Perceptrons

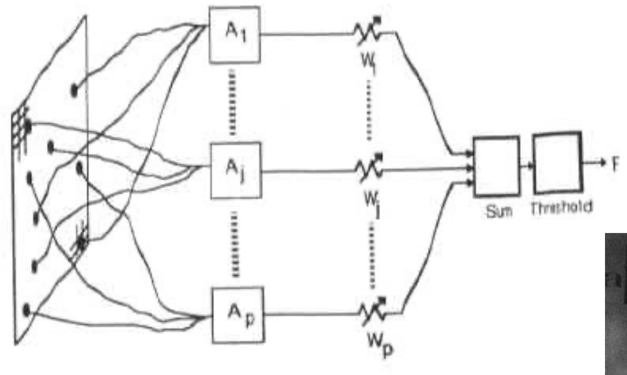
Stuart Russel and Peter Norvig, *Artificial Intelligence: A modern approach*, 4th edition, Pearson Education, 2020
http://aima.cs.berkeley.edu/ – chapter 18

Michael Negnevitsky, Artificial Intelligence: A guide to Intelligent Systems, 2nd edition, Pearson Education, 2004 – chapter 6

Ernesto Costa e Anabela Simões, *Inteligência Artificial: Fundamentos e Aplicações* (2ª edição) FCA, Set. 2008 – cap. 5

1

The Rosenblatt's perceptron, 1957



2

A supervised learning rule

- Given
  - a finite training set with two linearly separable classes
    $T = \{(x_i, d_i) \mid i=1,\ldots,M\}$ with d_i in $\{0,1\}$
  - and a learning rate η, s.t. $0 < \eta \leq 1$

Estimate the weights W that correctly classify the elements of T

3

A supervised learning rule

- Learning algorithm

1. k=0
2. init weights W[k=0] with *small* random values
3. Repeat until convergence
   1. Randomly select a training example from T, say (x_j, d_i)
   2. Compute the output: o_j =perceptron(x_j)
   3. Upate the weights: $W[k+1] = W[k] + \eta\,(d_i - o_j)\,x_j$
   4. k=k+1

4

Rosenblatt convergence theorem

- The algorithm converges to the correct classification
  - if the training data is linearly separable
  - and η is sufficiently small
  - The data set is finite, and
  - patterns are random but repeatidly presented

i.e., after N iterations

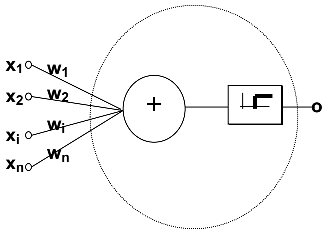
W[N] = W[N+1] = W[N+2] = ...

where W is the hiperplan that correctly separated the classes

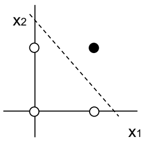
16-Nov-23 http://tiny.cc/ia2023-24

5

TLU



Example: **AND**



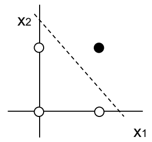
$$x_2 = -\frac{w_1}{w_2}x_1 + \frac{\theta}{w_2}$$

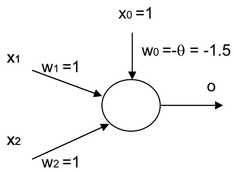
16-Nov-23 http://tiny.cc/ia2023-24

6

Examples of linearly separable classes

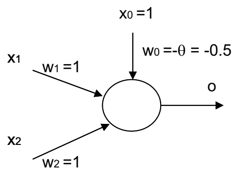
AND





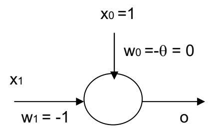
OR

x_1	x_2	o
0	0	0
0	1	1
1	0	1
1	1	1



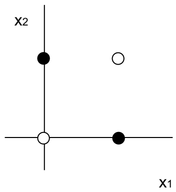
NOT

x_1	o
0	1
1	0



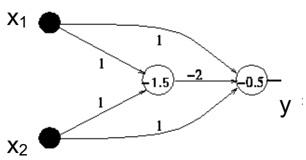
XOR: A non-linearly separable problem

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



No line can separate these two classes (Minsky, Papert, 1969)

XOR using 2 TLU:



How to learn the weights?

Two possible approaches:
- Unsupervised learning (ex: Hebb rule)
- Supervised learning (ex: Backprogation)

9

Hebb's postulate

"When an axon of cell A is near enough to excite cell B or repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased".

Donald O. Hebb (1949)

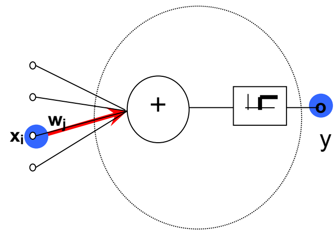
"*cells that fire together, wire together*"

16-Nov-23 http://tiny.cc/ia2023-24

10

Hebbian Learning

- Consider a weight w_j of a neuron with pre-and post-synaptic signal denoted by x_j and y respectively. The adjustment to the weight w_j at a given time step is given by:

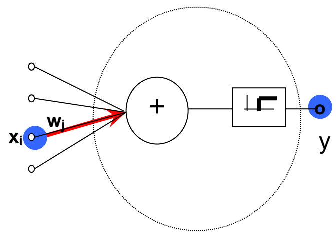


$$\Delta w_j = F(y, x_j)$$

where F is a function of both signals.

11

Hebbian Learning – basic form



- Plain Hebb rule: (AKA the *activity product rule*):

$$\Delta w_j = \eta \; y \, x_j$$

where η is a *learning rate* parameter

12

The δ rule

Stuart Russel and Peter Norvig, *Artificial Intelligence: A modern approach*, 2nd edition, Pearson Education, 2003
http://aima.cs.berkeley.edu/ – chapter 18

Michael Negnevitsky, Artificial Intelligence: A guide to Intelligent Systems, 2nd edition, Pearson Education, 2004 – chapter 6

Ernesto Costa e Anabela Simões, *Inteligência Artificial: Fundamentos e Aplicações* (2ª edição) FCA, Set. 2008 – cap. 5

13

Recalling the gradient method

14

Problem formulation

- Find the x* such that minimizes

$$f : R^n \to R$$

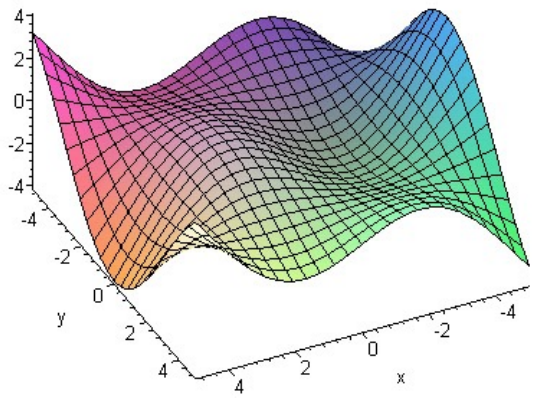
where f is a smooth C^1 function

The gradient

$$f : R^n \to R$$

$$\nabla f(x_1,\ldots,x_n) := \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)^T$$

Application example



$$f := (x, y) \rightarrow \cos\left(\frac{1}{2}x\right)\cos\left(\frac{1}{2}y\right)x$$

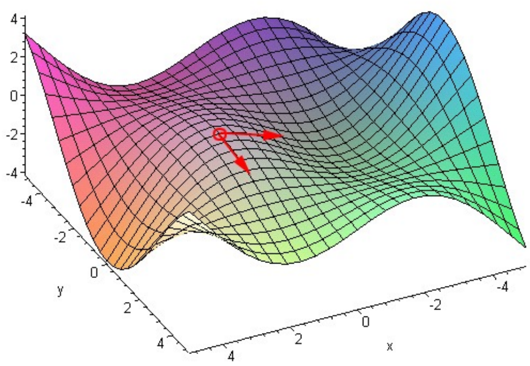
16-Nov-23 http://tiny.cc/ia2023-24

17

Directional derivatives

$$\frac{\partial f(x,y)}{\partial y}$$

$$\frac{\partial f(x,y)}{\partial x}$$

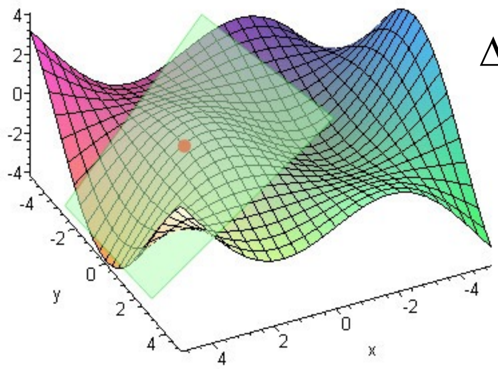


16-Nov-23 http://tiny.cc/ia2023-24

18

The gradient properties

- The gradient defines a (hyper) plane approximating the function infinitesimally



$$\Delta z = \frac{\partial f}{\partial x} \cdot \Delta x + \frac{\partial f}{\partial y} \cdot \Delta y$$

19

The gradient properties

- **Proposition**: let $f : R^n \to R$ be a smooth C^1 function around p, if f has local minimum (maximum) at p then,

$$(\nabla f)_p = \overline{0}$$

20

Gradient (steepest descent) algorithm

$$x_{i+1} = x_i + \eta \cdot h_i$$

i=0

$x_0 \in R^n$

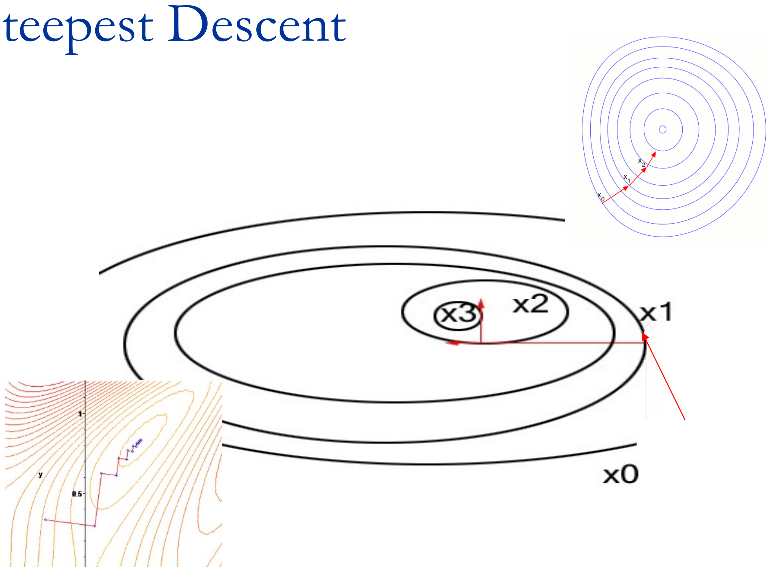
Repeat until $\nabla f(x_i) = 0$

1. compute *search direction* $h_i = -\nabla f(x_i)$
2. update

16-Nov-23 http://tiny.cc/ia2023-24

21

Steepest Descent



16-Nov-23 http://tiny.cc/ia2023-24

22

Summary

- Rosenblatt Perceptron
  - Limits of the Perceptron
- The Hebb rule
- Recalling the gradient method