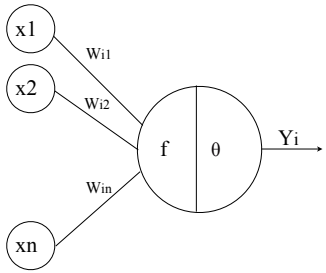
simple formal neuron

Formal neuron i



...to a simple formal neuron

Formal neuron i

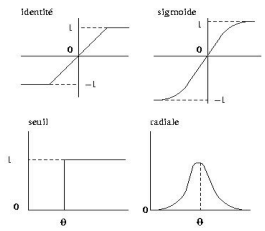
- calculate the incoming potentials, the Xi [-1,1]:
- summation in an internal potential Poti [-1,1]
- Threshold [-1,1]
- deliver one output Yi [-1,1]
- we have seen that the information transmission between two neurons is made by emission of neurotransmitters.
- Coefficient W [-1,1] of the incoming potential

...to a simple formal neuron

Formal neuron i

- Internal potential of neuron i : poti = $\sum$ wij.xj
- activation of output : Yi = f(poti)
- with f, the transfer function according to the model

For example :

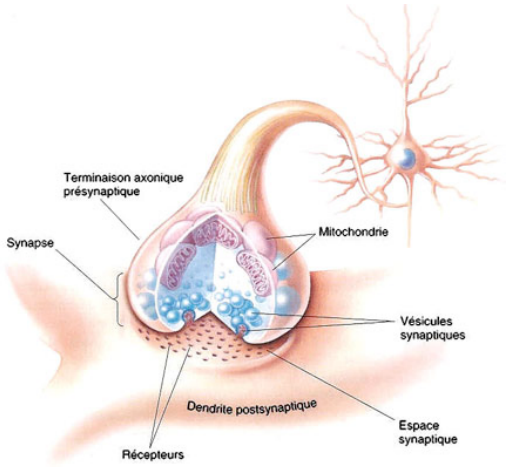


identity function

$$f(x) = \begin{cases} 0 \text{ if } x \leq \alpha \\ 1 \text{ if } x > \beta \\ x \text{ if } x \in [\alpha,\beta] \end{cases}$$

with $\theta = \beta - \alpha$

Learning



Learning

- nervous circuits : also crucial for memory (no dedicated centralised structure).

- Observation : an informal experience is correlated to measurable neuro-chemical and neuro-anatomical modifications in the brain

- Physiological modification of synapses:

- Pré-synaptic : increase release of neurotransmitters

- Post-synaptic : increase sensitiveness of the receptive membrane

- Structural modifications : the frequent “use” of a circuit induce an increase of the synaptic contacts

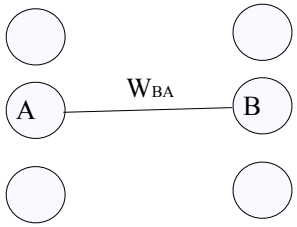
Learning

Hebb Rule [Hebb 49] :

“when cell A excites by its axon cell B and, in a repeated and persistent manner, it triggers impulsion of B, a process of metabolic change happens in one or two of both cells, driving to an significant increase of the efficiency of A to generate an impulsion in B, among the other cells”

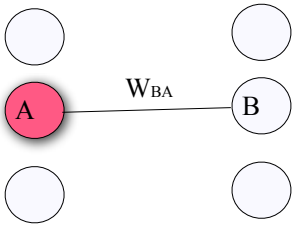
Learning

Hebb Rule [Hebb 49] :



Learning

Hebb Rule [Hebb 49] :

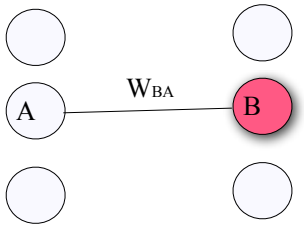


The sole A activity is not enough to induce B activation

$A^*W_{BA} < \theta_B$

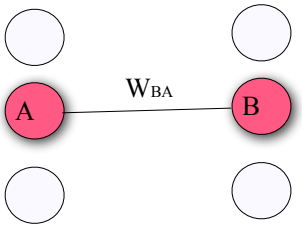
Learning

Hebb Rule [Hebb 49] :



Learning

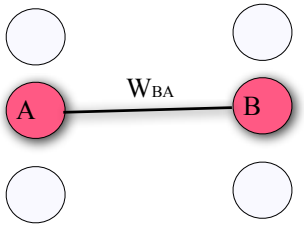
Hebb Rule [Hebb 49] :



For some reason... coactivation of A and B...

Learning

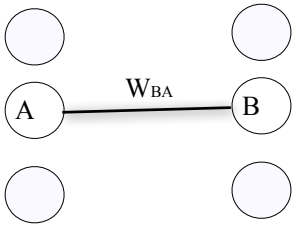
Hebb Rule [Hebb 49] :



For some reason... coactivation of A and B...repeatedly increase of the connection W_{BA}

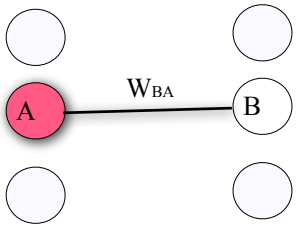
Learning

Hebb Rule [Hebb 49] :



Learning

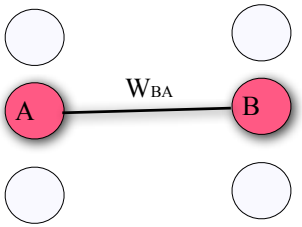
Hebb Rule [Hebb 49] :



The sole A activity is now enough ...

Learning

Hebb Rule [Hebb 49] :



The sole A activity is not enough to overshoot B threashold

$A^*W_{BA} > \theta_B$

Learning

- Hebb rule

$$w_{ij}(t+1) = w_{ij}(t) + eps.Y_j.X_i$$

i.e :

$$\Delta w_{ij} = eps.Y_j.X_i$$

with :

eps : learning speed in [0,1] (rather small)

Overview

Introduction : from biology to the formal neuron

Part I : supervised learning

- perceptron
  - simple rule
  - Widrow Hoff rule
  - limitations
- associative memories
- multi-layer perceptron
- backpropagation

Part II : unsupervised learning

- brain mechanisms
  - competition and cooperation
  - WTA
- Self Organizing Maps
  - Kohonen maps
  - K-means (analogy)
- Let's put it all together
  - ART

Supervised learning

- goal :
- make the network learn to categorize inputs
  - pattern recognition
  - class separation
- method :
  - train the network from different inputs
  - test the network generalization

Supervised learning

- goal :
- make the network learn to categorize inputs
  - pattern recognition
  - class separation
- method :
  - train the network from different inputs
  - test the network generalization

Supervised learning

The network will learn

- structural changes
- modification of Wij in order to obtain the correct output
- the expected result is a correct categorization
- learning is iterative: not too fast, not too slow
- learning rate epsilon
- test the generalization

Supervised learning

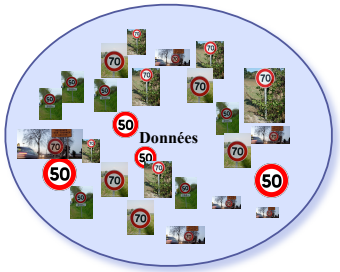
How to guide the learning ?

- fundamental notion of **error**
- supervised : we expect a given answer
- at each time step, we calculate the error
- error : (desired_ouput - output)
- while there is an error : we change the weights

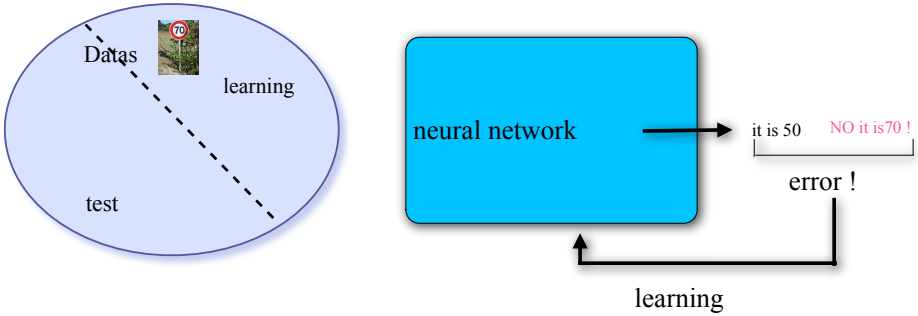
Supervised learning

Example

Supervised learning



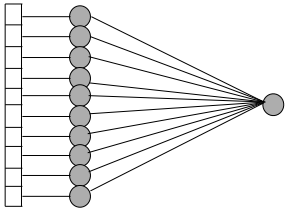
Supervised learning



Perceptron

Architecture

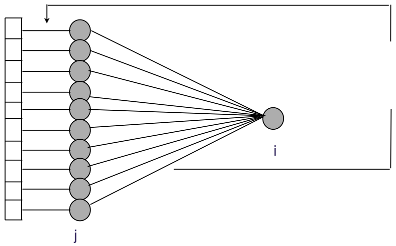
- Input, the retina : raw numerical information
- A first layer of neurons : one-one connections with the retina (normalization only)
- A last layer, called decision layer : the output of the system.



Perceptron

Architecture

- Input, the retina : raw numerical information
- A first layer of neurons : one-one connections with the retina (normalization only)
- A last layer, called decision layer : the output of the system.

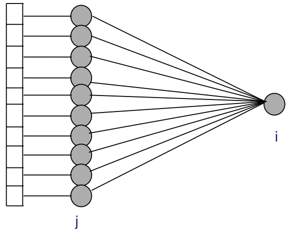


- non-modifiable weight. direct transmission, W=1/(nb_unit_in_input)
- modifiable weights. learning !
- Wij= random between [-1/n,1/n]

Perceptron

Learning : the simple rule

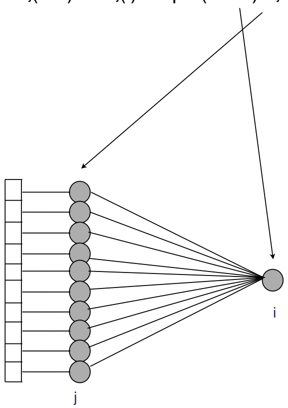
- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d - Y_i).X_j$



Perceptron

Learning : the simple rule

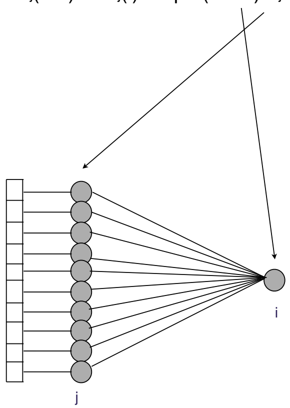
- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d - Y_i).X_j$



Perceptron

Learning : the simple rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d - Y_i).X_j$

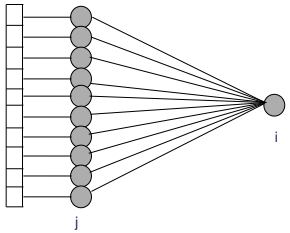


Perceptron

Learning : the simple rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.\underline{(Y_d - Y_i)}.X_j$

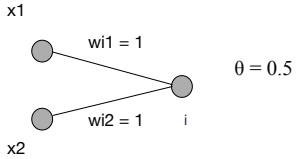
error



Perceptron

Graphical interpretation

- suppose the following simple perceptron :



- Draw the plane (here, the line) that separate the space in two categories.
- what happend if wi1 is equal to 2 ?
- what happend if θ is equal to 1 ?

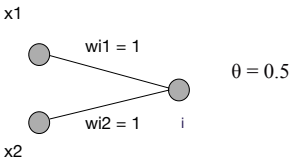
Perceptron

Graphical interpretation

- suppose the following simple perceptron :

Remember (lecture 1):

1. compute potential of neuron i : $pot_i = \sum w_{ij}.xj$
2. activation of output : $Yi = f(pot_i)$
- with f, the transfer function according to the model (heaviside : 1 if $pot_i > \theta$; 0 otherwise)



- Draw the plane (here, the line) that separate the space in two categories.
- what happend if wi1 is equal to 2 ?
- what happend if θ is equal to 1 ?

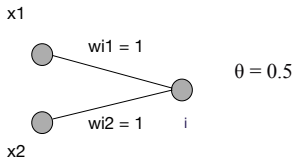
Perceptron

Graphical interpretation

- suppose the following simple perceptron :

Remember (lecture 1):

1. compute potential of neuron i : $pot_i = \sum w_{ij}.xj$
2. activation of output : $Yi = f(pot_i)$
- with f, the transfer function according to the model (heaviside : 1 if $pot_i > \theta$; 0 otherwise)



let start with x=(0,0)

Perceptron

Graphical interpretation

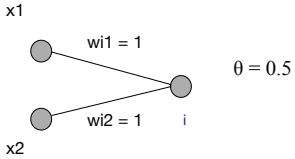
- suppose the following simple perceptron :

Remember (lecture 1):

1. compute potential of neuron i : $pot_i = \sum w_{ij}.xj$

2. activation of output : $Yi = f(pot_i)$

- with f, the transfer function according to the model (heaviside : 1 if $pot_i > \theta$; 0 otherwise)



$\theta = 0.5$

let start with x=(0,0)

poti = 0*1+0*1=0

Perceptron

Graphical interpretation

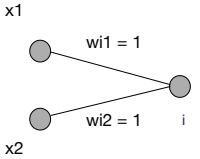
- suppose the following simple perceptron :

Remember (lecture 1):

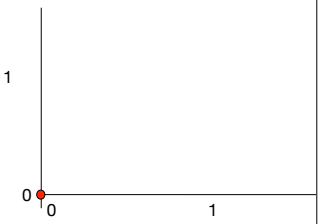
1. compute potential of neuron i : $pot_i = \sum w_{ij}.xj$

2. activation of output : $Yi = f(pot_i)$

- with f, the transfer function according to the model (heaviside : 1 if $pot_i > \theta$; 0 otherwise)



$\theta = 0.5$



let start with x=(0,0)

poti = 0*1+0*1=0

Yi = 0 because 0<0,5

Perceptron

Graphical interpretation

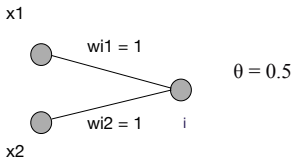
- suppose the following simple perceptron :

Remember (lecture 1):

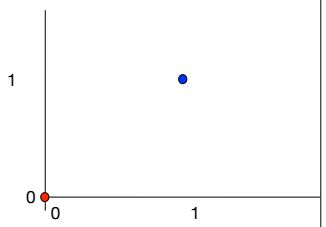
1. compute potential of neuron i : $pot_i = \sum w_{ij}.xj$

2. activation of output : $Yi = f(pot_i)$

- with f, the transfer function according to the model (heaviside : 1 if $pot_i > \theta$; 0 otherwise)



$\theta = 0.5$



let test with x=(1,1)

Perceptron

Graphical interpretation

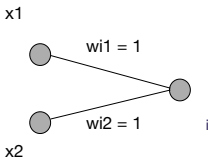
- suppose the following simple perceptron :

Remember (lecture 1):

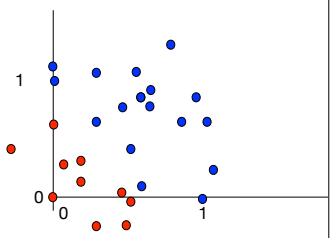
1. compute potential of neuron i : $pot_i = \sum w_{ij}.xj$

2. activation of output : $Yi = f(pot_i)$

- with f, the transfer function according to the model (heaviside : 1 if $pot_i > \theta$; 0 otherwise)



$\theta = 0.5$



and so on.....

Perceptron

Graphical interpretation

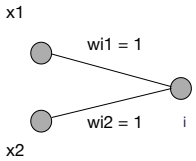
- suppose the following simple perceptron :

Remember (lecture 1):

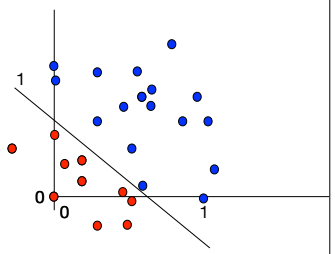
1. compute potential of neuron i : $pot_i = \sum w_{ij}.xj$

2. activation of output : $Yi = f(pot_i)$

- with f, the transfer function according to the model (heaviside : 1 if $pot_i > \theta$; 0 otherwise)



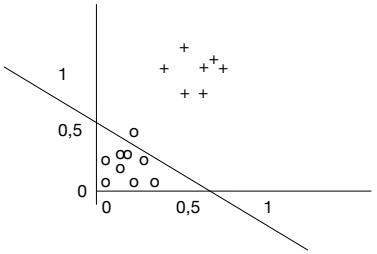
$\theta = 0.5$



the weights W1=1 and W2=2 draw a frontier between two categories

Perceptron

Graphical interpretation

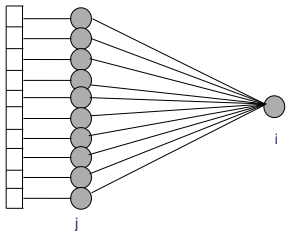


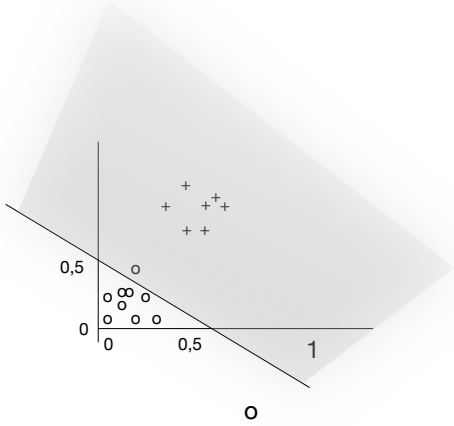
- input space separation
- categorization -> two areas, two categories in the input space
- change Wij -> change the frontier -> change the output -> change the error

Perceptron

Learning : the simple rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d-Y_i).X_j$

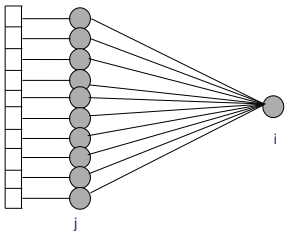


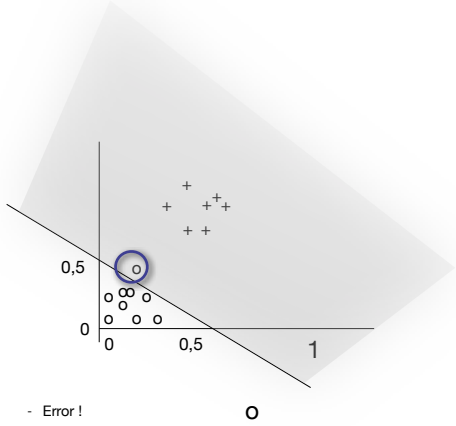


Perceptron

Learning : the simple rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d-Y_i).X_j$



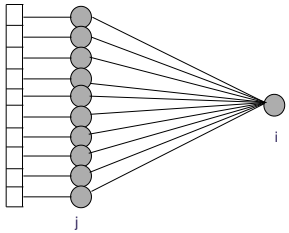


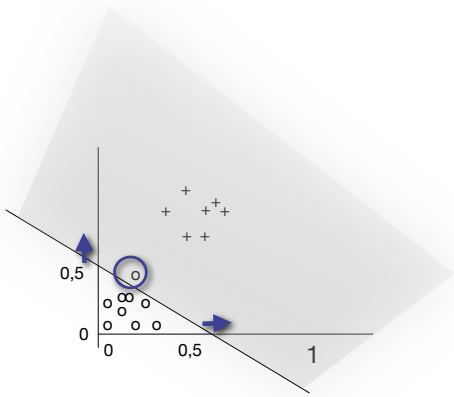
- Error !
- Yi is 1 and Yd is 0
- error is -1, both weights are lowered by eps

Perceptron

Learning : the simple rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d-Y_i).X_j$





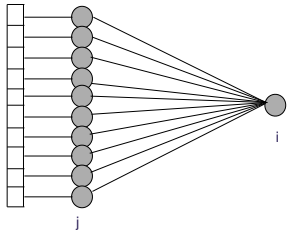
- Error !
- Yi is 1 and Yd is 0
- error is -1, both weights are lowered by eps

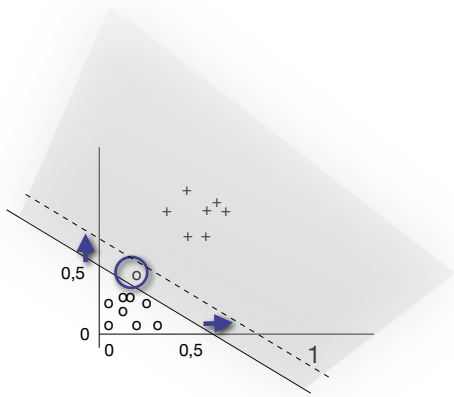
o

Perceptron

Learning : the simple rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d-Y_i).X_j$





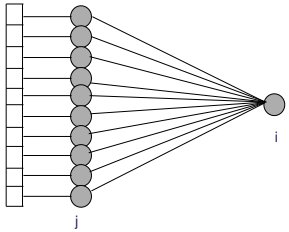
- Error !
- Yi is 1 and Yd is 0
- error is -1, both weights are lowered by eps

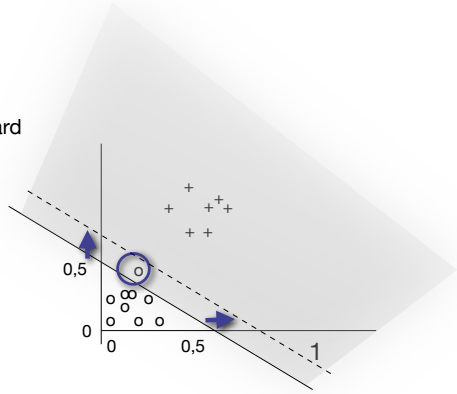
o

Perceptron

Learning : the simple rule

- Step by step (eps) progress toward space separation
- not accurate
- learning stops once no error





- Error !
- Yi is 1 and Yd is 0
- error is -1, both weights are lowered by eps

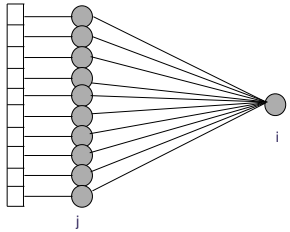
o

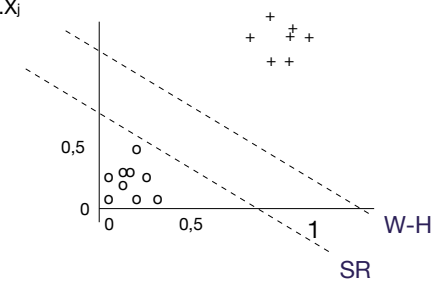
Perceptron

Learning : Widrow-Hoff rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d-\sum(wij.Xj)).X_j$

error





Mathematical explanation

Widrow-Hoff rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d - \sum(wij.Xj)).X_j$

  error

- Principle : we want our network to always provide the right answer

Mathematical explanation

Widrow-Hoff rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d - \sum(wij.Xj)).X_j$

  error

- Principle : we want our network to allways provide the right answer
- means : always provide the lowest error as possible
  - For all inputs examples
  - toward 0

Mathematical explanation

Linear regression and least mean square rule

- we introduce the global quadratic error : sum of the errors of all examples : $E^2 = \sum e^2$
- with e : perceptron error with a single example : $e = (Yd - \sum(wij.Xj))$
-

Mathematical explanation

Linear regression and least mean square rule

- we want to minimize E but :
  - At each time we only have acces to e
  - we can only change W
- So, we have to express E according to e and then according to W

- if $E = \sum (Yd - \sum(wij.Xj))^2$ then we will minimise E by descending the gradient of e, at each time step, changing W
- with $e = u(w)^2$ and $e' = 2u' . u(W)$
- $e' = 2 \ (\sum(wij.Xj)) .(Yd - \sum(wij.Xj)) = 2.Xi. (Yd - \sum(wij.Xj))$

Mathematical explanation

Linear regression and least mean square rule

- introducing eps being a learning speed parameter, we get :

$$\Delta wij = eps .Xi. (Yd - \sum(wij.Xj))$$

- that is to say :

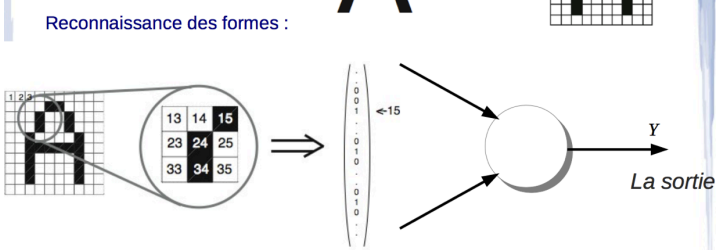
$$wij (t+1) = wij(t) + eps .Xi. (Yd - \sum(wij.Xj))$$

- that is sufficient to ensure a step by step minimization of E
- = descent of the error's gradient

Application : pattern recognition

Applications

Reconnaissance des formes :

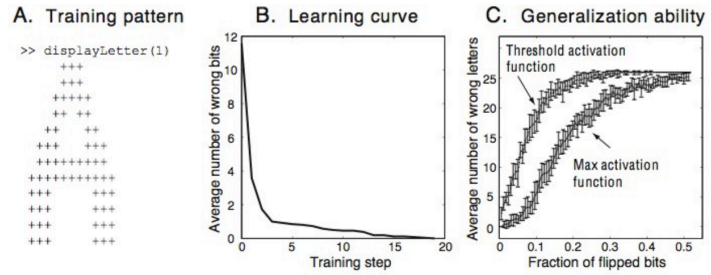


Les entrées

Optical character recognition: Predict meaning from features.
E.g., given features **x**, what is the character **y**

Application : pattern recognition

Applications 1

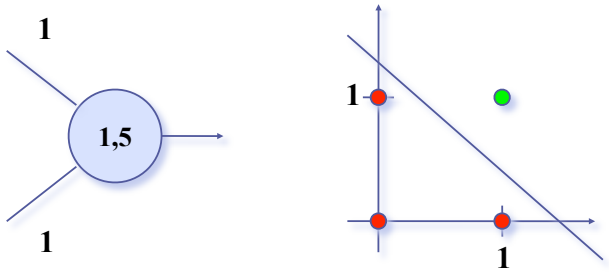


The network will learn

- structural changes
- modification of Wij in order to obtain the correct output
- the expected result is a correct categorization
- learning is iterative: not too fast, not too slow
- learning rate epsilon
- test the generalization

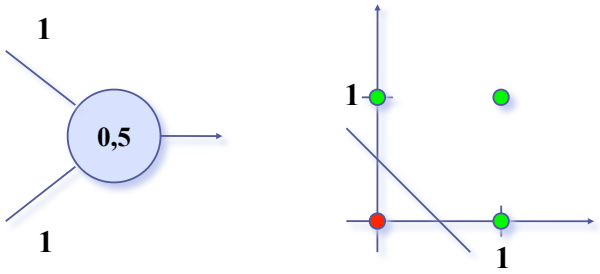
Perceptron : limitations

- logical AND



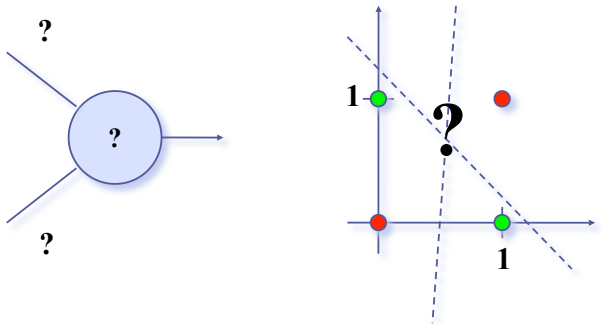
Perceptron : limitations

- logical OR



Perceptron : limitations

- XOR ?

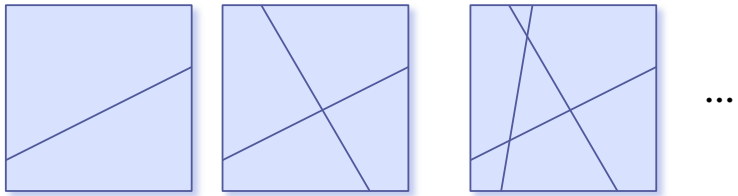


Perceptron : limitations

- limited to linear separation
- **what to do ?**
  - increase the number of layers and combine the outputs

Perceptron : limitations

- limited to linear separation
- **what to ?**
  - increase the number of layers and combine the outputs



1 layer 2 layers 3 layers ...

multi-layer perceptron

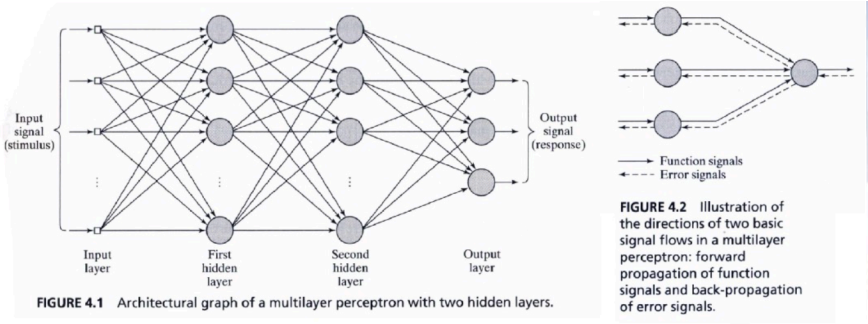


FIGURE 4.1 Architectural graph of a multilayer perceptron with two hidden layers.

FIGURE 4.2 Illustration of the directions of two basic signal flows in a multilayer perceptron: forward propagation of function signals and back-propagation of error signals.

multi-layer perceptron

The backpropgation algorithm looks for the minimum of the error function in weight space using the method of gradient descent

- the combination of weights that minimize this error = solution
- computation of the gradient of error at each time step
- switch to continuous activation function = continuous error function

$s_c : \mathbb{R} \to (0,1)$ defined by the expression

$$s_c(x) = \frac{1}{1+e^{-cx}}.$$

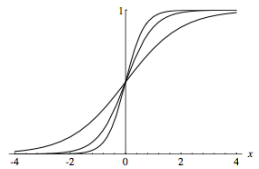


Fig. 7.1. Three sigmoids (for $c = 1$, $c = 2$ and $c = 3$)

back - propagation

Initialize all weights to small random numbers.

Until satisfied, Do

- For each training example, Do
  1. Input the training example to the network and compute the network outputs
  2. For each output unit j
     $\delta_j \leftarrow y_j(1-y_j)(d_j - y_j)$
  3. For each hidden unit h
     $\delta_h \leftarrow y_h(1-y_h)\sum_{j \in outputs} w_{jh}\delta_j$
  4. Update each network weight $w_{i,j}$
     $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$ where
     $\Delta w_{ji} = \eta \delta_j x_i$.

Note: w_{ji} is the weight from i to j (i.e., $w_{j \leftarrow i}$).

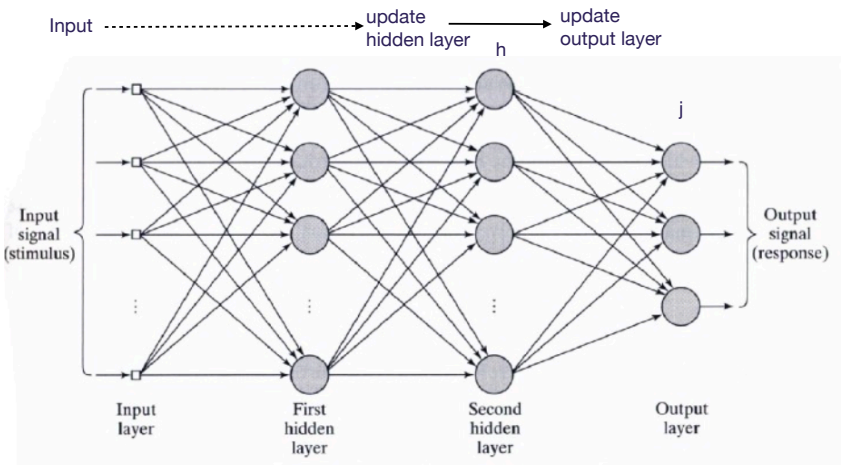
back - propagation

It mean that we will have to propagate the error backward to calculate the weights

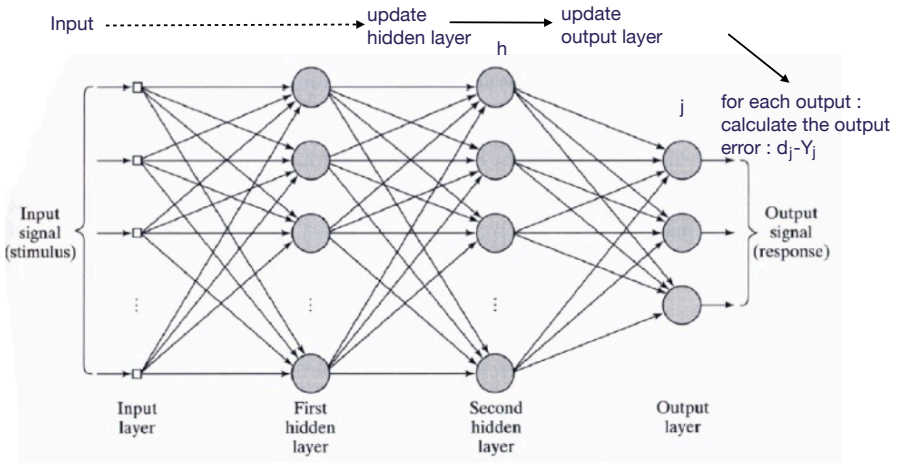
Activiy is computed forward

learning is computed backward

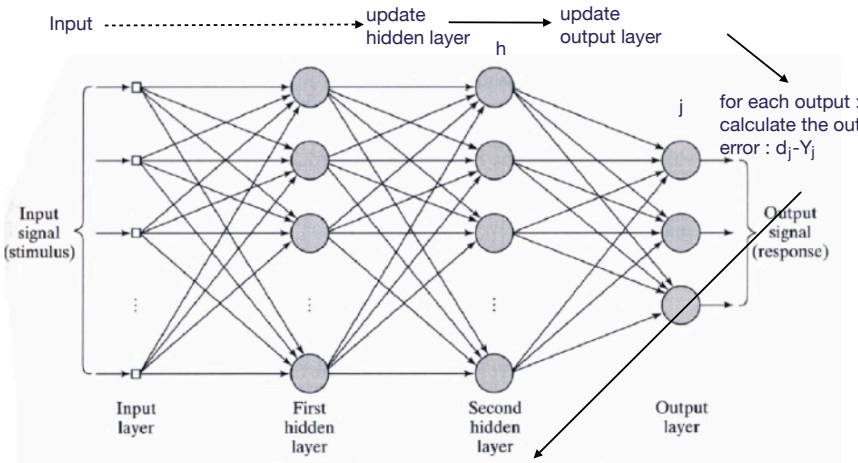
back - propagation



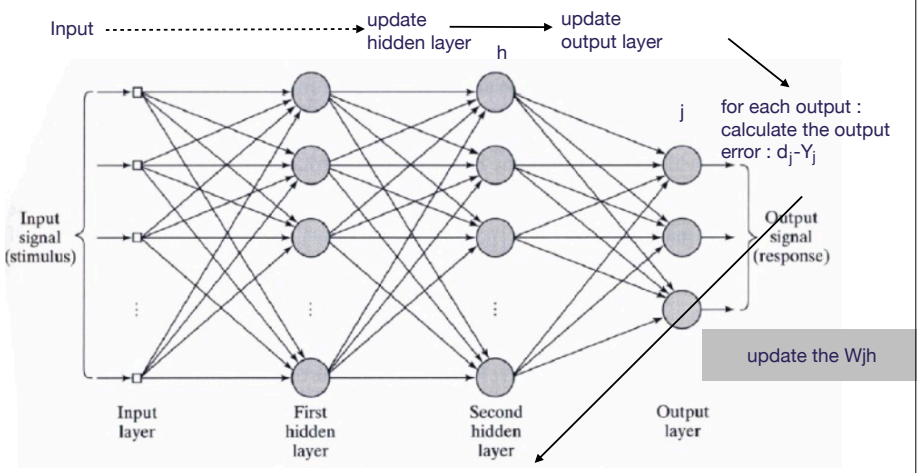
back - propagation



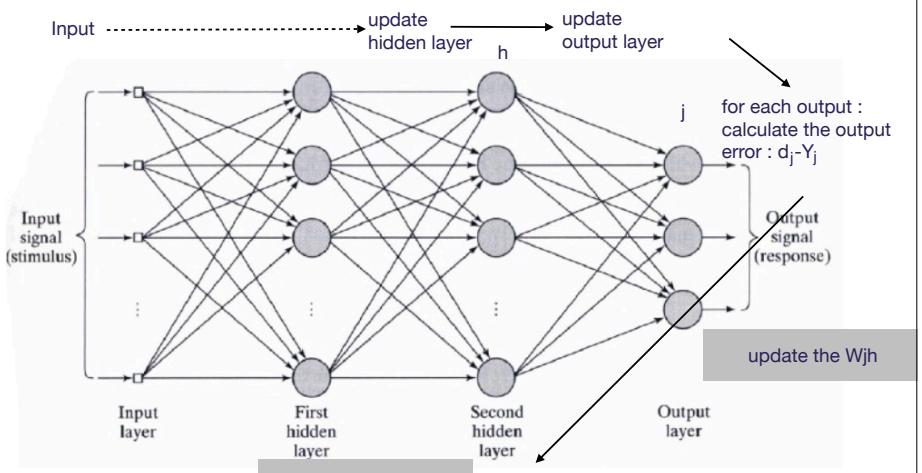
back - propagation



back - propagation



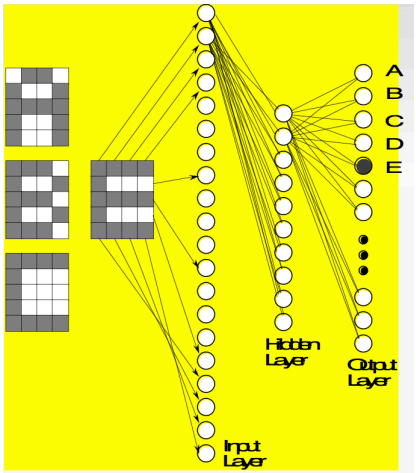
back - propagation



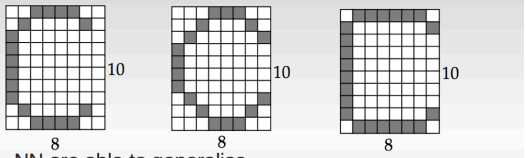
Applications

- The properties of neural networks define where they are useful.
  - Can learn complex mappings from inputs to outputs, based solely on samples
  - •Difficult to analyse: firm predictions about neural network behaviour difficult:
    - •Unsuitable for safety-critical applications.
  - Require limited understanding from trainer, who can be guided by heuristics.

Applications



Applications



- NN are able to generalise
- learning involves generating a partitioning of the input space
- for single layer network input space must be linearly separable
- what is the dimension of this input space?
- how many points in the input space?
- this network is binary(uses binary values)
- networks may also be continuous

Applications

Engine management

- The behaviour of a car engine is influenced by a large number of parameters
  - temperature at various points
  - fuel/air mixture
  - lubricant viscosity.
- Major companies have used neural networks to dynamically tune an engine depending on current settings.

Applications

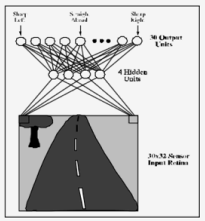
ALVINN

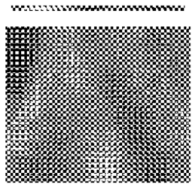
Drives 70 mph on a public highway

30 outputs for steering

4 hidden units

30x32 pixels as inputs





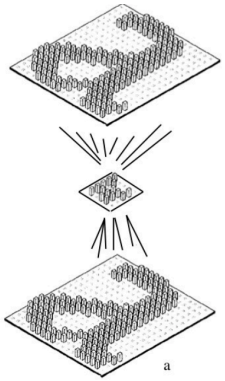
30x32 weights into one out of four hidden unit

Applications

Stock market prediction

- "Technical trading" refers to trading based solely on known statistical parameters; e.g. previous price
- Neural networks have been used to attempt to predict changes in prices.
- Difficult to assess success since companies using these techniques are reluctant to disclose information.

Applications



Compression