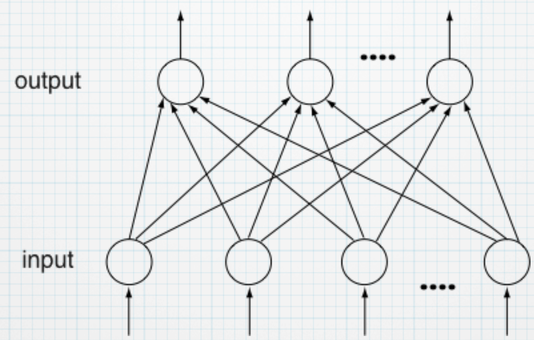
The Perceptron Algorithm

Perceptron (Frank Rosenblatt, 1957)

- First learning algorithm for neural networks;
- Originally introduced for character classification, where each character is represented as an image;

Perceptron (contd.)



Total input to output node: $\sum_{j=1}^{n} w_j x_j$

Output unit performs the function: (activation function):

$$H(x) = \begin{cases} 1 & \textit{if } x \geq 0 \\ 0 & \textit{if } x < 0 \end{cases}$$

Perceptron: Learning Algorithm

- **Goal**: we want to define a learning algorithm for the weights in order to compute a mapping from the inputs to the outputs;
- **Example**: two class character recognition problem.
  - **Training set**: set of images representing either the character 'a' or the character 'b' (supervised learning);
  - **Learning Task**: Learn the weights so that when a new unlabelled image comes in, the network can predict its label.
  - Settings:
    Class 'a' ➔ 1 (class C1)
    Class 'b' ➔ 0 (class C2)
    n input units (intensity level of a pixel)
    1 output unit

The perceptron needs to learn $f : \Re^n \rightarrow \{0,1\}$

Perceptron: Learning Algorithm

The algorithm proceeds as follows:

- Initial random setting of weights;
- The input is a random sequence $\{\boldsymbol{x}_k\}_{k\in\aleph}$
- For each element of class C1, if output = 1 (correct) do nothing, otherwise update weights;
- For each element of class C2, if output = 0 (correct) do nothing, otherwise update weights.

Perceptron: Learning Algorithm

A bit more formally:

$\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ $\qquad$ $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$

θ **:** Threshold of the output unit

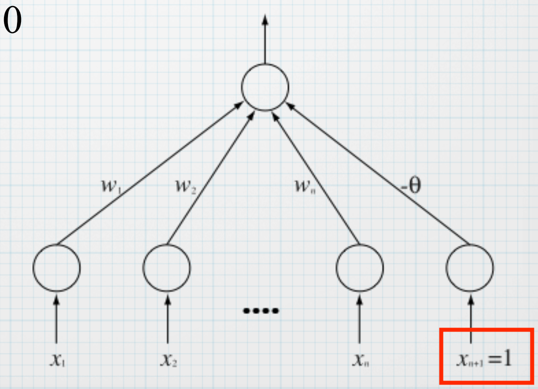
$$\boldsymbol{w}\boldsymbol{x}^T = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

Output is 1 if $\boldsymbol{w}\boldsymbol{x}^T - \theta \geq 0$

To eliminate the explicit dependence on θ :

Output is 1 if:

$$\hat{\boldsymbol{w}}\hat{\boldsymbol{x}}^T = \sum_{i=1}^{n+1} w_i x_i \geq 0$$



Perceptron: Learning Algorithm

- We want to learn values of the weights so that the perceptron correctly discriminate elements of C1 from elements of C2:
- Given x in input, if x is classified correctly, weights are unchanged, otherwise:

$$\boldsymbol{w}' = \begin{cases} \boldsymbol{w} + \boldsymbol{x} & \text{if an element of class } C_1 \text{ (1) was classified as in } C_2 \\ \boldsymbol{w} - \boldsymbol{x} & \text{if an element of class } C_2 \text{ (0) was classified as in } C_1 \end{cases}$$

Perceptron: Learning Algorithm

$$\boldsymbol{w}' = \begin{cases} \boldsymbol{w} + \boldsymbol{x} & \text{if an element of class } C_1 \text{ (1) was classified as in } C_2 \\ \boldsymbol{w} - \boldsymbol{x} & \text{if an element of class } C_2 \text{ (0) was classified as in } C_1 \end{cases}$$

- 1st case: $\boldsymbol{x} \in C_1$ ***and was classified in*** C_2

The correct answer is 1, which corresponds to: $\hat{\boldsymbol{w}}\hat{\boldsymbol{x}}^T \geq 0$

We have instead: $\hat{\boldsymbol{w}}\hat{\boldsymbol{x}}^T < 0$

We want to get closer to the correct answer: $\boldsymbol{w}\boldsymbol{x}^T < \boldsymbol{w}'\boldsymbol{x}^T$

$$\boldsymbol{w}\boldsymbol{x}^T < \boldsymbol{w}'\boldsymbol{x}^T \quad \text{iff} \quad \boldsymbol{w}\boldsymbol{x}^T < (\boldsymbol{w} + \boldsymbol{x})\boldsymbol{x}^T$$

$$(\boldsymbol{w} + \boldsymbol{x})\boldsymbol{x}^T = \boldsymbol{w}\boldsymbol{x}^T + \boldsymbol{x}\boldsymbol{x}^T = \boldsymbol{w}\boldsymbol{x}^T + \|\boldsymbol{x}\|^2$$

$$\text{because } \|\boldsymbol{x}\|^2 \geq 0, \text{ the condition is verified}$$

Perceptron: Learning Algorithm

$$w' = \begin{cases} w + x & \text{if an element of class } C_1 \text{ (1) was classified as in } C_2 \\ w - x & \text{if an element of class } C_2 \text{ (0) was classified as in } C_1 \end{cases}$$

- 2nd case: $x \in C_2$ *and was classified in* C_1

The correct answer is 0, which corresponds to: $\hat{w}\hat{x}^T < 0$
We have instead: $\hat{w}\hat{x}^T \geq 0$

We want to get closer to the correct answer: $wx^T > w'x^T$

$$wx^T > w'x^T \quad \text{iff} \quad wx^T > (w - x)x^T$$

$$(w - x)x^T = wx^T - xx^T = wx^T - \|x\|^2$$

$$\text{because } \|x\|^2 \geq 0, \text{ the condition is verified}$$

The previous rule allows the network to get closer to the correct answer when it performs an error.

Perceptron: Learning Algorithm

- **In summary**:

1. A random sequence $x_1, x_2, \cdots, x_k, \cdots$ is generated such that $x_i \in C_1 \cup C_2$
2. If x_k is correctly classified, then $w_{k+1} = w_k$ otherwise

$$w_{k+1} = \begin{cases} w_k + x_k & \text{if } x_k \in C_1 \\ w_k - x_k & \text{if } x_k \in C_2 \end{cases}$$

Perceptron: Learning Algorithm

Does the learning algorithm converge?

Convergence theorem: Regardless of the initial choice of weights, if the two classes are linearly separable, i.e. there exist $\boldsymbol{w}$ s.t.

$$\begin{cases} \hat{\boldsymbol{w}}\hat{\boldsymbol{x}}^T \geq 0 \;\; \boldsymbol{if} \;\; \boldsymbol{x} \in C_1 \\ \hat{\boldsymbol{w}}\hat{\boldsymbol{x}}^T < 0 \;\; \boldsymbol{if} \;\; \boldsymbol{x} \in C_2 \end{cases}$$

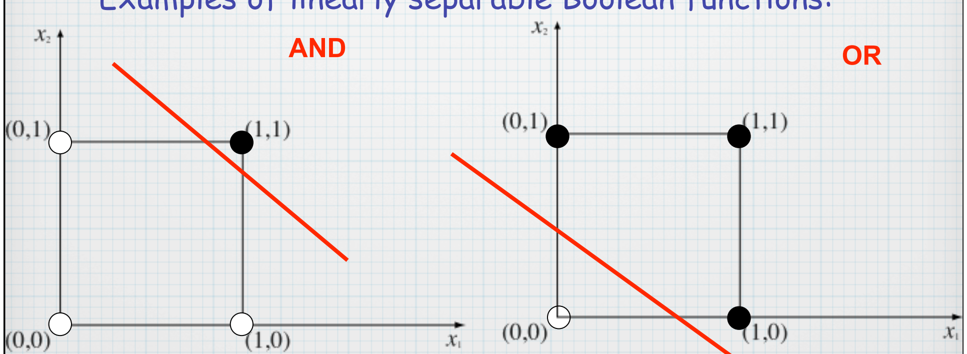
then the learning rule will find such solution after a finite number of steps.

Representational Power of Perceptrons

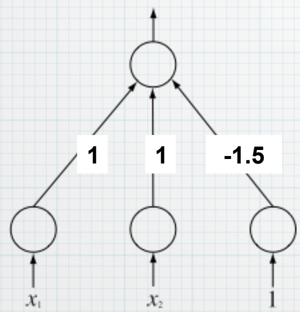
- Marvin Minsky and Seymour Papert, "Perceptrons" 1969:

  "The perceptron can solve only problems with linearly separable classes."

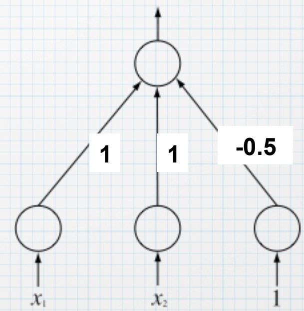
- Examples of linearly separable Boolean functions:



Representational Power of Perceptrons



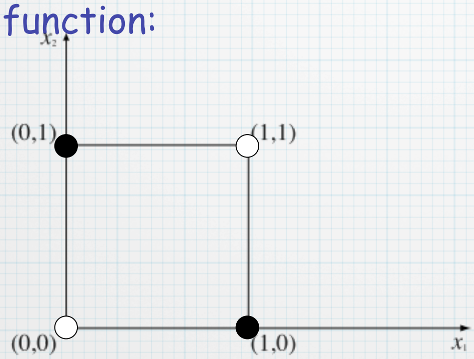
Perceptron that computes the AND function



Perceptron that computes the OR function

Representational Power of Perceptrons

- Example of a non linearly separable Boolean function:



EX-OR

The EX-OR function **cannot** be computed by a perceptron