

Data Mining (Mineria de Dades)

Lluís A. Belanche
belanche@lsi.upc.edu

Soft Computing Research Group

Dept. de Llenguatges i Sistemes Informàtics (Software department)

Universitat Politècnica de Catalunya

2011-2012

Data Mining (Part II)

Contents

1. Association Rules
2. Bayesian decision theory
3. Bayesian classifiers: LDA/QDA
4. Bayesian classifiers: Naïve Bayes
5. Bayesian classifiers: Nearest Neighbours
6. Neural Networks: Multilayer Perceptrons
7. Support Vector Machines
8. Ensemble Learning

Artificial neural networks (I)

Computational Intelligence methods are inspired (or motivated) biologically and oriented to problem-solving: fuzzy systems, artificial neural networks, evolutionary algorithms, swarm intelligence, artificial immune systems, rough systems, ant colony optimization, etc.

CI methods are an abstraction of known or working principles of Nature.

A **directed graph** (DG) is a structure composed by a set of nodes and a set of labelled directed segments (vertexes) that connect the nodes.

An artificial neural network (ANN) is a information-processing structure, distributed and parallel, that takes the form of a DG.

Introduction to ANNs

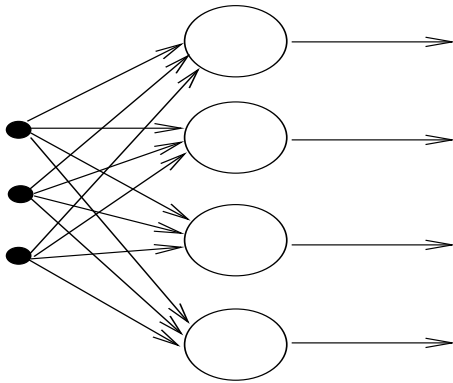
Jargon

Graph	ANN
node	neuron/unit
vertex	connection
label	weight/parameter
layout	architecture

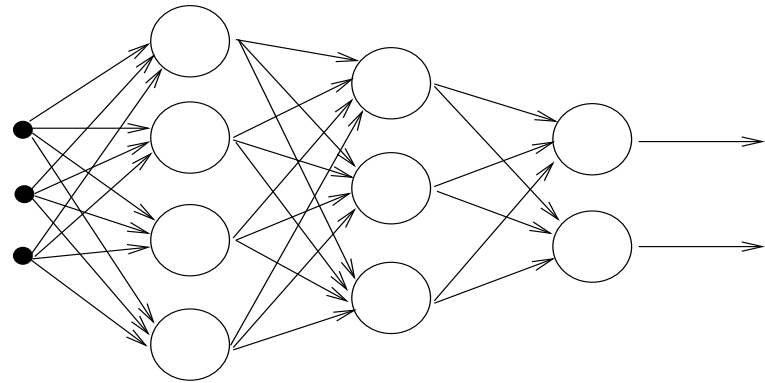
Problems tackled by ANNs include: associative memories, regression, classification, clustering, time-series prediction, function optimization, dimensionality reduction and function identification.

Introduction to ANNs

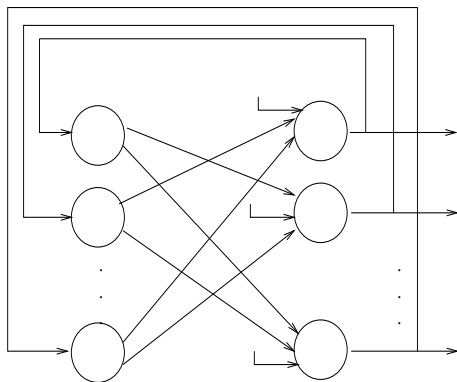
The architecture (I)



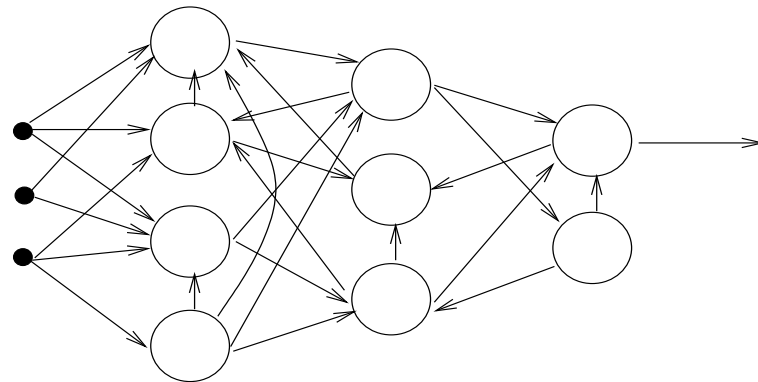
single-layer feed-forward



multi-layer feed-forward



single-layer recurrent



multi-layer recurrent

Introduction to ANNs

The architecture (II)

A **layer** is a collection of neurons sharing a common input vector (and usually computing the same parametric function). If they are *not* connected with one another we speak of a *feed-forward* layer; otherwise it is a *recurrent* layer.

An ANN is **recurrent** if it contains a recurrent layer; otherwise it is a **feed-forward** network. A recurrent network represents a dynamical system; a feed-forward network represents a function. Alternate characterization of feed-forward network: its graph contains no cycles.

Inter-layer connectivity can be total or partial.

Introduction to ANNs

The architecture (and III)

We say that a feed-forward network is **simple** (and call it SFFN) if it can be decomposed in consecutive layers, each of which is fully connected to the next one and there are no other connections (called *skip-layer*).

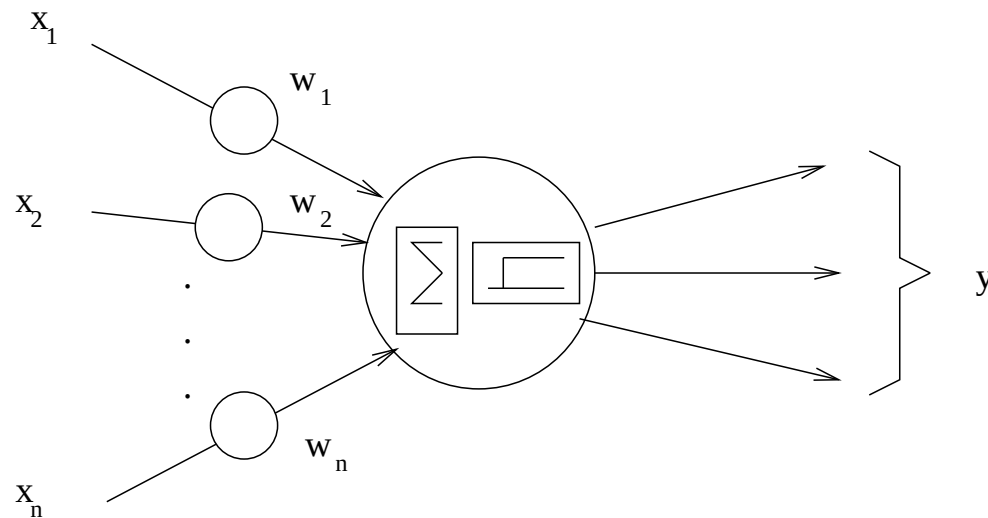
The **output layer** is the last layer in the direction of the arrows. All other layers in a SFFN are called hidden.

A **hidden** neuron is a neuron of a hidden layer.

Introduction to ANNs

The neuron

An (artificial) neuron is a local computing device that gets an *input* vector, combines this vector with a vector of local parameters (its *weights*) and with local information (e.g., the neuron's previous state) and *outputs* a scalar quantity as a result.



This output can be delivered as part of the input of another neuron or to the neuron itself (self connection).

Artificial neural networks (I)

The simplest choice is a linear combination of the inputs:

$$y(\mathbf{x}) = \sum_{i=1}^n w_i x_i + w_0 \quad \text{What kind of network gives rise to this function?}$$

Which can be extended to multiple outputs

$$y_k(\mathbf{x}) = \sum_{i=1}^n w_{ki} x_i + w_{k0}, k = 1, \dots, m \quad \text{And now?}$$

Finally, let us add a non-linearity to the output:

$$y_k(\mathbf{x}) = g \left(\sum_{i=1}^n w_{ki} x_i + w_{k0} \right), k = 1, \dots, m \quad \text{And now?}$$

Artificial neural networks (I)

For convenience, define $\mathbf{x} = (1, x_1, \dots, x_n)^t$, $\mathbf{w}_k = (w_{k0}, w_{k1}, \dots, w_{kn})^t$.

We have $y_k(\mathbf{x}) = g\left(\sum_{j=0}^n w_{kj}x_j\right)$, $1 \leq k \leq m$.

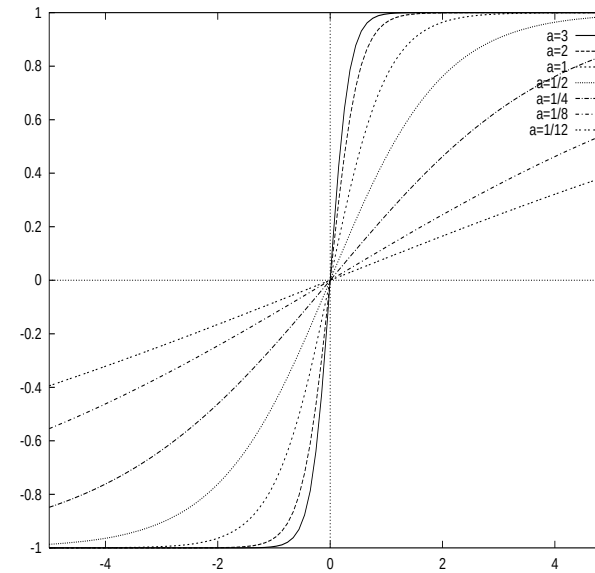
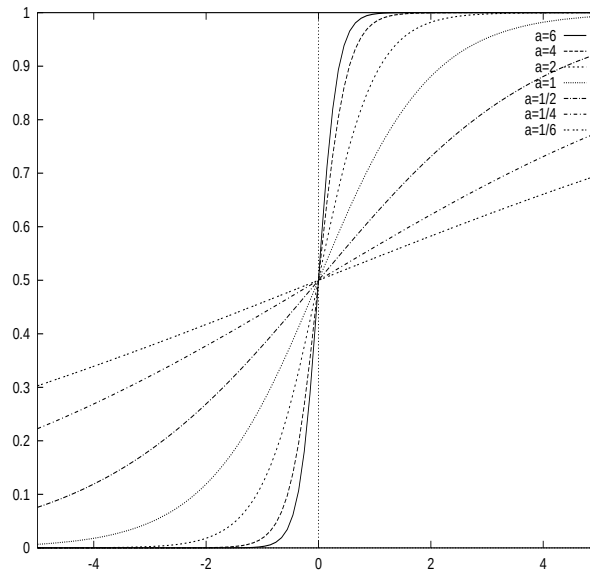
Define now the weight matrix $W_{(n+1) \times m}$ by gathering all the weight vectors by columns and introduce the notation $g[\cdot]$ to mean that g is applied component-wise. The network then computes $y(\mathbf{x}) = g[W^t \mathbf{x}]$.

The activation function g is most often a sigmoidal one (differentiable, non-negative (or non-positive) bell-shaped first derivative; horizontal asymptotes in $\pm\infty$), as:

logistic : $l_\beta(z) = \frac{1}{1+e^{-\beta z}} \in (0, 1)$

tanh : $\tanh_\beta(z) = \frac{e^{\beta z} - e^{-\beta z}}{e^{\beta z} + e^{-\beta z}} \in (-1, 1), \beta > 0$

Artificial neural networks (I)



Left: logistic

Right: tanh

(Note a is β)

Artificial neural networks (I)

The Delta rule

We wish to fit this function to a set of learning examples $S = \{(\mathbf{x}_\mu, \mathbf{t}_\mu)\}_{\mu=1,\dots,p}$, where $\mathbf{x}_\mu \in \mathbb{R}^n, \mathbf{t}_\mu \in \mathbb{R}^m$.

Define the (empirical) mean-square error of the network as:

$$E_{emp}(W) = \frac{1}{2} \sum_{\mu=1}^p \sum_{k=1}^m (t_{\mu k} - y_k(\mathbf{x}_\mu))^2$$

Artificial neural networks (I)

The Delta rule

Let $f : \mathbb{R}^r \rightarrow \mathbb{R}$ differentiable; we wish to minimize it by making changes in its variables. Then the increment in each variable is proportional to the corresponding derivative: $x_i(t+1) = x_i(t) + \Delta x_i(t)$, with

$$\Delta x_i(t) = -\alpha \left. \frac{\partial f}{\partial x_i} \right|_{\mathbf{x}=\mathbf{x}(t)}, \quad \alpha > 0, i = 1, \dots, r$$

Illustration with $r = 1$. Let $f(x) = 3x^2 + x - 4$ and take $\alpha = 0,05$. We have $f'(x) = 6x + 1$. Then $x(0) = 1, x(1) = x(0) - \alpha f'(1) = 1 - 0,05 \cdot 7 = 0,65, x(2) = 0,65 - 0,05 \cdot 4,9 = 0,405, \dots$. We find $\lim_{i \rightarrow \infty} x(i) = -\frac{1}{6}$.

Artificial neural networks (I)

The Delta rule

In our case, the function to be minimized is the empirical error and the variables are the weights $\mathbf{w}$ of the network:

$$\Delta w_{ij}(t) = -\alpha \left. \frac{\partial E_{emp}(W)}{\partial w_{ij}} \right|_{\mathbf{w}=\mathbf{w}(t)}, \quad \alpha > 0, i = 1, \dots, m, j = 0, \dots, n$$

We have

$$\frac{\partial E_{emp}(W)}{\partial w_{ij}} = - \sum_{\mu=1}^p (t_{\mu i} - y_i(\mathbf{x}_{\mu})) g'(\hat{y}_{\mu i}) x_{\mu j}$$

where $\delta_{\mu i} \equiv t_{\mu i} - y_i(\mathbf{x}_{\mu})$ is called the *delta* and $\hat{y}_{\mu i} \equiv \sum_{j=0}^n w_{ij} x_{\mu j}$ (sometimes called the *net input*); in other words, $y_i(\mathbf{x}_{\mu}) = g(\hat{y}_{\mu i})$

Artificial neural networks (I)

The Delta rule

Therefore $\Delta w_{ij}(t) = \alpha \sum_{\mu=1}^p (t_{\mu i} - y_i(\mathbf{x}_{\mu})) g'(\hat{y}_{\mu i}) x_{\mu j}$ (evaluated at $\mathbf{w}(t)$) is the Delta rule (aka α -LMS Learning Rule)

The network thus represents a linear regressor where the regression coefficients are estimated iteratively.

This is a form of learning (because of the adaptation to the example data) but as yet it is not incremental: we need all the examples from the beginning (this is sometimes referred to as a “batch” rule).

This rule represents the most analyzed & applied simple learning rule.

Artificial neural networks (I)

The Delta rule

In the “on-line” version of the rule, we begin with W arbitrary and apply:

$$\Delta w_{ij}(t) = \alpha_t (t_{\mu i} - y_i(\mathbf{x}_\mu)) g'(\hat{y}_{\mu i}) x_{\mu j}$$

evaluated at $\mathbf{w}(t)$. At each learning step t , the input vector $\mathbf{x}_\mu$ is drawn at random from $\{\mathbf{x}_1, \dots, \mathbf{x}_p\}$. It can be shown, that if $\sum_{t \geq 0} \alpha_t = \infty$ and $\sum_{t \geq 0} \alpha_t^2 < \infty$, then $\mathbf{w}(t)$ converges to the global minimum $\mathbf{w}^*$ asymptotically in the mean square sense: $\lim_{t \rightarrow \infty} \|\mathbf{w}(t) - \mathbf{w}^*\|^2 = 0$.

One such procedure is $\alpha_t = \frac{\alpha}{t+1}$, with $\alpha > 0$

Artificial neural networks (I)

The Delta rule

The function $E_{emp}(W)$ is quadratic in W because of the linear relation between y_i and w_i . In fact, it defines a convex hyperparaboloidal surface with a single minimum W^* (the global minimum).

The constant α controls the stability and speed of convergence. If chosen sufficiently small, the gradient descent procedure asymptotically converges towards W^* , regardless of the initial matrix.

A sufficient condition is $0 < \alpha < \frac{2}{\eta_{max}}$, where η_{max} is the largest eigenvalue of the input autocorrelation matrix $E[\mathbf{x}\mathbf{x}^t] \approx \mathbf{X}\mathbf{X}^t$. This number may not be feasible; alternatively, one may use $\alpha < \frac{2}{\sum_{\mu=1}^p \|\mathbf{x}_{\mu}\|^2}$,

since $\eta_{max} < Tr(E[\mathbf{x}\mathbf{x}^t]) \approx Tr(\mathbf{X}\mathbf{X}^t) = \sum_{\mu=1}^p \|\mathbf{x}_{\mu}\|^2$

Artificial neural networks (I)

The Delta rule

The fit can be under tighter control using regularization:

$$E_{emp}^{\lambda}(W) = E_{emp}(W) + \lambda \|W\|^2, \quad \lambda > 0, \quad \|W\|^2 = \sum_{ij} w_{ij}^2$$

In this context, this technique is also known as “weight decay”, because it leads to the new updating receipt:

$$\Delta w_{ij}(t) = -\alpha \left[\left. \frac{\partial E_{emp}(W)}{\partial w_{ij}} \right|_{\mathbf{w}=\mathbf{w}(t)} + \lambda w_{ij}(t) \right], \quad \alpha, \lambda > 0, \quad i = 1, \dots, m, \quad j = 0, \dots, n$$

Artificial neural networks (I)

The Delta rule

The value of λ is quite often chosen by resampling techniques.

How do we stop the process in practice?

1. When the number of iterations t reaches a predetermined maximum
2. When the relative error reaches a predetermined minimum