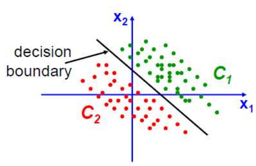
## Perceptron

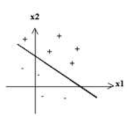
- Introduced by Frank Rosenblatt (American psychologist) in 1958
- Simple form of neural network used to classify examples linearly separable
- It is possible to find a hyperplane s.t. on the one side there are all the examples classified in one way, and on the other all those classified in another way.

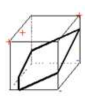
1

## Perceptron

Ex:

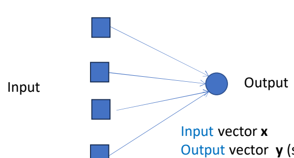


2

## Perceptron




Input vector **x**
Output vector **y** (single value if only a neuron)
Weight vector: **w**

Training set = insieme di coppie (input, output desiderato):

[(input$_1$,outputdesiderato$_1$),

(input$_2$,outputdesiderato$_2$),

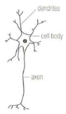
……

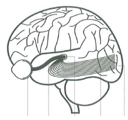
(input$_t$,outputdesiderato$_t$)]

3

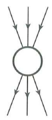
## Basic priciples of neural networks

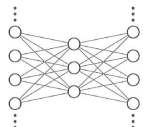
- In the brain the basic computational unit is the neuron
- There are 10 milliard neurons 60 millions of millions of synapses..



- In the same way the basic computational unit of the neural network is the neuron. Several neurons are connected to each other by synapses.

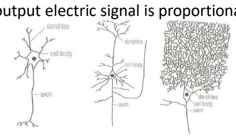




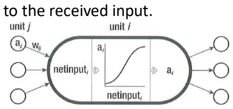
4

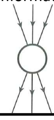
## Basic principles of neural networks

- In the brain, the role of a neuron is that of collecting, elaborating, propagating electric signals
- The output electric signal is proportional to the received input



- In the same way, the formal neuron receives adds and transmits to successive neurons the received information. The output is proportional to the received input.

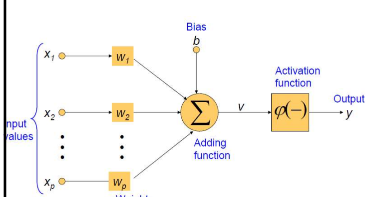


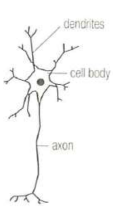



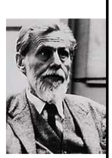
The operations of the single neuron are very simple:e.g., deciding if the total input is higher than a threshold or not

5

## Neuron (McCulloch Pitts 1943)



McCulloch-neuroscientist

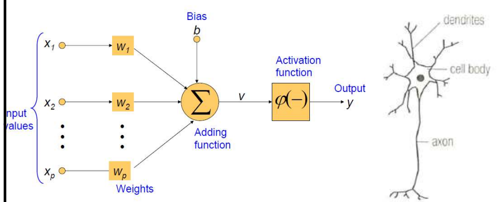
Pitts-Logician

Input vector $\mathbf{x} = \begin{bmatrix} x_1, \\ \dots \\ x_p \end{bmatrix}$

Weight vector $\mathbf{w} = \begin{bmatrix} w_1 \\ \dots \\ w_p \end{bmatrix}$

6

## Neuron (McCulloch Pitts 1943)



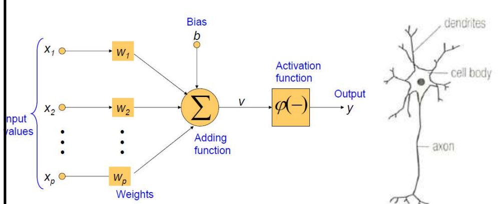

For each input value $x_i$ what reaches the neuron j is $w_{ji}x_i$

Receiver Sender

•$w_{ji}$ is the weight of the synapse from i to j. A synapse can be excitatory or inhibitory, so the weight can be > 0 or < 0 (or = 0).

7

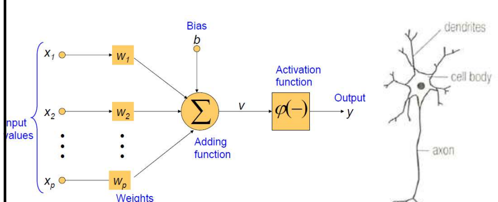
## Neuron




•Network input (netinput) to j = $\mathbf{x}\cdot\mathbf{w}$ (= $\mathbf{x}^T\mathbf{w} = \sum_{i=1}^{p} x_i w_i$)

8

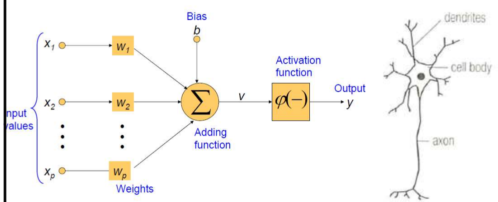
## Neuron



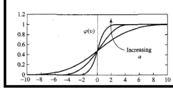

•Examples of input values:
- •pixel values
- •Phonemes
- •Measurements
- •Encoding of higher level concepts
- •Encoding of single words
- •Outputs from previous neurons.

9

## Neuron

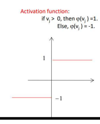



•The output of the neuron will depend on netinput and bias. yj = φ(uj + b), where uj + b the activation potential or local field of j.



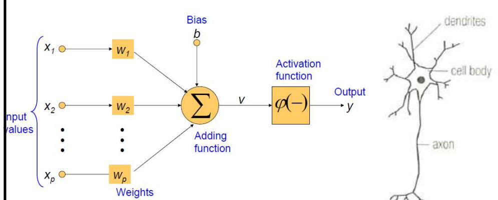

Examples of activation function.

For the perceptron:




10

## Neuron



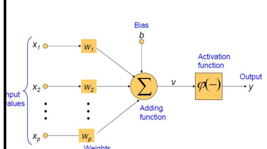

•All neurons have **a bias** that represents its spontaneous activation.

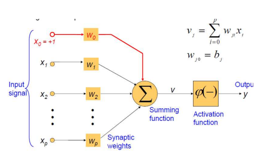
•If bias > 0,the neuron is often active, also with low inputs.

•If bias < 0, for the neuron to be active, incoming input values must be high (it is indeed a threshold).

11

## Neuron



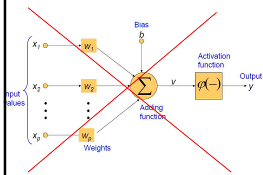

•To simplify the calculi, the bias of neuron j can be treated as an extra input element $x_0 = 1$ with $w_0 = b$.

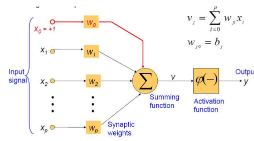
•In this case, $v_j = \sum_{j=0}^{p} w_{ji} x_i$

•We will use this architecture

12

## Neuron




•To simplify the calculi, the bias of neuron j can be treated as an extra input element $x_0 = 1$ with $w_0 = b$.

•In this case, $v_j = \sum_{j=0}^{p} w_{ji}x_i$

•We will use this architecture

13

## Example

OR

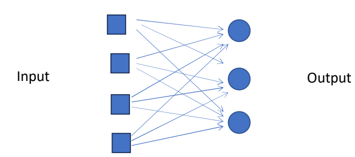
| |
|---|
| ([1,1], 1) |
| ([1,0], 1) |
| ([0,1], 1) |
| ([0,0],-1) |

Transformed into:

| |
|---|
| ([1,1,1], 1) |
| ([1,1,0], 1) |
| ([1,0,1], 1) |
| ([1,0,0],-1) |

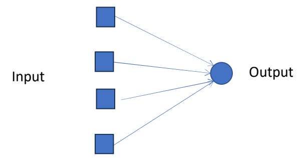
14

## Perceptron




15

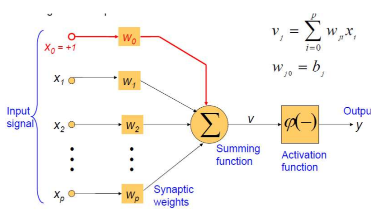
## Perceptron

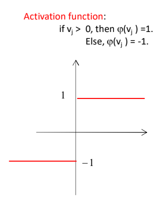



16

## Perceptron

- Activation function:
  - The output of the perceptron will be 1 if $\sum_{i=0}^{p} w_{ji} x_i > 0$ (i.e. **x**·**w** > 0)
  - The output of the perceptron will be -1 if $\sum_{i=0}^{p} w_{ji} x_i \le 0$ (i.e. **x**· **w** ≤ 0)




17

## Perceptron

- The perceptron's output for a given input depends on the weight vector

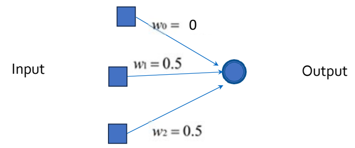
18

## Perceptron

[1,1,1]
[1,1, 0]
[1,0, 1]
[1,0, 0]

Given a certain weights configuration $\mathbf{w}$,

- output is 1 for all input vectors $\mathbf{x}$ s.t. $\mathbf{x} \cdot \mathbf{w} > 0$
- output is -1 for all input vectors $\mathbf{x}$ s.t. $\mathbf{x} \cdot \mathbf{w} \leq 0$



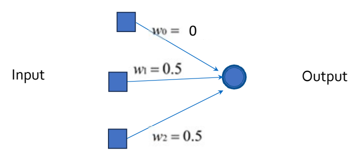

19

## Perceptron

OR ([1,1,1] , 1)
([1,1, 0], 1)
([1,0, 1], 1)
([1,0, 0], -1)

Given a certain weights configuration $\mathbf{w}$,

- output is 1 for all input vectors $\mathbf{x}$ s.t. $\mathbf{x} \cdot \mathbf{w} > 0$
- output is -1 for all input vectors $\mathbf{x}$ s.t. $\mathbf{x} \cdot \mathbf{w} \leq 0$



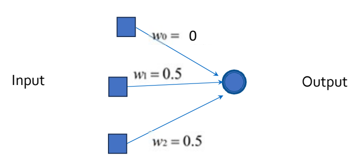

20

## Perceptron

OR ([1,1,1] , 1)
([1,1, 0], 1)
([1,0, 1], 1)
([1,0, 0], -1)

Given a certain weights configuration $\mathbf{w}$,

- output is 1 for all input vectors $\mathbf{x}$ s.t. $\mathbf{x} \cdot \mathbf{w} > 0$
- output is -1 for all input vectors $\mathbf{x}$ s.t. $\mathbf{x} \cdot \mathbf{w} \leq 0$




The decision boundary is made by all $\mathbf{x}$ s.t. $\mathbf{x} \cdot \mathbf{w} = 0$

21

## Perceptron

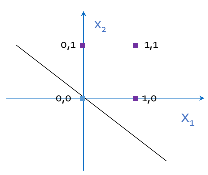
In dimension 3 (with $w_0 = 0$) { $\mathbf{x}$ s.t $\mathbf{x} \cdot \mathbf{w} = 0$} = { $\mathbf{x}$ s.t. $x_1w_1 + x_2w_2 = 0$}.

$w_1x_1 + w_2x_2 = 0$ is the equation of the decision boundary line.

22

## Perceptron

Example, a possible solution for OR is: $w_1=w_2=0.5$, $w_0 = 0$

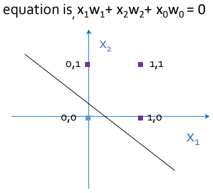



$0.5\ x_1 + 0.5\ x_2 = 0$

23

## Perceptron

If $w_0$ not 0, the equation is, $x_1w_1 + x_2w_2 + x_0w_0 = 0$




$$0.5 \cdot x_1 + 0.5 \cdot x_2 - 0.1 = 0$$

➡ Analogous considerations hold for inputs with more dimensions.

24

## Perceptron

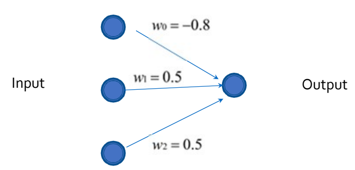
Exercise.

1) Find a configuration for AND
([1,1,1] , 1)
([1,1, 0], -1)
([1,0, 1], -1)
([1,0, 0], -1)

25

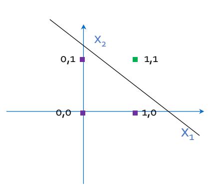
## Perceptron

AND([1,1,1] , 1)
([1,1, 0], -1)
([1,0, 1], -1)
([1,0, 0], -1)




26

## Perceptron




$0.5x_1 + 0.5x_2 - 0.8 = 0$

27

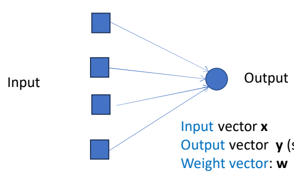
## Perceptron

- The perceptron learns to classify examples linearly separable by adjusting the weights.
- Examples (or ‘patterns’ o ‘instances’) are pairs (input, desired output) that constitute the Training Set.

Eg: ([1,1,1] , 1)
([1,1, 0], 1)
([1,0, 1], 1)
([1,0, 0], -1)

28

## Perceptron




Input vector **x**
Output vector **y** (single value if only a neuron)
Weight vector: **w**

Training set = set of pairs (input, desired output)

Each element of a training set= pattern[1]
There can also be a Test set (in more complex networks) to test the network learning on a slightly different training set.

[1]Terminology: ‘pattern’ is used to denote a whole pair (input, desired output) of the training set or only its input component

29

## Learning algorithm

- Adjust the weights to obtain the desired classification
- ➡ Learning algorithm based on the correction of the error (for each misclassified pattern)

30

## Learning algorithm

**Rule for the correction of weights at the n-th iteration:**

- w(n+1) = w(n) if output correct
- *If output incorrect:*
  - w(n+1) = w(n) - η(n)x(n) if x(n) ·w(n) > 0 and x(n) ∈ *C2* ← *(Output too high)*
  - w(n+1) = w(n) + η(n)x(n) if x(n) · w(n) ≤ 0 and x(n) ∈ *C1* ← *(Output too low)*

31

## Learning algorithm

- The algorithm continues until there are elements not correctly classified

Pseudo-code:

```
n=0;
   initialize w(n) randomly;
   while (there are misclassified training examples)
           Select a misclassified augmented example (x(n),d(n))
            if(d(n) = 1), w(n+1) = w(n) + ηx(n);
           if(d(n) = -1), w(n+1) = w(n) - ηx(n);
           n = n+1;
   end-while;
```

An epoch is a iteration over all elements of the training set
Here weight update is pattern-by-pattern: weights are updated for each pattern misclassified.

32

## Learning algorithm

- The algorithm continues until there are elements not correctly classified

Pseudo-code:

```
n=1;
   initialize w(n) randomly;
   while (there are misclassified training examples)
           Select a misclassified augmented example (x(n),d(n))
            if(d(n) = 1), w(n+1) = w(n) + ηx(n);
           if(d(n) = -1), w(n+1) = w(n) - ηx(n);      → w(n+1) = w(n) + ηd(n)x(n);
           n = n+1;
   end-while;
```

33

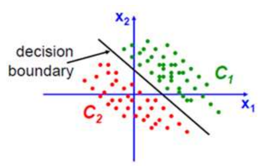
## Exercise

- Apply the learning algorithm to a perceptron with w(0) = 0, η = 0.5 so that it correctly classifies the following instances (AND):
  ([1,1,1] , 1)
  ([1,1, 0], -1)
  ([1,0, 1], -1)
  ([1,0, 0], -1)

34

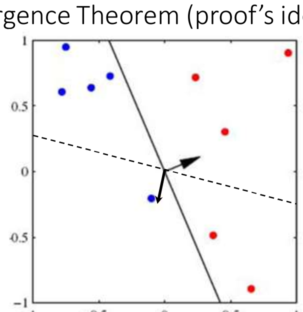
## Convergence Theorem

- If there is a solution (i.e., if the problem is linearly separable), the algorithm finds it.




35

## Convergence Theorem (proof's idea)

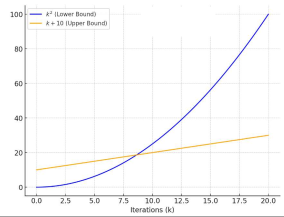



At each iteration, the weight vector is modified and, as a consequence, the decision boundary.
The Convergence Theorem guarantees that weight adjustement terminates

36

## Convergence Theorem (proof's idea)

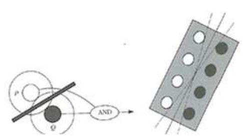
- $||w(k+1)||^2 \geq k^2\alpha^2 / ||w^*||^2$ (A) Lower Bound
- $||w(k+1)||^2 \leq k\,\beta$ (B) Upper Bound
- (A) and (B) are compatibile only if $k^2\alpha^2 / ||w^*||^2 \leq k\,\beta$, i.e., $k \leq \beta\, ||w^*||^2 / \alpha^2$

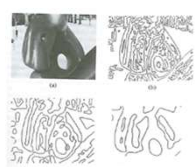



37

## Biological inspiration for the perceptron

- Perceptron was presented as a pattern recognition device (although it is used for more general problems)
- The perceptron mimicks a part of what is known of the mammalian visual system
- In the early 60s David Hubel Torsten Wiesel proposed an explanation of early stages of vision based on experiments on cats.
- They identified specific groups of cells that responded in a fixed way to events that occurred in a limited area of the retina (their receptive field)
- EX: groups of cells that respond to the existence of a line in a specific direction
- (specialized cells--Marr—primal sketch).




38

## Perceptron

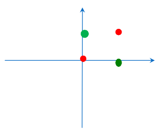
Inspired by Huber and Wiesel discovery of simple cells in cats' visual cortex

Show video (mins 1 to 2)
https://www.youtube.com/watch?v=jw6nBWo21Zk

39

## Limits of perceptron

- Solves linearly separable problems, as AND and OR
- It does not solve non linearly separable problems, as XOR



- ([1,0], 1)          ([1,1], -1]
- ([0, 1], 1)          ([0,0], -1)

*Perceprons di* Minsky e Papert ,del 1969

40

## Limits of perceptron

- *Perceptrons by* Minsky e Papert, 1969 opens the crisis of research in neural networks.

41

## Other problems that cannot be solved by 1-layer perceptrons

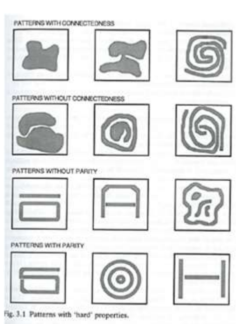


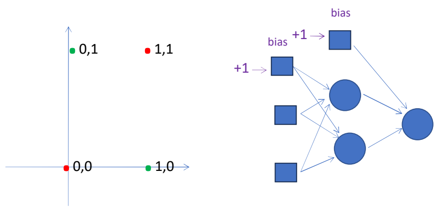
Fig. 3.1 Patterns with 'hard' properties.

Minsky Papert
Perceptrons 1969

42

## Multilayer networks

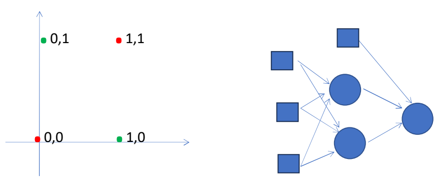
- To solve XOR we must introduce hidden units: each unit solves part of the problem, then solutions are combined.




43

## Multilayer networks

- To solve XOR we must introduce hidden units: each unit solves part of the problem, then solutions are combined.



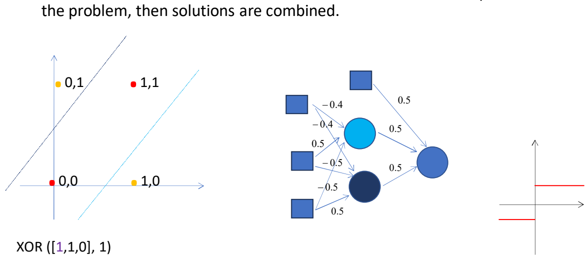

XOR ([1,1,0], 1)
([1,0,1], 1)
([1,0,0],0)
([1,1,1],0)

x1 XOR x2 ≡ (x1 ∧ ¬x2) ∨ (¬x1 ∧ x2)

44

## Multilayer networks

- To solve XOR we must introduce hidden units: each unit solves part of the problem, then solutions are combined.

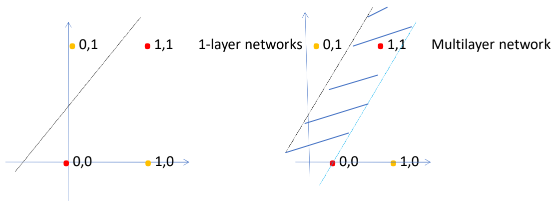



XOR ([1,1,0], 1)
([1,0,1], 1)
([1,0,0],0)
([1,1,1],0)

x1 XOR x2 ≡ (x1 ∧ ¬x2) ∨ (¬x1 ∧ x2)

45

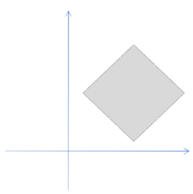
## Multilayer networks




46

## Multilayer networks

- If enough hidden units, the network can discriminate a convex zone.



47