VC 11/12 – T15
Neural Networks & SVMs

Mestrado em Ciência de Computadores

Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos

Miguel Tavares Coimbra

U. PORTO FC

Outline

- Introduction to soft computing
- Neural Networks
- Support Vector Machines

Topic: Introduction to soft computing

- Introduction to soft computing
- Neural Networks
- Support Vector Machines

http://www.wallpaperbase.com/wallpapers/celebsm/michaeljordan/michael_jordan_1.jpg

Humans make great decisions

Soft-Computing

- Aim:

  "To exploit the tolerance for imprecision uncertainty, approximate reasoning and partial truth to achieve tractability, robustness, low solution cost, and **close resemblance with human like decision making**" [L.A.Zadeh]

- To find an approximate solution to an imprecisely/precisely formulated problem.

Simple problem: Parking a car

- Easy for a human to park a car.
- Why?
  - Position is not specified exactly.
- Otherwise:
  - Need precise measurements.
  - Need precise control.
  - Need a lot of time.

High precision carries a high cost!!

We park cars quite well

- We exploit the **tolerance for imprecision**.
- We search for an **acceptable solution**.
- We choose the acceptable solution with the **lowest cost**.

These are the **guiding principles** of **soft computing**!

Soft-Computing revisited

- Collection of methodologies which, in one form or another, reflect their defined guiding principles achieving:
  - – Tractability
    - We can handle otherwise ‘impossible’ problems.
  - – Robustness
    - We can handle imprecision and ambiguity.
  - – Close resemblance with human like decision making.

Principal constituent methodologies

- Fuzzy Systems
- Neural Networks
- Evolutionary Computation
- Machine Learning
- Probabilistic Reasoning

Complementary rather than competitive

Topic: Neural Networks

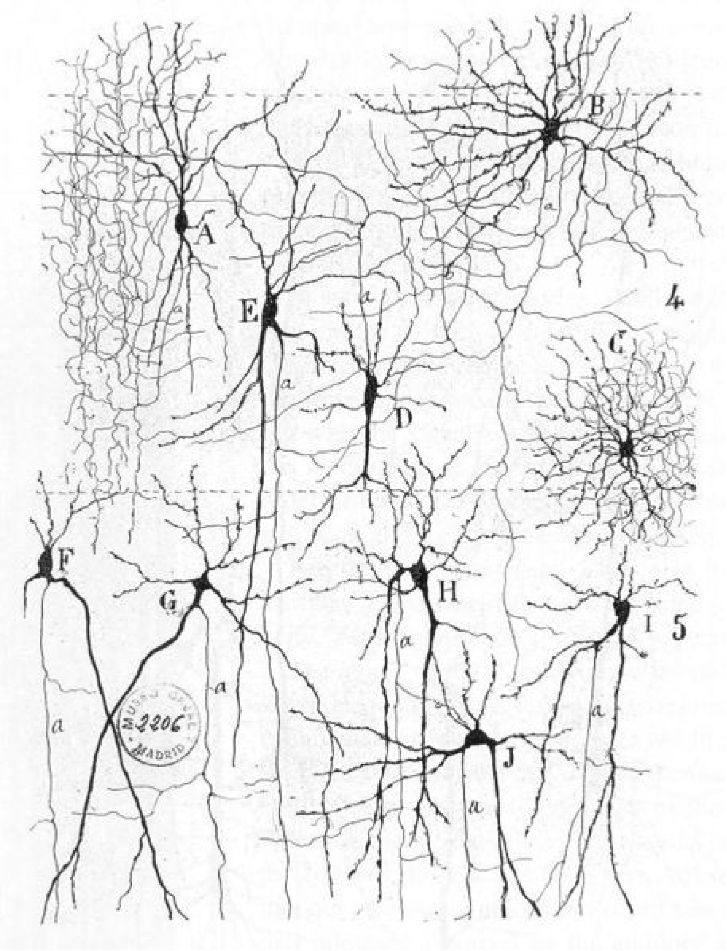
- Introduction to soft computing
- Neural Networks
- Support Vector Machines

If you can't beat it.... Copy it!

http://managementcraft.typepad.com/photos/uncategorized/brain.jpg

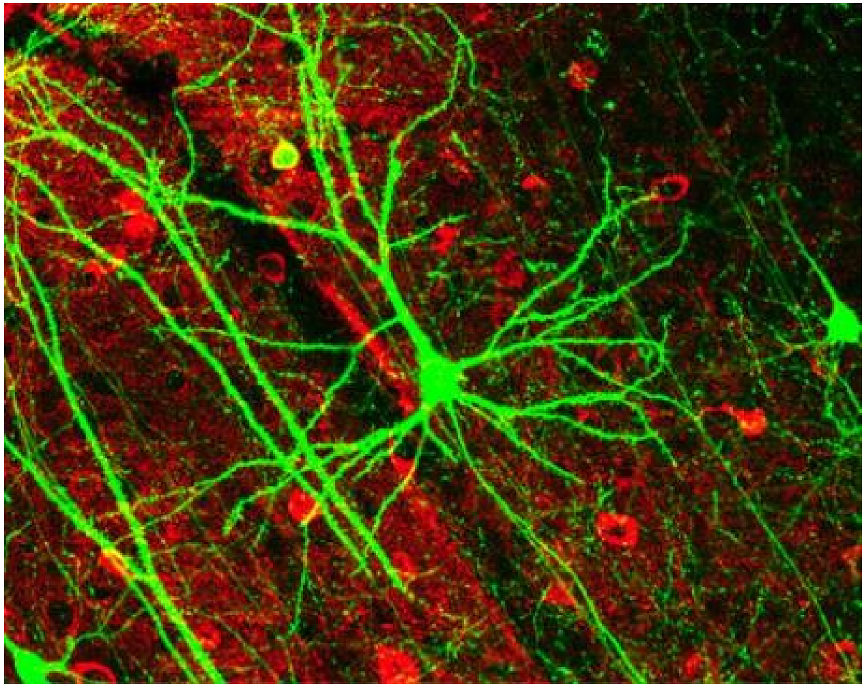
Biological Neural Networks

- Neuroscience:
  - Population of physically inter-connected neurons.
- Includes:
  - Biological **Neurons**
  - Connecting **Synapses**
- The human brain:
  - 100 billion neurons
  - 100 trillion synapses



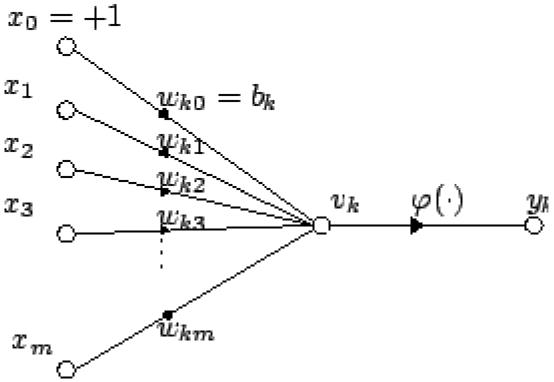
Biological Neuron

- Neurons:
  - Have *K* inputs (*dendrites*).
  - Have 1 output (*axon*).
  - If the sum of the input signals surpasses a *threshold*, sends an *action potential* to the axon.
- Synapses
  - Transmit electrical signals between neurons.



Artificial Neuron

- Also called the **McCulloch-Pitts neuron.**
- Passes a **weighted sum of inputs**, to an **activation function**, which produces an **output** value.



$$y_k = \varphi\left(\sum_{j=0}^{m} w_{kj} x_j\right)$$

McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 7:115 - 133.

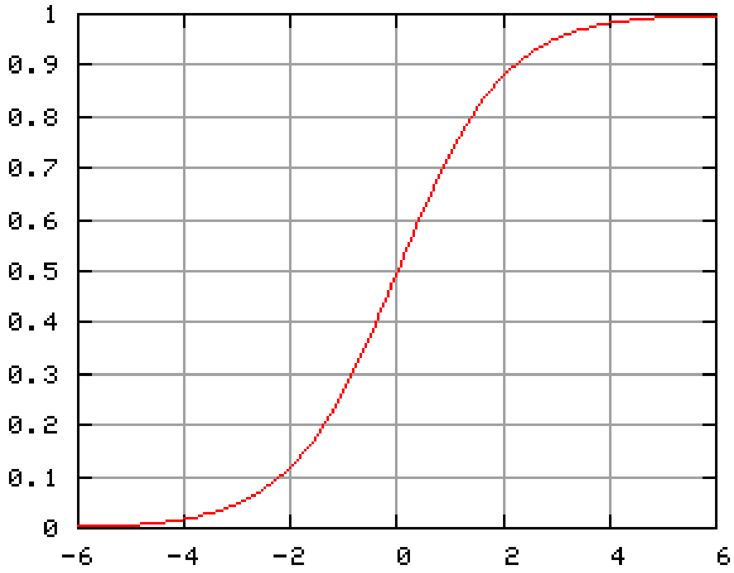
Sample activation functions

- Step function

$$y = \begin{cases} 1 & \Leftarrow u \geq k \\ 0 & \Leftarrow u < k \end{cases} \qquad u = \sum_{i=1}^{n} w_i x_i$$

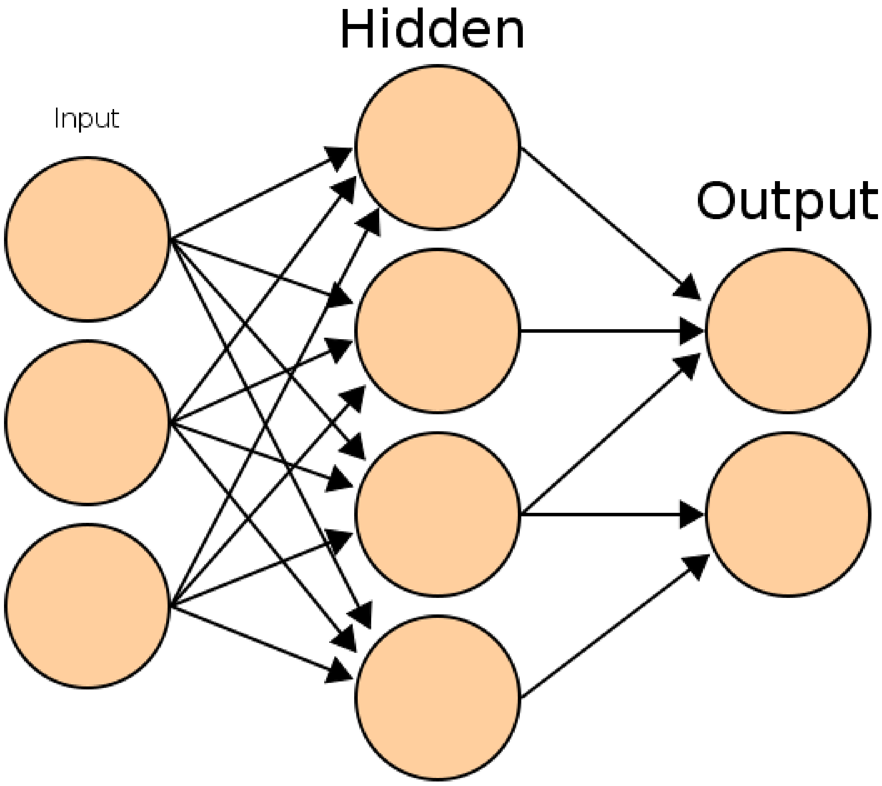
- Sigmoid function

$$y = \frac{1}{1 + e^{-u}}$$



Artificial Neural Network

- Commonly refered as **Neural Network**.
- Basic principles:
  - One neuron can perform a simple decision.
  - Many **connected** neurons can make more **complex decisions**.



Characteristics of a NN

- Network configuration
  - How are the neurons inter-connected?
  - We typically use *layers* of neurons (input, output, hidden).
- Individual Neuron parameters
  - Weights associated with inputs.
  - Activation function.
  - Decision *thresholds*.

How do we find these values?

Learning paradigms

- We can define the network configuration.
- How do we define neuron ***weights*** and ***decision thresholds***?
  - **Learning** step.
  - We **train** the NN to classify what we want.
- Different learning paradigms
  - Supervised learning.
  - Unsupervised learning.
  - Reinforcement learning.

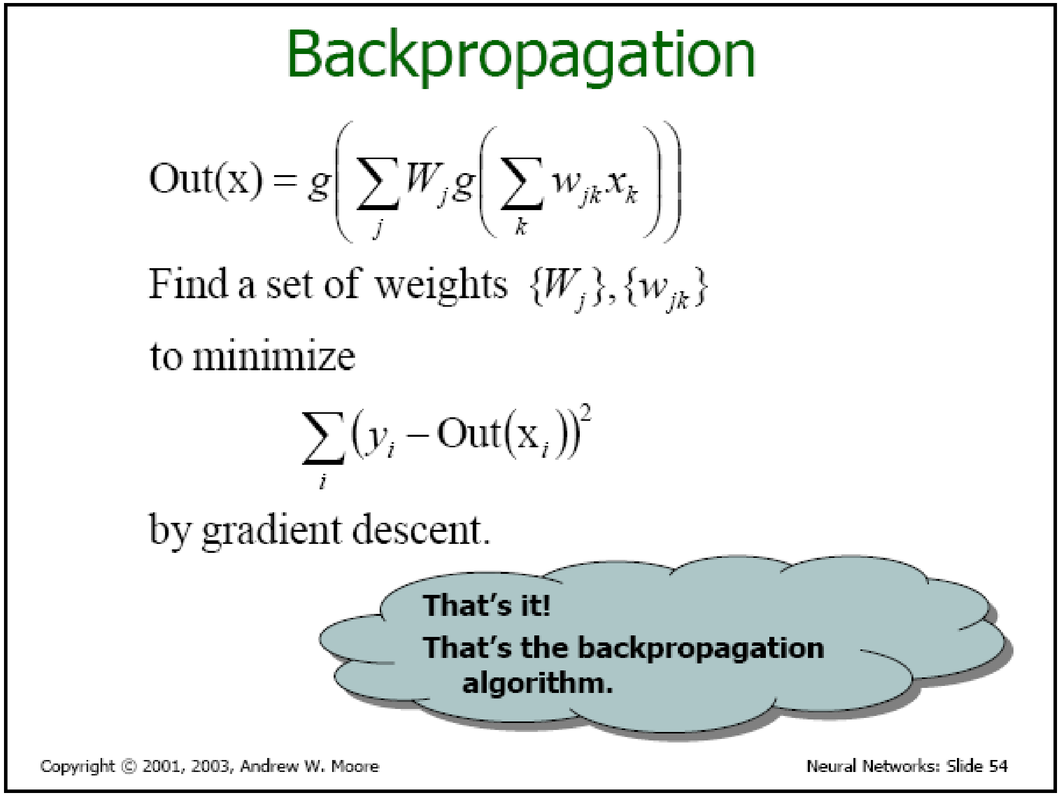
Appropriate for **Pattern Recognition**.

Learning

- We want to obtain an **optimal solution** given a set of **observations**.
- A **cost function** measures how close our solution is to the **optimal solution**.
- Objective of our learning step:
  - Minimize the **cost function**.

Backpropagation Algorithm

Backpropagation



Backpropagation

$$\text{Out}(\text{x}) = g\left(\sum_j W_j g\left(\sum_k w_{jk} x_k\right)\right)$$

Find a set of weights $\{W_j\},\{w_{jk}\}$

to minimize

$$\sum_i (y_i - \text{Out}(\text{x}_i))^2$$

by gradient descent.

That's it!
That's the backpropagation algorithm.

Copyright © 2001, 2003, Andrew W. Moore

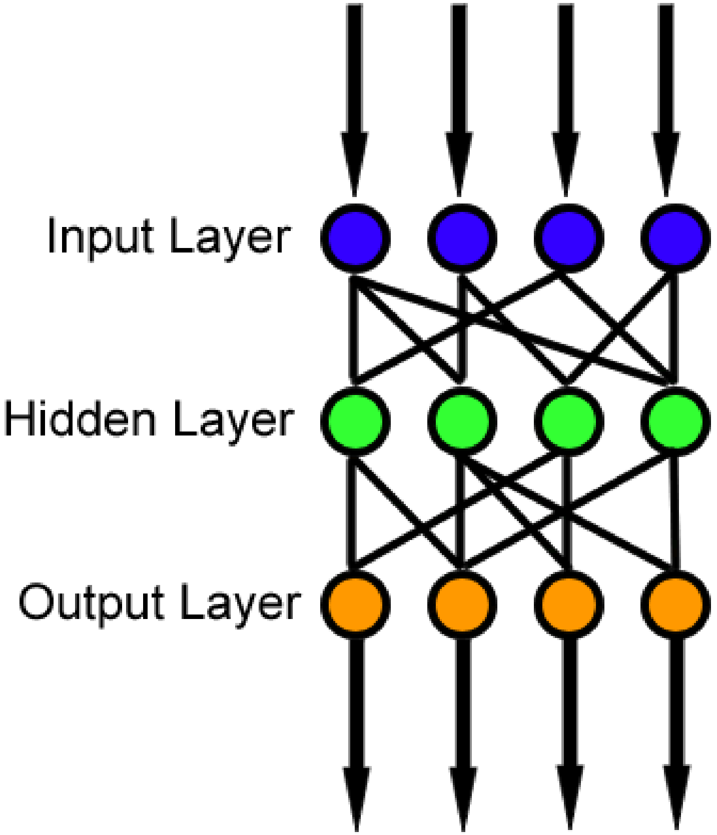
Neural Networks: Slide 54

For more details please study Dr. Andrew Moore's excellent tutorial.

http://www.autonlab.org/tutorials/neural13.pdf

Feedforward neural network

- Simplest type of NN.
- Has no *cycles*.
- Input layer
  - Need as many neurons as coefficients of my *feature vector*.
- Hidden layers.
- Output layer
  - Classification results.



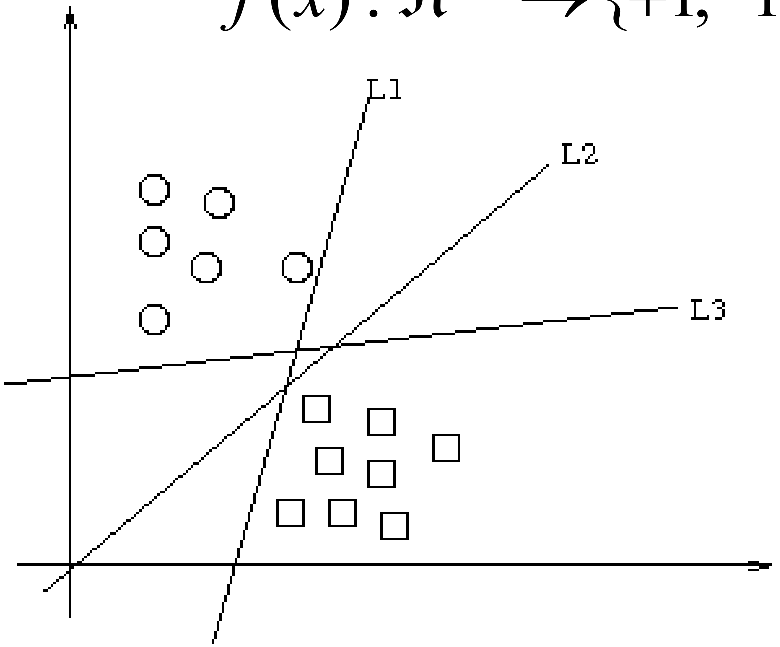
Topic: Support Vector Machines

- Introduction to soft computing
- Neural Networks
- Support Vector Machines

Maximum-margin hyperplane

- There are many planes that can separate our **classes** in **feature space**.
- Only one **maximizes the separation margin**.
- Of course that classes need to be separable in the first place...

$$f(x):\Re^M \rightarrow \{+1,-1\}$$

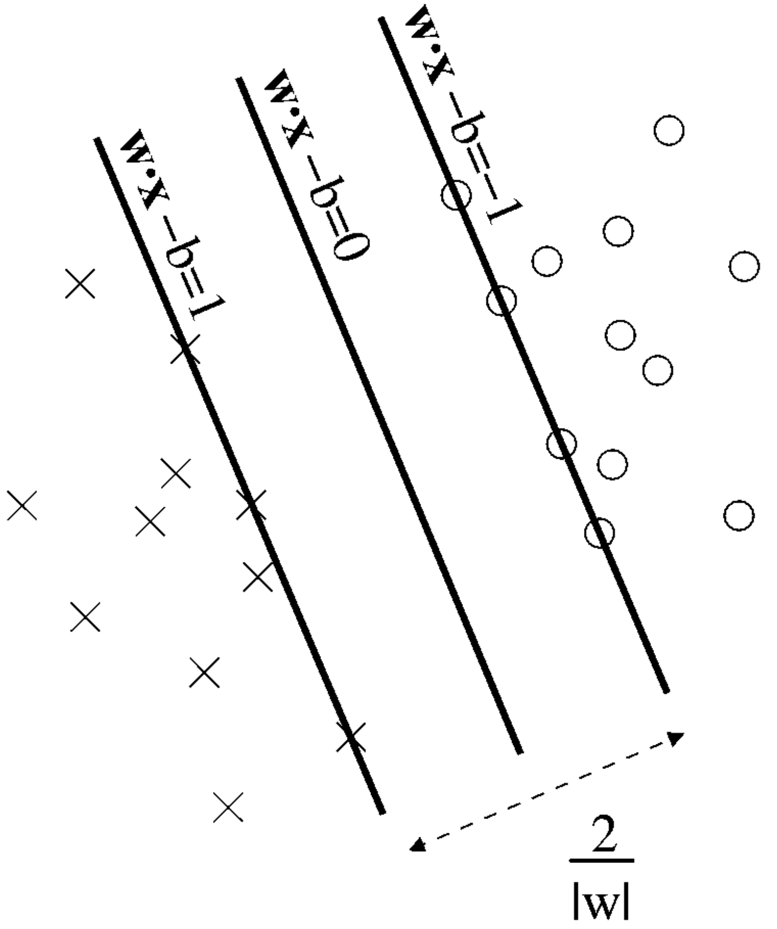


$$C \in \{+1,-1\}$$

$$V = \{v_1, v_2, \ldots, v_M\}, v_i \in \Re^M$$

Support vectors

- The **maximum-margin hyperplane** is limited by some vectors.
- These are called **support vectors**.
- Other vectors are irrelevant for my decision.



Decision

- I map a new **observation** into my **feature space**.
- Decision hyperplane:

$$(w.x)+b=0, w\in\Re^{N}, b\in\Re$$

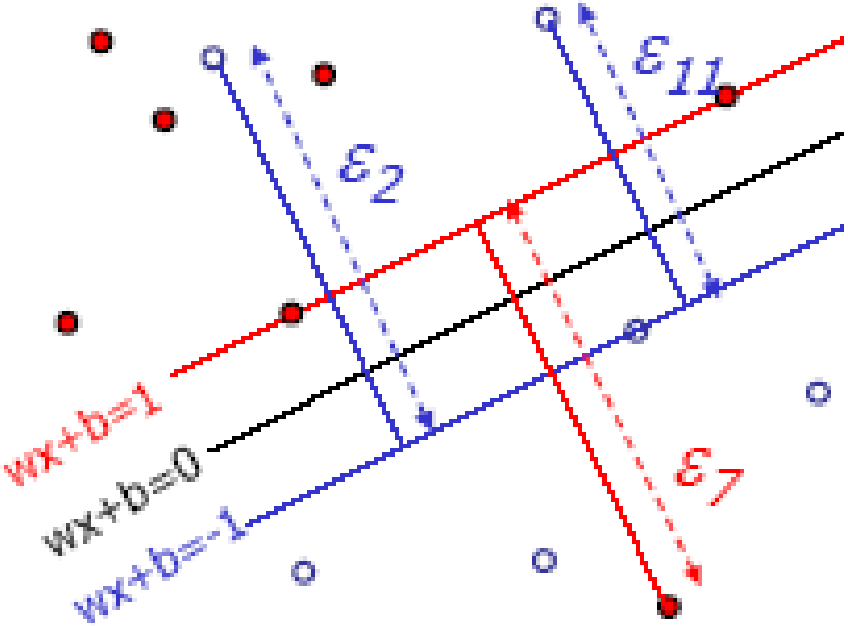
- Decision function:

$$f(x)=sign((w.x)+b)$$

A vector is either **above** or **below** the hyperplane.

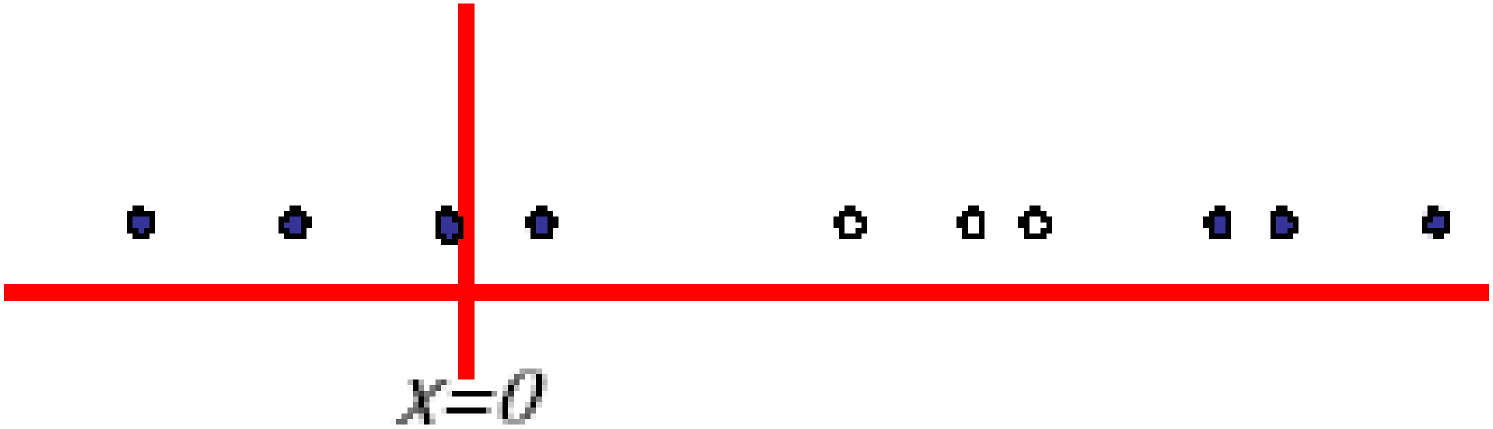
Slack variables

- Most **feature spaces** cannot be segmented so easily by a hyperplane.
- Solution:
  - Use slack variables.
  - ‘Wrong’ points ‘pull’ the margin in their direction.
  - Classification errors!



But this doesn't work in most situations...

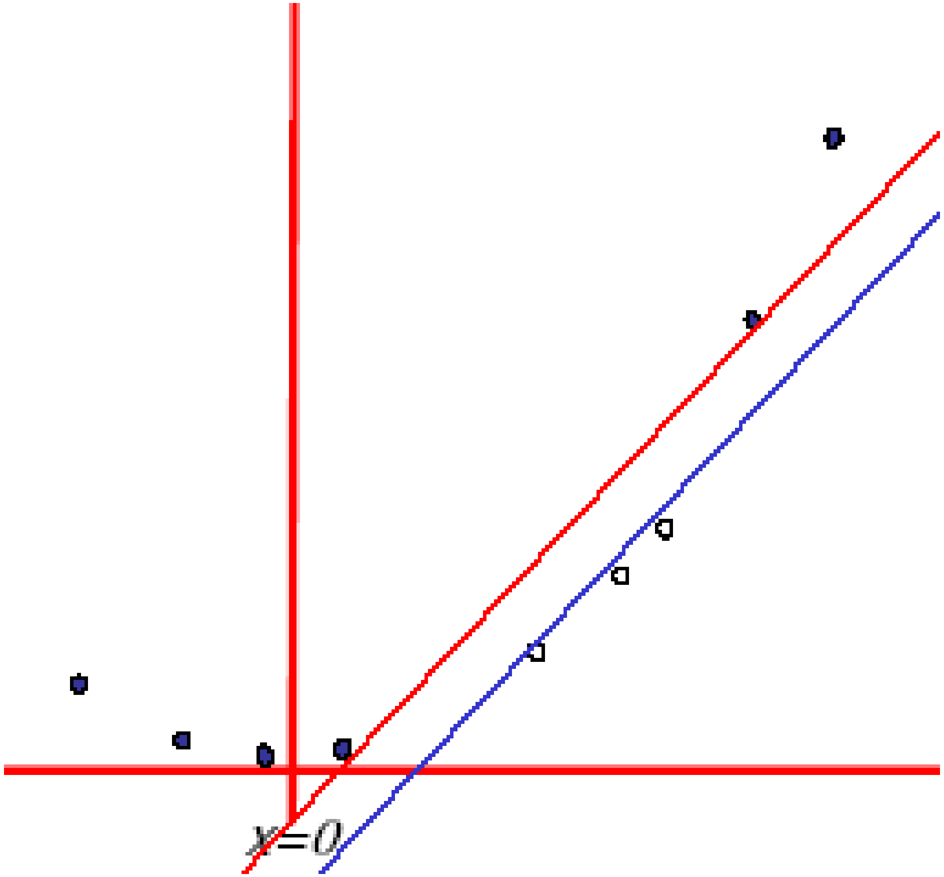
- Still, how do I find a **Maximum-margin hyperplane** for some situations?



- Most real situations face this problem...

Solution: Send it to hyperspace!

- Take the previous case: **f(x) = x**
- Create a new higher-dimensional function: **$g(x^2) = (x, x^2)$**
- A **kernel function** is responsible for this transformation.



Typical kernel functions

Linear	$K(x,y) = x.y + 1$
Polynomial	$K(x,y) = (x.y + 1)^p$
Radial-Base Functions	$K(x,y) = e^{-\lVert x-y \rVert^2 / 2\sigma^2}$
Sigmoid	$K(x,y) = \tanh(kx.y - \delta)$

Classification

- Training stage:
  - Obtain kernel parameters.
  - Obtain maximum-margin hyperplane.
- Given a new **observation**:
  - Transform it using the kernel.
  - Compare it to the hyperspace.
- Works very well indeed. Typically outperforms all known classifiers.

Resources

- Andrew Moore, "Statistical Data Mining Tutorials", http://www.autonlab.org/tutorials/
- C.J. Burges, "A tutorial on support vector machines for pattern recognition", in Knowledge Discovery Data Mining, vol.2, no.2, 1998, pp.1-43.