BABEŞ-BOLYAI UNIVERSITY
Faculty of Computer Science and Mathematics

ARTIFICIAL INTELLIGENCE

Intelligent systems

Machine learning

Artificial Neural Networks

Topics

A. Short introduction in Artificial Intelligence (AI)

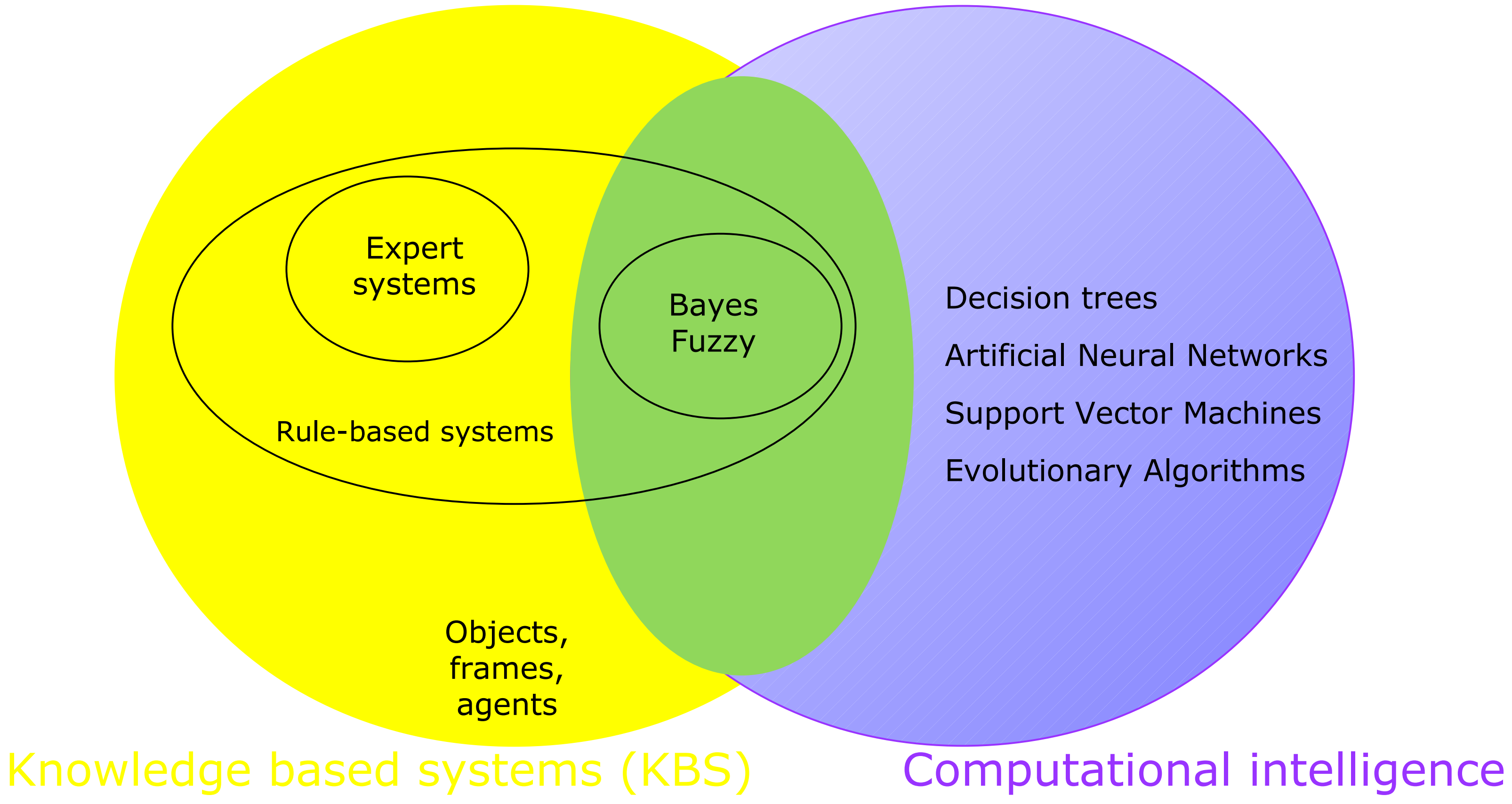
B. Solving search problems
   A. Definition of search problems
   B. Search strategies
      A. Uninformed search strategies
      B. Informed search strategies
      C. Local search strategies (Hill Climbing, Simulated Annealing, Tabu Search, Evolutionary algorithms, PSO, ACO)
      D. Adversarial search strategies

C. **Intelligent systems**
   A. Rule-based systems in certain environments
   B. Rule-based systems in uncertain environments (Bayes, Fuzzy)
   C. **Learning systems**
      A. Decision Trees
      B. **Artificial Neural Networks**
      C. Support Vector Machines
      D. Evolutionary algorithms
   D. Hybrid systems

Useful information

- Chapter VI (19) of *S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995*
- Chapter 8 of *Adrian A. Hopgood, Intelligent Systems for Engineers and Scientists, CRC Press, 2001*
- Chapters 12 and 13 of *C. Groşan, A. Abraham, Intelligent Systems: A Modern Approach, Springer, 2011*
- Chapter V of *D. J. C. MacKey, Information Theory, Inference and Learning Algorithms, Cambridge University Press, 2003*
- Chapter 4 of *T. M. Mitchell, Machine Learning, McGraw-Hill Science, 1997*

Intelligent systems



Intelligent systems – Machine Learning

- Typology
  - Experience criteria:
    - Supervised learning
    - Unsupervised learning
    - Active learning
    - Reinforcement learning
  - Algorithm criteria
    - Decision trees
    - **Artificial Neural Networks**
    - Evolutionary Algorithms
    - Support Vector Machines
    - Hidden Markov Models

Intelligent Systems - ANNs

- Artificial Neural Networks (ANNs)
  - Aim
  - Definition
  - Solved problems
  - Characteristics
  - Example
  - Design
  - Evaluation
  - Typology

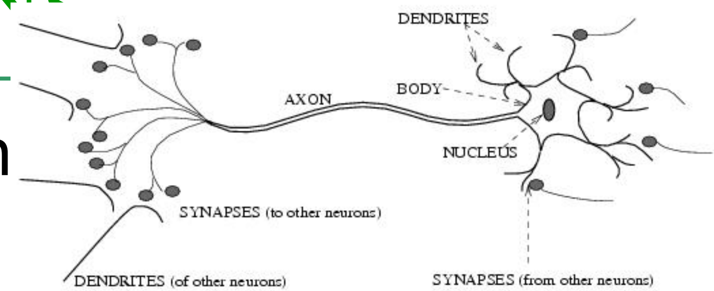
Intelligent Systems - ANNs

- Aim
  - Binary classification for any input data (discrete or continuous)
    - Data can be separated by:
      - A line → $ax + by + c = 0$ (if $m = 2$)
      - A plan → $ax + by + cz + d = 0$ (if $m = 3$)
      - A hyper plan $\sum a_i x_i + b = 0$ (if $m > 3$)
    - How do we identify the optimal values of a, b, c, d, a_i?
      - Artificial Neural Networks (ANNs)
      - Support Vector Machines (SVMs)
  - Why ANN?
  - Who does the brain learn?

Intelligent Systems - ANNs

- Aim → why ANN?
  - Some tasks can be easy done by humans, but they are difficult to be encoded as algorithms
    - Shape recognition
      - Old friends
      - Handwritten
      - Voice
    - Rational processes
      - Car driving
      - Piano playing
      - Basketball playing
      - Swimming
  - Such tasks are to difficult to be formalized and done by a rational process

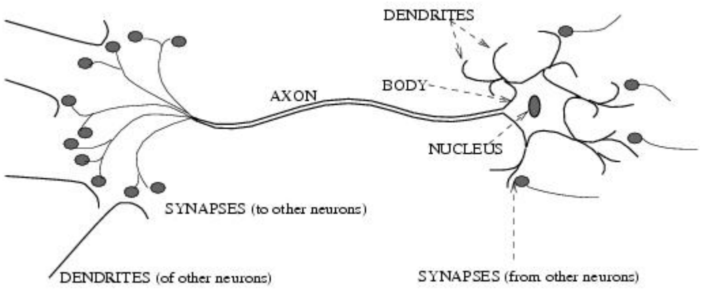
Intelligent Systems - ANNs



- Aim → how does the brain learn
  - Human brain – components
    - ~10.000.000.000 of neurons connected through synapses
    - Each neuron
      - Has a body (soma_, an axon and more dendrites
      - Can be in a given state
        - Active state – if the input information is over a given stimulation threshold
        - Passive state – otherwise
    - Synapse
      - Link between the axon of a neuron and the dendrites of other neurons
      - Take part to information exchange between neurons
      - 5.000 connections/neuron (average)
    - During a life, new connections can appear

Intelligent Systems - ANNs

- Aim → how does the brain learn?
  - How does learning (information processing) take place?
    - Useful connections become permanent (others are eliminated)
    - The brain is interested about news
    - Models for information processing
      - Learning
      - Storing
      - Memorizing
    - Memory
      - Typology
        - Short time memory
        - Immediately → 30 sec.
        - Working memory
        - Long term memory
      - Capacity
        - Increasing along life
        - Limited → learning a poetry strophe by strophe
      - Influenced by emotional states
    - Brain
      - Neuron network
      - Complex system, non-linear and parallel that processes information
    - Information is stored and processed by the entire network, not only by a part of network → global information and processing
    - Basic characteristic of a neural network → learning → artificial neural network

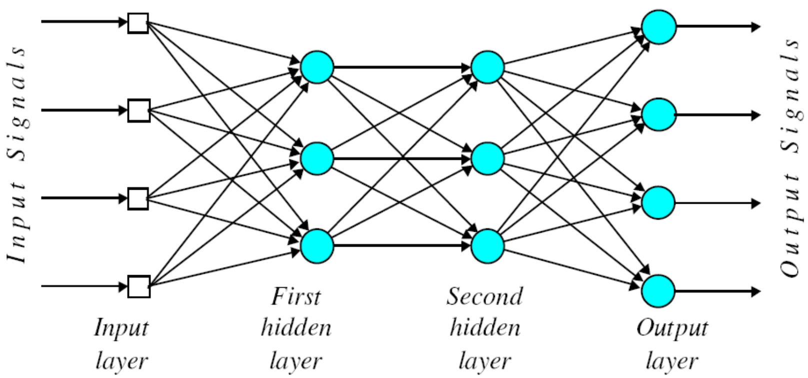


Intelligent Systems - ANNs

- Definition
  - What is an ANN?
  - Biological NN vs. artificial NN
  - How does network learn?

Intelligent Systems - ANNs

- Definition → what is an ANN?
  - A structure similar to a biological NN
  - A set of nodes (units, neurons, processing elements) located in a graph with more layers
    - Nodes
      - Have inputs and outputs
      - Perform a simple computing through an activation function
      - Connected by weighted links
        - Links between nodes give the network structure
        - Links influence the performed computations
    - Layers
      - Input layer
        - Contains m nodes (m - # of attributes of a data)
      - Output layer
        - Contains r nodes (r - # of outputs)
      - Intermediate layers
        - Different structures
        - Different sizes



Intelligent Systems - ANNs

- Definition → biological NNs vs. artificial NNs

BNN	ANN
Soma	Node
Dendrite	Input
Axon	Output
Activation	Processing
Synapse	Weighted connection

- Definition → how does network learn?
  - A training data set of n data

$$((x_{p1}, x_{p2}, \ldots, x_{pm}, y_{p1}, y_{p2}, \ldots, y_{pr},))$$

  with $p = 1, 2, \ldots, n$, m – #attributes, r – #outputs
  - Form an ANN with m input nodes, r output nodes and an internal structure
    - Some hidden layers, each layer having a given number of nodes
    - With weighted connections between every 2 nodes of consecutive layers
  - Determine the optimal weights by minimising the error
    - Difference between the real output y and the output computed by the network

Intelligent Systems - ANNs

- Problems solved by an ANN
  - Problem data can be represented by pairs (attribute-value)
  - Objective function can be:
    - Single or multi-criteria
    - Discrete or continuous (real values)
  - Training data can be noised
  - A large training time

Intelligent Systems - ANNs

- Design
  - ANN construction (for solving problem P)
  - Initialisation of ANN's parameters
  - ANN training
  - ANN testing

Intelligent Systems - ANNs

- Design
  - ANN construction (for solving problem P)
    - For a classification problem:
      - (x^d, t^d), cu:
      - $x^d \in \mathbf{R}^m \rightarrow x^d=(x^d_1, x^d_2,\ldots, x^d_m)$
      - $t^d \in \mathbf{R}^R \rightarrow t^d=(t^d_1, t^d_2,\ldots, t^d_R)$,
      - with $d = 1,2,\ldots,n,n+1,n+2,\ldots,N$
    - First n data are used for training
    - Last N-n data are used for testing

    - Construct the ANN:
      - Input layer has m nodes (each node reads an attribute of an input data – $x^d_1, x^d_2,\ldots, x^d_m$)
      - Output layer can contain R nodes (each node provides an output attribute – $t^d_1, t^d_2,\ldots, t^d_R$)
      - One or more hidden layers with one or more neurons on each layer

Intelligent Systems - ANNs

- Design
  - ANN construction (for solving problem P)
  - **Initialisation of ANN’s parameters**
  - **ANN training**
  - ANN testing

Intelligent Systems - ANNs

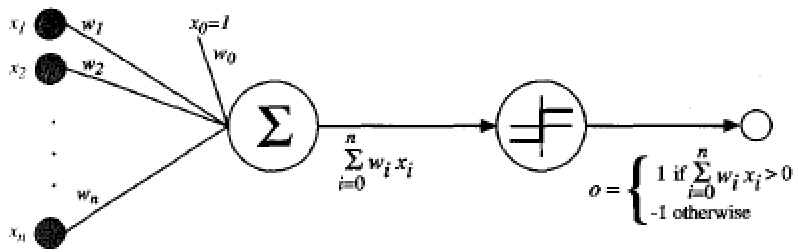
Design

- Initialisation of ANN’s parameters
  - Initialisation of weights (between nodes from consecutive layers)
  - Establish the activation function of each neuron of hidden layers

- ANN’s training
- Aim
  - Identify the optimal weights
- Algorithm
  - Search the optimal weights by minimising the errors (difference between the real output and the computed output)
- How does ANN learn?
  - Network = set of primitive computational units
    - Network learning = ∪ primitive unit learning
  - Primitive computational units
    - Perceptron
    - Linear unit
    - Sigmoidal unit

Intelligent Systems - ANNs

- Design → ANN's training → How does ANN learn?
  - Neuron as a simple computing element
    - Neuron structure
      - Each node has inputs and outputs
      - Each node perform a simple computation
    - Neuron processing
      - Information is transmitted to the neuron
      - Neuron processes the information
      - The answer of neuron is read
    - Neuron learning –algorithm of learning the weights that correctly process the information
      - Start by some initial weights
      - While not stopCondition
        - Process the information and establish the quality of current weights
        - Modify the weights such to obtain better results

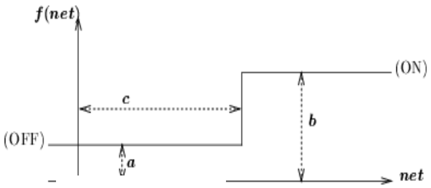


Intelligent Systems - ANNs

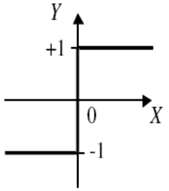
- Design → ANN’s training → How does ANN learn?
  - Neuron as simple computing element
    - Neuron structure
      - Each node has inputs and outputs
      - Each node performs a simple computation through an activation function
    - Neuron processing $net = \sum_{i=1}^{n} x_i w_i$
      - Information is transmitted to the neuron → compute the weighted sum of inputs
      - Neuron processes the information → by using an activation function
        - Constant function
        - Step function
        - Slope function
        - Linear function
        - Sigmoid function
        - Gaussian function

Intelligent Systems - ANNs

- Design → ANN's training → How does ANN learn?
  - Activation function of a neuron
    - Constant function f(net) = const
    - Step function (c - threshold)

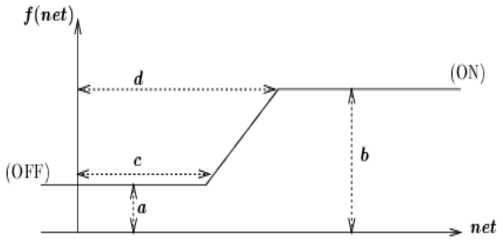


$$f(net) = \begin{cases} a, & \text{if } net < c \\ b, & \text{if } net > c \end{cases}$$



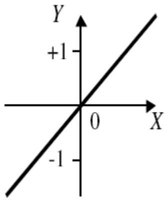
      - For a=+1, b =-1 and c = 0 → sign function
      - Discontinuous function
    - Slope function

$$f(net) = \begin{cases} a, & \text{if } net \le c \\ b, & \text{if } net \ge d \\ a + \dfrac{(net-c)(b-a)}{d-c}, & \text{otherwise} \end{cases}$$



    - Funcţia liniară

$$f(net) = a * net + b$$

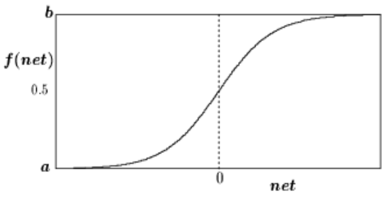


      - For a = 1 and b = 0 → identity function f(net)=net
      - Continuous function

Intelligent Systems - ANNs

- Design → ANN's training → How does ANN learn?
  - Activation function of a neuron
    - Sigmoid function
      - Shape of S
      - Continuous and differentiable
      - Rotational symmetric to a given point (net = c)

$$\lim_{net \to -\infty} f(net) = a \qquad \lim_{net \to \infty} f(net) = b$$



      - Examples:

$$f(net) = z + \frac{1}{1 + \exp(-x \cdot net + y)}$$

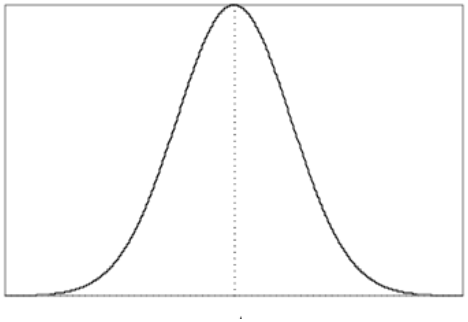
$$f(net) = \tanh(x \cdot net - y) + z \qquad \text{where} \quad \tanh(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$$

      - for y=0 and z = 0 → a=0, b = 1, c=0
      - For y=0 and z = -0.5 → a=-0.5, b = 0.5, c=0
      - The x is greater, the curve is steeper

Intelligent Systems - ANNs

- Design → ANN's training → How does ANN learn?
  - Activation function of a neuron
    - Gaussian function
      - Bell shape
      - Continuous

$$\lim_{net \to \infty} f(net) = a$$



      - Has a single optimum point (maximum) – for net = μ
      - Example

$$f(net) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{net-\mu}{\sigma}\right)^2\right]$$

Intelligent Systems - ANNs

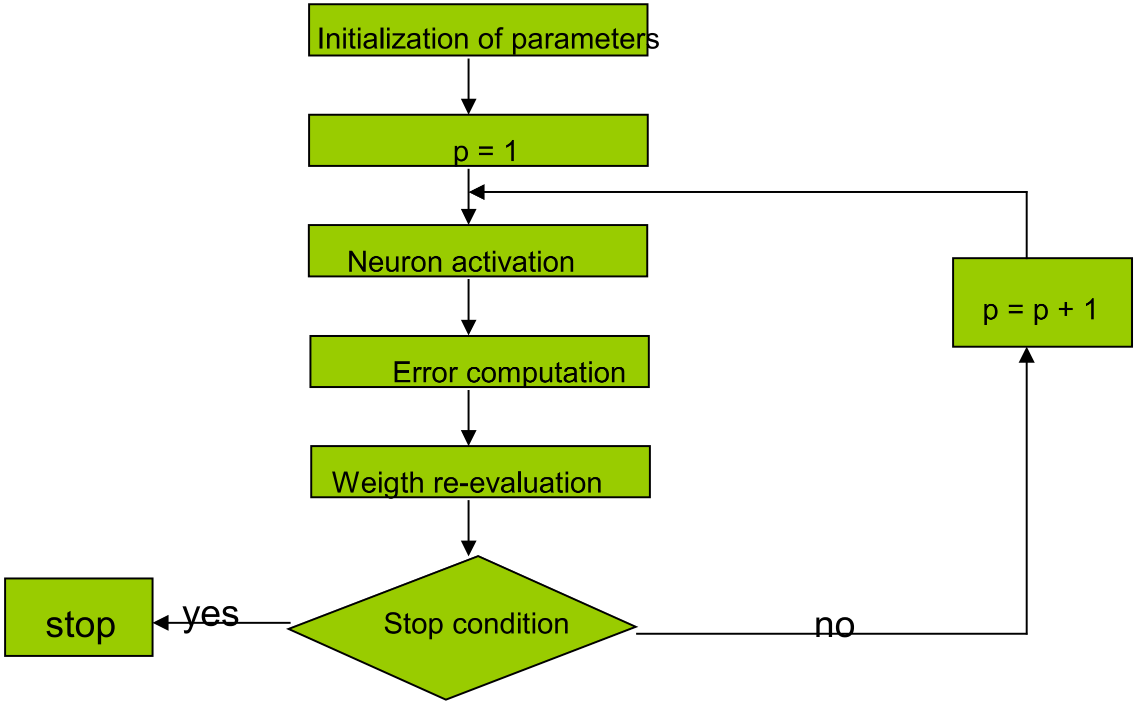
- Design → ANN's training → How does ANN learn?
  - Neuron as simple computation element
    - Neuron structure
    - Neuron processing
      - Information is transmitted to the neuron → compute the weighted sum of inputs

$$net = \sum_{i=1}^{n} x_i w_i$$

      - Neuron processes the information → using an activation function
        - Constant function
        - Step function
        - Slope function
        - Linear function
        - Sigmoid function
        - Gaussian function
      - Read the neuron's answer → determine is the computed result is the same with the real one

Intelligent Systems - ANNs

- Design → ANN's training → How does ANN learn?
  - Neuron as simple computation element
    - Neuron structure
    - Neuron processing
    - Neuron learning
      - Algorithm



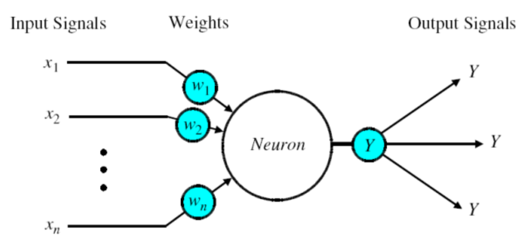
Intelligent Systems - ANNs

- Design → ANN's training → How does ANN learn?
  - Neuron learning
    - 2 basic rules
      - Perceptron's rule → perceptron's algorithm
        1. Start by some random weights
        2. Determine the quality of the model create for these weights for a single input data
        3. Re-compute the weights based on the model's quality
        4. Repeat (from step 2) until a maximum quality is obtained
      - Delta's rule → algorithm of gradient descent
        1. Start by some random weights
        2. Determine the quality of the model create for these weights for all input data
        3. Re-compute the weights based on the model's quality
        4. Repeat (from step 2) until a maximum quality is obtained

        - Similar to perceptron's rule, but the model's quality is established based on all data (all training data)

Intelligent Systems - ANNs

- Design → ANN's training → How does ANN learn?
  - Neuron learning
    - Suppose we have a train data set:
      - (x^d, t^d), cu:
        - $x^d \in \mathbf{R}^m$ ➔ $x^d=(x^d_1, x^d_2, \ldots, x^d_m)$
        - $t^d \in \mathbf{R}^R$ ➔ $t^d=(t^d_1, t^d_2, \ldots, t^d_R)$, and $R = 1$ (➔ $t^d=(t^d_1)$)
        - With $d = 1,2,\ldots,n$
      - ANN = elementary computational primitive (a neuron) ➔ a network:
        - m output nodes
        - linked to the computing neuron through weights w_i, $i =1,2,\ldots,m$ and
        - an output node



Intelligent Systems - ANNs

- Design → ANN's training → How does ANN learn?
  - Neuron learning
    - Perceptron's algorithm
      - Based on error minimisation associated to an instance of train data
      - Modify the weights based on error associated to an instance of train data

Initialisation of network weights

w_i = random(a,b), where i=1,2,...,m

d = 1

While there are incorrect classified examples

Activate the neuron and determine the output

Perceptron → sign activation function

$$o^d = sign(\mathbf{wx}) = sign(\sum_{i=1}^{m} w_i x_i)$$

Determine the weight modification

$$\Delta w_i = \eta(t^d - o^d)x_i^d, \text{ unde } i = 1,2,\ldots,m$$

where η - learning rate

Modify the weights

$$w_i = w_i + \Delta w_i$$

if d < n then d++

otherwise d = 1

EndWhile

Intelligent Systems - ANNs

- Design → ANN's training → How does ANN learn?
  - Neuron learning
    - Gradient descent algorithm
      - Based on the error associated to the entire set of train data
      - Modify the weights in the direction of the steepest slope of error reduction E(**w**) for the entire set of train data

$$E(\mathbf{w}) = \frac{1}{2}\sum_{d=1}^{n}(t^d - o^d)^2$$

      - How the steepest slope is determined? → derive E based on w (establish the gradient of error E)

$$\nabla E(\mathbf{w}) = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_m}\right)$$

      - Error's gradient is computed based on activation function of neuron (that must be differentiable → continuous)
        - Linear function $f(net) = \sum_{i=1}^{m} w_i x_i^d$
        - Sigmoid function $f(net) = \frac{1}{1+e^{-\mathbf{wx}}} = \frac{1}{1+e^{-\sum_{i=1}^{m} w_i x_i^d}}$
      - How the weights are modified?

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}, \text{ where } i = 1,2,\ldots,m$$

Intelligent Systems - ANNs

- Design → ANN's training → How does ANN learn?
  - Neuron learning
    - Descent gradient algorithm → error's gradient computation
      - Linear function

$$f(net) = \sum_{i=1}^{m} w_i x_i^d$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial \frac{1}{2}\sum_{d=1}^{n}\left(t^d - o^d\right)^2}{\partial w_i} = \frac{1}{2}\sum_{d=1}^{n}\frac{\partial\left(t^d - o^d\right)^2}{\partial w_i} = \frac{1}{2}\sum_{d=1}^{n} 2\left(t^d - o^d\right)\frac{\partial\left(t^d - \mathbf{wx}^d\right)}{\partial w_i}$$

$$\frac{\partial E}{\partial w_i} = \sum_{d=1}^{n}\left(t^d - o^d\right)\frac{\partial\left(t^d - w_1 x_1^d - w_2 x_2^d - \ldots - w_m x_m^d\right)}{\partial w_i} = \sum_{d=1}^{n}\left(t^d - o^d\right)\left(-x_i^d\right)$$

$$\Delta w_i = -\eta\frac{\partial E}{\partial w_i} = \eta\sum_{d=1}^{n}\left(t^d - o^d\right)x_i^d$$

      - Sigmoid function

$$f(net) = \frac{1}{1+e^{-\mathbf{wx}}} = \frac{1}{1+e^{-\sum_{i=1}^{m} w_i x_i^d}} \qquad y = s(z) = \frac{1}{1+e^{-z}} \Rightarrow \frac{\partial s(z)}{\partial z} = s(z)\left(1 - s(z)\right)$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial \frac{1}{2}\sum_{d=1}^{n}\left(t^d - o^d\right)^2}{\partial w_i} = \frac{1}{2}\sum_{d=1}^{n}\frac{\partial\left(t^d - o^d\right)^2}{\partial w_i} = \frac{1}{2}\sum_{d=1}^{n} 2\left(t^d - o^d\right)\frac{\partial\left(t^d - sig(\mathbf{wx}^d)\right)}{\partial w_i} = \sum_{d=1}^{n}\left(t^d - o^d\right)\left(1 - o^d\right)o^d\left(-x_i^d\right)$$

$$\Delta w_i = -\eta\frac{\partial E}{\partial w_i} = \eta\sum_{d=1}^{n}\left(t^d - o^d\right)\left(1 - o^d\right)o^d x_i^d$$

Intelligent Systems - ANNs

- Design → ANN's training → How does ANN learn?
  - Neuron learning
    - Gradient descent algorithm (GDA)

Simple GDA	Stochastic GDA
Initialisation of network weights wi = random(a,b), where i=1,2,...,m While not stop condition Δwi=0, unde i=1,2,...,m For each train example (xd,td), where d=1,2,...,n Activate the neuron and determine the output od Linear activation → od=wxd Sigmoid activation → od=sig(wxd) For each weight wi, where i =1,2,...,m Determine the weight modification $\Delta w_i = \Delta w_i - \eta \frac{\partial E}{\partial w_i}$ where η - learning rate For each weight wi, where i =1,2,...,m Modify the weights wi $w_i = w_i + \Delta w_i$ EndWhile	Initialisation of network weights wi = random(a,b), where i=1,2,...,m While not stop condition Δwi=0, unde i=1,2,...,m For each train example (xd,td), where d=1,2,...,n Activate the neuron and determine the output od Linear activation → od=wxd Sigmoid activation → od=sig(wxd) For each weight wi, where i =1,2,...,m Determine the weight modification $\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$ where η - learning rate Modify the weights wi $w_i = w_i + \Delta w_i$ EndWhile

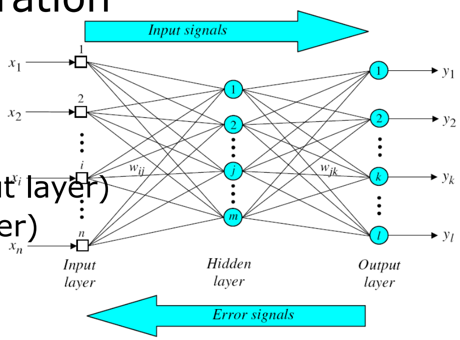
Intelligent Systems - ANNs

Design → ANN's training → How does ANN learn?
Neuron learning

Differences	Perceptron's algorithm	Gradient descent algorithm (delta rule)
What does o^d represent?	$o^d=\mathrm{sign}(\mathbf{wx}^d)$	$o^d=\mathbf{wx}^d$ or $o^d=\mathrm{sig}(\mathbf{wx}^d)$
Convergence	After a finite # of steps (until the perfect separation)	Asymptotic (to minimum error)
Solved problems	With linear separable data	Any data (linear separable or non-linear)
Neuron's output	Discrete and with threshold	Continue and without threshold

Intelligent Systems - ANNs

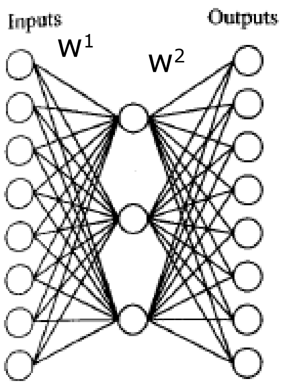
- Design → ANN's training
  - Network learning
    - Network = set of primitive computational units that are interconnected ➔
      - Net learning = ∪ primitive learning
    - Net with neurons located on one or more layers → ANN is able of learning a complex model (not a linear one only) for data separation
    - Algorithm for learning the weights → backpropagation
      - Based on descent gradient algorithm
      - Improvements
        - Information is forward propagated (from input layer to output layer)
        - Error is backward propagated (from output layer to input layer)



Initialisation of network weights
While not stop condition
    For each train example (x^d,t^d), where d=1,2,...,n
        Activate the neuron and determine the output o^d
            forward propagate the information and determine the output of each neuron
        Modify the weights
            Establish and backward propagate the error
                Establish the errors of neurons from the output layer
                Backward propagate the errors in the entire network → distribute the errors on all connections of the network
            Modify the weights
EndWhile

Intelligent Systems - ANNs

- Design → ANN's training
  - How does network learn?
    - Suppose a train data:
      - (x^d, t^d), with:
        - $x^d \in \mathbf{R}^m$ → $x^d=(x^d_1, x^d_2,..., x^d_m)$
        - $t^d \in \mathbf{R}^R$ → $t^d=(t^d_1, t^d_2,..., t^d_R)$
        - With d = 1,2,...,n
    - Consider 2 ANNs
      - An ANN with a single hidden layer with H neurons → ANN_1
        - m neurons on the input layer
        - R neurons on the output layer
        - H neurons on the hidden layer
        - Weights between input and hidden layers w^1_{ih} with i=1,2,...,m, h = 1,2,...,H
        - Weights between hidden and output layers w^2_{hr} with h = 1,2,...,H, r = 1,2,...,R



      - An ANN with P hidden layers, each layer having H_i (i =1,2,...,P) neurons → ANN_p
        - m neurons on the input layer
        - R neurons on the output layer
        - P hidden layers
        - H_p neurons on the p^{th} hidden layer, p =1,2,...,P
        - Weights between input layer and first hidden layer $w^1_{ih_1}$ with i=1,2,...,m, h_1 = 1,2,...,H_1
        - Weights between first hidden layer and second hidden layer $w^2_{h_1h_2}$ with h_1 = 1,2,...,H_1 , h_2 = 1,2,...,H_2
        - Weights between second hidden layer and third hidden layer $w^3_{h_2h_3}$ with h_2 = 1,2,...,H_2, h_3 = 1,2,...,H_3
        - ...
        - Weights between $(P-1)^{th}$ hidden layer and last hidden layer $w^p_{h_{p-1}h_p}$ with h_{P-1} = 1,2,...,H_{P-1} şi h_P = 1,2,...,H_P
        - Weights between last hidden layer and output layer $w^{p+1}_{h_p r}$ with h_P = 1,2,...,H_P, r = 1,2,...,R

Intelligent Systems - ANNs

- Design → ANN's training ->How does network learn?
  - Backpropagation algorithm for ANN_1

Initialisation of network weights w^1_{ih} and w^2_{hr} with i=1,2,…,m, h = 1,2,…,H, r = 1,2,…,R

While not stop condition

For each train example (x^d,t^d), where d=1,2,…,n

Activate the neuron and determine the output o^d

Forward propagate the information and determine the output of each neuron

$$o^d_h = \sum_{i=1}^{m} w^1_{ih} x^d_i \text{ sau } o^d_h = sig\left(\sum_{i=1}^{m} w^1_{ih} x^d_i\right), \text{cu } h = 1,2,\ldots,H$$

$$o^d_r = \sum_{h=1}^{H} w^2_{hr} o^d_h \text{ sau } o^d_r = sig\left(\sum_{h=1}^{H} w^2_{hr} o^d_h\right), \text{cu } r = 1,2,\ldots,R$$

Modify the weights

Establish and backward propagate the error

Establish the errors of neurons from the output layer

$$\delta^d_r = t^d_r - o^d_r \text{ or } \delta^d_r = o^d_r(1 - o^d_r)(t^d_r - o^d_r), \text{ with } r = 1,2,\ldots,R$$

Modify the weights between hidden layer and output layer

$$w^2_{hr} = w^2_{hr} + \eta \delta^d_r o^d_h, \text{ where h} = 1,2,\ldots,H, \text{ r} = 1,2,\ldots,R$$

Backward propagate the errors in the entire network → distribute the errors on all connections of the network

$$\delta^d_h = \sum_{r=1}^{R} w^2_{hr} \delta^d_r \text{ sau } \delta^d_h = o^d_h\left(1 - o^d_h\right)\sum_{r=1}^{R} w^2_{hr} \delta^d_r$$

Modify the weights between input layer and hidden layer

$$w^1_{ih} = w^1_{ih} + \eta \delta^d_h x^d_i, \text{ where } i = 1,2,\ldots,m,\ h = 1,2,\ldots,H$$

EndWhile

Intelligent Systems - ANNs

- Design → ANN's training ->How does network learn?
  - Backpropagation algorithm for ANN_P

Initialisation of network weights $w^1_{ih_1}, w^2_{h_1h_2}, \ldots, w^p_{h_{p-1}h_p}, w^{p+1}_{h_pr}$

While not stop condition

For each train example (x^d, t^d), where d=1,2,...,n

Activate the neuron and determine the output

Forward propagate the information and determine the output of each neuron

$$o^d_{h_1} = \sum_{i=1}^{m} w^1_{ih_1} x^d_i \text{ or } o^d_{h_1} = sig\left(\sum_{i=1}^{m} w^1_{ih_1} x^d_i\right), \text{with } h_1 = 1,2,\ldots,H_1$$

$$o^d_{h_2} = \sum_{h_1=1}^{H_1} w^2_{h_1h_2} o^d_{h_1} \text{ or } o^d_{h_2} = sig\left(\sum_{h_1=1}^{H_1} w^2_{h_1h_2} o^d_{h_1}\right), \text{with } h_2 = 1,2,\ldots,H_2$$

...

$$o^d_{h_p} = \sum_{h_{p-1}=1}^{H_{p-1}} w^p_{h_{p-1}h_p} o^d_{h_{p-1}} \text{ or } o^d_{h_p} = sig\left(\sum_{h_{p-1}=1}^{H_{p-1}} w^p_{h_{p-1}h_p} o^d_{h_{p-1}}\right), \text{with } h_p = 1,2,\ldots,H_p$$

$$o^d_r = \sum_{h_p=1}^{H_p} w^{p+1}_{h_pr} o^d_{h_p} \text{ or } o^d_r = sig\left(\sum_{h_p=1}^{H_p} w^{p+1}_{h_pr} o^d_{h_p}\right), \text{with } r = 1,2,\ldots,R$$

Intelligent Systems - ANNs

- Design → ANN's training ->How does network learn?
  - Backpropagation algorithm for ANN_P

Initialisation of network weights $w^1_{ih_1}, w^2_{h_1h_2}, \ldots, w^p_{h_{p-1}h_p}, w^{p+1}_{h_pr}$

While not stop condition

For each train example (x^d, t^d), where d=1,2,...,n

Activate the neuron and determine the output

Forward propagate the information and determine the output of each neuron

Modify the weights

Establish and backward propagate the error

Establish the errors of neurons from output layer

$$\delta^d_r = t^d_r - o^d_r \text{ or } \delta^d_r = o^d_r(1-o^d_r)(t^d_r - o^d_r), \text{ with } r = 1,2,\ldots,R$$

Modify the weights between the last hidden layer and the output layer

$$w^{p+1}_{h_pr} = w^{p+1}_{h_pr} + \eta\delta^d_r o^d_{h_p}, \text{ where } \mathrm{h_p} = 1,2,\ldots,H_p, \mathrm{r} = 1,2,\ldots,R$$

Intelligent Systems - ANNs

- Design → ANN's training ->How does network learn?
  - Backpropagation algorithm for ANN_P

Initialisation of network weights $w^1_{ih_1}, w^2_{h_1h_2}, \ldots, w^p_{h_{p-1}h_p}, w^{p+1}_{h_pr}$

While not stop condition

For each train example (x^d, t^d), where d=1,2,...,n

Activate the neuron and determine the output

Forward propagate the information and determine the output of each neuron

Modify the weights

Establish and backward propagate the error

Establish the errors of neurons from output layer

Modify the weights between the last hidden layer and the output layer

Backward propagate (on each layer) the errors in the entire network → distribute the errors on all connections proportional to the weights and modify the weights

$$\delta^d_{h_p} = \sum_{r=1}^{R} w^{p+1}_{h_pr}\delta^d_r \text{ or } \delta^d_{h_p} = o^d_{h_p}\left(1-o^d_{h_p}\right)\sum_{r=1}^{R} w^{p+1}_{h_pr}\delta^d_r$$

$$w^{p+1}_{h_pr} = w^{p+1}_{h_pr} + \eta\delta^d_r o^d_{h_p}, \text{ where } \mathrm{h_p} = 1,2,\ldots,H_p, \ \mathrm{r} = 1,2,\ldots,R$$

$$\delta^d_{h_{p-1}} = \sum_{h_p=1}^{H_p} w^p_{h_{p-1}h_p}\delta^d_{h_p} \text{ or } \delta^d_{h_{p-1}} = o^d_{h_{p-1}}\left(1-o^d_{h_{p-1}}\right)\sum_{h_p=1}^{H_p} w^p_{h_{p-1}h_p}\delta^d_{h_p}$$

$$w^p_{h_{p-1}h_p} = w^p_{h_{p-1}h_p} + \eta\delta^d_{h_p} o^d_{h_{p-1}}, \text{ where } h_{\text{p-1}} = 1,2,\ldots,H_{p-1} \text{ and } h_p = 1,2,\ldots,H_p$$

...

$$\delta^d_{h_1} = \sum_{h_2=1}^{H_2} w^2_{h_1h_2}\delta^d_{h_2} \text{ or } \delta^d_{h_1} = o^d_{h_1}\left(1-o^d_{h_1}\right)\sum_{h_2=1}^{H_2} w^2_{h_1h_2}\delta^d_{h_2}$$

$$w^1_{ih_1} = w^1_{ih_1} + \eta\delta^d_{h_1} x^d_i, \text{ where } i = 1,2,\ldots,m, \ h_1 = 1,2,\ldots,H_1$$

Intelligent Systems - ANNs

- Design → ANN's training ->How does network learn?
  - Backpropagation algorithm
    - Stop conditions
      - Error is 0
      - After a given number of iterations
        - During an iteration, a single example is processed
        - n iterations = an epoch

Intelligent Systems - ANNs

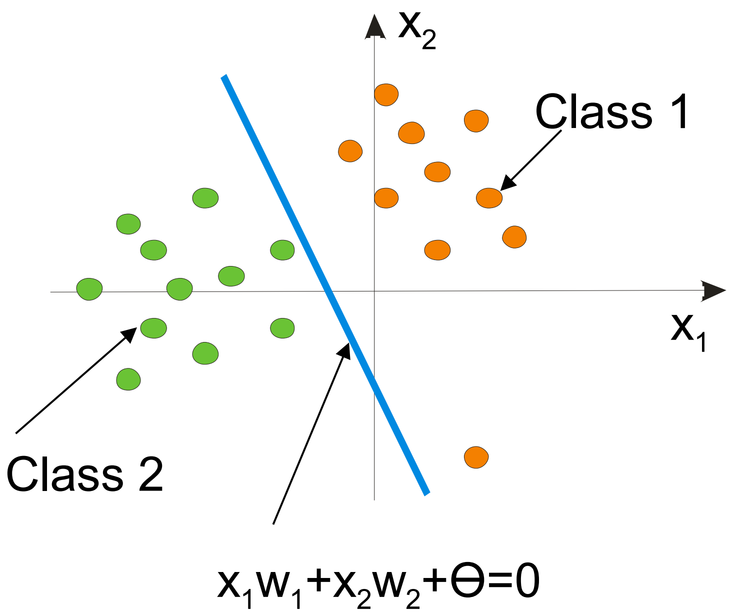
- Design
  - ANN construction (for solving problem P)
  - Initialisation of ANN's parameters
  - ANN training
  - **ANN testing**

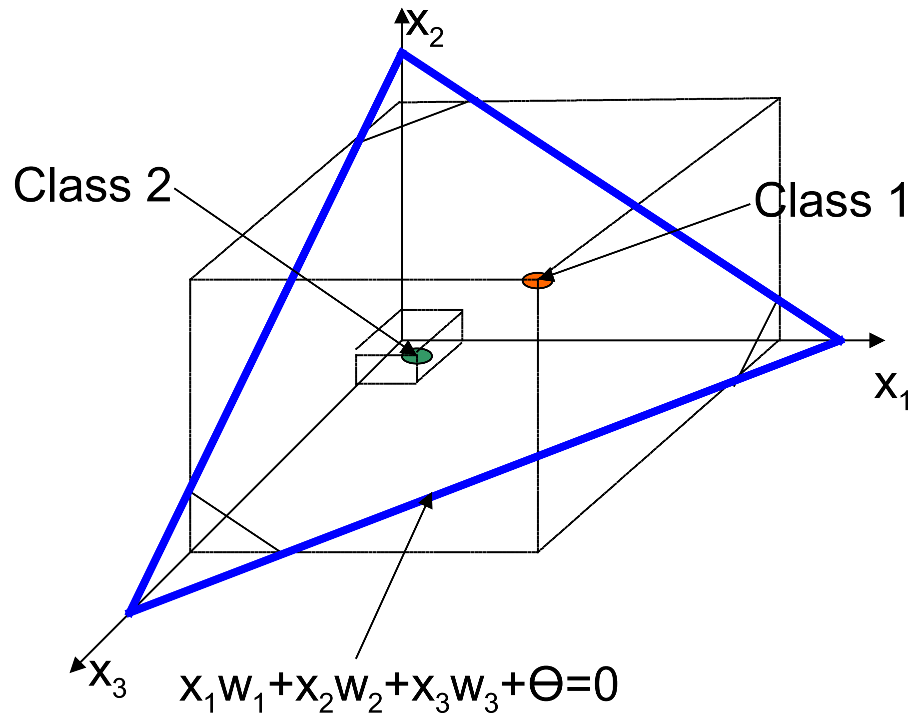
Intelligent Systems - ANNs

- Design
  - ANN testing
    - Decode the model learn by ANN
      - By combining the weights and inputs
      - Taking into account the activation function of neurons and the structure of the network

Intelligent Systems - ANNs

- Example





Binary classification with m=2 input attributes

Binary classification with m=3 input attributes

Intelligent Systems - ANNs

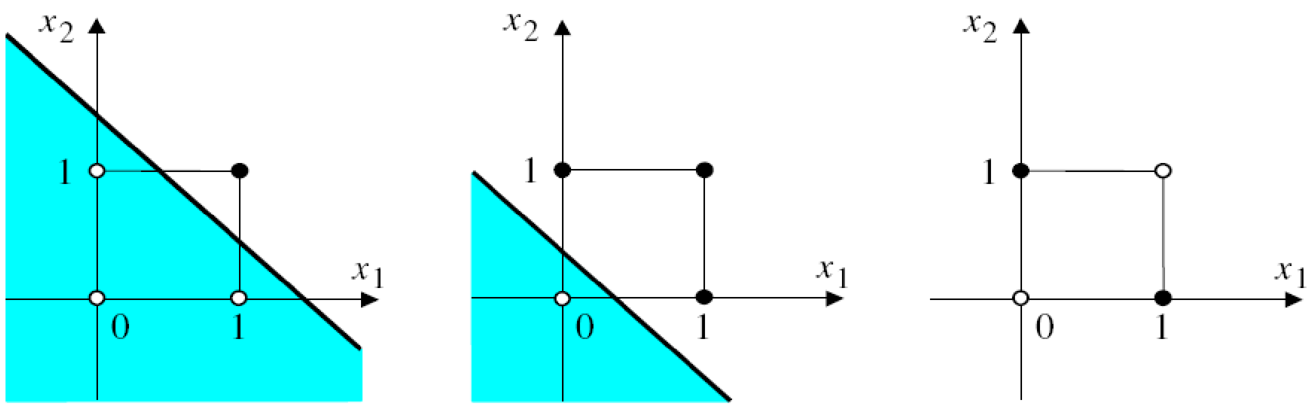
- Example
  - Perceptron for solving *logic And* problem

Epoch	Inputs x_1	Inputs x_2	Desired output Y_d	Initial weights w_1	Initial weights w_2	Actual output Y	Error e	Final weights w_1	Final weights w_2
1	0	0	0	0.3	−0.1	0	0	0.3	−0.1
	0	1	0	0.3	−0.1	0	0	0.3	−0.1
	1	0	0	0.3	−0.1	1	−1	0.2	−0.1
	1	1	1	0.2	−0.1	0	1	0.3	0.0
2	0	0	0	0.3	0.0	0	0	0.3	0.0
	0	1	0	0.3	0.0	0	0	0.3	0.0
	1	0	0	0.3	0.0	1	−1	0.2	0.0
	1	1	1	0.2	0.0	1	0	0.2	0.0
3	0	0	0	0.2	0.0	0	0	0.2	0.0
	0	1	0	0.2	0.0	0	0	0.2	0.0
	1	0	0	0.2	0.0	1	−1	0.1	0.0
	1	1	1	0.1	0.0	0	1	0.2	0.1
4	0	0	0	0.2	0.1	0	0	0.2	0.1
	0	1	0	0.2	0.1	0	0	0.2	0.1
	1	0	0	0.2	0.1	1	−1	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1
5	0	0	0	0.1	0.1	0	0	0.1	0.1
	0	1	0	0.1	0.1	0	0	0.1	0.1
	1	0	0	0.1	0.1	0	0	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1

Threshold: $\theta = 0.2$; learning rate: $\alpha = 0.1$

Intelligent Systems - ANNs

- Example
  - Perceptron – limits
    - A perceptron can learn And and OR operations, but it can not learn XOR operation (it is not linear separable)



(a) $AND\ (x_1 \cap x_2)$ (b) $OR\ (x_1 \cup x_2)$ (c) $Exclusive\text{-}OR\ (x_1 \oplus x_2)$

- Non-linear separable data can not be classified
  - Solutions
    - Neuron with continuous threshold
    - More neurons

Intelligent Systems - ANNs

- Typology
  - feed-forward ANNs
    - Information is processed and passed from a layer to another layer
    - Node connections do not form cycles
    - Utilized for supervised learning, especially
    - Activation function → linear, sigmoid, Gaussian
  - Recurrent ANNs (with feedback)
    - Can contain connections between nodes of the same layer
    - Node connections can form cycles
    - Jordan ANNs
    - Elman ANNs
    - Hopfield ANNs

    for supervised learning
    - Self-organized ANNs → for unsupervised learning
      - Hebbian ANNs
      - Kohonen ANNs (*Self organised maps*)

Intelligent Systems - ANNs

- Advantages
  - Can solve supervised and unsupervised learning problems
  - Can identify dynamic and non-linear relations among data
  - Can solve multi-class classification problems
  - Can compute very fast (parallel and distribute computing)

- Limits
  - ANNs have over-fitting problems (even if cross-validation is performed)
  - ANNs can find, sometimes, local optima only (without identifying the global optimum)

Review

- Automatic learning systems
  - Artificial Neural Networks
    - Computational models inspired by biological neural networks
    - Special graphs with nodes located in layers
      - Input layer → read the input data of the problem
      - Output layer → provides results
      - Hidden layer(s) → perform computations
    - Nodes (neurons)
      - Have weighted inputs
      - Have activation functions (linear, sigmoid, etc)
      - Require training → algorithms like:
        - Perceptron
        - Gradient descent
    - Training algorithm for an entire ANN → Backpropagation
      - Information is forward propagated
      - Errors are backward propagated

Next lecture

A. Short introduction in Artificial Intelligence (AI)

B. Solving search problems
   A. Definition of search problems
   B. Search strategies
      A. Uninformed search strategies
      B. Informed search strategies
      C. Local search strategies (Hill Climbing, Simulated Annealing, Tabu Search, Evolutionary algorithms, PSO, ACO)
      D. Adversarial search strategies

C. **Intelligent systems**
   A. Rule-based systems in certain environments
   B. Rule-based systems in uncertain environments (Bayes, Fuzzy)
   C. **Learning systems**
      A. Decision Trees
      B. Artificial Neural Networks
      C. **Support Vector Machines**
      D. Evolutionary algorithms
   D. Hybrid systems

Next lecture – useful information

- Chapter 15 of *C. Groşan, A. Abraham, Intelligent Systems: A Modern Approach, Springer, 2011*
- Chapter 9 of *T. M. Mitchell, Machine Learning, McGraw-Hill Science, 1997*
- Documents from *12_svm* and *13_GP folders*

- Presented information have been inspired from different bibliographic sources, but also from past AI lectures taught by:
  - PhD. Assoc. Prof. Mihai Oltean – www.cs.ubbcluj.ro/~moltean
  - PhD. Assoc. Prof. Crina Groşan - www.cs.ubbcluj.ro/~cgrosan
  - PhD. Prof. Horia F. Pop - www.cs.ubbcluj.ro/~hfpop