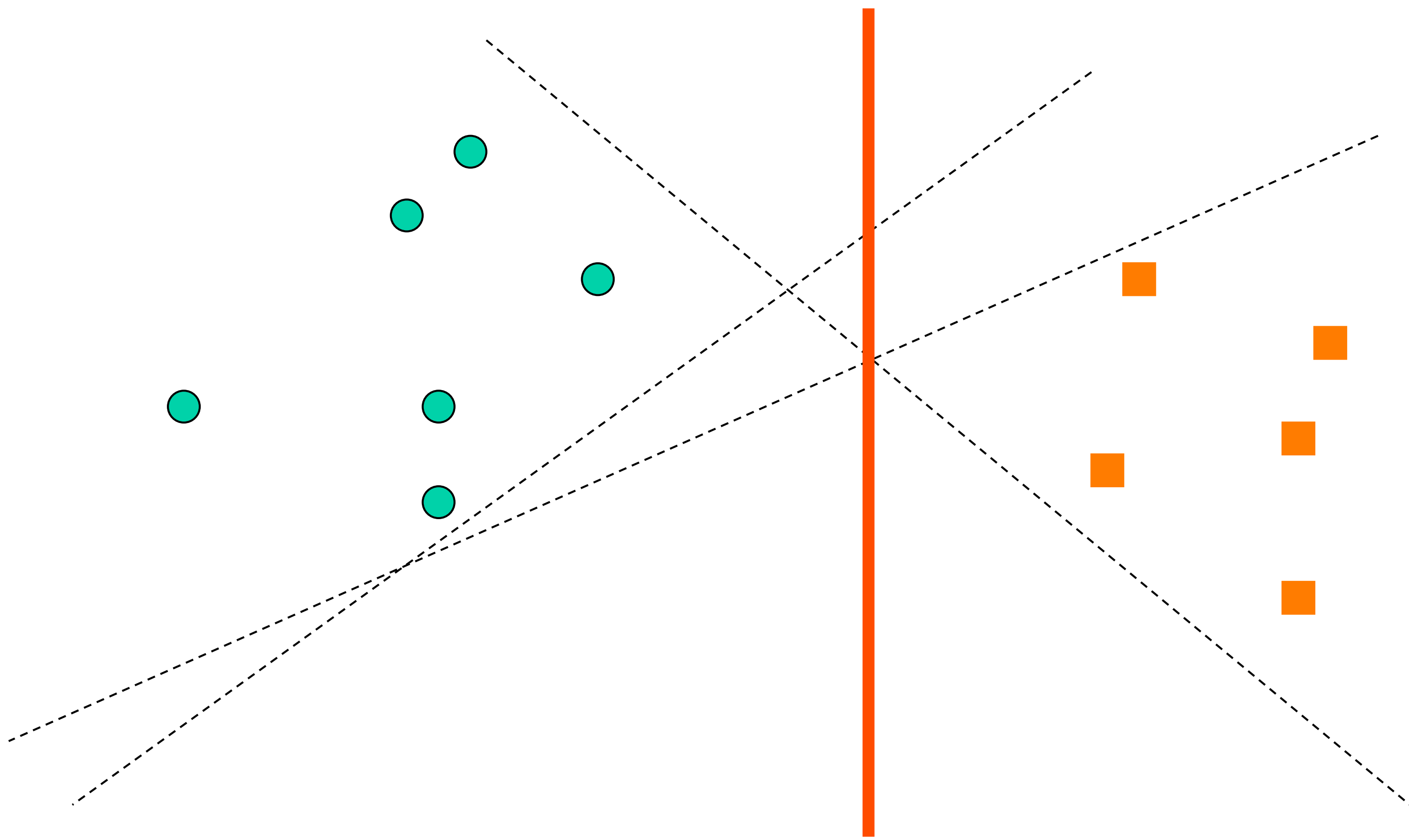


Geometric Margin



Hyperplane Margin

Maximal margin vs other margins



Perceptron training on a data set (on-line algorithm)

$\vec{w}_0 \leftarrow \vec{0}; b_0 \leftarrow 0; k \leftarrow 0; R \leftarrow \max_{1\le i\le l} \|\vec{x}_i\|$

Repeat

  for i = 1 to m

   if $y_i(\vec{w}_k \cdot \vec{x}_i + b_k) \le 0$ then

    $\vec{w}_{k+1} = \vec{w}_k + \eta y_i \vec{x}_i$

    $b_{k+1} = b_k + \eta y_i R^2$

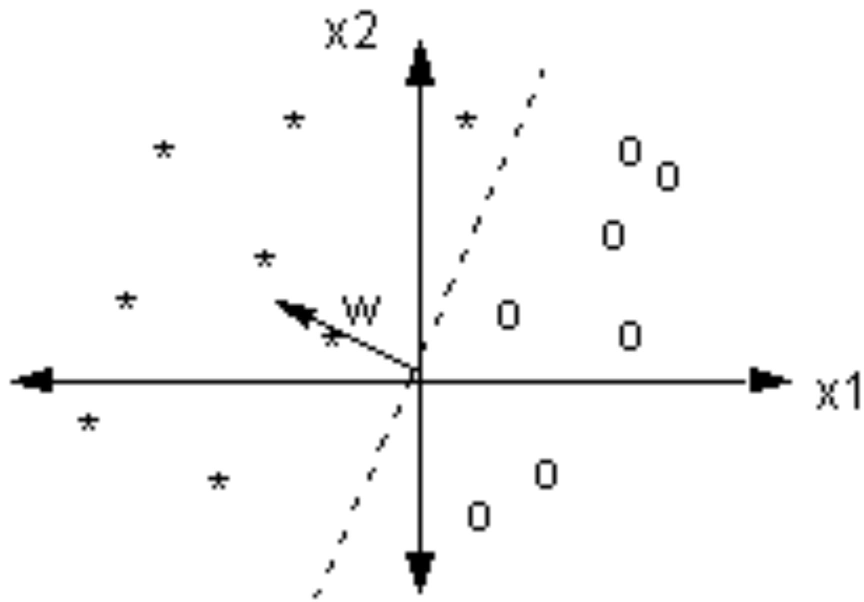
    k = k + 1

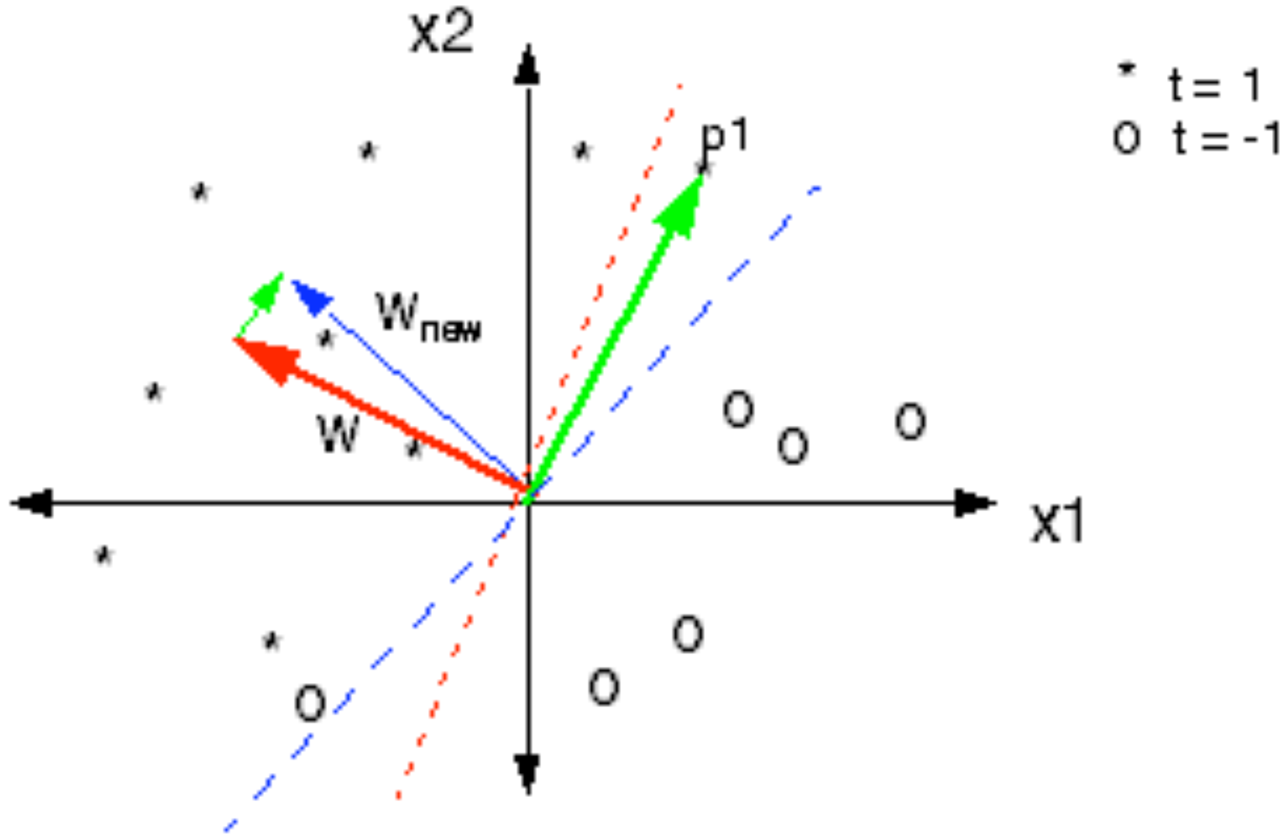
   endif

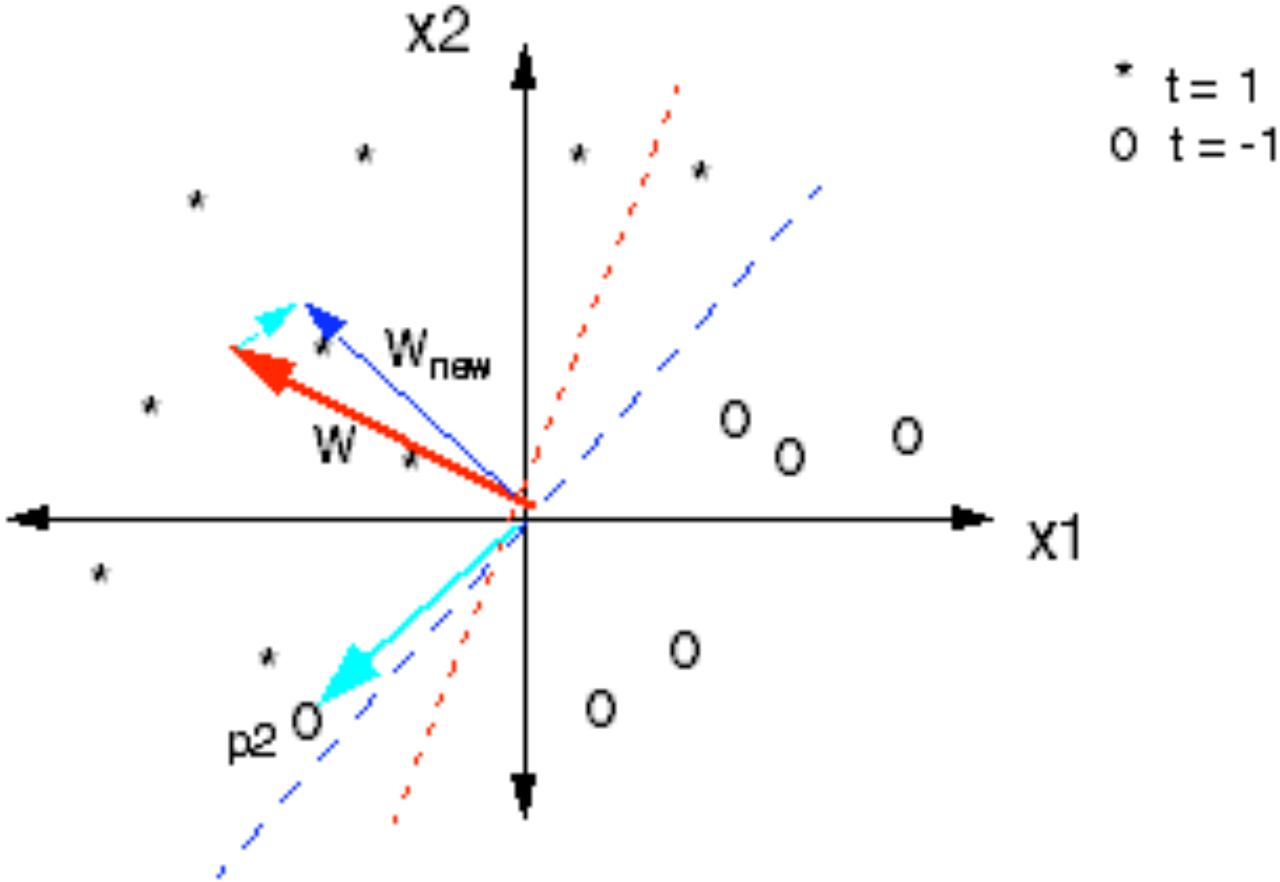
  endfor

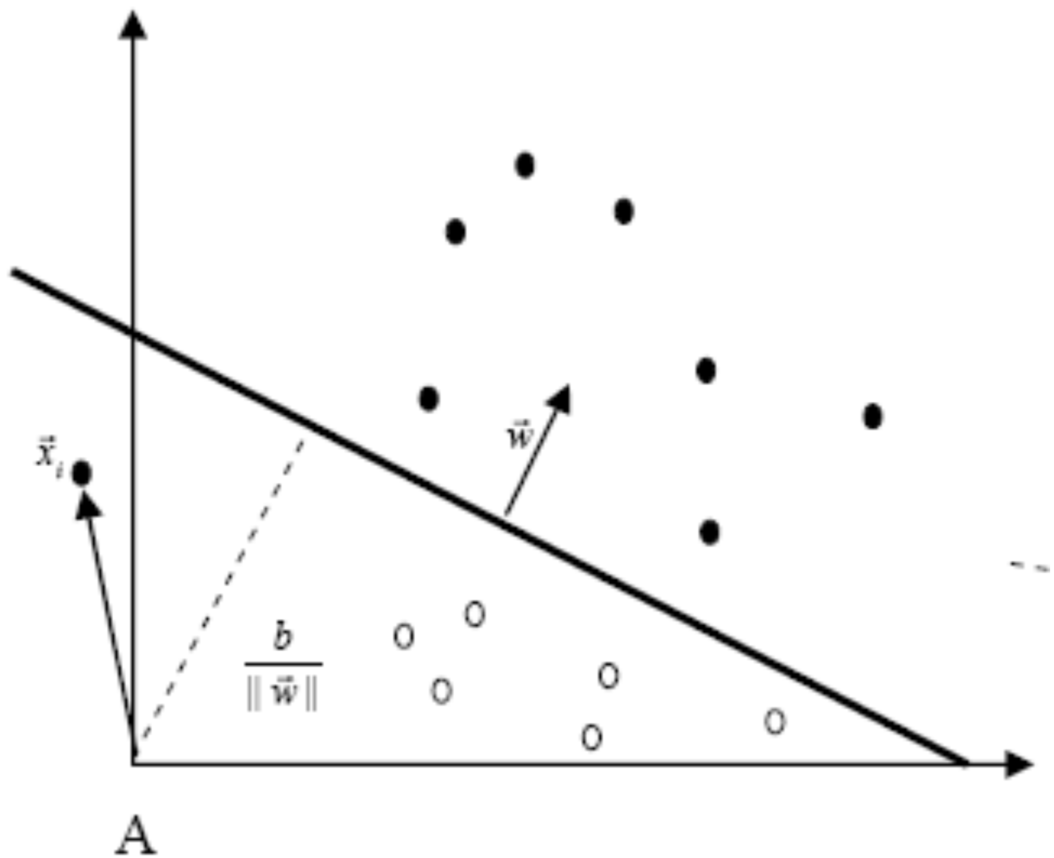
until no error is found

return k, $(\vec{w}_k, b_k)$

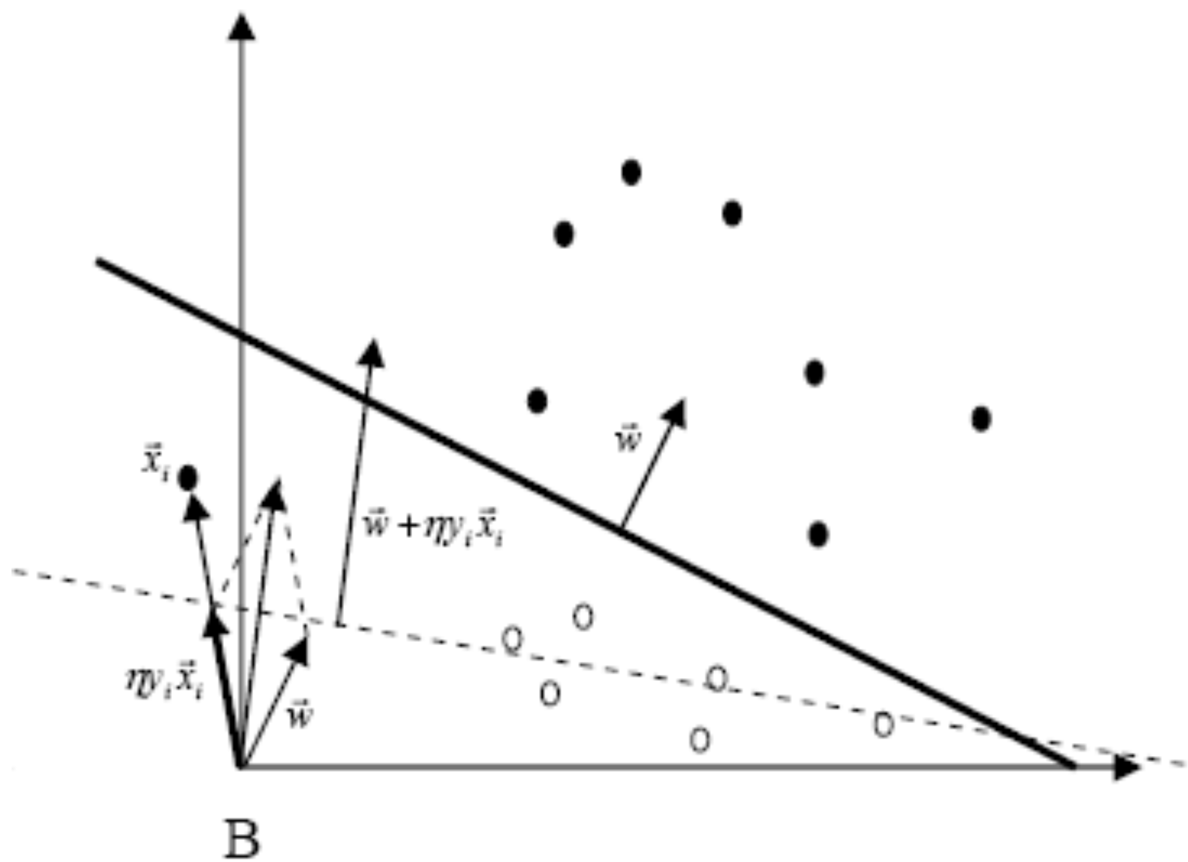

x2
x1
w

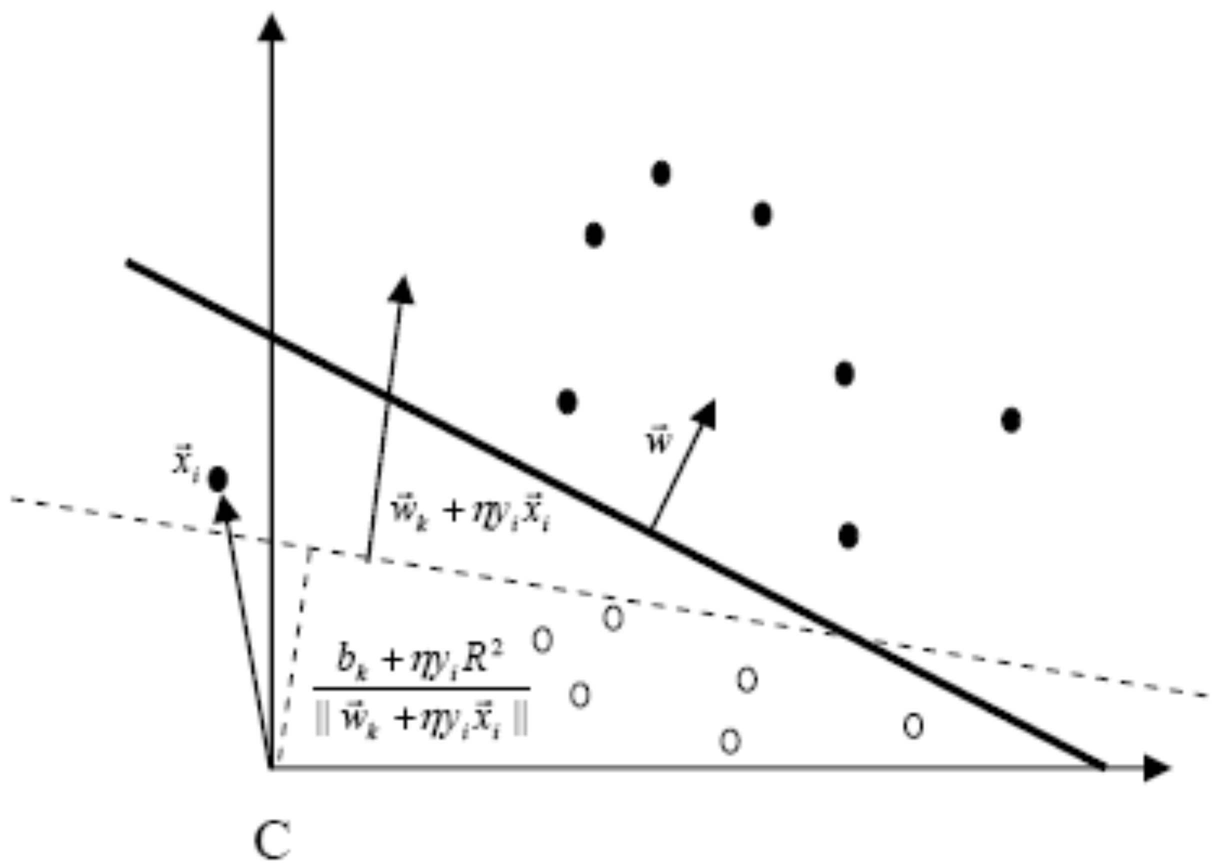

x2
x1
p1
W_{new}
W
* t = 1
o t = -1


x2
x1
W
Wnew
p2
* t = 1
o t = -1



$\vec{x}_i$
$\vec{w}$
$\frac{b}{\|\vec{w}\|}$
A


$\vec{x}_i$
$\vec{w}$
$\vec{w} + \eta y_i \vec{x}_i$
$\eta y_i \vec{x}_i$
$\vec{w}$
B



$\vec{x}_i$
$\vec{w}$
$\vec{w}_k + \eta y_i \vec{x}_i$
$\dfrac{b_k + \eta y_i R^2}{\| \vec{w}_k + \eta y_i \vec{x}_i \|}$
C

Novikoff's Theorem

Let S be a non-trivial training-set and let

$$R = \max_{i=1,..,m} \| x_i \| .$$

Let us suppose there is a vector $\mathbf{w}^*, \| \mathbf{w}^* \| = 1$ and

$$y_i(\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*) \geq \gamma, \quad i = 1,..,m,$$

with $\gamma > 0$. Then the maximum number of errors of the perceptron is:

$$t^* = \left(\frac{2R}{\gamma} \right)^2,$$

Observations

- The theorem states that independently of the margin size, if data is linearly separable the perceptron algorithm finds the solution in a finite amount of steps.
- This number is inversely proportional to the square of the margin.
- The bound is invariant with respect to the scale of the *patterns* (i.e. only the relative distances count).
- The learning rate is not essential for the convergence.

Dual Representation

- The decision function can be rewritten as:

$$h(x) = \mathrm{sgn}(\vec{w} \cdot \vec{x} + b) = \mathrm{sgn}\Big(\sum_{j=1..m} \alpha_j y_j \vec{x}_j \cdot \vec{x} + b\Big) =$$

$$\mathrm{sgn}\Big(\sum_{i=1..m} \alpha_j y_j \vec{x}_j \cdot \vec{x} + b\Big)$$

- as well as the updating function

$$\text{if } y_i\Big(\sum_{j=1..m} \alpha_j y_j \vec{x}_j \cdot \vec{x}_i + b\Big) \leq 0 \text{ then } \alpha_i = \alpha_i + \eta$$

- The learning rate η only affects the re-scaling of the hyperplane, it does not affect the algorithm, so we can fix $\eta = 1$.

First properties of SVMs

- DUALITY is the first feature of Support Vector Machines
- SVMs are learning machines using the following function:

$$f(x) = \mathrm{sgn}(\vec{w} \cdot \vec{x} + b) = \mathrm{sgn}\Big(\sum_{j=1..m} \alpha_j y_j \vec{x}_j \cdot \vec{x} + b\Big)$$

- Note that data appears only as scalar product (for both testing and learning phases)
- The Matrix $G = \left(\vec{x}_i \cdot \vec{x}_j\right)_{i,j=1}^{m}$ is called Gram matrix

Limits of Linear Classifiers

- Data must be linearly separable
- Noise (almost all classifier types)
- Data must be in vectorial format

Solutions

- Multi-Layers Neural Network: back-propagation learning algorithm.
- SVMs: kernel methods.

  The learning algorithm is decoupled by the application domain which is encoded by a kernel function