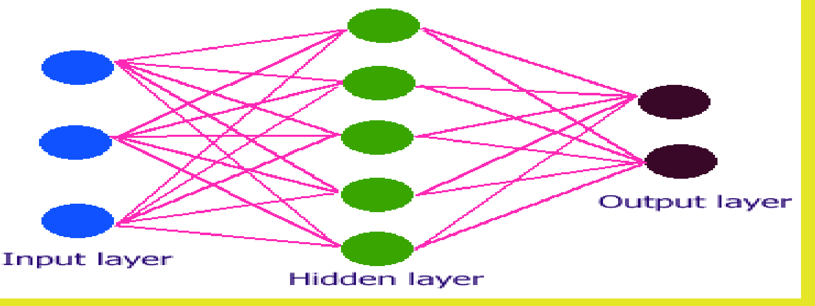
Statistical Data Mining



Artificial Neural Networks

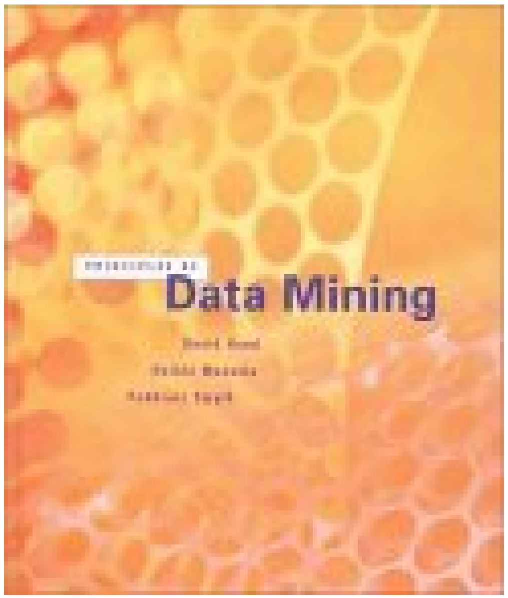
Professor Dr. Gholamreza Nakhaeizadeh

Content

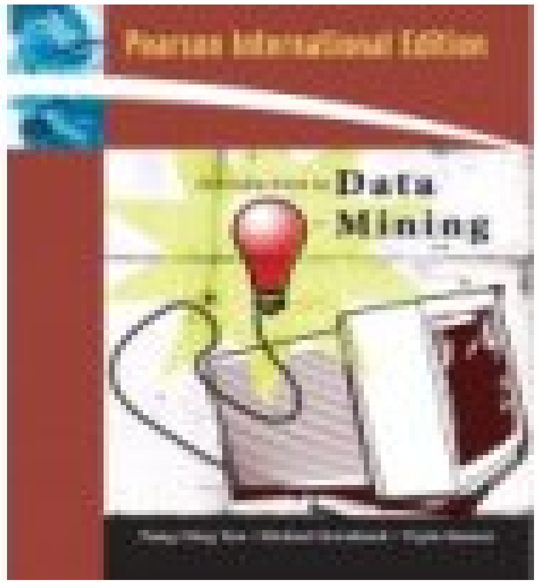
Introduction

- **Rise and fall of Neural Networks**
- **Input function of neuron**
- **Activation function of neuron**
- **Output (function) of neuron**
- **Feed-Forward networks**
- **Feedback networks**
- **Learning process**
- **Coding and decoding methods**
- **Perceptron**
- **Backpropagation**
- **Weakness and Strength of ANN**

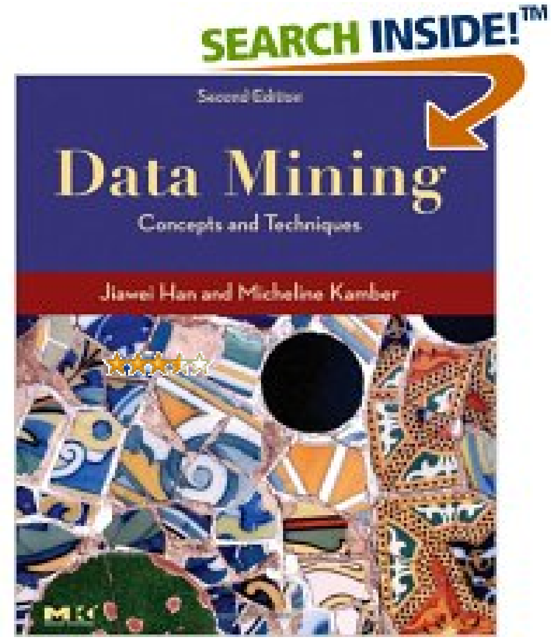
Literatur used (1)



Principles of Data Mining
David J. Hand, Heikki Mannila, Padhraic Smyth



Pang-Ning Tan, Michael Steinbach, Vipin Kumar



Jiawei Han and Micheline Kamber

Literature Used (2)

http://cse.stanford.edu/class/sophomore-college/projects-00/neural-networks/

http://www.cs.cmu.edu/~awm/tutorials

http://www.crisp-dm.org/CRISPwP-0800.pdf

http://en.wikipedia.org/wiki/Feedforward_neural_network

http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#Feedback%20networks

http://www.dmreview.com/

http://www.planet-source-code.com/vb/scripts/ShowCode.asp?lngWId=5&txtCodeId=378

http://download-uk.oracle.com/docs/html/B13915_02/i_olap_chapter.htm#BABCBDFA

http://download-uk.oracle.com/docs/html/B13915_02/i_rel_chapter.htm#BABGFCFG

http://training.inet.com/OLAP/home.htm

http://www.doc.gold.ac.uk/~mas01ds/cis338/index.html

http://wwwmaths.anu.edu.au/~steve/pdcn.pdf

www.kdnuggets.com

The Data Warehouse Toolkit by Ralph Kimball (John Wiley and Sons, 1996)

Building the Data Warehouse by William Inmon (John Wiley and Sons, 1996)

Studenmund, A. H. (2006). Using Econometrics, A practical Guide. Pearson International Edition

Artificial Neural Networks

Data Mining Algorithms

Machine Learning

- Rule Based Induction
- Decision Trees
- Neural Networks
- Conceptional clustering
- …….

Statistics

- Discriminant Analysis
- Cluster Analysis
- Regression Analysis
- Logistic Regression Analysis
- …….

Database Technology

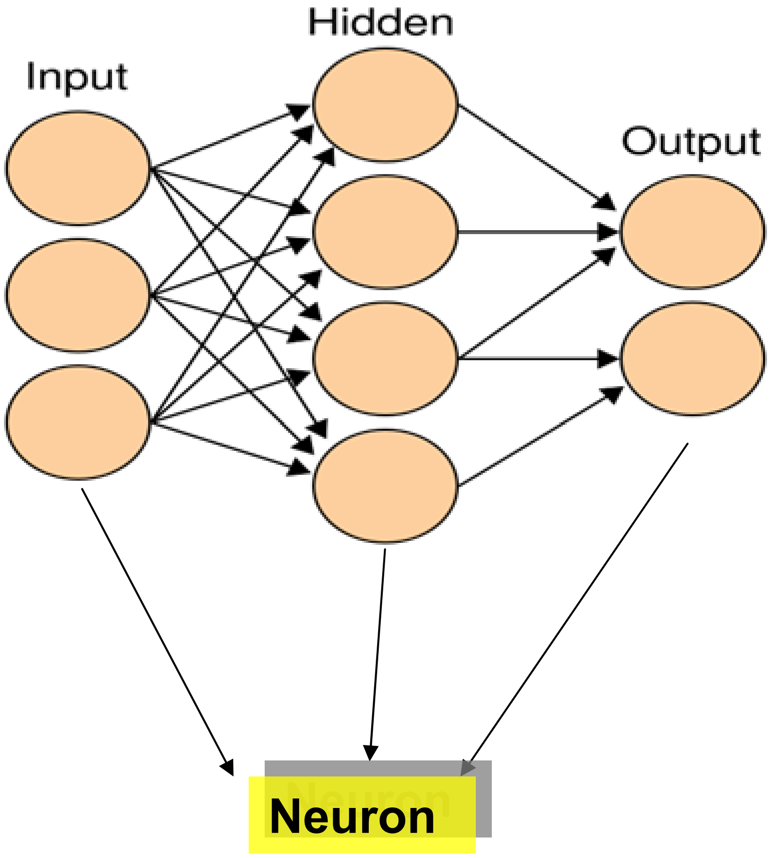
- Association Rules
- ….

Artificial Neural Networks

Artificial Neural Networks (ANN) **Introduction**

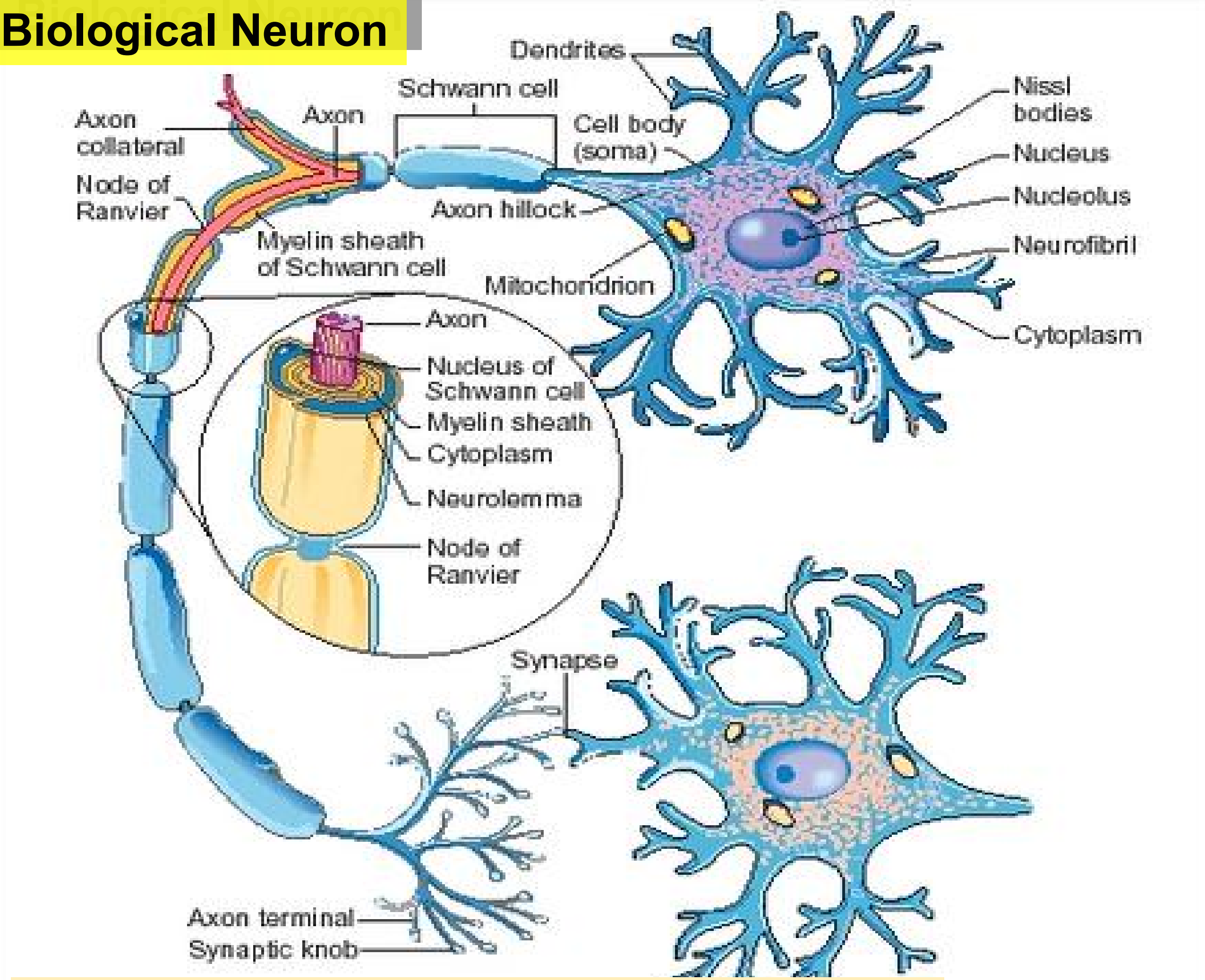
- **Inspired by the way biological nervous systems e.g. the brain, process information**

- **An ANN consist of an input layer, an output layer and one or more hidden layer(s):**
  - **exchanges and process information**
  - **has learning capability**
  - **is an adaptive system that changes its structure during the learning phase**



Artificial Neural Networks

Biological Neuron



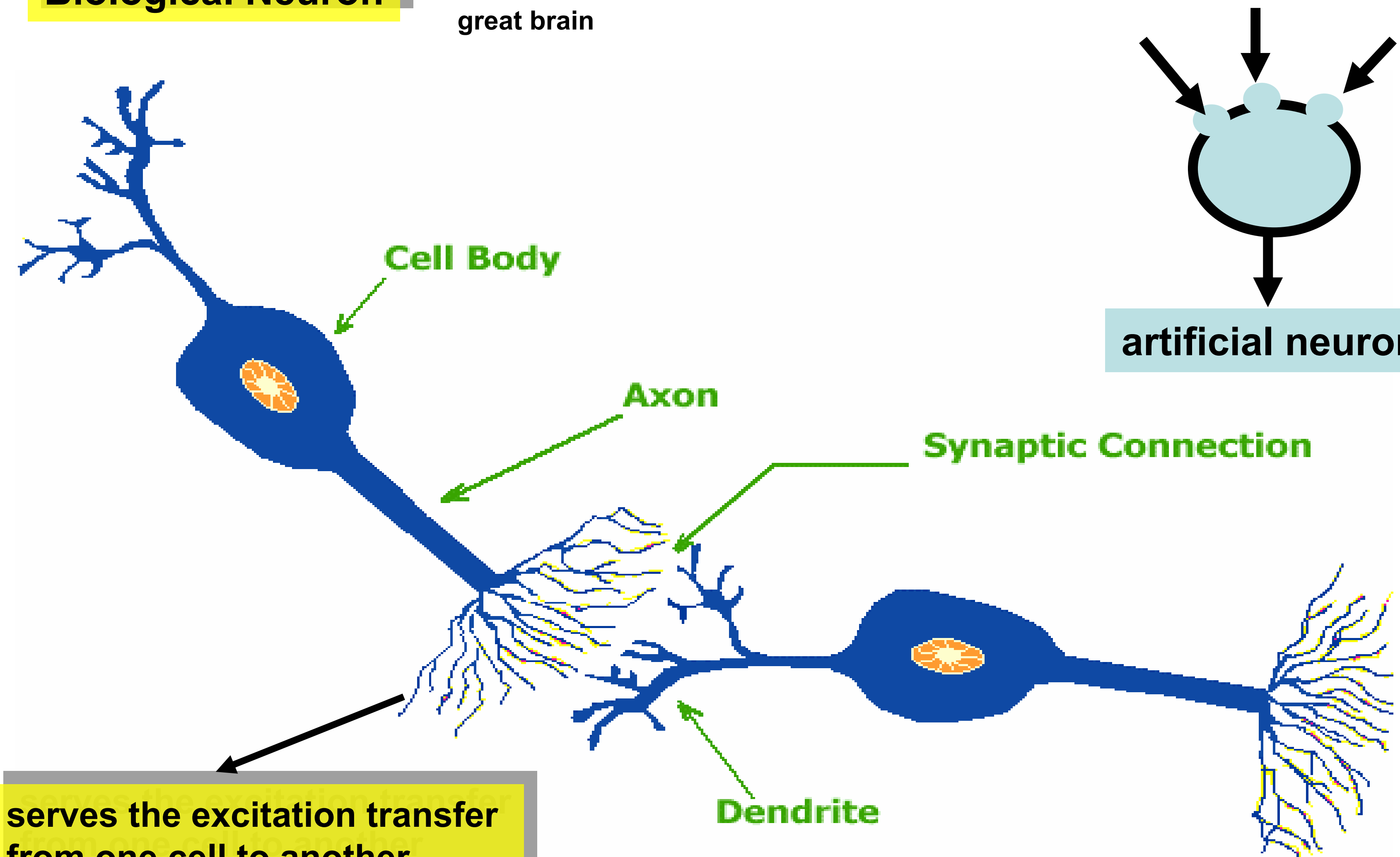
Neuron. (Diagram by Hans & Cassidy. Courtesy of Gale Group.)

http://www.healthline.com/galeimage?contentId=genh_03_00566&id=genh_03_img0308

Artificial Neural Networks

Biological Neuron

Source. http://www.intelligentsolutionsinc.com/part1.htm
great brain



artificial neuron

serves the excitation transfer from one cell to another

Artificial Neural Networks

Rise and fall of Neural Networks

Fall

- In 1969 Minsky and Papert showed the limitations of single layer Perceptrons
- Their results caused that a lot of researchers loose their interest

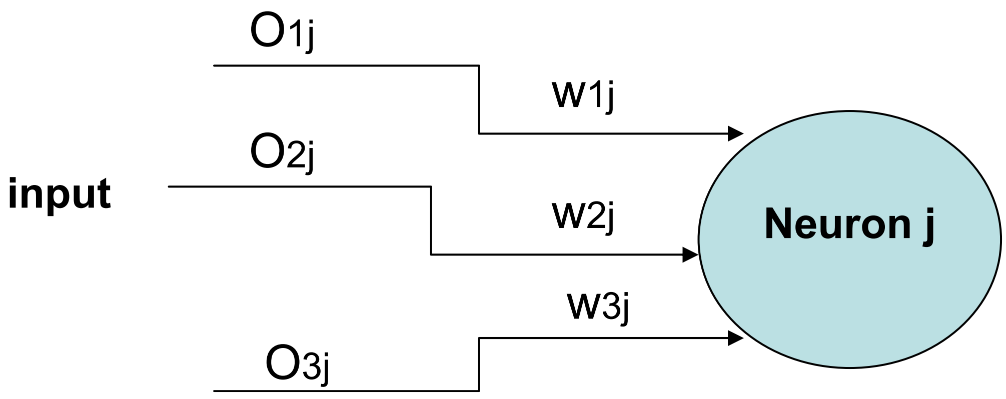
Rise

- In the 70’s and 80's, it was shown that multilevel perceptrons don’t have These shortcomings
  - Paul J. Werbos invented 1974 the back-propagation having the ability to perform classification tasks beyond simple Perceptrons
  - Back-propagation was independently rediscovered in 1980s by David Rumelhat and David Parker

Artificial Neural Networks

Artificial Neural Networks

Input function of neuron



The input value is multiplied by the weight before entering the neuron these weighted inputs are then added together and generate the :

$$\sum_{i=1}^{m} O_{ij}\, w_{ij} \longrightarrow \text{input function}$$

O_i : Input of the neuron i from the pervious layer

Artificial Neural Networks

Artificial Neural Networks | **Activation Function of neuron**

- **Biological neurons can have only two activation states (active, not active)**
- **Artificial neurons, however, can have different activation states range between (-1, 1) or (0,1)**
- **Lower bounds -1 or 0 stand for total inactivation**
- **Upper bound 1 stands for total activation**

Example of usual activation functions
Sigmoid (logistic) function:

Output of the unit i of the pervious layer

$$O_j = \frac{1}{1 + e^{-aZ_j}}$$

with $Z_j = \sum_{i=1}^{m} O_{ij}\, w_{ij} + \Theta_j$ **where Θ_j is the bias**

Oj is the output of the unit j of the actuel layer.
The parameter **a** is a real number
and constant. Usually in ANN applications
a is selected between 0.5 and 2

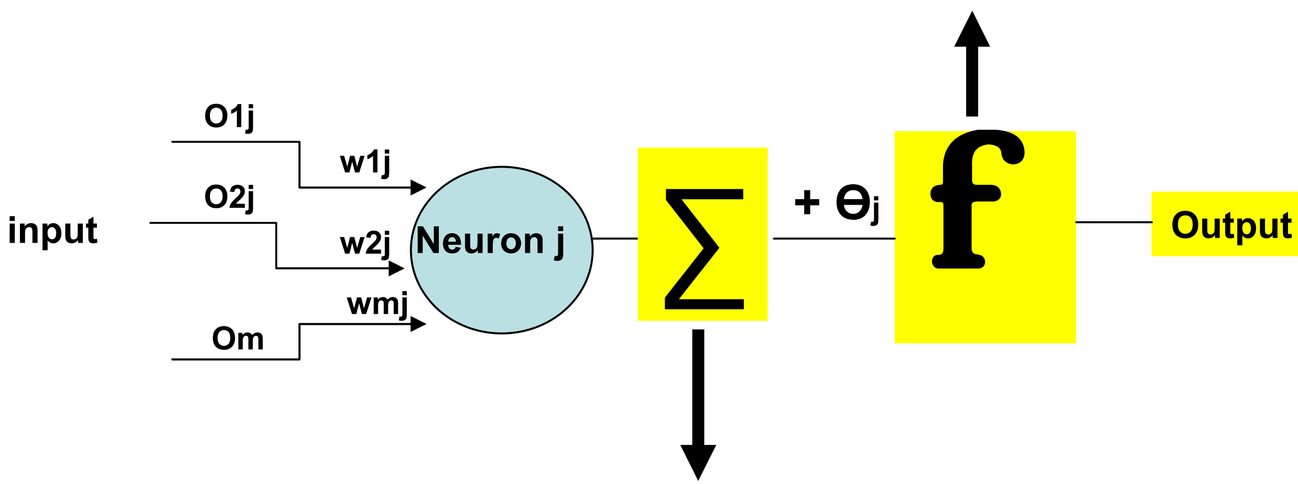
Net input

Artificial Neural Networks

Artificial Neural Networks

Output (function) of neuron

$$\frac{1}{1 + e^{-aZ_j}}$$



$$\text{with } Z_j = \sum_{i=1}^{n} O_i w_{ij} + \Theta_j$$

Artificial Neural Networks

Artificial Neural Networks **Activation Function of neuron**

Why nonlinear activation functions are needed ?

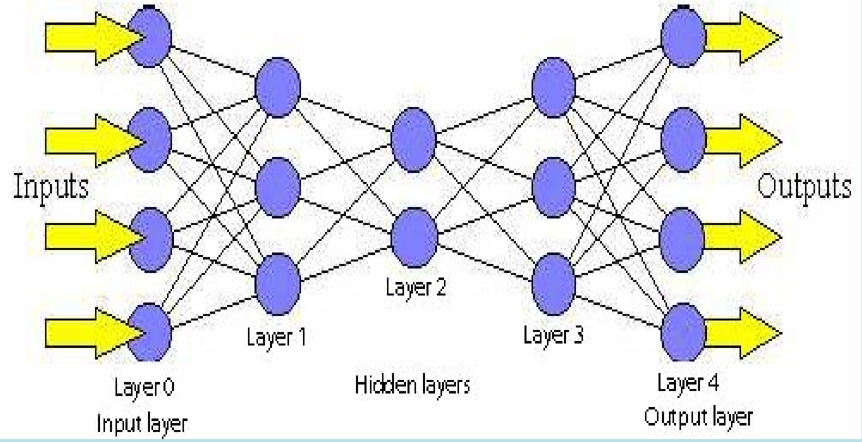
To catch the nonlinearity aspects in the data, otherwise
the hidden units could not make the ANN more powerful than just
Simple ***perceptrons*** without any hidden units.

The nonlinearity makes representing of nonlinear functions possible
and is the most powerful adjective of multilayer ANN

Artificial Neural Networks

Artificial Neural Networks | **Architecture** | **Feed-Forward networks (FNN)**

- **FNN (known also as multi-layer *perceptrons*) are widely used models specially in practical applications**
- **They were the first and simplest type of ANN developed**

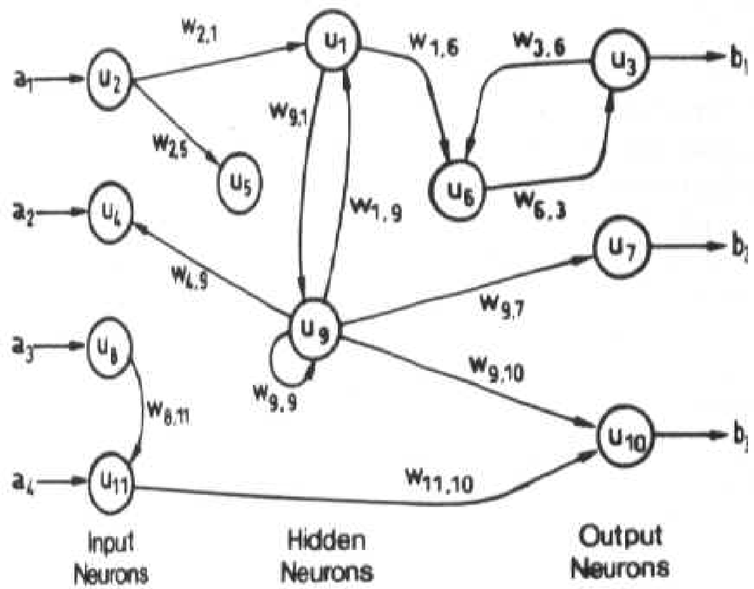


In a FFN
- **the information moves in only one direction**
- **information at a later level never backpropagates to the previous levels**
- **there are no cycles or loops in the network**

Artificial Neural Networks

Artificial Neural Networks | **Architecture** | **Feedback networks (FBN)**

The architecture of FBN (called also as interactive or recurrent Networks) is designed in a manner that they can send signals in both directions or in the loops



FBN are generally more difficult to train than FFN

Artificial Neural Networks

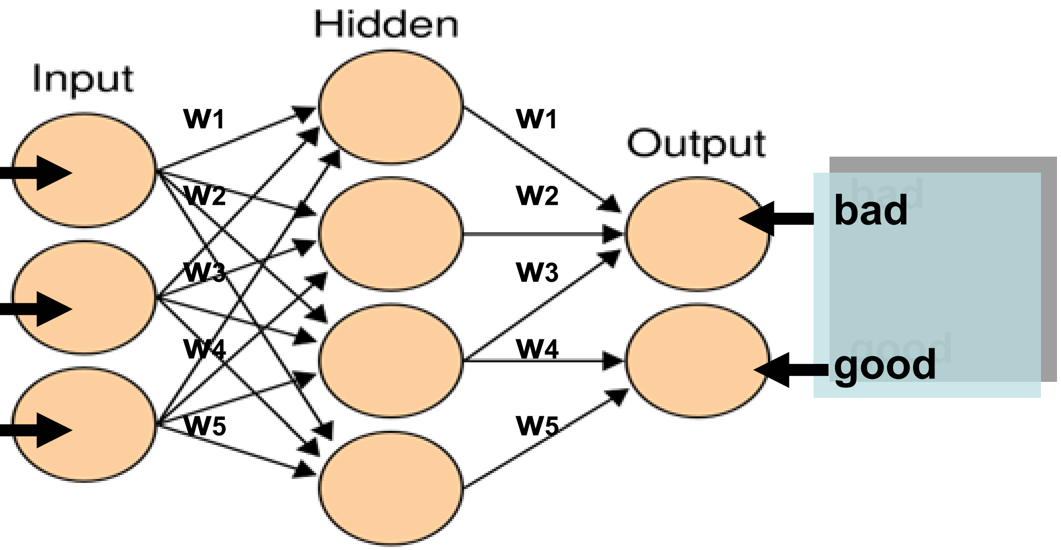
Artificial Neural Networks | Learning Process | Supervised Learning

	Income >2000	Car	Gender	Credit Rating
1	no	yes	F	bad
2	no state	no	F	bad
3	no state	yes	M	good
4	no	yes	M	bad
5	yes	yes	M	good
6	yes	yes	F	good
7	no state	yes	F	good
8	yes	no	F	good
9	no state	no	M	bad
10	no	no	F	bad

Artificial Neural Networks

Artificial Neural Networks | Learning Process

	Income >2000	Car	Gender
1	no	yes	F
2	no state	no	F
3	no state	yes	M
4	no	yes	M
5	yes	yes	M
6	yes	yes	F
7	no state	yes	F
8	yes	no	F
9	no state	no	M
10	no	no	F



Assumption: During the learning process, the net topology as well as input, activation and output functions remain constant. Only the weights change. Learning happens by weights adaptation.

The weights are adapted so long that the calculated net output is the correct desired output

Artificial Neural Networks

Artificial Neural Networks | Learning Process

Ripley (1996), (p.354)

Training set:

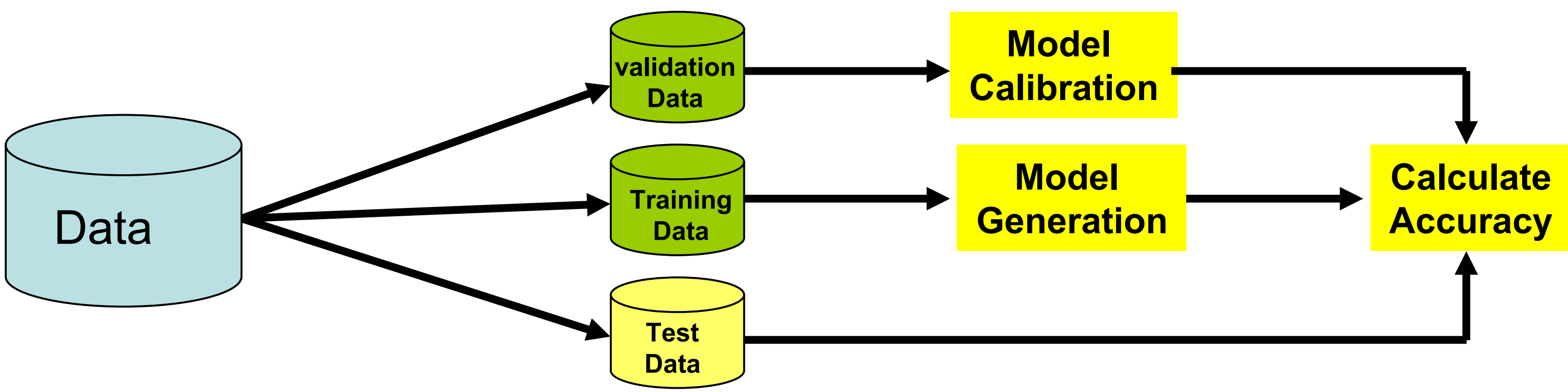
A set of examples used for learning, that is to fit the parameters [i.e., weights] of the classifier.

Validation set:

A set of examples used to tune the parameters [i.e., architecture, not weights] of a classifier, for example to choose the number of hidden units in a neural network.

Test set:

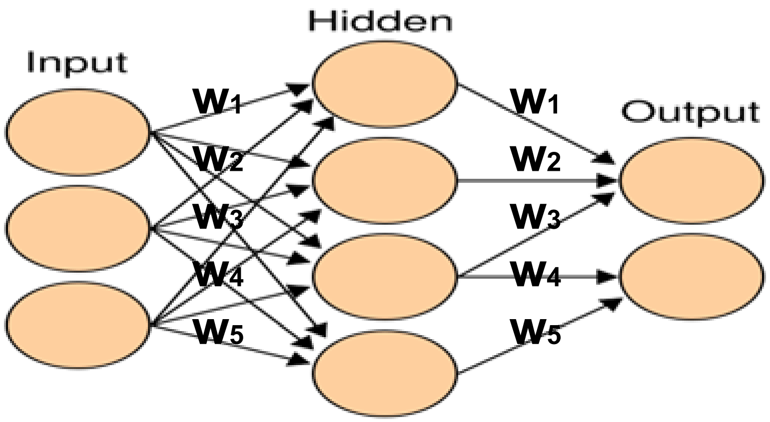
A set of examples used only to assess the performance [generalization] of a fully-specified classifier.



Artificial Neural Networks

Artificial Neural Networks | **Learning Process** | **Learning Rule**

How should we calculate the weight changing ?



new weight = old weight + weight change

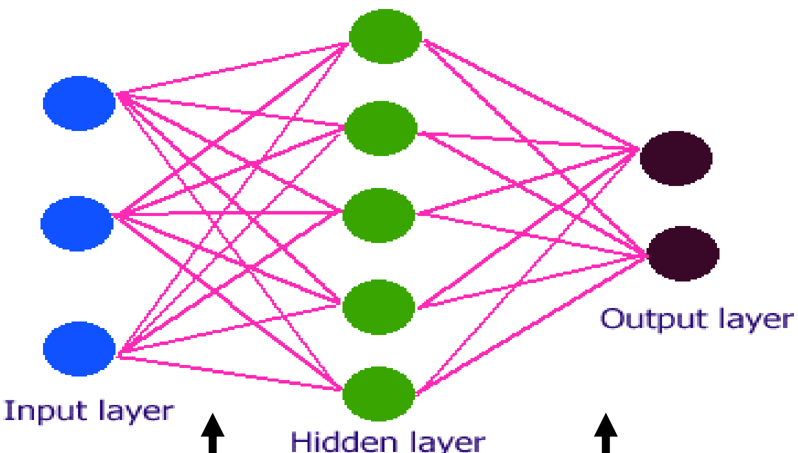
$w(t+1) = w(t) + \Delta w(t)$ ⟶ **Learning Rule**

In the last years several Learning Rules have been invented

Artificial Neural Networks

Artificial Neural Networks | **Learning Process** | **Learning Rule**

For each training Tuple:
The weights of the whole networks are modified in a manner to minimize the mean squared error (difference between the predicted and observed target value)



$$\textbf{Min: } E = \frac{1}{2} \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2$$

In most cases the output is a nonlinear function of weights and it is difficult to find the minimum of E analytically

Gradient descent method

$$w_j \rightarrow w_j - \beta * \partial E(\mathbf{w}) / \partial w_j$$

ß: Learning Rate between 0 and 1

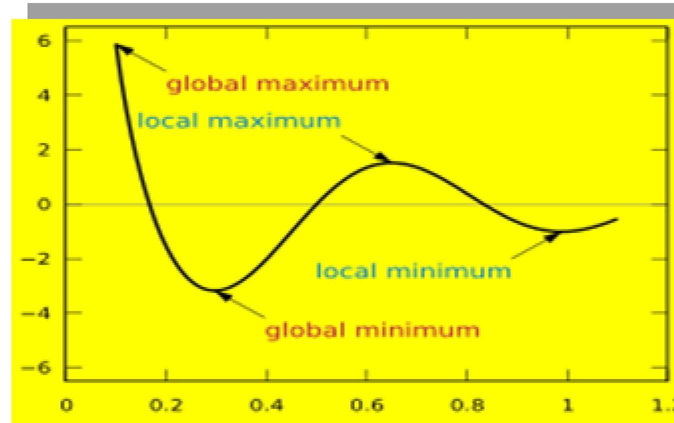
GD Learning Rule can be used to learn the weights of the outputs and hidden neurons

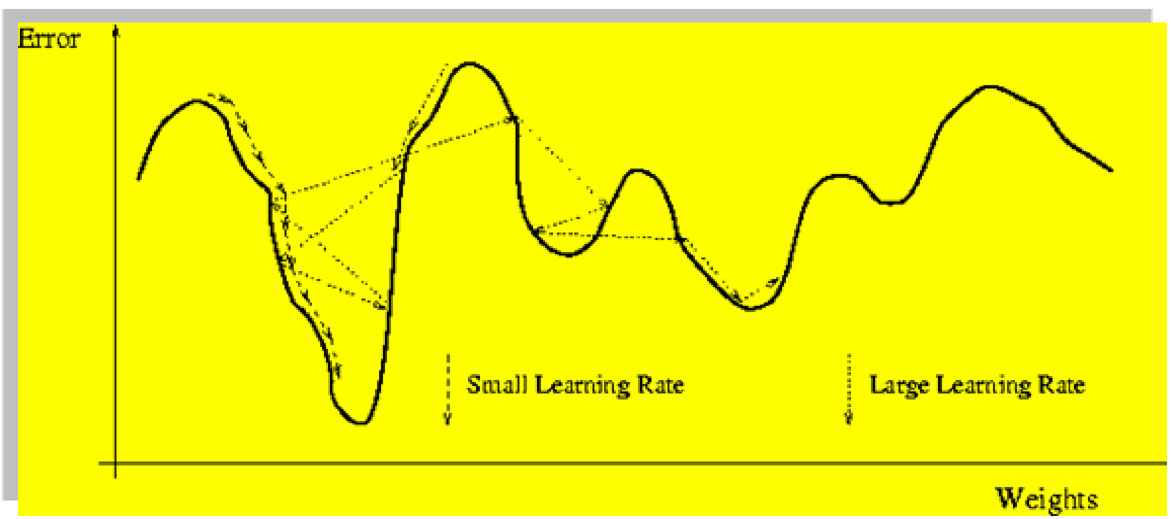
Artificial Neural Networks

Artificial Neural Networks | **Perceptron** | **Learning in Perceptron**

Disadvantages of Gradient descent method:

- Getting Stuck in a local optima
- Small gradient → small change (slow)
- Big gradient → big change (cavort)



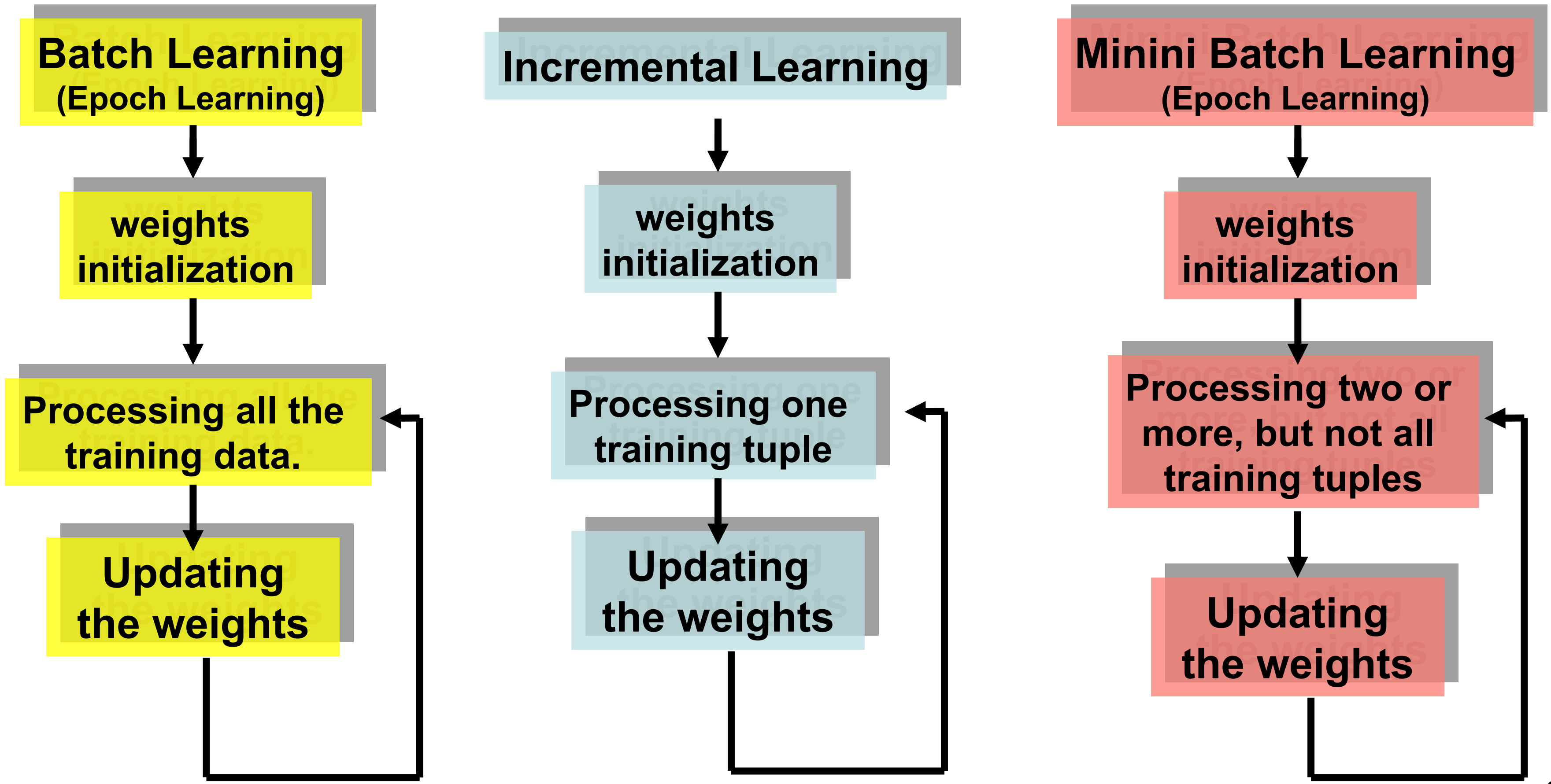


Source: http://www.teco.edu/~albrecht/neuro/html/node18.html#SECTION00630000000000000000

Artificial Neural Networks

Artificial Neural Networks | **Learning Process** | **Learning Methods**

Categorization of learning methods



Artificial Neural Networks

Artificial Neural Networks | **Learning Process** | **Training Stop**

- **Overfitting generates a relatively small error on the training dataset but larger error on new data**

- **To prevent Overfitting, it is sometime necessary to stop training by using a validation dataset**

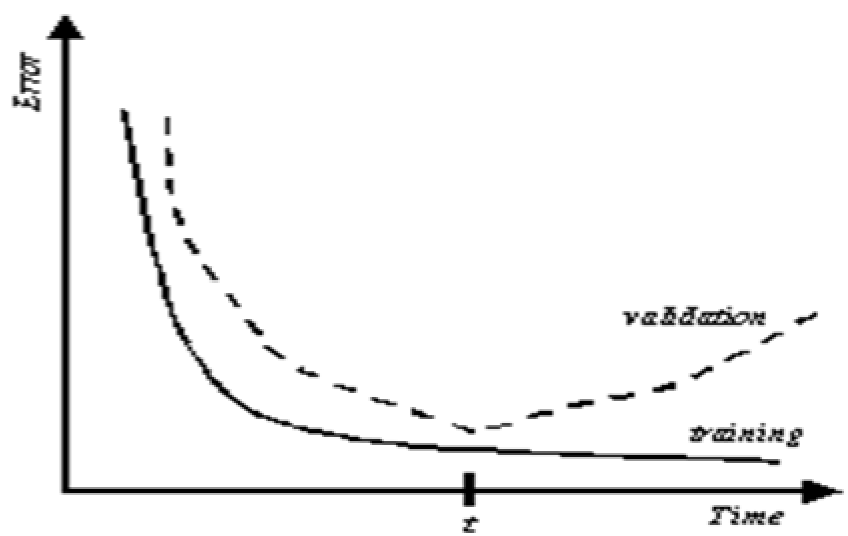
When stop training ?

- **After a few epochs the ANN should be tested by using the validation dataset**

Error on validation dataset increases higher than the last time error ?

↓

Training Stop



Artificial Neural Networks

Artificial Neural Networks | **Neuron's values**

there are two types of neurons:

- **Binary neurons** — **can only takes values from the set {0,1} or {1, -1}**
- **Real-valued neurons** — **can take values in the intervals [0,1] or [1, -1]**

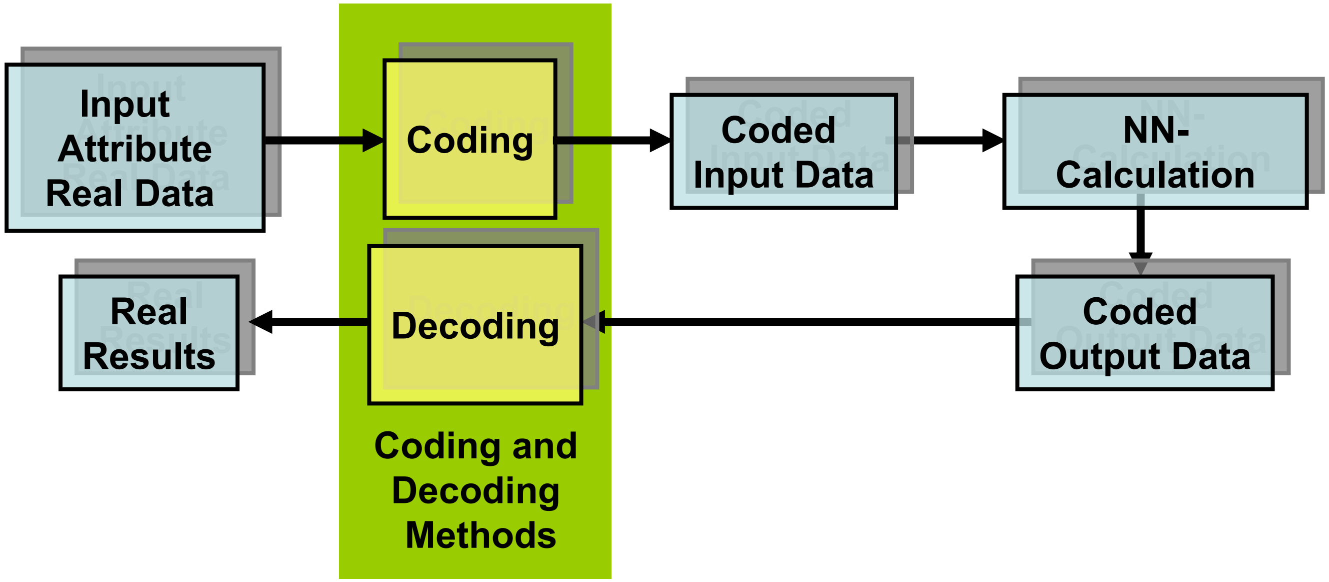
Artificial Neural Networks

Artificial Neural Networks | **Coding and decoding Methods**

- **Coding: transfer the continuous-valued or categorical data to the value domains:**

[0,1], [-1, 1] or {0,1}, { -1, 1}

- **Decoding : vice versa**



Artificial Neural Networks

Artificial Neural Networks

Coding and decoding Methods

Coding of continuous-valued data by using transformation function

original value	10	25	30	50

→ [0 , 1] →

new transf. value	0	0.375	0.50	1

X_n^{old} : original value (n=1, 2, … , N)

X_n^{new} : new, transferred value (n=1, 2, … , N)

X_{max}^{old} : maximal original value

X_{min}^{old} : minimal original value

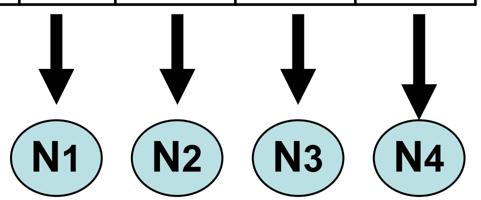
$$X_n^{new} = A * X_n^{old} + B$$

$$A = \frac{X_{max}^{new} - X_{min}^{new}}{X_{max}^{old} - X_{min}^{old}}$$

$$B = X_{min}^{new} - A * X_{min}^{old}$$

Artificial Neural Networks

Marital status				
single	**1**	0	0	0
married	0	**1**	0	0
widowed	0	0	**1**	0
divorced	0	0	0	**1**

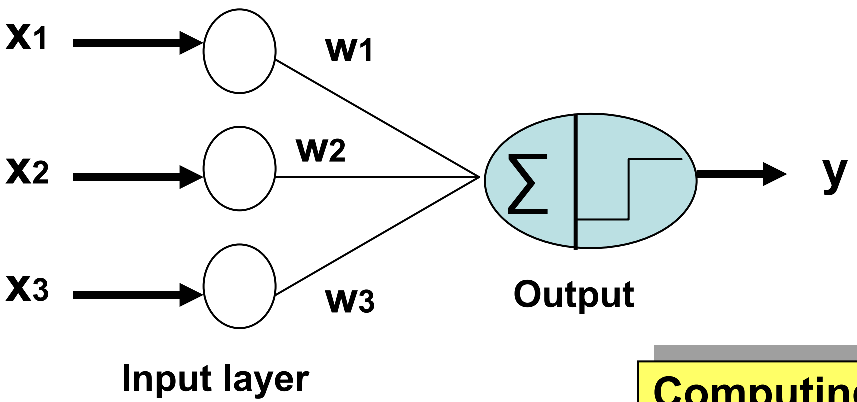


For each value of a nominal attribute an input neuron is necessary

Many nominal attributes with several values lead to a large number of input neurons

Perceptron:

A simple ANN architecture consists only of input and output nodes (no hidden layer):



Computing the output y in Perceptron:

$$\hat{y} = \begin{cases} 1, & \text{if } w_1 x_1 + w_2 x_2 + w_3 x_3 - t > 0 \\ -1, & \text{if } w_1 x_1 + w_2 x_2 + w_3 x_3 - t =< 0 \end{cases}$$

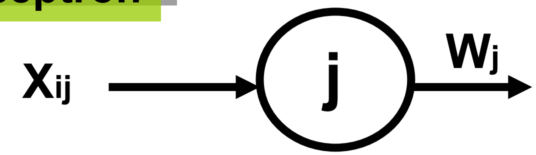
t : Bias factor

Generally:

$$\hat{y} = \text{sign}\,(w_1 x_1 + w_2 x_2 + \ldots w_j x_j + \ldots w_m x_m - t)$$

Artificial Neural Networks

Artificial Neural Networks | **Perceptron** | **Learning in Perceptron**



x_{11}	x_{12}	....	x_{1j}	....	x_{1m}	y_1
....						
x_{i1}	x_{i2}	....	x_{ij}	......	x_{im}	y_i
....						
x_{n1}	x_{n2}	....	x_{nj}	....	x_{nm}	y_n

$$w_j^{(k+1)} = w_j^{(k)} + ß\,(y_i - \hat{y}_i^{(k)})\,x_{ij}$$

$w_j^{(k+1)}$: new weight in iteration k + 1 of neuron j

$w_j^{(k)}$: old weight in iteration k of neuron j

ß : learning rate

y_i : observed output of the tuple i

$\hat{y}_i^{(k)}$: calculated output of the tuple i in iteration k

x_{ij} : value of attribute j of the tuple i

The above Learning Rule is based on Gradient Descent method by minimizing

$$\text{Min: } E = \tfrac{1}{2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Artificial Neural Networks

Artificial Neural Networks | **Perceptron** | **Learning in Perceptron**

$$\text{Min: } E = \frac{1}{2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$ **for tuple i** $$\text{Min: } E = \frac{1}{2} (y_i - \hat{y}_i)^2$$

$$\hat{y}_i = \sum_{j=1}^{m} w_{ij} \, x_{ij} \qquad \partial \hat{y}_i / \partial w_{ij} = x_{ij}$$

$$\partial E / \partial w_j = - (y_i - \hat{y}_i) * \partial \hat{y}_i / w_{ij} = - (y_i - \hat{y}_i) \, x_{ij} \qquad (1)$$

$$w_j \longrightarrow w_j - ß * \partial E(\mathbf{w}) / \partial w_j \qquad (2)$$

(1) and (2) lead to

$$\boxed{w_j^{(k+1)} = w_j^{(k)} + ß \, (y_i - \hat{y}_i^{(k)}) \, x_{ij}}$$

Artificial Neural Networks

Artificial Neural Networks | **Perceptron** | **Learning in Perceptron**

The relation $w_1 x_1 + w_2 x_2 + \ldots w_j x_j + \ldots + w_m x_m$ **is linear in w and x**

For linearly separable classification problems, the learning algorithm

$$w_j^{(k+1)} = w_j^{(k)} + ß\,(y_i - \hat{y}_i^{(k)})\,x_{ij}$$

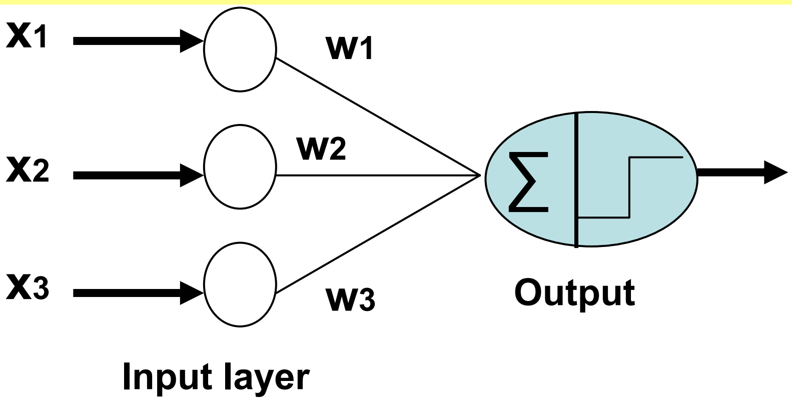
converges if the learning rate ß is sufficiently small

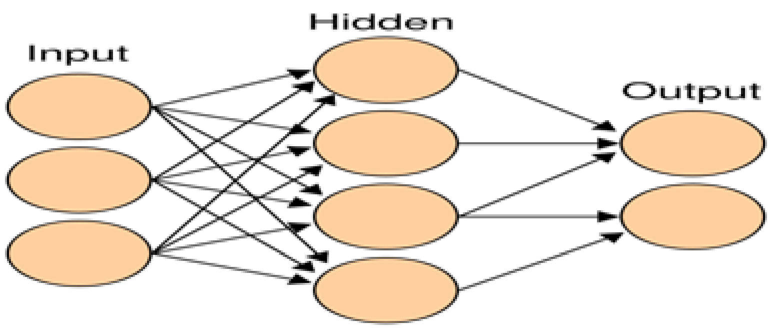
If the classes are not linearly separable, the above learning rule does not converge .

For such tasks ANN with hidden layers are necessary. Backpropagation is such an alternative.

Artificial Neural Networks

Backpropagation (backprop , BP) is the mostly used multilayer feed forward ANN, specially in the praxis . As we have seen, Perceptron can not handle the nonlinearly separable classes, but, BP can





The modification of the weights going to a hidden unit can not be performed by the method used in Perceptron because – in contrast to the output units – we have no information about the observed outputs of the hidden units. It means, it is not possible to determine the error term related to each hidden unit.

Solution: propagation of the errors of the output units backwards ⟶ Backpropagation

Artificial Neural Networks

Artificial Neural Networks | **Backpropagation Algorithms** | **Learning Steps**

Weight initialization : network weights are initialized randomly, usually from the interval [-1 , 1] or [- 0.5 , 0.5]

Calculation of the net input: As we have seen before the net Input of the neuron j is

$$Z_j = \sum_{i=1}^{p} X_{ij} w_{ij} + \theta_j \qquad (1)$$

Calculation of the output function: Using a logistic function we have got

$$O_j = \frac{1}{1 + e^{-aZ_j}} \qquad (2)$$

This function is differentiable and nonlinear

Repeat (1) and (2) until we the output layer is reached and included

Artificial Neural Networks

Artificial Neural Networks | Backpropagation Algorithms | Learning Steps

Backpropagate the error

δ_j

E: Error of the unit j

$$\partial E / \partial w_{ij} = \partial E / \partial z_j \ * \partial z_j / \partial w_{ij} = x_{ij} \ * \partial E / \partial z_j = x_{ij}\, \delta_j$$

A- Neuron *j* belongs to the *Output Layer*

Contribution to E by j : $E = \frac{1}{2} (T_j - O_j)^2$ **T_j: target Value of the neuron j**

$$\delta_j = \partial / \partial z_j \ E = - (T_j - O_j)\, \partial O_j / \partial z_j$$

and for a =1

$$\delta_j = - (T_j - O_j)\ (1 - O_j)\, O_j$$

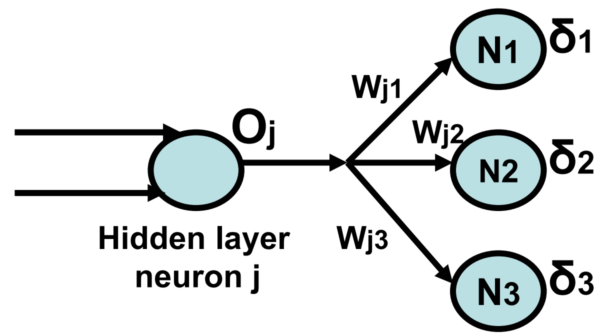
Therefore : regarding the Gradient Descent Learning Rule : $W_{ij} \longrightarrow W_{ij} + ß\, \delta_j\, X_{ij}$

Artificial Neural Networks

Artificial Neural Networks | **Backpropagation Algorithms** | **Learning Steps**

Backpropagate the error

Case B- Neuron *j* belongs to a *Hidden Layer*



$$E_j = \delta_j = O_j\,(1 - O_j)\sum_k \delta_k W_{jk}$$

O_j : Oput of the neuron j

δ_k : Error of the neuron k in the next layer

Wjk : weight of the connection from the neuron j to a neuron k in the next layer

Gradient Descent Learning Rule : $W_{ij} \rightarrow W_{ij} + ß\,\delta_j X_{ij}$

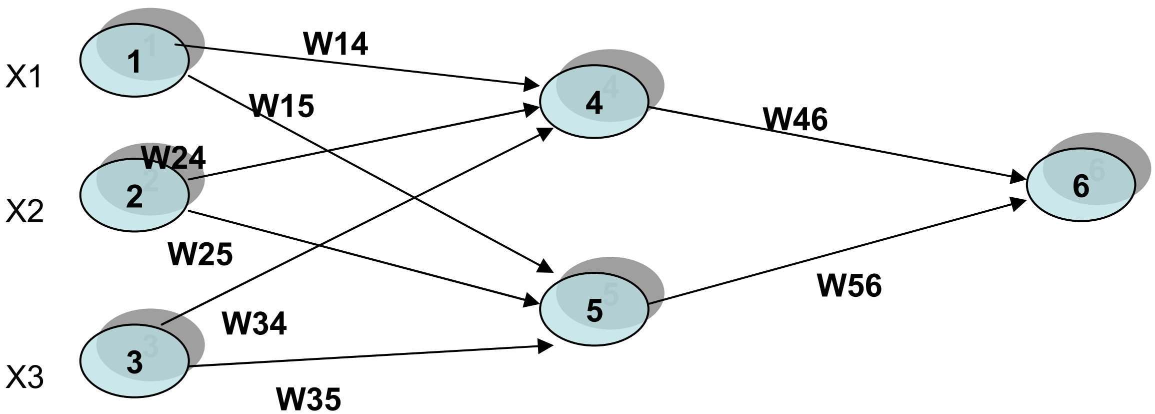
Bias updating for Neuron J : $\Theta j = \Theta j + ß\, Ej$

Artificial Neural Networks

Artificial Neural Networks Backpropagation Algorithms

Example

(source: Han et al (2006))



x1	x2	x3	w14	w15	w24	w25	w34	w35	w46	w56	Θ4	Θ5	Θ6
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

Unit	Net Input, Zj	Output, Oj
4	0.2 + 0 -0.5 -0.4 = -0.7	$1/(1 + e^{0.7}) = 0.332$
5	-0.3 + 0 +0.2 + 0.2 = 0.1	$1/(1 + e^{-0.1}) = 0.525$
6	(-0.3) (0.332) – (02) (0.525) + 0.1 = -0.105	$1/(1 + e^{0.105}) = 0.474$

Artificial Neural Networks

Artificial Neural Networks | **Backpropagation Algorithms** | **Example**

(source: Han et al (2006))

Unit j	Error j
6	(0.474) (1- 0.474) (1- 0.474) = 0.1311
5	(0.525) (1-0.525) (0.1311) (-0.2) = - 0.0065
4	(0.332) (1- 0.332) (0.1311) (-0.3) = - 0.0087

Weight or bias	New value
W46	-0.3 + (0.9) (0.1311) (0.332) = - 0.261
W56	- 0.2 + (0.9) (0.1311) (0.525) = - 0.138
W14	0.2 + (0.9) (-0.0087) (1) = 0.192
W15	- 0.3 + (0.9) (-0.0065) (1) = -0.306
W24	0.4 + (0.9) (-0.0087) (0) = 0.4
W25	0.1 + (0.9) (-0.0065) (0) = 0.1
W34	-0.5 + (0.9) (-0.0087) (1) = - 0.508
W35	0.2 + (0.9) (-0.0065) (1) = 0.194
Θ6	0.1 + (0.9) (0.1311) = 0.218
Θ5	0.2 + (0.9) (-0.0065) = 0.194
Θ4	-0.4 + (0.9) (-0.0087) = - 0.408

Weakness and Strength of ANN

Based on: http://www-sal.cs.uiuc.edu/~hanj/bk2/

- **Strength**
  - High tolerance to noisy data
  - Well-suited for continuous-valued inputs and outputs
  - Successful on a wide array of real-world data
  - Techniques have recently been developed for the extraction of rules from trained neural networks

- **Weakness**
  - Long training time, specially for the datasets with a large number of categorical attributes
  - Require a number of parameters typically best determined empirically, e.g., the network topology or ``structure."
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of ``hidden units" in the network

- Introduction
- Rise and fall of Neural Networks
- Input function of neuron
- Activation function of neuron
- Output (function) of neuron
- Feed-Forward networks
- Feedback networks
- Learning process
- Coding and decoding methods
- Perceptron
- Backpropagation
- Weakness and Strength of ANN