Predictive Learning: Neural Networks & Support Vector Machines

João Gama
jgama@fep.up.pt

BANCO DE PORTUGAL
EUROSISTEMA

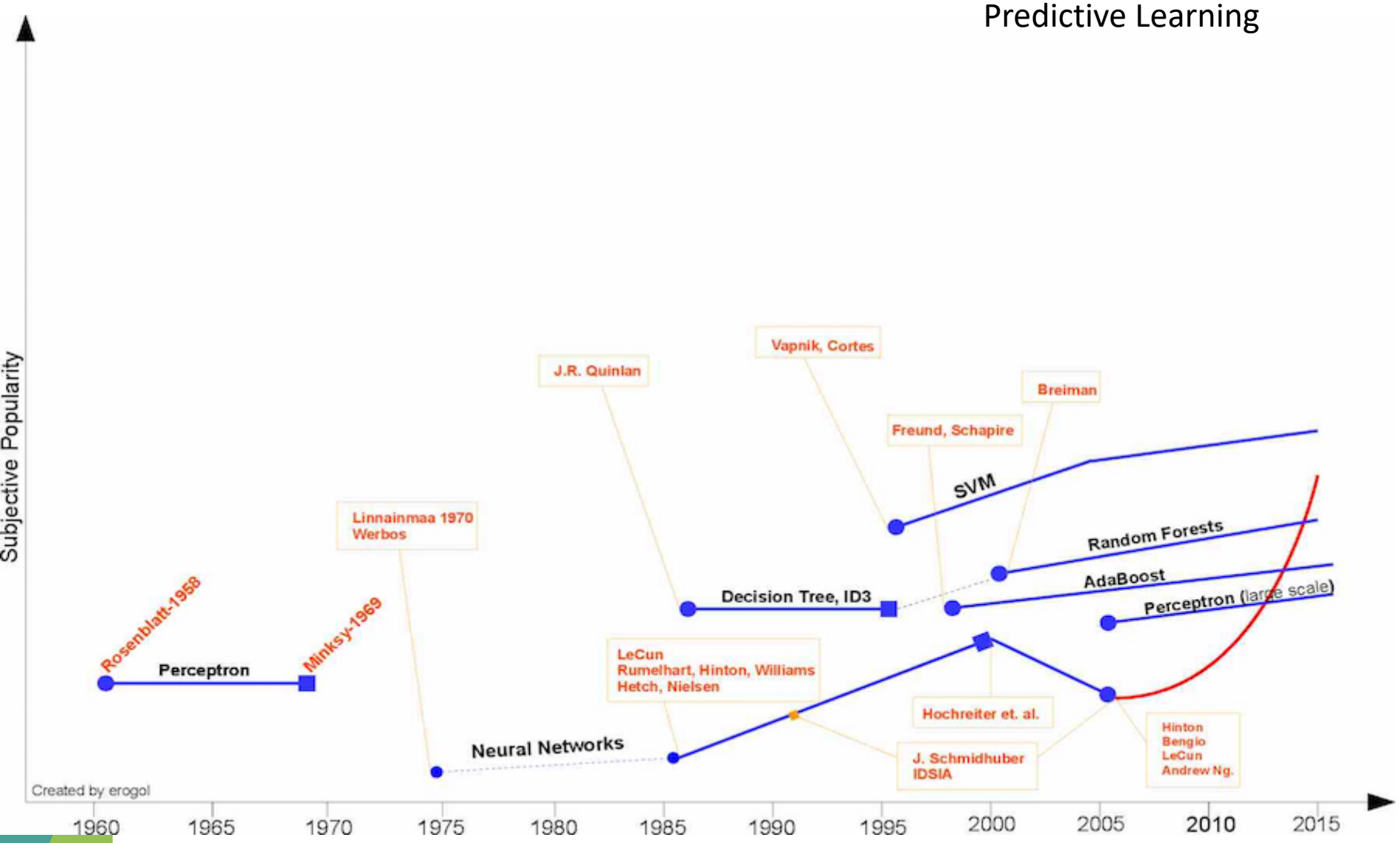
ACADEMIA
BANCO DE PORTUGAL

Big Data and Machine Learning

BANCO DE PORTUGAL
EUROSISTEMA

Developments of Machine Learning

Predictive Learning



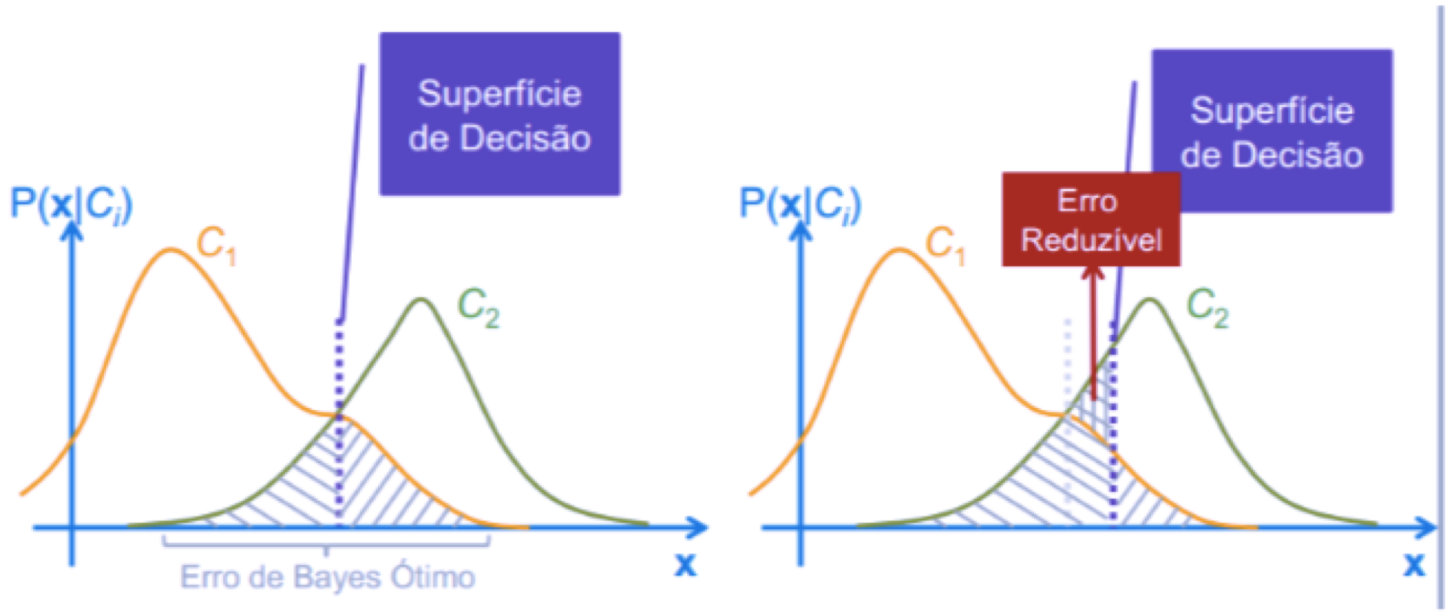
Loss Functions

- Assume we have an unknown function $f(.)$ that label the examples: $y = f(x)$.
  A labeled example is a pair of the form (x_i, y_i);
- Using a set of labeled examples, we learn an approximation function $\hat{f}(x)$.
- A loss function measures the goodness of that approximation.
  How to quantify the goodness of fit?

- Assume an observation x.
  The true label (unknown) is $y = f(x)$.
- The learned function assigns the label $\hat{y} = \hat{f}(x)$.
- Dependent on the domain of y, we use different loss functions to quantify the matching between y and $\hat{y}$.
  For classification problems, $y \in \{y_1, \ldots, y_k\}$, for regression problems $y \in \Re$.

- Classification: 0-1 loss function:
  $loss(y, \hat{y}) = 1$ iif $y \neq \hat{y}$; 0 iif $y = \hat{y}$
- Regression:
  - Squared error: $loss(y, \hat{y}) = (y - \hat{y})^2$
  - Absolute error: $loss(y, \hat{y}) = |y - \hat{y}|$

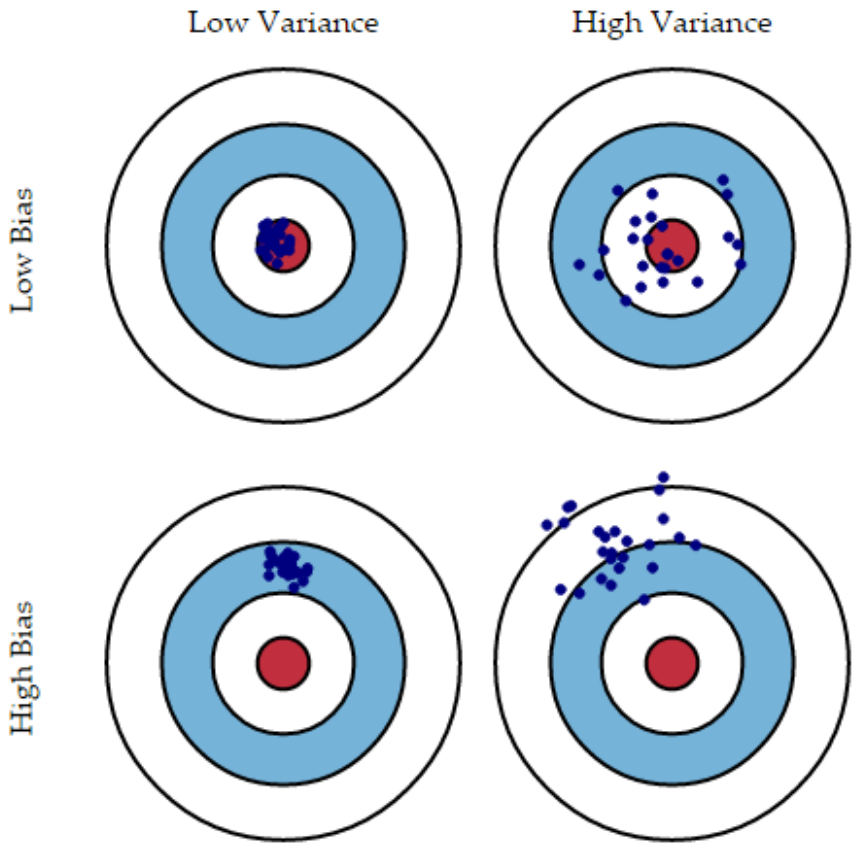
Optimal Bayes-Error



BANCO DE PORTUGAL
EUROSISTEMA

Bias & Variance Analysis

$$E[(t-y)^2] = (t-\overline{y})^2 + E[(\overline{y}-y)^2]$$
$$\text{Exp. loss} = \text{Bias} + \text{Variance}$$

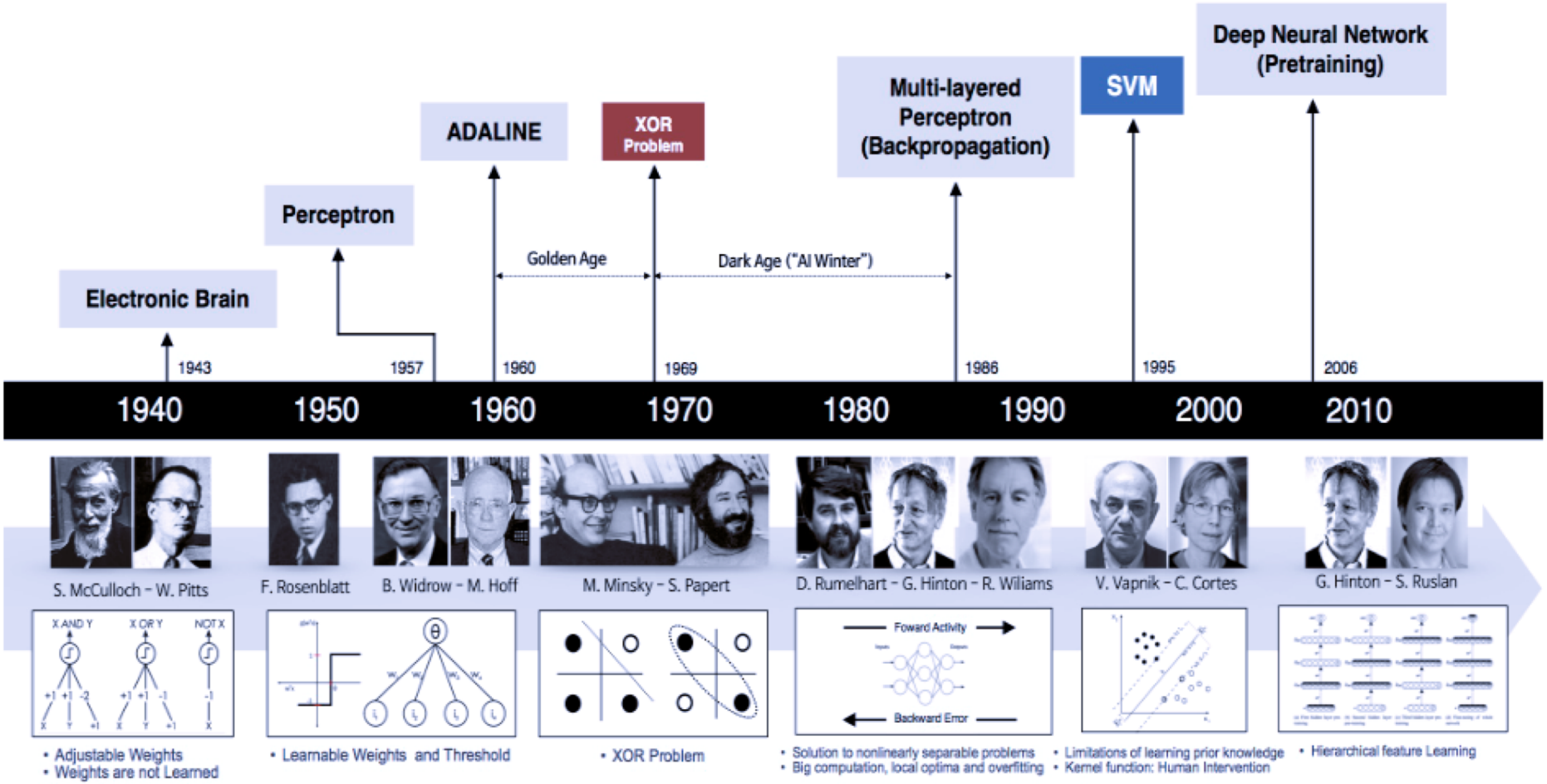


Neural Networks

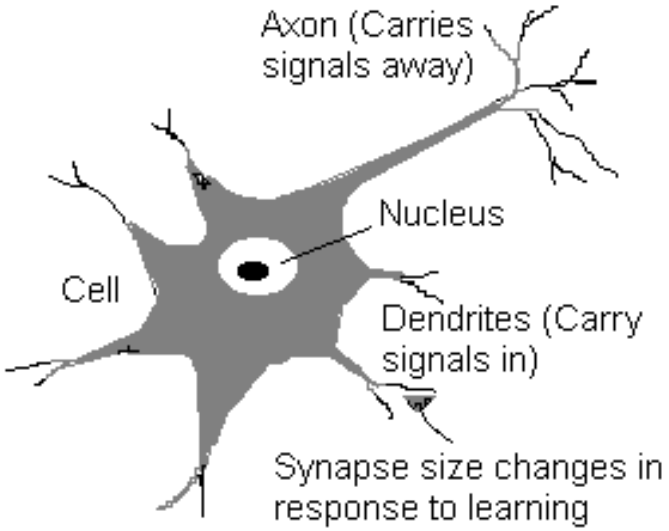
João Gama
jgama@fep.up.pt

Developments of Neural Networks

Milestones in the Development of Neural Networks

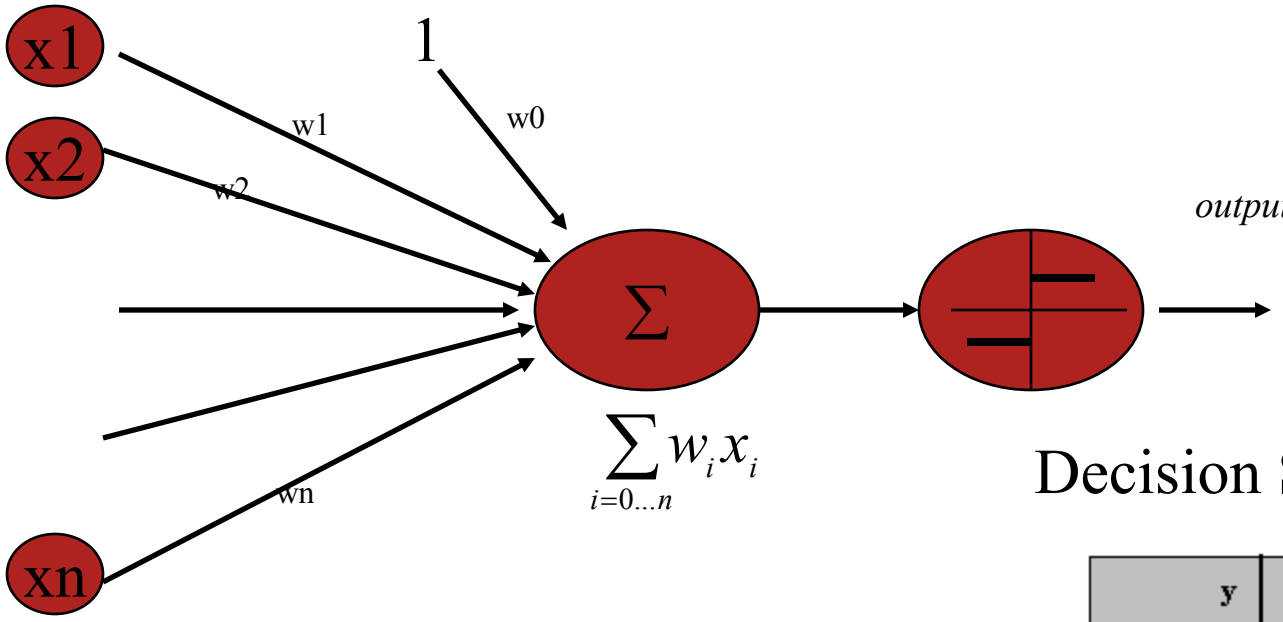


- A neuron: many-inputs / one-output unit output can be *excited* or *not excited*
- incoming signals from other neurons determine if the neuron shall excite ("fire")
- Output subject to attenuation in the *synapses,* which are junction parts of the neuron

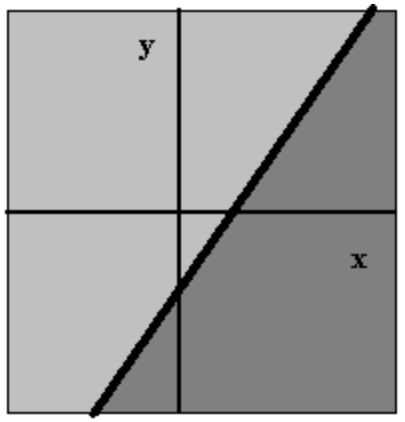


BANCO DE PORTUGAL
EUROSISTEMA

Linear Machine: $w_o+w_1x_1+w_2x_2+\ldots+w_nx_n$



Decision Surface:

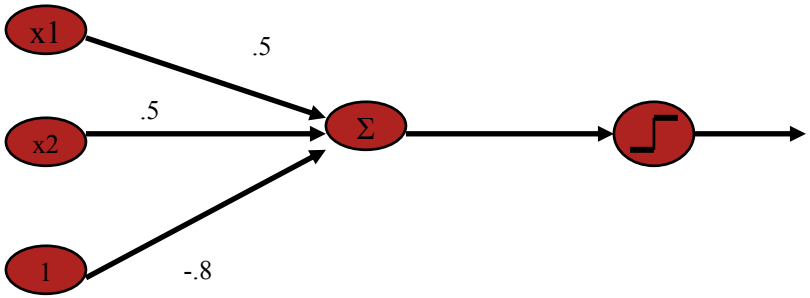


Vectorial Representation:

$$o(\vec{x}) = sig(\vec{w}.\vec{x})$$

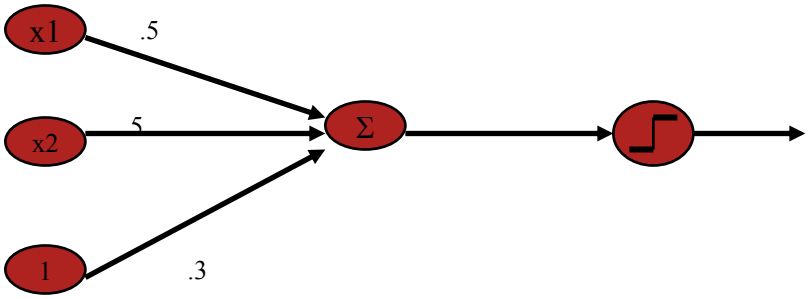
AND

Representation of Boolean functions



	A	B	C	D	E	F	G	H	I
1	**x1**	**x2**	**y**		**w0**	**w1**	**w2**		
2	-1	-1	-1		-0,8	0,5	0,5		-1,8
3	-1	1	-1						-1,8
4	1	-1	-1						-0,8
5	1	1	1						0,2
6									
7									
8	**x1**	**x2**	**y**		**w0**	**w1**	**w2**		
9	-1	-1	-1		0,3	0,5	0,5		-0,7
10	-1	1	1						0,3
11	1	-1	1						0,3
12	1	1	1						1,3

OR



How to learn w_i?

The idea behind the Algorithm:

1. **Initialize the weights with a small random number**
2. **For each example x**
   1. Compute the result of the linear machine: **sign(x . w)**
   2. If the result is correct, go to next example
   3. If predict is 1 and the observed value is 0 (false positive)
      1. Update the weights by subtracting a delta
   4. If predict is 0 and the observed value is 1 (false negative)
      1. Update the weights by adding a delta
3. **If any example has been misclassified go to step 1**
4. **Otherwise, return the current value of the weights**

– **Updating the weights (delta rule):**

$$w_i(t+1) = w_i(t) + \eta(Observed - \Pr edicted)x_i$$

$$where\ \eta \text{ is the learning rate}$$

Assume a linear machine:

$w_0+w_1*x_1+w_2*x_2+...+w_n*x_n$

The goal is to learn the coefficients w_i that minimize the square error:

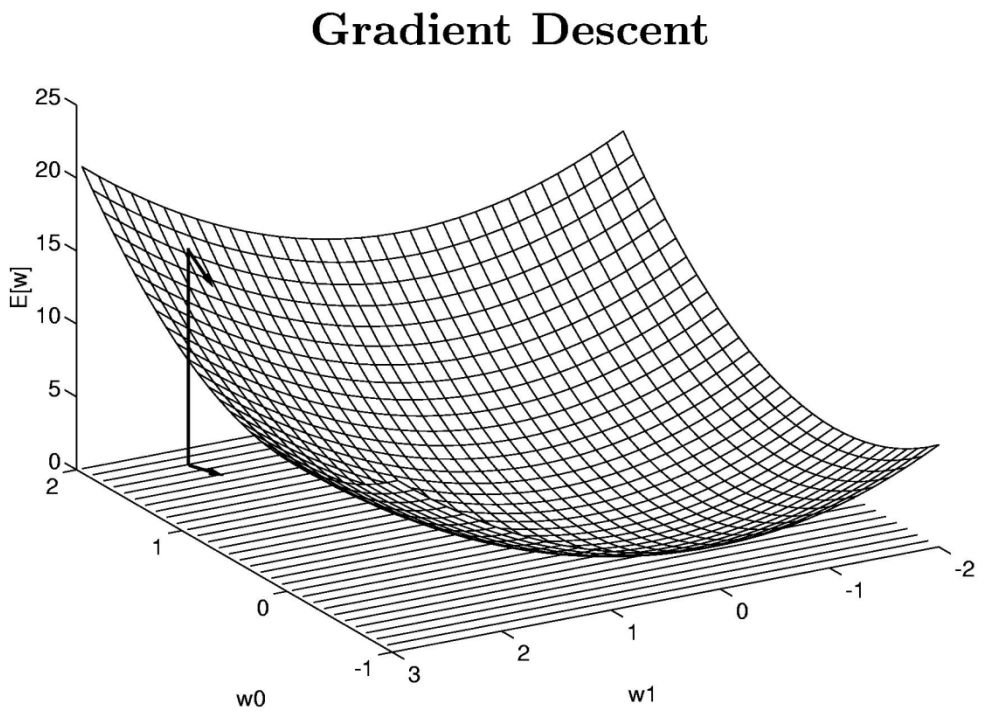
$$E(w_i) = \frac{1}{2} \Sigma(t_d-p_d)^2$$

Sum of the square of the differences bwtween the observed value (t_d) and the predict value (p_d)

Basic idea:

Changing the coefficients to reduce the error following the downward direction of the gradient.

- Error surface in the parameters space:
  - The square error defines a parabolic surface.



BANCO DE PORTUGAL
EUROSISTEMA

Deriving the Delta Rule

The partial derivative of E(w) (with respect to each coefficient) can be computed as:

Gradient:

$$\nabla E(\vec{w}) = \left[\frac{\partial E}{\partial w_0}, \ldots, \frac{\partial E}{\partial w_n} \right]$$

Learning Rule:

$$\vec{w} \leftarrow \vec{w} + \Delta \vec{w}$$

$$\Delta \vec{w} = -\eta \nabla E(\vec{w})$$

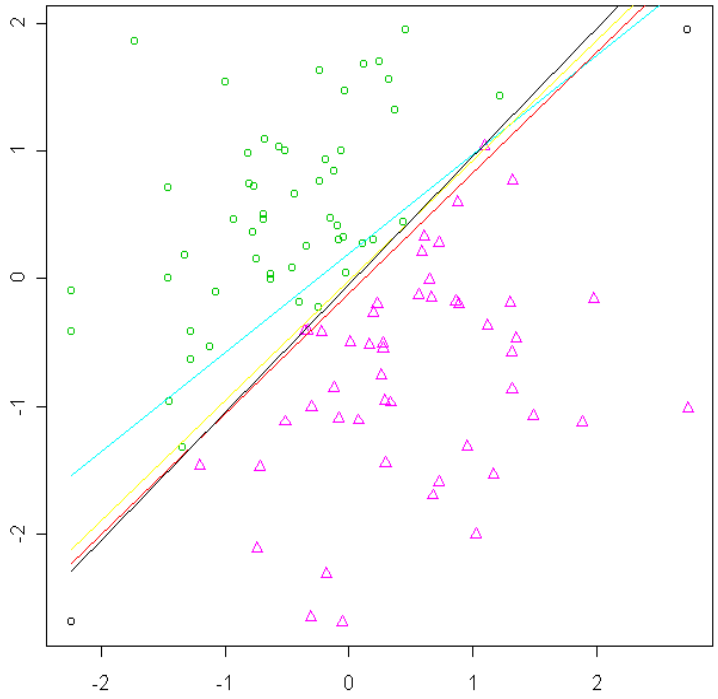
$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - p_d)^2$$

$$\frac{\partial E}{\partial w_i} = \sum (t_d - p_d)(-x_{id})$$

$$\Delta w_i = \eta \sum_{d \in D} (t_d - p_d) x_{id}$$

- **Does not assume any distribution for the data**
- **Converge to a solution if the samples are linearly separable.**
- **The figure illustrates the convergence of a linear machine.**
- **A linear machine defines decision surfaces that are hyperplanes.**
  - It is able to represent AND, OR, and other Boolean functions.
  - It is not capable of representing XOR

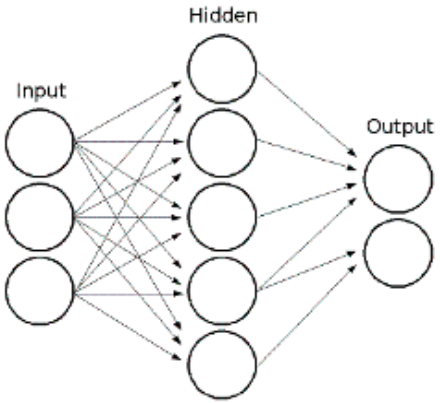


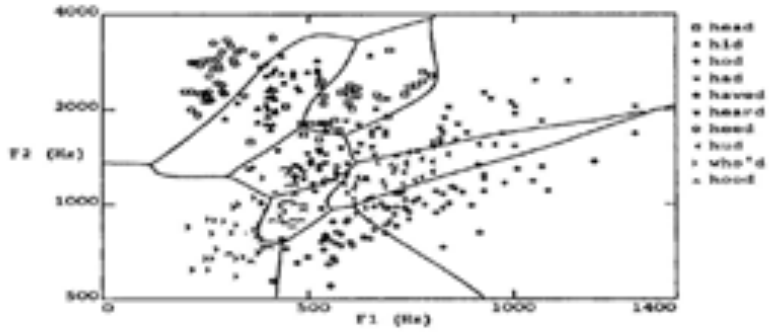
Multilayer Perceptrons

BANCO DE PORTUGAL
EUROSISTEMA

Multi-layer perceptrons

- **Linear machines (as well as discriminant functions) only define linear decision surfaces.**
- **Structuring linear machines in layers is possible to define non-linear decision surfaces**
  - The combination of the linear units is also linear
  - Non-linear unit : the sigmoid function.

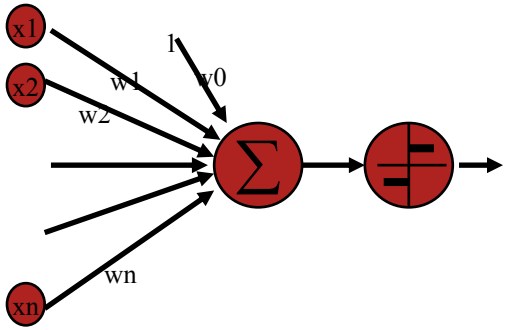




BANCO DE PORTUGAL
EUROSISTEMA

The sigmoid function

$$o(x) = \begin{cases} 1 \text{ sse } x > 0 \\ -1 \text{ sse } x < 0 \end{cases}$$

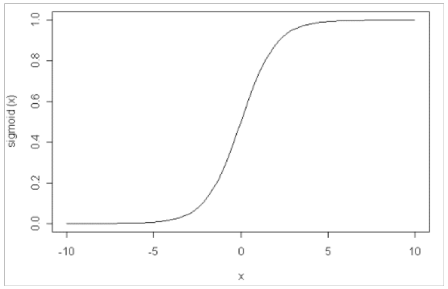


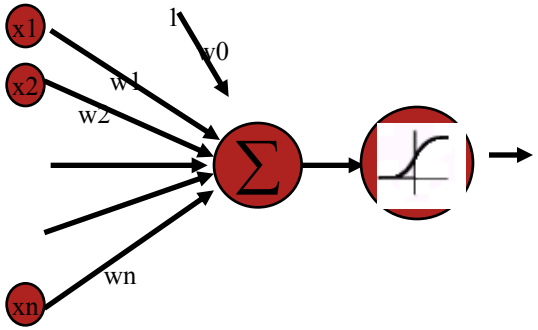
The sigmoid function:

$$o(x) = \frac{1}{1+e^{-x}}$$

$$x \in \Re$$

$$o(x) \in [0;1]$$

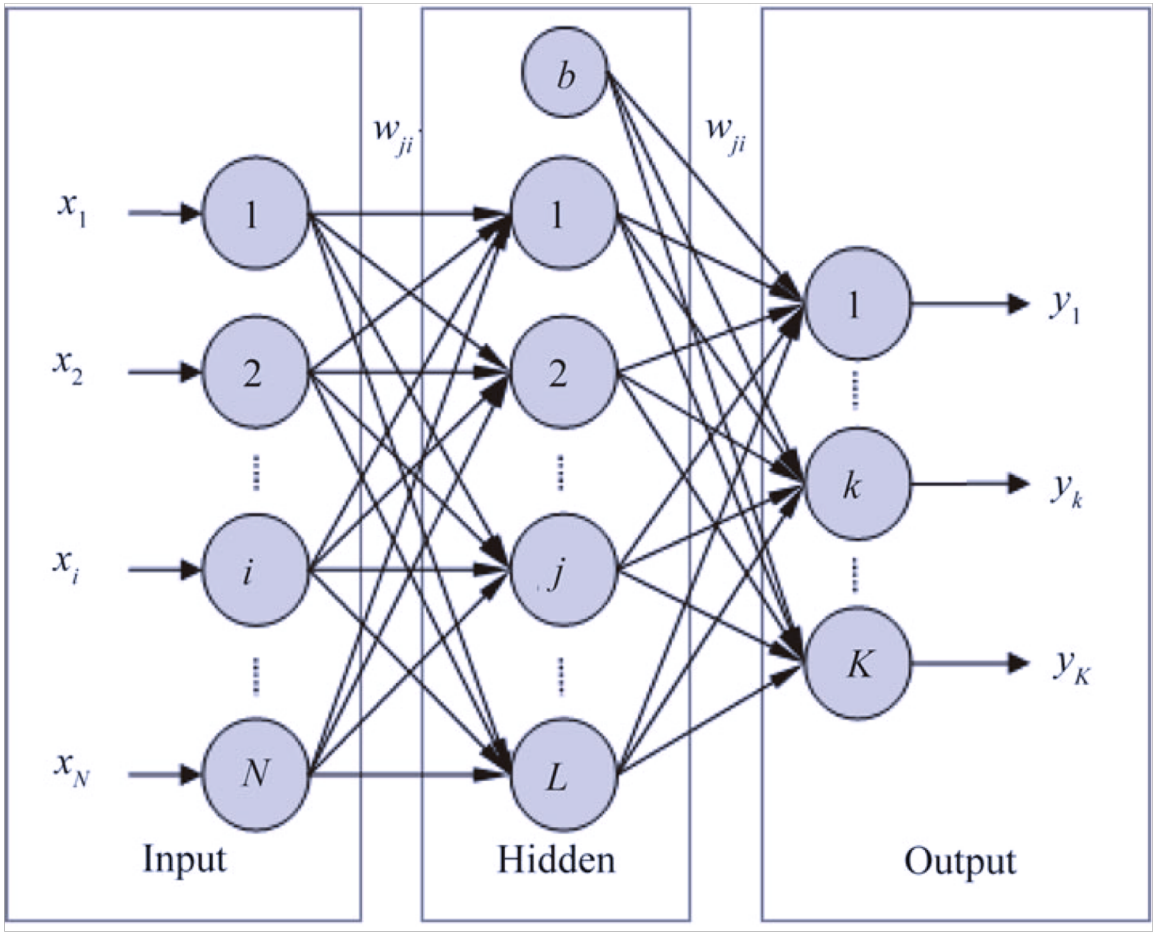




Interesting property:

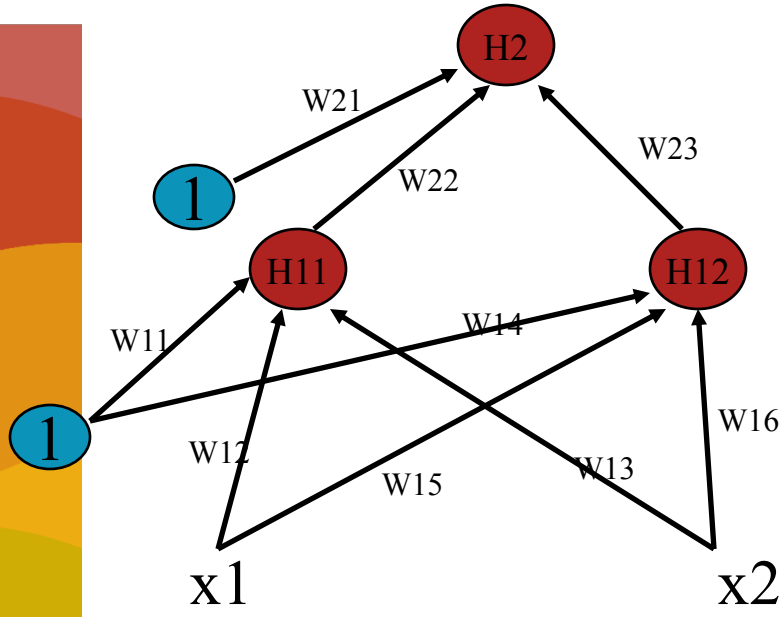
$$\frac{\partial o(x)}{\partial x} = o(x)(1 - o(x))$$

Adaptive interaction between individual neurons
Power: collective behavior of interconnected neurons

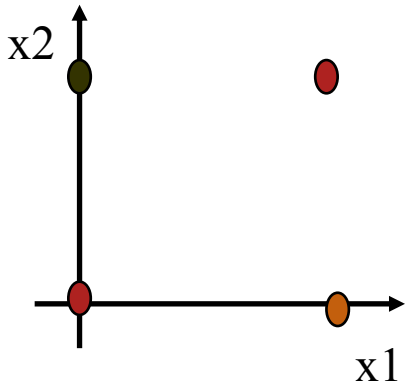


BANCO DE PORTUGAL
EUROSISTEMA

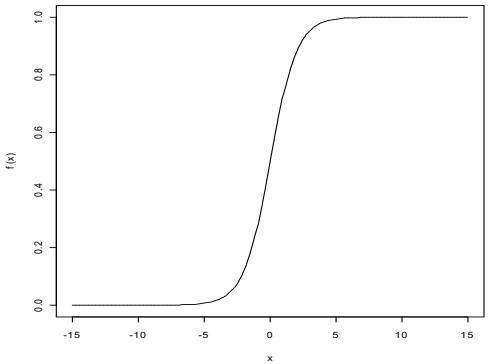
Solving XOR

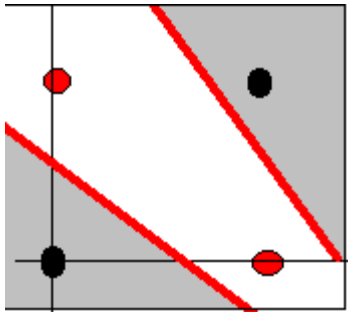


x1	x2	Y
0	0	0
0	+1	+1
+1	0	+1
+1	+1	0



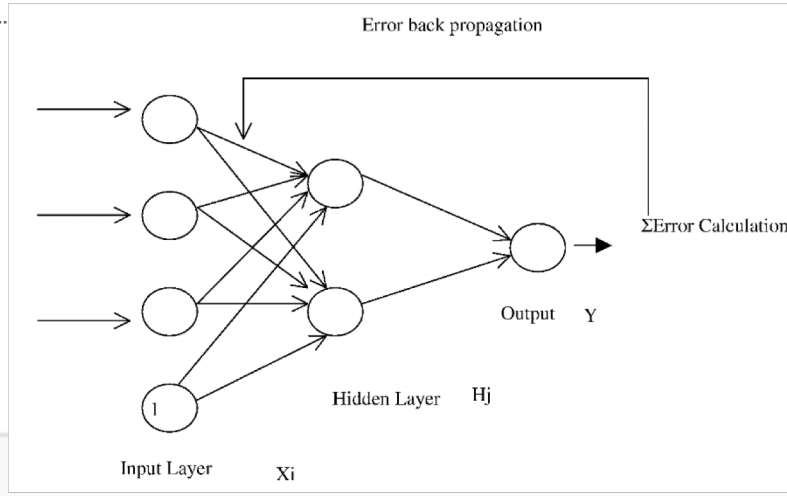
$$sigmoide(x) = \frac{1}{1+\exp(-x)}$$





BANCO DE PORTUGAL
EUROSISTEMA

The Backpropagation Algorithm



Algorithm for a MLP with 3 layers

Actual algorithm for a 3-layer network (only one hidden layer):

```
initialize network weights (often small random values)
do
   forEach training example ex
      prediction = neural-net-output(network, ex)  // forward pass
      actual = teacher-output(ex)
      compute error (prediction - actual) at the output units
      compute Δw_h for all weights from hidden layer to output layer  // backward pass
      compute Δw_i for all weights from input layer to hidden layer   // backward pass continued
      update network weights
until all examples classified correctly or stopping criterion satisfied
return the network
```

BANCO DE PORTUGAL
EUROSISTEMA

The Backpropagation Algorithm

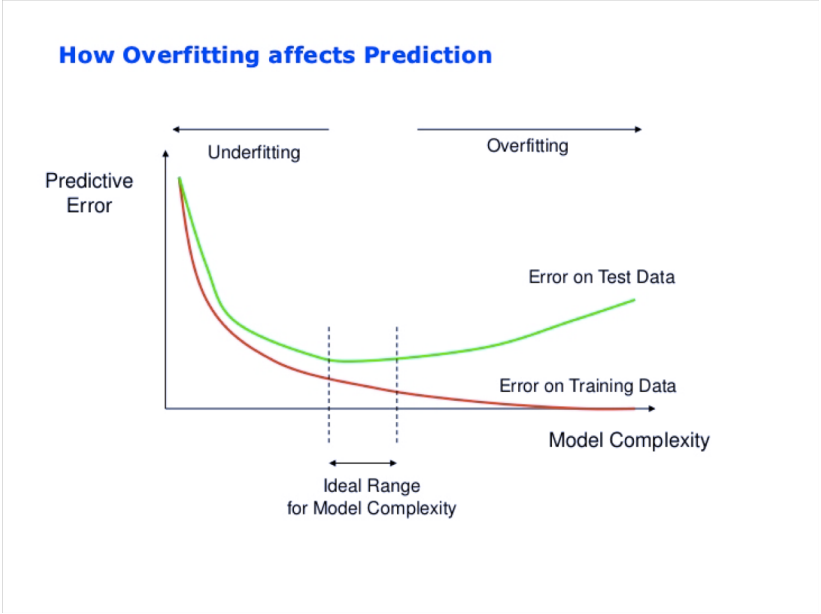
The backpropagation algorithm implements a search using gradient descent in the space defined by the weights of the network.

- In multi-layer networks of the error surface may contain several local minima.
- Not guarantee to find a global minimum.
- In practice, often works well (can run multiple times)
- Incremental (stochastic) version: correction of weights after seeing an example
- More effective to escape from local minima
- Minimizes error over training examples
- Will it generalize well to subsequent examples?
- Training can take thousands of iterations !
- slow!
- Using network after training is very fast

Overfiting

When should we finish training the network?

- Stopping Criteria:
  - If stopping too early we run the risk of getting a network not yet trained.
  - If stopping too late: danger of overfiting (adjustment to noise in the data)
- Usual criteria:
  - Based on the error in the training set
    - When the error in the training set is below a certain limit.
  - Error based on a evaluation set (independent from the training set)
    - When the error on the validation set has reached a minimum.



The definition of the network topology can be problematic

The number of nodes in the hidden layer

Few nodes: underfitting

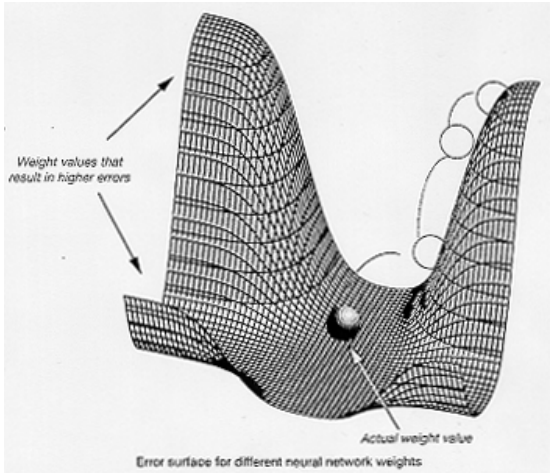
Many nodes: overfitting

There are no criteria for defining the number of nodes in the hidden layer

Effect of learning rate

A learning rate

- Little has the effect of learning times higher
- High may lead to non-convergence.



Error surface for different neural network weights

BANCO DE PORTUGAL
EUROSISTEMA

When to Consider Neural Networks

Use when:

Input is high-dimensional discrete or real-valued (e.g. raw sensor input)

Output is discrete or real valued

Output is a vector of values

Possibly noisy data

Form of target function is unknown

Human readability of result is unimportant

Examples:

Speech phoneme recognition [Waibel]

Image classification [Kanade, Baluja, Rowley]

Financial prediction

BANCO DE PORTUGAL
EUROSISTEMA

Generalization vs. specialization

Optimal number of hidden neurons

Too many hidden neurons: you get an over fit, training set is memorized, thus making the network useless on new data sets

Not enough hidden neurons:
network is unable to learn problem concept

Overtraining:

Too much examples, the ANN memorizes the examples instead of the general idea

Generalization vs. specialization trade-off:

 # hidden nodes & training samples

Expressive Capabilities

Representation

The multilayer networks (MLP) can approach any function:

Boolean Functions

- Any Boolean function can be represented by a network with one hidden layer

Continuous functions

- Any function can be approximated remains limited (with an arbitrarily small error) for a network with one hidden layer (using sigmoid) and a drive (not run) output.

Arbitrary functions

- Any function can be approximated (with an arbitrarily small error) by a network with two hidden layers.

Capacity

the amount of information that can be stored in the network

The capacity of a neural network is *dense*.

ability to learn any function given enough data

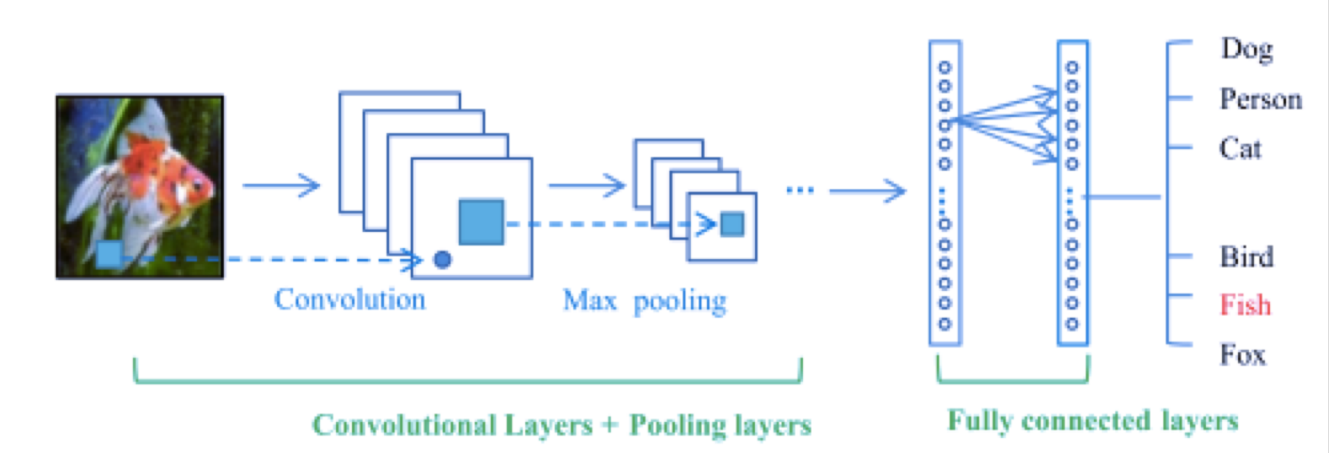
Deep Learning

Aims to discover multiple levels of distributed representations. It relies on hierarchical architectures to learn high-level abstractions in data

Raw Input

Feature Extraction Construction

Classifier



General CNN architecture (Guo et al., 2016)

Support Vector Machines

João Gama
jgama@fep.up.pt
November 2017

(based on Andrew Moore slides)

Introduction to Support Vector Machines (SVM)

Linear SVM

Linear separable data

Non-linear separable data

Non-Linear SVM

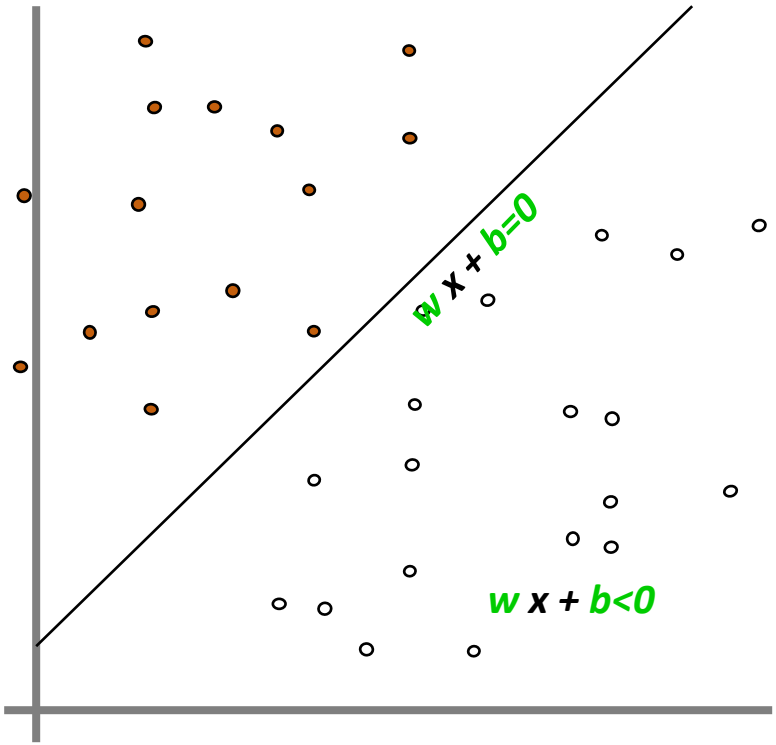
Kernels

Properties of SVM

Linear Classifiers

- denotes +1
- denotes -1

$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}\,\mathbf{x} + b)$

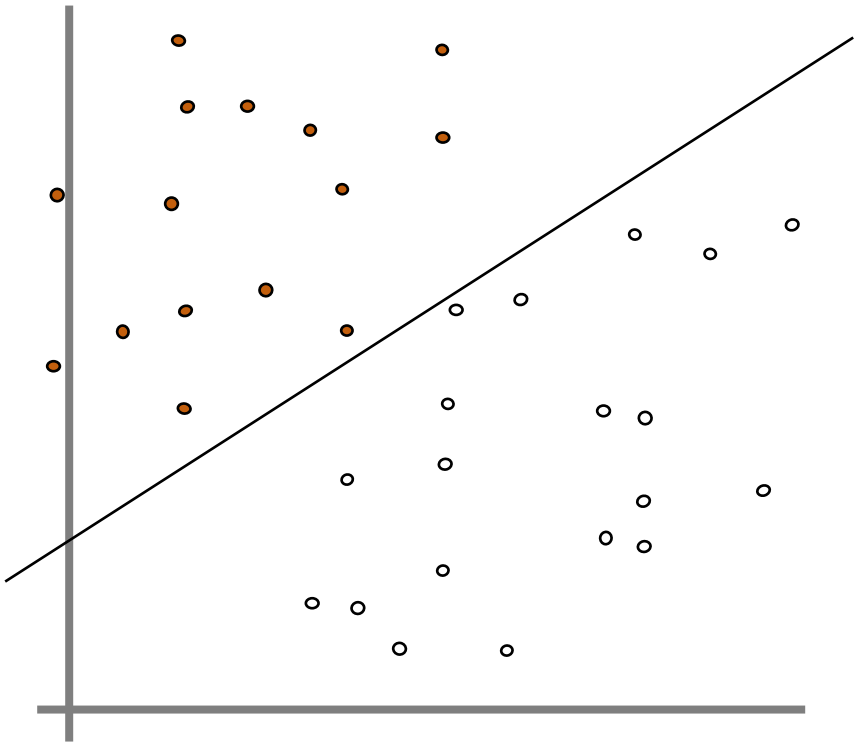


Linear Classifiers

$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}\,\mathbf{x} + b)$

denotes +1

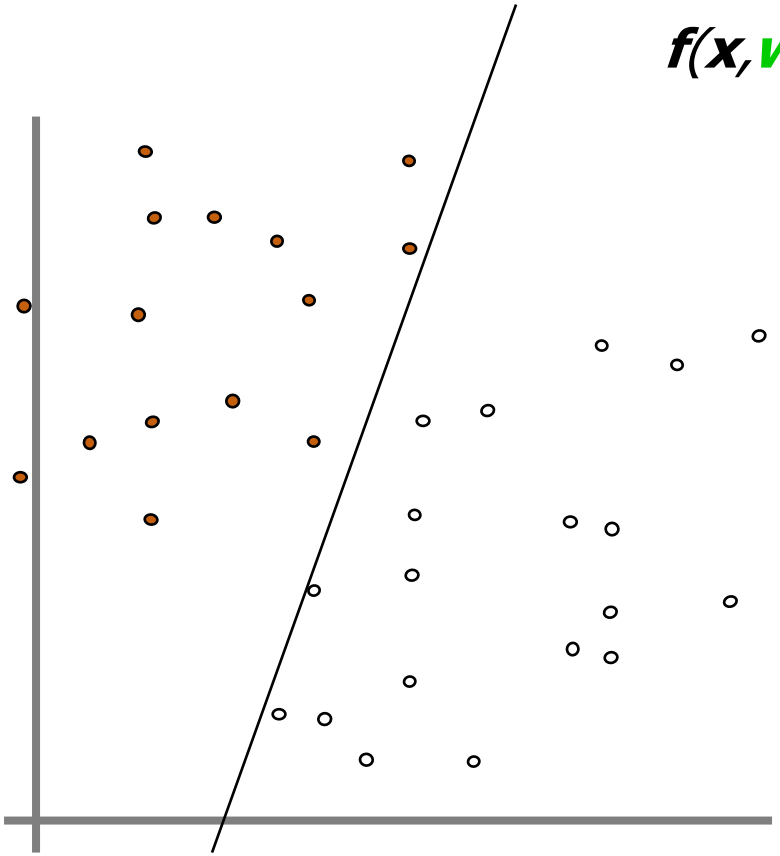
denotes -1



Linear Classifiers

denotes +1

denotes -1



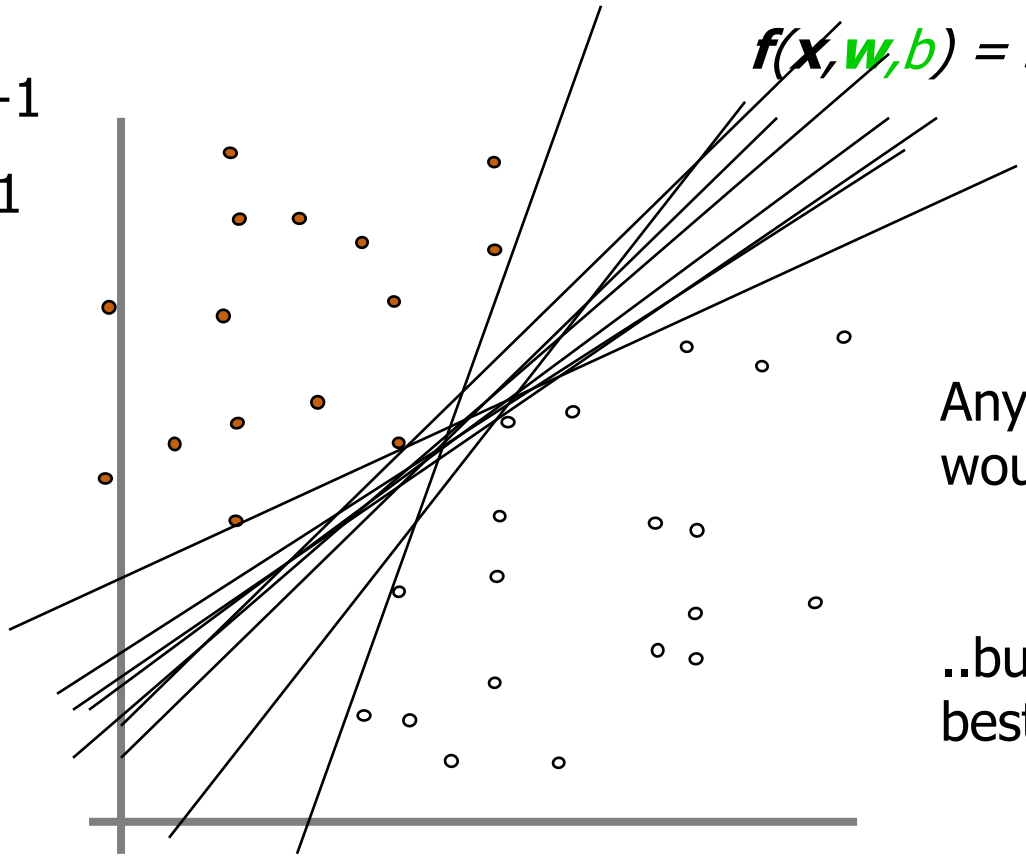
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$

Linear Classifiers

$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}\,\mathbf{x} + b)$

denotes +1

denotes -1



Any of these would be fine..

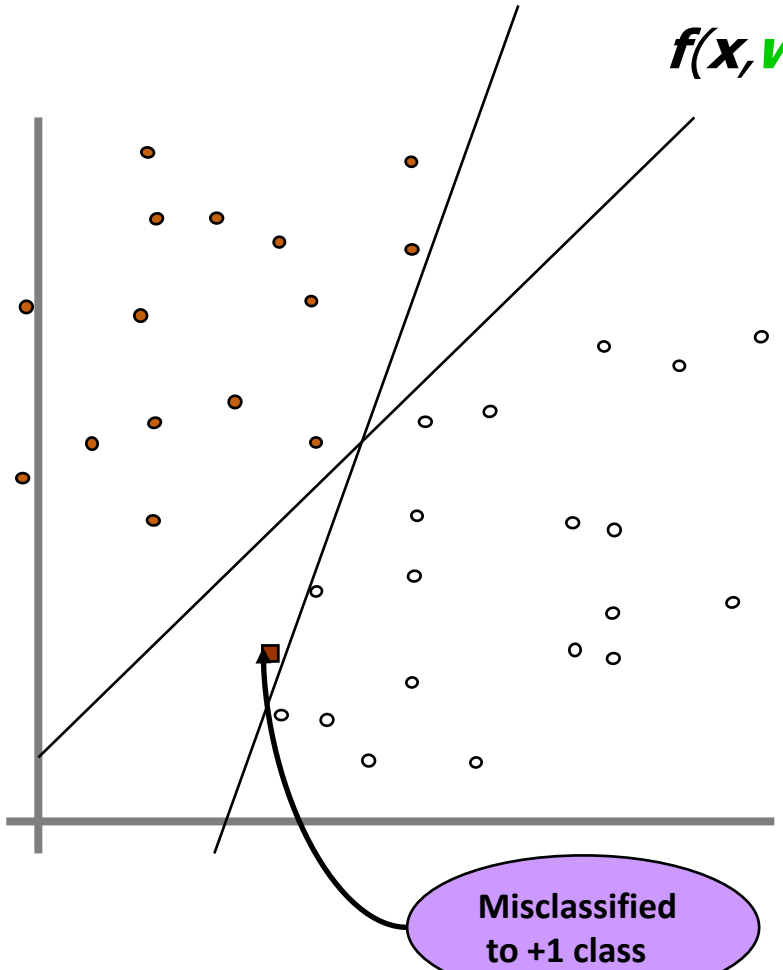
..but which is best?

Linear Classifiers

$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$

denotes +1

denotes -1



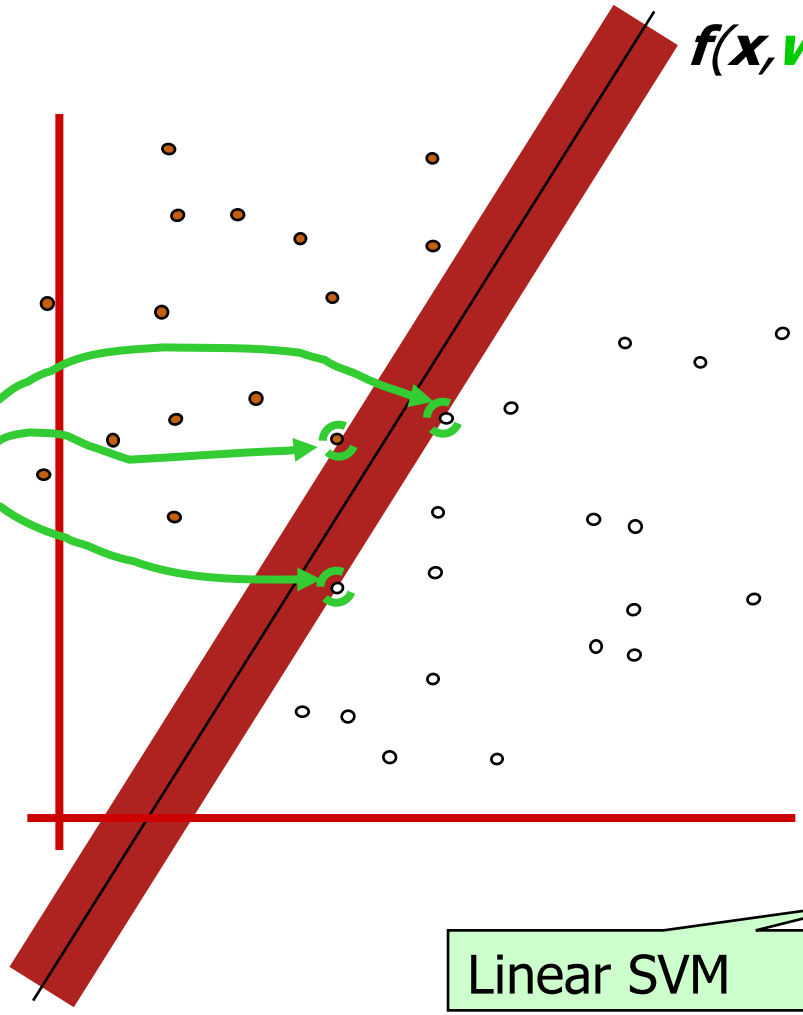
How would you classify this data?

Maximum Margin

$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}\,\mathbf{x} + b)$

denotes +1

denotes -1



Support Vectors are those datapoints that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the maximum margin.

This is the simplest kind of SVM (Called an LSVM)

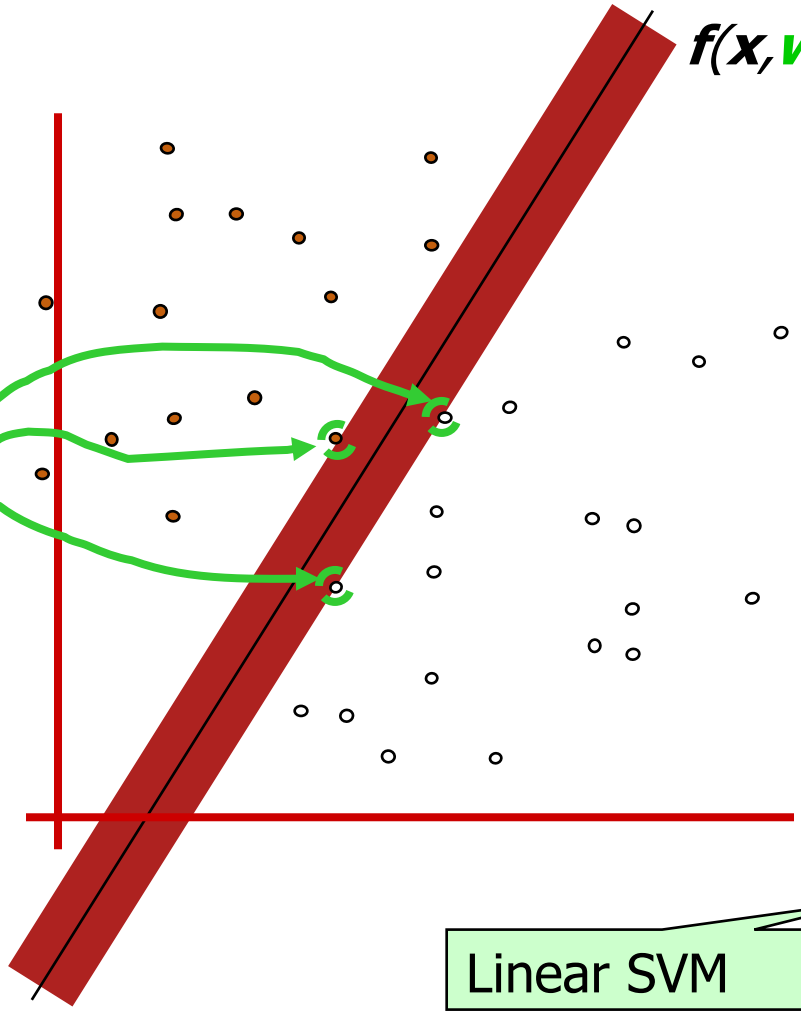
Linear SVM

Maximum Margin

$f(x, w, b) = sign(w \cdot x + b)$

denotes +1

denotes -1



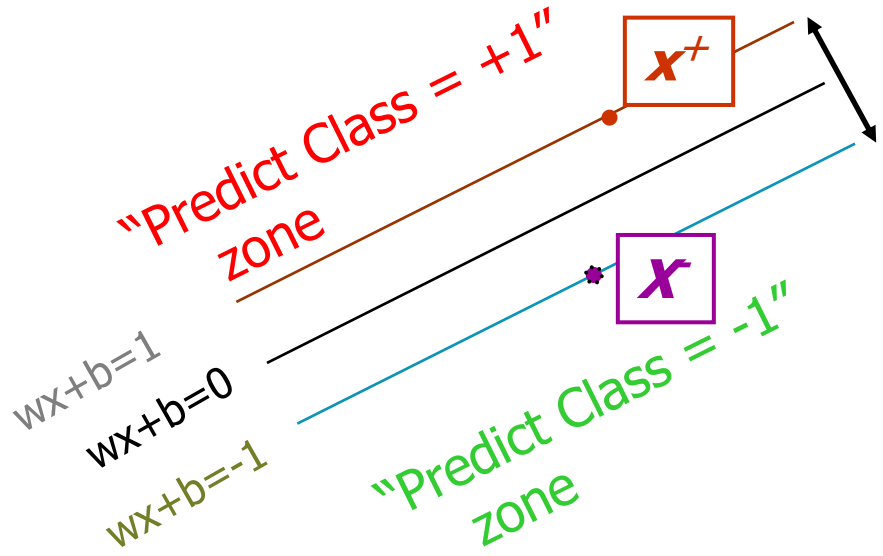
Support Vectors are those datapoints that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the maximum margin.

Only support vectors are important; other training examples are ignorable.

Linear SVM

Linear SVM Mathematically



M=Margin Width

What we know:

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

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

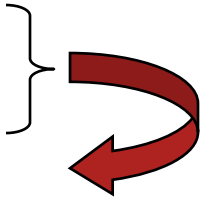
$w \cdot (x^+ - x^-) = 2$

$$M = \frac{(x^+ - x^-) \cdot w}{|w|} = \frac{2}{|w|}$$

Linear SVM Mathematically

- Goal: **1) Correctly classify all training data**

$$wx_i + b \geq 1 \quad \text{if } y_i = +1$$
$$wx_i + b \leq -1 \quad \text{if } y_i = -1$$
$$y_i(wx_i + b) \geq 1 \quad \text{for all i}$$



2) Maximize the Margin $M = \frac{2}{|w|}$

same as minimize $\frac{1}{2} w^t w$

- We can formulate a quadratic optimization problem and solve for w and b

- Minimize $\Phi(w) = \frac{1}{2} w^t w$

subject to $y_i(wx_i + b) \geq 1 \qquad \forall i$

Solving the Optimization Problem

Find $\mathbf{w}$ and b such that
$\Phi(\mathbf{w}) = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- Need to optimize a *quadratic* function subject to *linear* constraints.
  - Quadratic optimization problems are a well-known class of mathematical programming problems, and many algorithms exist for solving them.
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\sigma\alpha_i\alpha_j y_i y_j (\mathbf{x_i}\cdot\mathbf{x_j})$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

- The solution has the form:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w}^T \mathbf{x_k} \text{ for any } \mathbf{x_k} \text{ such that } \alpha_k \neq 0$$

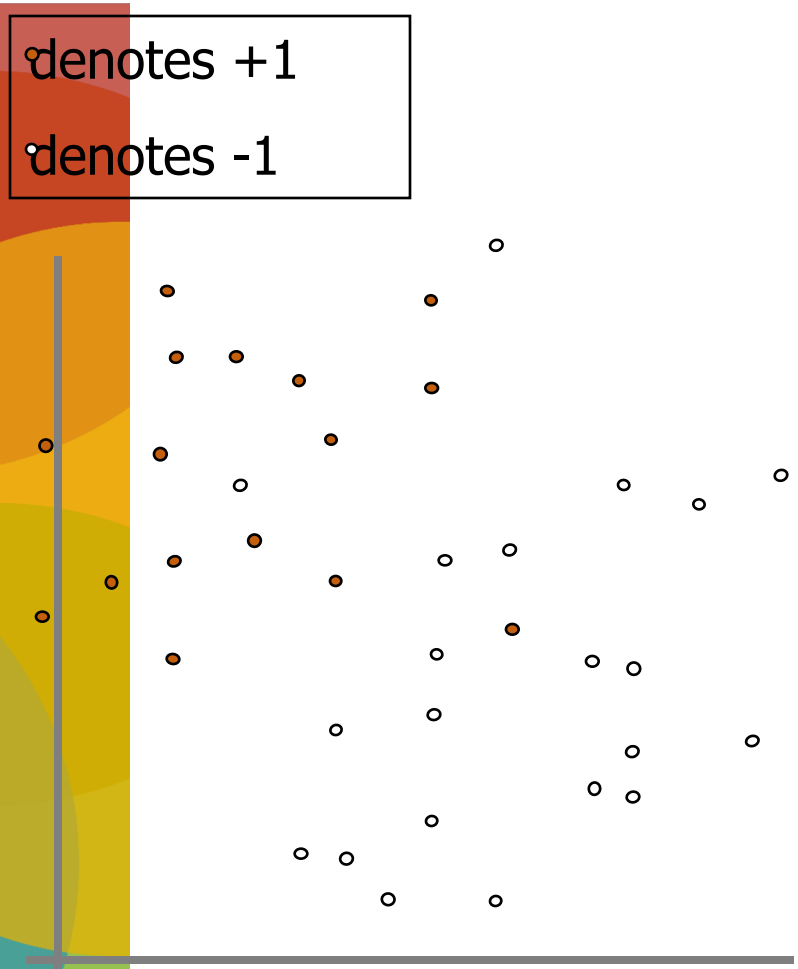
- Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a support vector.
- Then the classifying function will have the form:

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x_i}^T \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point $\mathbf{x}$ and the support vectors $\mathbf{x_i}$ – we will return to this later.
- Solving the optimization problem involves computing the inner products $\mathbf{x_i}^T \mathbf{x_j}$ between all pairs of training points.

BANCO DE PORTUGAL
EUROSISTEMA

Dataset with noise



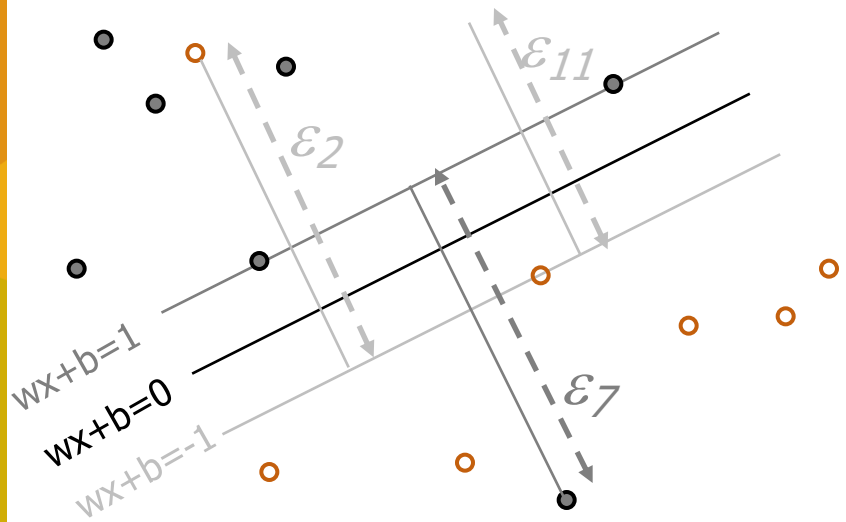
- **Hard Margin: So far we require all data points be classified correctly**

  - No training error

- **What if the training set is noisy?**
- **Solution:**
  - **Allow some errors**

Soft Margin Classification

***Slack variables ξi* can be added to allow misclassification of difficult or noisy examples.**



What should our quadratic optimization criterion be?

Minimize

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$

Hard Margin v.s. Soft Margin

- **The old formulation:**

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

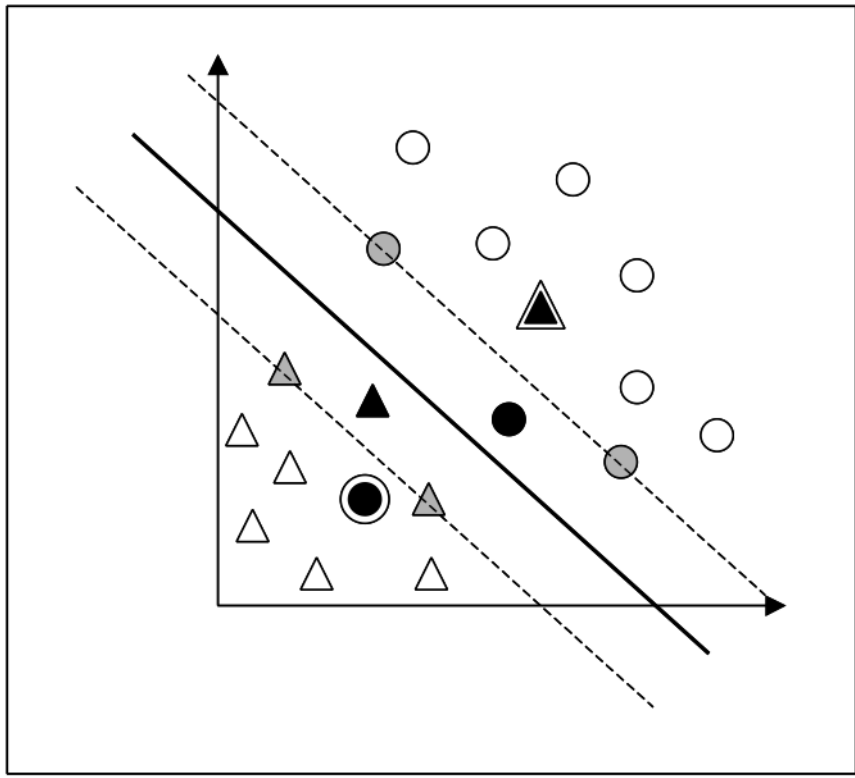
- **The new formulation incorporating slack variables:**

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- **Parameter *C* can be viewed as a way to control overfitting.**

Hard Margin v.s. Soft Margin

- As in hard margin SVM
  - x_i *where* $\boldsymbol{\alpha_i > 0}$ are the support vectors
- The SVs where $\boldsymbol{\alpha_i = C}$
  - If $\xi_i > 1$ are errors
  - If $0 < \xi_i < 1$ are correctly classified but between the margins
- Points where $\alpha_i = 0$ *and* $\xi_i = 0$ are correctly classified and are outside the margins



Linear SVMs: Overview

- The classifier is a *separating hyperplane.*
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points x_i are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution training points appear only inside dot products:

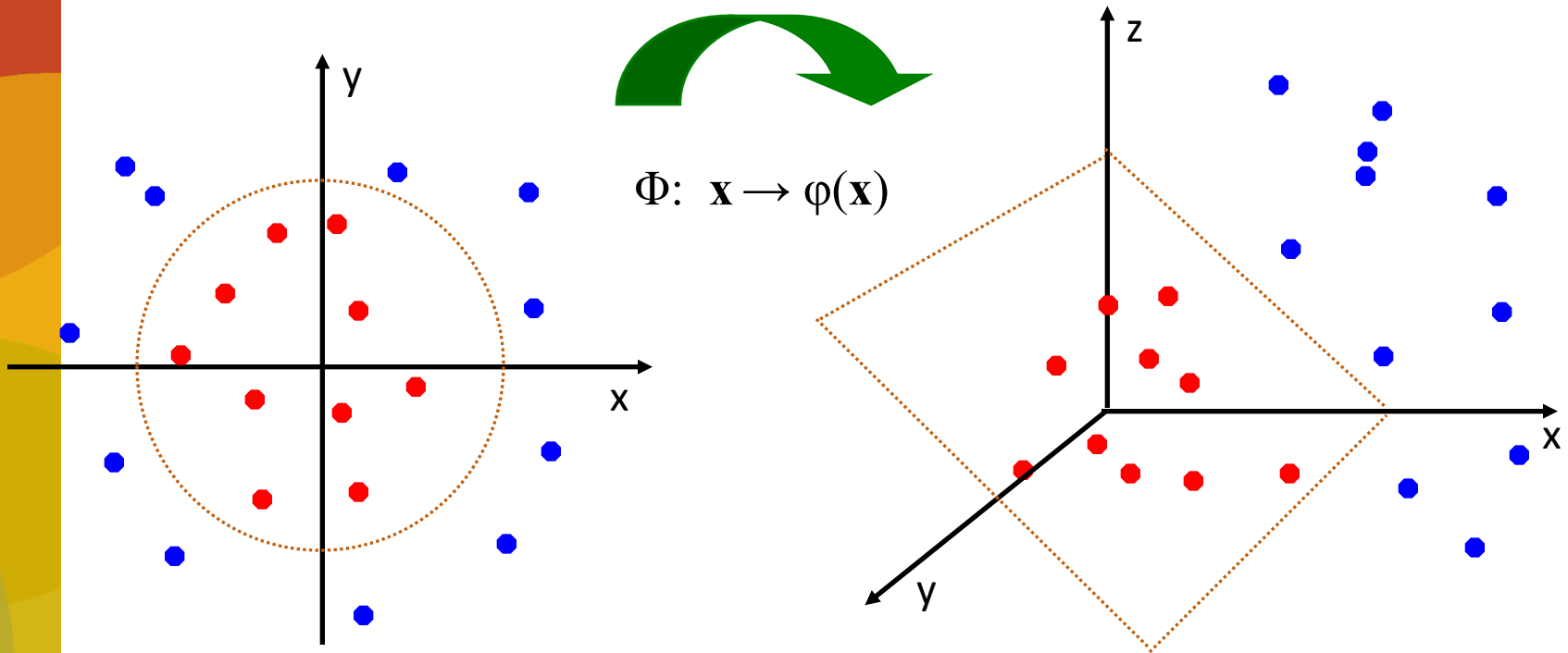
Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \le \alpha_i \le C$ for all α_i

$$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

Non-Linear SVM

Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Original instance space

Projected instance space

The "Kernel Trick"

- **The linear classifier relies on dot product between vectors: u.v**
- **If every data point is mapped into high-dimensional space via some transformation Φ: $x \rightarrow \varphi(x)$, the dot product becomes: $K(u,v) = \varphi(u) \cdot \varphi(v)$**
- **A *kernel function* is some function that receives 2 points in the instance space and computes the dot product in some expanded feature space.**
- **Example:**

2-dimensional vectors $u=[u_1 \ u_2]$ and $v=[v_1 \ v_2]$

let $K(u,v)=(1 + u \cdot v)^2$,

Need to show that $K(u,v)= \varphi(u) \cdot \varphi(v)$:

$$K(u,v)=(1 + u \cdot v)^2,$$

$$= 1+ u_1^2v_1^2 + 2\, u_1v_1\, u_2v_2 + u_2^2v_2^2 + 2u_1v_1 + 2u_2v_2$$

which corresponds to the inner product:

$$= [1 \ \ u_1^2 \ \ \sqrt{2}\, u_1u_2 \ \ u_2^2 \ \ \sqrt{2}u_1 \ \ \sqrt{2}u_2] \cdot [1 \ \ v_1^2 \ \ \sqrt{2}\, v_1v_2 \ \ v_2^2 \ \ \sqrt{2}v_1 \ \ \sqrt{2}v_2]$$

$$= \varphi(u)^{T}\varphi(v), \quad \text{where } \varphi(u) = [1 \ \ u_1^2 \ \ \sqrt{2}\, u_1u_2 \ \ u_2^2 \ \ \sqrt{2}u_1 \ \ \sqrt{2}u_2]$$

The dot product of vectors [*a*, *b*] and [*c*, *d*] is *ac* + *bd*.

What Functions are Kernels?

- **For some functions *K*(u,v) checking that**

  $K(u,v) = \varphi(u) \cdot \varphi(v)$ can be cumbersome.

- **Mercer's theorem:**

Every semi-positive definite symmetric function is a kernel

- **All eigenvalues are positive**

K=

$K(u_1,v_1)$	$K(u_1,v_2)$	$K(u_1,v_3)$	…	$K(u_1,v_N)$
$K(u_2,v_1)$	$K(u_2,v_2)$	$K(u_2,v_3)$		$K(u_2,v_N)$
…	…	…	…	…
$K(u_N,v_1)$	$K(u_N,v_2)$	$K(u_N,v_3)$	…	$K(u_N,v_N)$

Examples of Kernel Functions

- Linear: $K(\mathbf{u},\mathbf{v}) = \mathbf{u} \cdot \mathbf{v}$
- Polynomial of power p: $K(u,v) = (1 + u \cdot v)^p$
- Gaussian (radial-basis function network):

$$K(u,v) = \exp\left(-\frac{\|u-v\|^2}{2\sigma^2}\right)$$

- Sigmoid: $K(\mathbf{u},\mathbf{v}) = \tanh(\beta_0\, \mathbf{u} \cdot \mathbf{v} + \beta_1)$

Non-linear SVMs Mathematically

- **Dual problem formulation:**

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(x_i, x_j)$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

- **The solution is:**

$$f(x) = \Sigma\alpha_i y_i K(x_i, x_j) + b$$

- **Optimization techniques for finding α_i's remain the same!**

Nonlinear SVM - Overview

- SVM locates a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product in the feature space.

Properties of SVM

- Flexibility in choosing a similarity function
- Sparseness of solution when dealing with large data sets
  - only support vectors are used to specify the separating hyperplane
- Ability to handle large feature spaces
  - complexity does not depend on the dimensionality of the feature space
- Overfitting can be controlled by soft margin approach
- Nice math property:
  - a simple convex optimization problem which is guaranteed to converge to a single global solution
- Instance Selection

Weakness of SVM

SVM is sensitive to noise

A relatively small number of mislabeled examples can dramatically decrease the performance

It only considers two classes
- how to do multi-class classification with SVM?
- Answer:
1) with output arity m, learn m SVM's

SVM 1 learns "Output==1" vs "Output != 1"

SVM 2 learns "Output==2" vs "Output != 2"

:

SVM m learns "Output==m" vs "Output != m"

2)To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

Bibliography

Extração de Conhecimento de Dados- Data Mining (3ª Edição)
de João Gama, Ana Lorena, Katti, e André Carvalho, Silabo, 2017

Machine Learning, Tom Mitchel, McGrawHill

Data Mining: Concepts and Techniques, 3rd edition, Jiawei Han, Micheline Kamber, and Jian Pei, Morgan Kaufmann, 2011.