COM 424 E: NEURAL NETWORKS

Lecture 03: The Perceptron

The Perceptron

- A single artificial neuron that computes its weighted input and uses a threshold activation function.
- It effectively separates the input space into two categories by

the hyperplane:

$w^T x + b = 0$

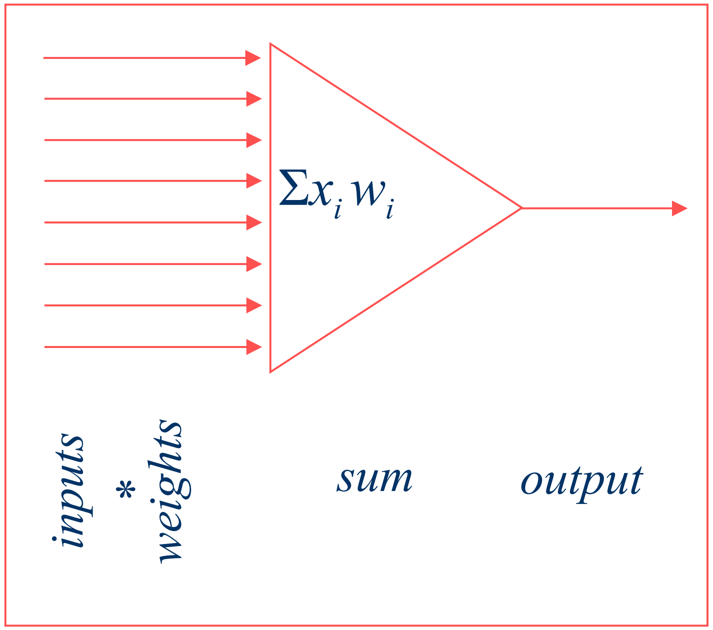
Perceptron: Applications

- The perceptron is used for classification: classify correctly a set of examples into one of the two classes C_1, C_2:

- ***If the output of the perceptron is +1 then the input is assigned to class C_1***
- ***If the output is -1 then the input is assigned to C_2***

The Perceptron

The *Perceptron* can automatically learn to categorise or classify input vectors into types.



It obeyed the following rule:

If the sum of the weighted inputs exceeds a threshold, output 1, else output -1.

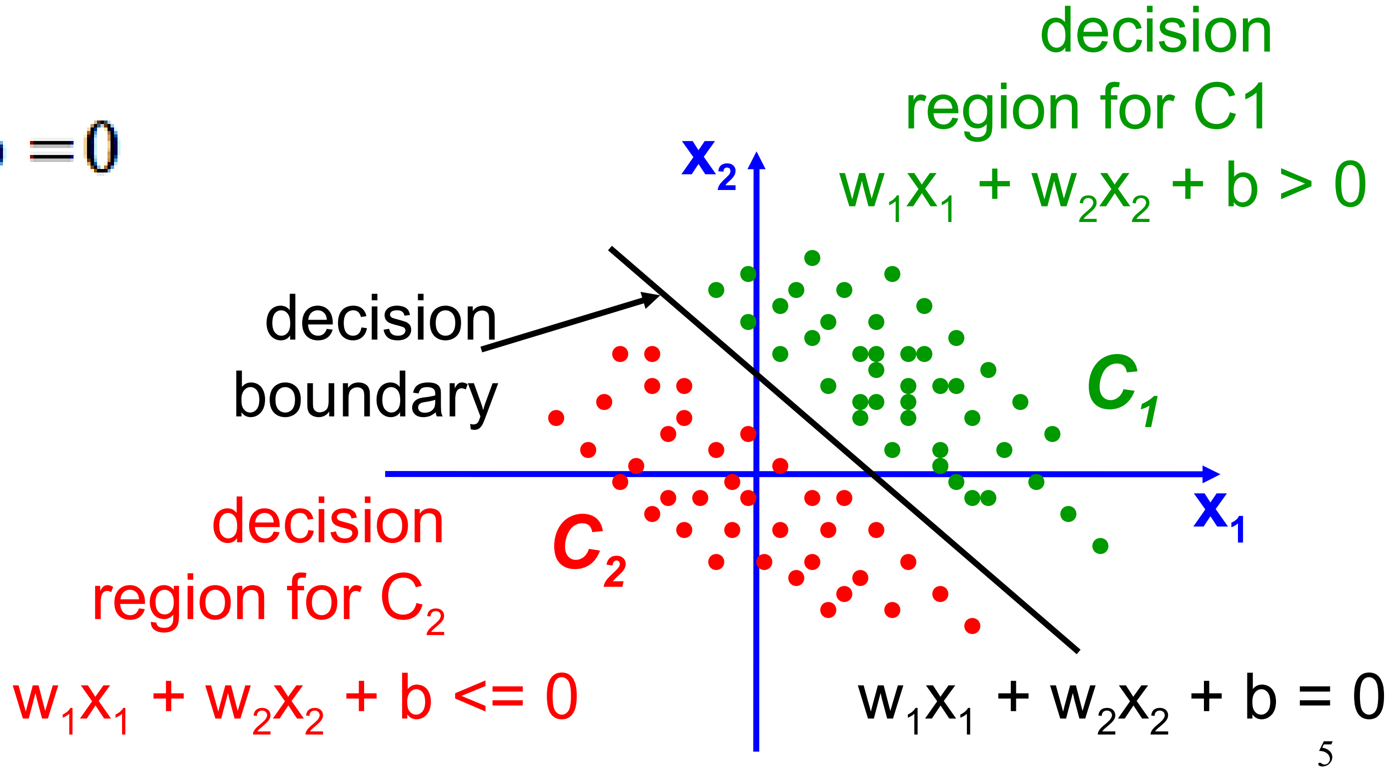
$$1 \text{ if } \Sigma\ input_i * weight_i > threshold$$

$$-1 \text{ if } \Sigma\ input_i * weight_i < threshold$$

Perceptron: Classification

The equation below describes a hyperplane in the input space. This hyperplane is used to separate the two classes C1 and C2

$$\sum_{i=1}^{m} w_i x_i + b = 0$$



decision region for C1
$w_1x_1 + w_2x_2 + b > 0$
x_2
decision boundary
C_1
x_1
C_2
decision region for C_2
$w_1x_1 + w_2x_2 + b <= 0$
$w_1x_1 + w_2x_2 + b = 0$

Perceptron: Limitations

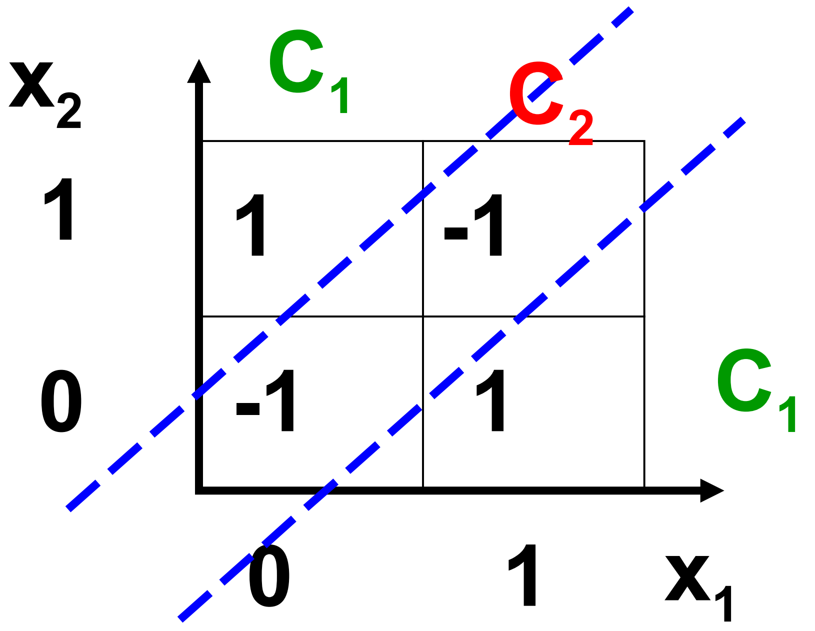
- The perceptron can only model linearly separable functions.
- The perceptron can be used to model the following Boolean functions:
- AND
- OR
- COMPLEMENT

But it cannot model the XOR. Why?

Perceptron: Limitations

The XOR is not linear separable

It is impossible to separate the classes C_1 and C_2 with only one line



Perceptron: Learning Algorithm

- Variables and parameters

  $\mathbf{x}(n)$ = input vector

  $= [+1, x_1(n), x_2(n), \ldots, x_m(n)]^T$

  $\mathbf{w}(n)$ = weight vector

  $= [b(n), w_1(n), w_2(n), \ldots, w_m(n)]^T$

  $b(n)$ = bias

  $y(n)$ = actual response

  $d(n)$ = desired response

  , = learning rate parameter

The fixed-increment learning algorithm

- Initialization: set $\mathbf{w}(0) = 0$
- Activation: activate perceptron by applying input example (vector $\mathbf{x}(n)$ and desired response $d(n)$)
- Compute actual response of perceptron:

$$y(n) = \mathrm{sgn}[\mathbf{w}^T(n)\mathbf{x}(n)]$$

- Adapt weight vector: if $d(n)$ and $y(n)$ are different then

$$\mathbf{w}(n+1) = \mathbf{w}(n) + ,\quad [d(n) - y(n)]\mathbf{x}(n)$$

$$\text{Where } d(n) = \begin{cases} +1 & \text{if } \mathbf{x}(n) \square C_1 \\ -1 & \text{if } \mathbf{x}(n) \square C_2 \end{cases}$$

- Continuation: increment time step n by 1 and go to Activation step

Example

Consider a training set $\{C_1 , C_2 \}$ where:

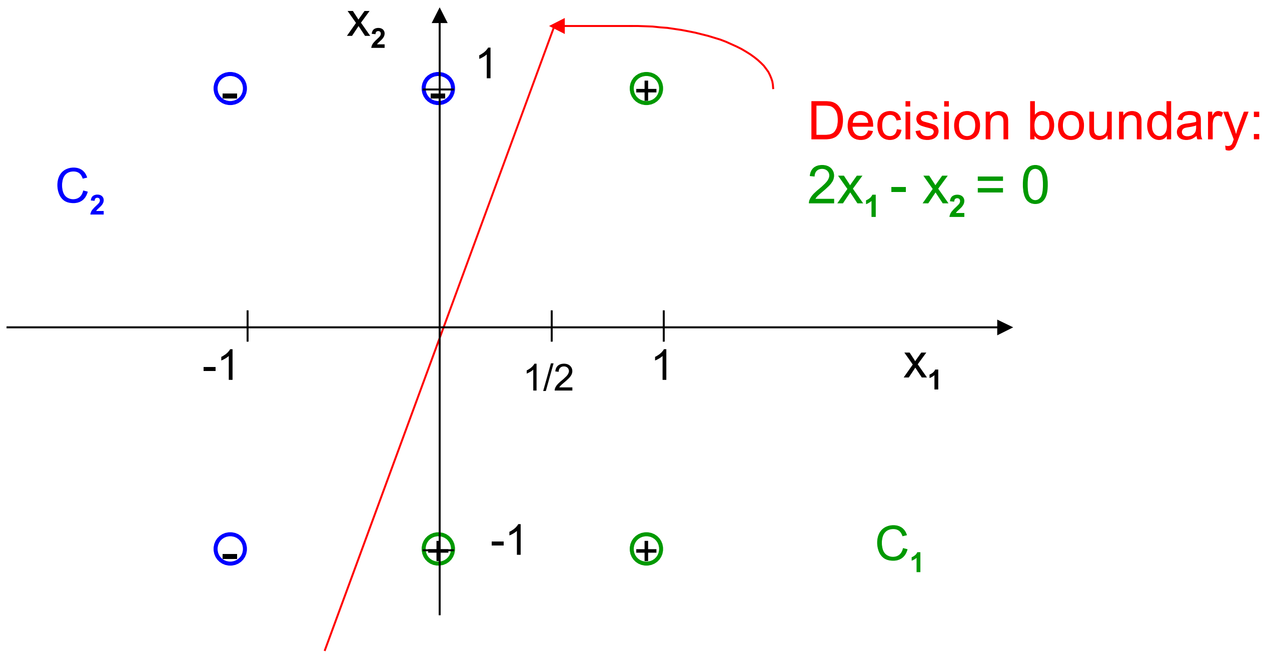
$C_1 = \{(1,1), (1, -1), (0, -1)\}$ elements of class 1

$C_2 = \{(-1,-1), (-1,1), (0,1)\}$ elements of class -1

Use the perceptron learning algorithm to classify these examples.

$\mathbf{w}(0) = [1, 0, 0]^T$

Example



Beyond perceptrons

- Need to learn the features, not just how to weight them to make a decision.
- Need to make use of recurrent connections, especially for modeling sequences.
  - » The network needs a memory (in the activities) for events that happened some time ago, and we cannot easily put an upper bound on this time.
- Need to learn representations without a teacher.
  - » This makes it much harder to define what the goal of learning is.