BIOLOGICALLY INSPIRED AI

Universidad Carlos III de Madrid

AI

Outline

Motivation

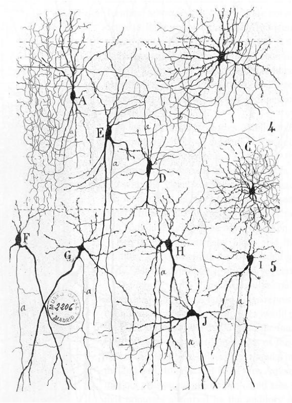
- Nature has always served as a source of inspiration for engineers and scientists
- The best problem solvers known in nature are:
  - the (human) brain
  - the evolution mechanism that created the human brain (Darwin)

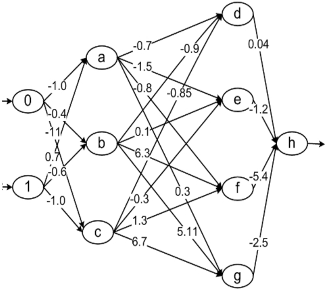
Motivation

- Nature has always served as a source of inspiration for engineers and scientists
- The best problem solvers known in nature are:
  - the (human) brain
  - the evolution mechanism that created the human brain (Darwin)
- Alternative 1: neurocomputing
- Alternative 2: evolutionary computation

Outline

Neural network



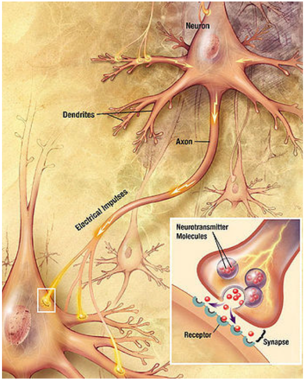


The brain

- Ten billion (10^{10}) neurons
- Neuron switching time $\sim 10^{-3}$ secs
- Face Recognition: 0.1 secs
- On average, each neuron has several thousand connections
- Hundreds of operations per second
- High degree of parallel computation
- Distributed representations
- Die off frequently (never replaced, though recently disputed)
- Compensated for problems by massive parallelism

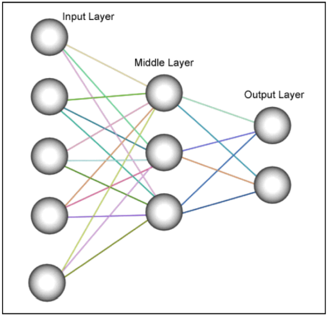
Neurons

- A neuron only fires if its input signal exceeds a certain amount (the threshold) in a short time period
- Synapses can be either excitatory or inhibitory
- A neuron has a cell body, a branching input structure (dendrite) and a branching output structure (axon)

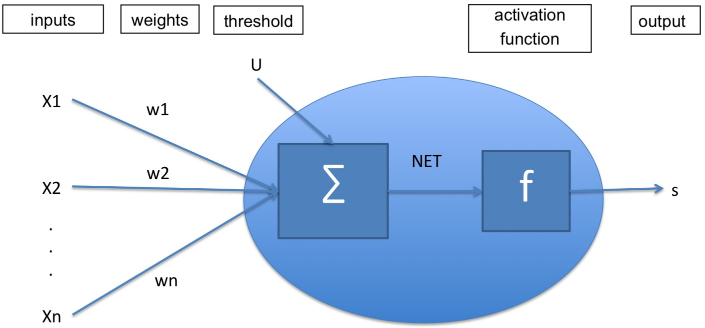


Artificial neural networks

- Inspired by neurobiology
- Many simple neuron-like units
- Many weighted interconnections among units
- Highly parallel, distributed processing
- Learning: by tuning the connection weights

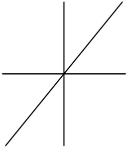


Neuron structure



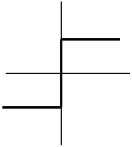
- $NET = U + w_1 * x_1 + w_2 * x_2 + \ldots + w_n * x_n = \sum_{i=0}^{n} w_i x_i$
- $S = f(NET)$

Types of functions



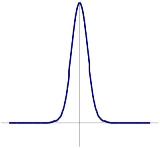
Linear function

$f(x)=x$



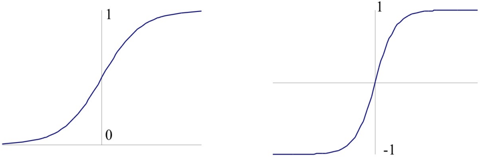
Threshold function

$$f(x) = \begin{cases} f1 \;/\; x > 0 \\ -f1 \;/\; x \le 0 \end{cases}$$



Gaussian function

$$f(x) = e^{\frac{-x^2}{2}}$$



Sigmoid functions

Function (0,1)

$$f(x) = \frac{1}{1+e^{-x}}$$

Function (-1,1)

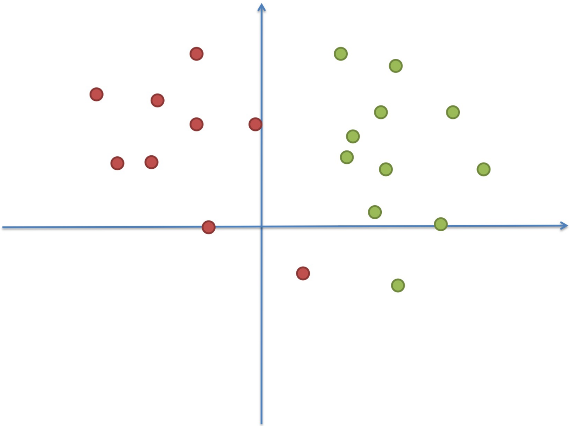
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Currently, $f(x) = max(0, x)$

Perceptron [Rosenblatt, 1958]

- Simplest form of neural network
- Inspired on McCulloch-Pitts cells and on studies on frogs vision system
- Supervised learning
- Linear classification: given a set of training instances (patterns), determine the discriminating hyperplane

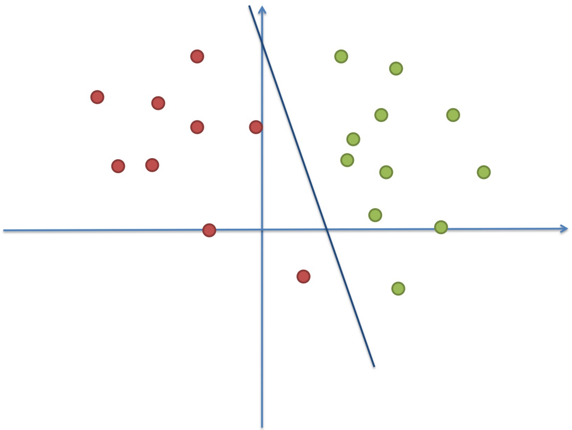
Objective. Perceptron [Rosenblatt, 1958]



Examples: vector (attributes)+class

$x = \langle \vec{x}, c(x) \rangle = \langle (x_1, x_2, \ldots, x_n), c(x) \rangle$

Objective. Perceptron [Rosenblatt, 1958]



Hyperplane (2 dimensions/attributes): $w_1 x_1 + w_2 x_2 + U = 0$

Classification in Perceptron

- Given an input vector, $\vec{x}$

$$NET(\vec{x}) = U + \sum_{i=1}^{n} w_i x_i = \sum_{i=0}^{n} w_i x_i$$

$$f(\vec{x}) = \begin{cases} 1 & \text{if } NET(\vec{x}) > 0 \\ -1 & \text{otherwise} \end{cases}$$

- If $f(\vec{x}) = 1$, it belongs to class +
- If $f(\vec{x}) = -1$, it belongs to class -

Learning in Perceptron

- Weights are randomly initialized w_i and $U = w_0$
- Repeat until termination

  At each iteration, weights are modified such that hyperplane completely separates examples

  - an example is chosen $\langle \vec{x}, c(x) \rangle = \langle (x_1, x_2, \ldots, x_n), c(x) \rangle$ where $c(x)$ is the class (1 or -1)
  - net output is computed $y(\vec{x}) = f(U + \sum_{i=1}^{n} w_i x_i)$
  - if $y(\vec{x}) \neq c(x)$ weights (and threshold) are updated

Learning rule in Perceptron

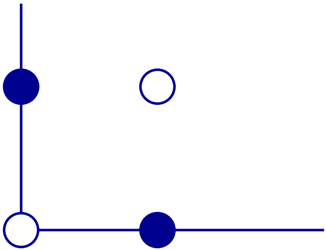
- If $U = w_0$ and $\vec{w} = (w_0, w_1, \ldots, w_n)$
- Weights are updated:

$$\vec{w}_{t+1} = \vec{w}_t + \sum_{x \text{ misclassified}} \begin{cases} \vec{x} & \text{if } y(\vec{x}) = -1 \text{ and } c(x) = +1 \\ -\vec{x} & \text{if } y(\vec{x}) = +1 \text{ and } c(x) = -1 \end{cases}$$

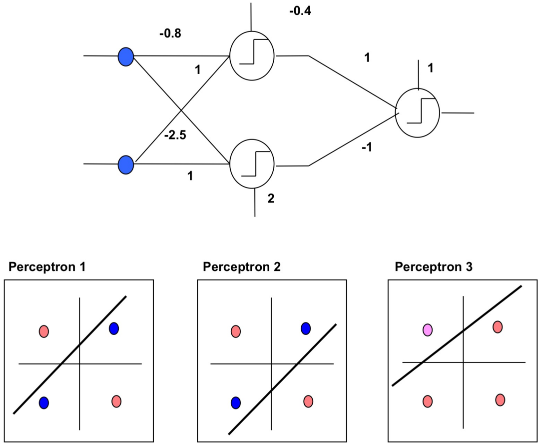
$$\vec{w}_{t+1} = \vec{w}_t + \sum_{x \text{ misclassified}} c(x)\vec{x}$$

What can a Perceptron learn?

- Classes that can be separated by an hyperplane
- Example of where it fails (XOR)



Multi-layer Perceptron



Classification in multi-layer Perceptron

- If a multi-layer perceptron has three layers: n input neurons, m output neurons and r neurons in the middle (hidden) layer
- The input layer receives a training pattern: $< \vec{x}, c(x) >, i = 1..n$
- The hidden layer computes the activation of each neuron:

$$b_j = f(U_j + \sum_{i=1}^{n} w_{ij} x_i), j = 1..r$$

- The output layer computes the activation of each neuron:

$$y_k = f(V_k + \sum_{j=1}^{r} w_{jk} b_j), k = 1..m$$

- We can extend this scheme to more layers...

Learning rule in multi-layer (backpropagation)

- On output layer:

$$\Delta w_{t+1} = \alpha(c(x) - y(\vec{x}))y(\vec{x})(1 - y(\vec{x}))\vec{x}$$

- On hidden layers:

$$\delta_k = o_k(1 - o_k) \sum_{j \in C(k)} w_j \delta_j$$

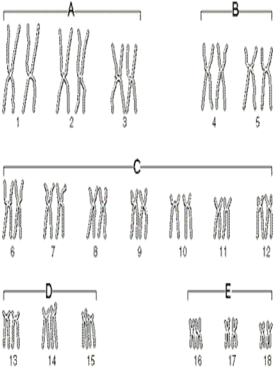
  - o_k output of neuron
  - $\delta_j = (c(x) - y(\vec{x}))y(\vec{x})(1 - y(\vec{x}))$ error on j neuron
  - $c(x)$ expected output (class of example)
  - $C(k)$ set of output neurons connected to k
  - α learning rate

Summary

- Brain-inspired machine learning
- Computation is collective, asynchronous, and parallel
- Fault tolerant, redundancy, and sharing of responsibilities
- Allows for uncertainty in examples
- Widely used

Outline

Biological inspiration



1000010101010101010
0010100101001010101
1001010101001011000
1010100101011010100
0010010101010101010
1010010100101001010
0101000100101010010
10100111110010100110
1010010010100101010
0010101001010111100
1001001001010101001
0010001010011000100

Natural evolution

Darwinian Evolution: Survival of the fittest

- All environments have finite resources
- They can only support a limited number of individuals
- Lifeforms have basic instinct/lifecycles geared towards reproduction
- Therefore, some kind of selection is inevitable
- Those individuals that compete for the resources most effectively have increased chance of reproduction

Evolutionary Computation metaphor

Evolution	**Problem solving**
Environment	Problem
Individual	Solution
Fitness	Quality

- Fitness: chances of survival and reproduction

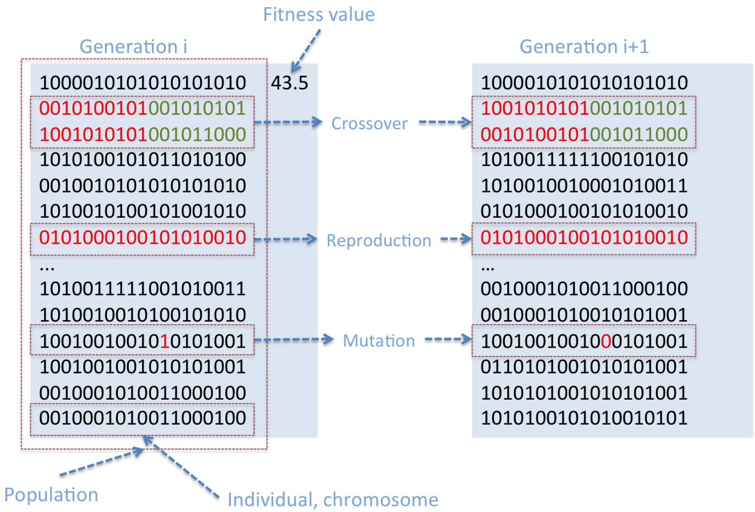
Representation spaces

- Candidate solutions (individuals) exist in phenotype space
  - search in a grid: each individual is a sequence of movement steps to go from an initial position to a goal
  - up, up, right, right, right
- They are encoded in chromosomes, which exist in genotype space
  - up: 00, down: 01, right: 10, left: 11
  - up, up, right, right, right: 0000101010
- Transformations:
  - Encoding: phenotype $\rightarrow$ genotype (not necessarily one to one)
  - Decoding: genotype $\rightarrow$ phenotype (must be one to one)

Representation

- Each EC iteration represents a generation
- At each generation, there is one population
- A population is composed of a set of p individuals
- Each individual represents a solution to the problem
- There is a set of genetic operators that transform individuals and generate the next population
- A fitness function evaluates each individual for its chances to survive and/or reproduce

Representation and evolution



Simple algorithm

Algorithm GA (F,TC,p,r,m)

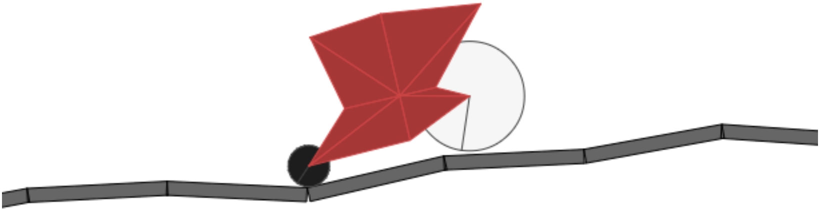
F: Fitness function
TC: Termination condition
p: number of individuals in population
r: replacement ratio
m: mutation ratio

$P \leftarrow$ generate p initial individuals, computing their $F(\cdot)$
While not TC
  select $P_s \leftarrow (1-r)p$ individuals according to their $F(\cdot)$
  crossover: randomly $(F(\cdot))$ select rp pairs from P_s
    $\forall h_1, h_2$, cross h_1 and h_2
    add new individuals to P_s
  mutate: change a random bit in $m\%$ of individuals in P_s
  update $P \leftarrow P_s$
  evaluate $\forall h \in P$, compute $F(h)$
Return best h $(F(\cdot))$

Alternative techniques

- Genetic algorithms (Holland): bit strings
- Evolutionary programs (Michalewicz): data structures
- Evolutionary strategies (Rechenberg and Schwefel): real numbers
- Evolutionary programming (Fogel): finite state automata
- Genetic programming (Koza): trees
- Classifier systems (Holland): rules

Example of cars configuration



http://rednuht.org/genetic_cars_2/

Bibliography

- R. Dawkins, “The Selfish Gene”, Oxford University Press, (2006)
- D.E. Goldberg, “Genetic Algorithms in Search, Optimization, and Machine Learning”. Addison Wesley, Reading Massachusetts, (1989)
- J.Holland, “Adaptation in Natural an Artificial Systems”. University of Michigan Press, Ann Arbor, (1975)
- J. R. Koza, “Genetic Programming: On the Programming of Computers by Means of Natural Selection”, MIT Press, (1992)
- M. Mitchell, “An Introduction to Genetic Algorithms”, MIT Press, Massachusetts, (1996)

References

F. Rosenblatt.
The perceptron: A probabilistic model for information storage and organization in the brain.
Psychological Review, 65(6):386–408, 1958.