

1 Basic concepts of Neural Networks and Fuzzy Logic Systems

Inspirations based on course material by Professors Heikki Koivo <http://www.control.hut.fi/Kurssi/AS-74.115/Material/> and David S. Touretzki <http://www-2.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15782-s04/slides/> are duly acknowledged

Neural networks and Fuzzy Logic Systems are often considered as a part of **Soft Computing** area:

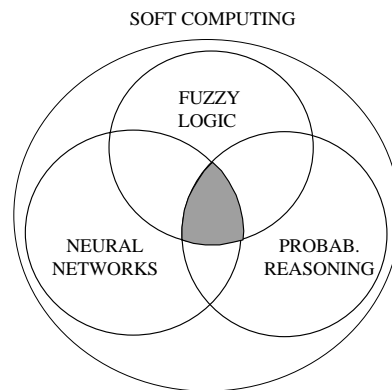


Figure 1-1: Soft computing as a composition of fuzzy logic, neural networks and probabilistic reasoning.

Intersections include

- neuro-fuzzy systems and techniques,
- probabilistic approaches to neural networks (especially classification networks) and fuzzy logic systems,
- and Bayesian reasoning.

Neuro-Fuzzy systems

- We may say that neural networks and fuzzy systems try to emulate the operation of human brain.
- Neural networks concentrate on the structure of human brain, i.e., on the “hardware” emulating the basic functions, whereas fuzzy logic systems concentrate on “software”, emulating fuzzy and symbolic reasoning.

Neuro-Fuzzy approach has a number of different connotations:

- The term “Neuro-Fuzzy” can be associated with hybrid systems which act on two distinct subproblems: a neural network is utilized in the first subproblem (e.g., in signal processing) and a fuzzy logic system is utilized in the second subproblem (e.g., in reasoning task).
- We will also consider system where both methods are closely coupled as in the Adaptive Neuro-Fuzzy Inference Systems (anfis)

Neural Network and Fuzzy System research is divided into two basic schools

- Modelling various aspects of human brain (structure, reasoning, learning, perception, etc)
- Modelling artificial systems and related data: pattern clustering and recognition, function approximation, system parameter estimation, etc.

Neural network research in particular can be divided into

- Computational Neuroscience
 - Understanding and modelling operations of single neurons or small neuronal circuits, e.g. minicolumns.
 - Modelling information processing in actual brain systems, e.g. auditory tract.
 - Modelling human perception and cognition.
- Artificial Neural Networks used in
 - Pattern recognition
 - adaptive control
 - time series prediction, etc.

Areas contributing to Artificial neural networks:

- Statistical Pattern recognition
- Computational Learning Theory
- Computational Neuroscience
- Dynamical systems theory
- Nonlinear optimisation

We can say that in general

- Neural networks and fuzzy logic systems are parameterised computational nonlinear algorithms for numerical processing of data (signals, images, stimuli).
- These algorithms can be either implemented on a general-purpose computer or built into a dedicated hardware.
- Knowledge is acquired by the network/system through a learning process.
- The acquired knowledge is stored in internal parameters (weights).

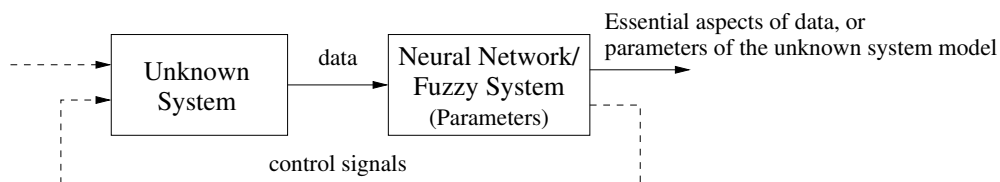
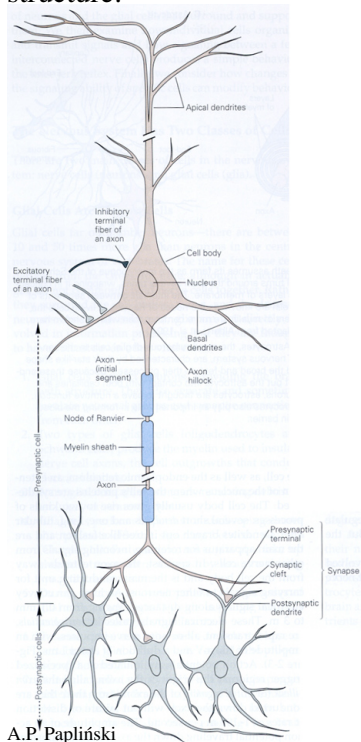


Figure 1-2: System as a data generator

1.1 Biological Fundamentals of Neural Networks

A typical neuron, or nerve cell, has the following structure:



from Kandel, Schwartz and Jessel, *Principles of Neural Science*

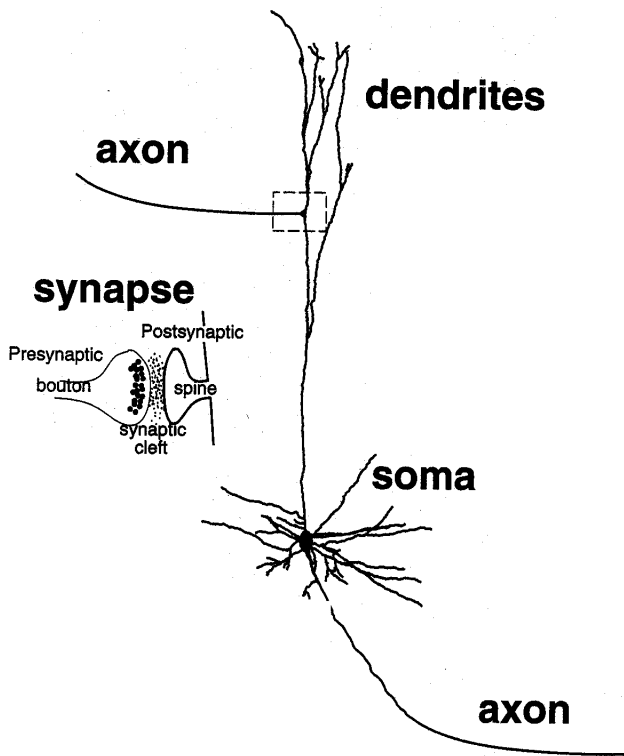
- Most neurons in the vertebrate nervous system have several main features in common.
- The **cell body** contains the nucleus, the storehouse of genetic information, and gives rise to two types of cell processes, **axons** and **dendrites**.
- Axons, the transmitting element of neurons, can vary greatly in length; some can extend more than 3m within the body. Most axons in the central nervous system are very thin ($0.2 \dots 20 \mu\text{m}$ in diameter) compared with the diameter of the cell body ($50 \mu\text{m}$ or more).
- Many axons are insulated by a fatty sheath of myelin that is interrupted at regular intervals by the nodes of Ranvier.
- The **action potential**, the cell's conducting signal, is initiated either at the **axon hillock**, the initial segment of the axon, or in some cases slightly farther down the axon at the first nod of Ranvier.
- Branches of the axon of one neuron (the presynaptic neuron) transmit signals to another neuron (the postsynaptic cell) at a site called the **synapse**.
- The branches of a single axon may form synapses with as many as 1000 other neurons.
- Whereas the axon is the output element of the neuron, the **dendrites** (apical and basal) are input elements of the neuron. Together with the cell body, they receive synaptic contacts from other neurons.

1-5

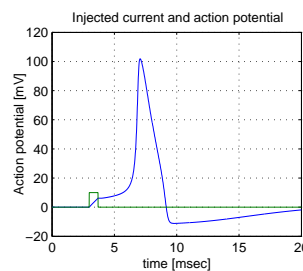
Simplified functions of these very complex in their nature “building blocks” of a neuron are as follow:

- The synapses are elementary signal processing devices.
 - A synapse is a biochemical device which converts a pre-synaptic electrical signal into a chemical signal and then back into a post-synaptic electrical signal.
 - The input pulse train has its amplitude modified by parameters stored in the synapse. The nature of this modification depends on the type of the synapse, which can be either inhibitory or excitatory.
- The postsynaptic signals are aggregated and transferred along the dendrites to the nerve cell body.
- The cell body generates the output neuronal signal, activation potential, which is transferred along the axon to the synaptic terminals of other neurons.
- The frequency of firing of a neuron is proportional to the total synaptic activities and is controlled by the synaptic parameters (weights).
- The pyramidal cell can receive 10^4 synaptic inputs and it can fan-out the output signal to thousands of target cells — the connectivity difficult to achieve in the artificial neural networks.

Another microscopic view of typical neuron of the mammalian cortex (a pyramidal cell):



- Note the cell body or soma, dendrites, synapses and the axon.
- Neuro-transmitters (information-carrying chemicals) are released pre-synaptically, floats across the synaptic cleft, and activate receptors postsynaptically.
- According to Calaj’s “neuron-doctrine” information carrying signals come into the dendrites through synapses, travel to the cell body, and activate the axon. Axonal signals are then supplied to synapses of other neurons.



1.2 A simplistic model of a biological neuron

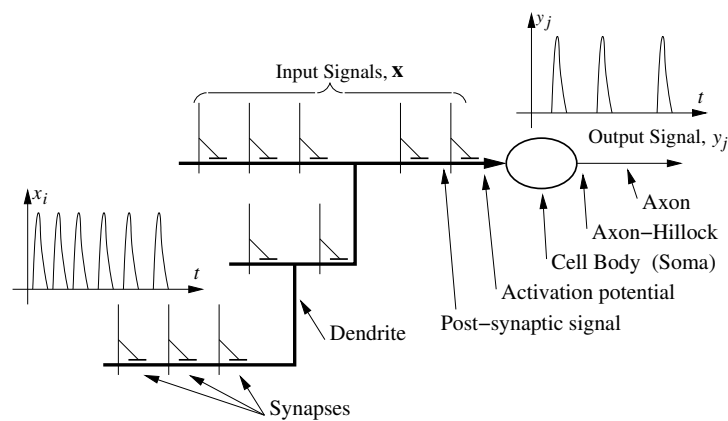


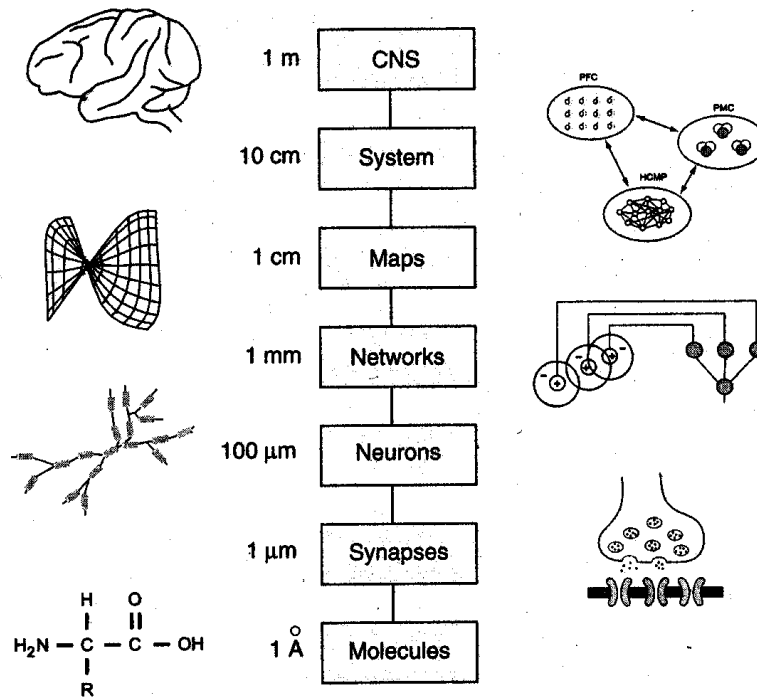
Figure 1-3: Conceptual structure of a biological neuron

Basic characteristics of a biological neuron:

- data is coded in a form of instantaneous frequency of pulses
- synapses are either excitatory or inhibitory
- Signals are aggregated (“summed”) when travel along dendritic trees
- The cell body (neuron output) generates the output pulse train of an average frequency proportional to the total (aggregated) post-synaptic activity (activation potential).

Levels of organization of the nervous system

from T.P.Trappenberg, *Fundamentals of Computational Neuroscience*, Oxford University Press, 2002



A.P. Papliński

1-9

A few good words about the brain

Adapted from S. Haykin, *Neural Networks – a Comprehensive Foundation*, Prentice Hall, 2nd ed., 1999

The brain is a highly complex, non-linear, parallel information processing system. It performs tasks like pattern recognition, perception, motor control, many times faster than the fastest digital computers.

- Biological neurons, the basic building blocks of the brain, are slower than silicon logic gates. The neurons operate in milliseconds which is about six–seven orders of magnitude slower than the silicon gates operating in the sub-nanosecond range.
- The brain makes up for the slow rate of operation with two factors:
 - a huge number of nerve cells (neurons) and interconnections between them. The number of neurons is estimated to be in the range of 10^{10} with $60 \cdot 10^{12}$ synapses (interconnections).
 - A function of a biological neuron seems to be much more complex than that of a logic gate.
- The brain is very energy efficient. It consumes only about 10^{-16} joules per operation per second, comparing with 10^{-6} J/oper-sec for a digital computer.

Brain plasticity:

- At the early stage of the human brain development (the first two years from birth) about 1 million synapses (hard-wired connections) are formed per second.
- Synapses are then modified through the learning process (plasticity of a neuron).
- In an adult brain plasticity may be accounted for by the above two mechanisms: creation of new synaptic connections between neurons, and modification of existing synapses.

A.P. Papliński

1-10

What can you do with Artificial Neural Networks (aNN)

Adapted from Neural Nets FAQ: <ftp://ftp.sas.com/pub/neural/FAQ.html>

- In principle, NNs can compute any computable function, i.e., they can do everything a normal digital computer can do.
- In practice, NNs are especially useful for **classification** and **function approximation**/mapping problems which are tolerant of some imprecision, which have lots of training data available, but to which hard and fast rules (such as those that might be used in an expert system) cannot easily be applied.
- Almost any mapping between vector spaces can be approximated to arbitrary precision by feedforward NNs (which are the type most often used in practical applications) if you have enough data and enough computing resources.
- To be somewhat more precise, feedforward networks with a single hidden layer, under certain practically-satisfiable assumptions are statistically consistent estimators of, among others, arbitrary measurable, square-integrable regression functions, and binary classification devices.
- NNs are, at least today, difficult to apply successfully to problems that concern manipulation of symbols and memory. And there are no methods for training NNs that can magically create information that is not contained in the training data.

Who is concerned with NNs?

- Computer scientists want to find out about the properties of non-symbolic information processing with neural nets and about learning systems in general.
- Statisticians use neural nets as flexible, nonlinear regression and classification models.
- Engineers of many kinds exploit the capabilities of neural networks in many areas, such as signal processing and automatic control.
- Cognitive scientists view neural networks as a possible apparatus to describe models of thinking and consciousness (high-level brain function).
- Neuro-physiologists use neural networks to describe and explore medium-level brain function (e.g. memory, sensory system, motorics).
- Physicists use neural networks to model phenomena in statistical mechanics and for a lot of other tasks.
- Biologists use Neural Networks to interpret nucleotide sequences.
- Philosophers and some other people may also be interested in Neural Networks for various reasons.

1.3 Taxonomy of neural networks

- From the point of view of their **active** or **decoding** phase, artificial neural networks can be classified into **feedforward** (static) and **feedback** (dynamic, recurrent) systems.
- From the point of view of their **learning** or **encoding** phase, artificial neural networks can be classified into **supervised** and **unsupervised** systems.
- Practical terminology often mixes up the above two aspects of neural nets.

Feedforward supervised networks

This networks are typically used for function approximation tasks. Specific examples include:

- Linear recursive least-mean-square (LMS) networks
- Multi Layer Perceptron (MLP) aka Backpropagation networks
- Radial Basis networks

Feedforward unsupervised networks

This networks are used to extract important properties of the input data and to map input data into a “representation” domain. Two basic groups of methods belong to this category

- Hebbian networks performing the **Principal Component Analysis** of the input data, also known as the Karhunen-Loeve Transform.
- Competitive networks used to performed **Learning Vector Quantization**, or tessellation/clustering of the input data set.
- Self-Organizing Kohonen Feature Maps also belong to this group.

Recurrent networks

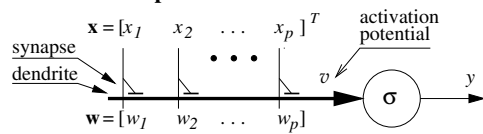
These networks are used to learn or process the temporal features of the input data and their internal state evolves with time. Specific examples include:

- Hopfield networks
- Associative Memories
- Adaptive Resonance networks

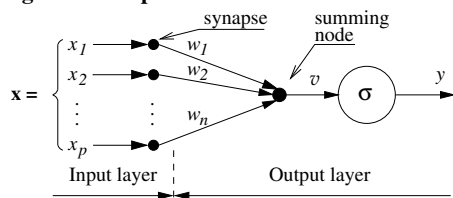
1.4 Models of artificial neurons

Three basic graphical representations of a single p -input (p -synapse) neuron:

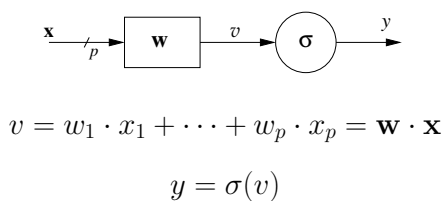
a. Dendritic representation



b. Signal flow representation



c. Block-diagram representation



Artificial neural networks are nonlinear information (signal) processing devices which are built from interconnected elementary processing devices called neurons.

An artificial neuron is a p -input single-output signal processing element which can be thought of as a simple model of a non-branching biological neuron.

From a **dendritic representation** of a single **neuron** we can identify p **synapses** arranged along a linear **dendrite** which aggregates the synaptic activities, and a neuron body or axon-hillock generating an output signal.

The **pre-synaptic activities** are represented by a p -element **column vector** of input (afferent) signals

$$\mathbf{x} = [x_1 \dots x_p]^T$$

In other words the space of input patterns is p -dimensional.

Synapses are characterised by adjustable parameters called weights or synaptic strength parameters. The weights are arranged in a p -element **row vector**:

$$\mathbf{w} = [w_1 \dots w_p]$$

- In a **signal flow** representation of a neuron p synapses are arranged in a layer of input **nodes**. A dendrite is replaced by a single summing node. Weights are now attributed to **branches** (connections) between input nodes and the summing node.
- Passing through synapses and a dendrite (or a summing node), input signals are aggregated (combined) into the **activation potential**, which describes the total **post-synaptic activity**.
- The activation potential is formed as a linear combination of input signals and synaptic strength parameters, that is, as an **inner product** of the weight and input vectors:

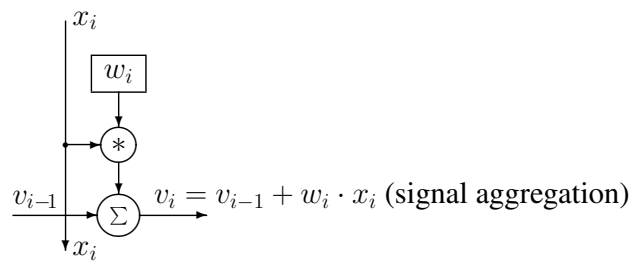
$$v = \sum_{i=1}^p w_i x_i = \mathbf{w} \cdot \mathbf{x} = [w_1 \ w_2 \ \dots \ w_p] \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \quad (1.1)$$

- Subsequently, the activation potential (the total post-synaptic activity) is passed through an **activation function**, $\sigma(\cdot)$, which generates the output (efferent) signal:

$$y = \sigma(v) \quad (1.2)$$

- The activation function is typically a saturating function which normalises the total post-synaptic activity to the standard values of output (axonal) signal.
- The **block-diagram** representation encapsulates basic operations of an artificial neuron, namely, aggregation of pre-synaptic activities, eqn (1.1), and generation of the output signal, eqn (1.2)

A **single synapse** in a dendritic representation of a neuron can be represented by the following block-diagram:



In the synapse model we can identify:

- a storage for the synaptic weight,
- augmentation (multiplication) of the pre-synaptic signal with the weight parameter, and
- the dendritic aggregation of the post-synaptic activities.

A synapse is classified as

- **excitatory**, if a corresponding weight is positive, $w_i > 0$, and as
- **inhibitory**, if a corresponding weight is negative, $w_i < 0$.

It is sometimes convenient to add an additional parameter called **threshold**, θ or **bias** $b = -\theta$. It can be done by fixing one input signal to be constant. Than we have

$$x_p = +1, \text{ and } w_p = b = -\theta$$

With this addition, the activation potential is calculated as:

$$\hat{v} = \sum_{i=1}^p w_i x_i = v - \theta, \quad v = \sum_{i=1}^{p-1} w_i x_i$$

where \hat{v} is the augmented activation potential.

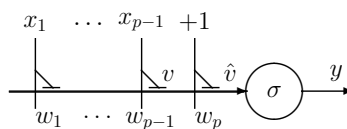


Figure 1-4: A single neuron with a biasing input

1.5 Types of activation functions

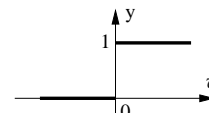
Typically, the activation function generates either **unipolar** or **bipolar** signals.

A linear function: $y = v$.

Such linear processing elements, sometimes called ADALINES, are studied in the theory of linear systems, for example, in the “traditional” signal processing and statistical regression analysis.

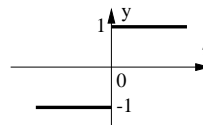
A step function

unipolar:

$$y = \sigma(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$


Such a processing element is traditionally called **perceptron**, and it works as a threshold element with a binary output.

bipolar:

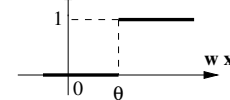
$$y = \sigma(v) = \begin{cases} +1 & \text{if } v \geq 0 \\ -1 & \text{if } v < 0 \end{cases}$$


A step function with bias

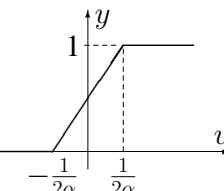
The bias (threshold) can be added to both, unipolar and bipolar step function. We then say that a neuron is “fired”, when the synaptic activity exceeds the threshold level, θ . For a unipolar case, we have:

$$y = \sigma(v) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} \geq \theta \\ 0 & \text{if } \mathbf{w} \cdot \mathbf{x} < \theta \end{cases}$$

(The McCulloch-Pitts perceptron — 1943)



A piecewise-linear function

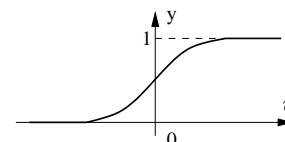
$$y = \sigma(v) = \begin{cases} 0 & \text{if } v \leq -\frac{1}{2\alpha} \\ \alpha v + \frac{1}{2} & \text{if } |v| < \frac{1}{2\alpha} \\ 1 & \text{if } v \geq \frac{1}{2\alpha} \end{cases}$$


- For small activation potential, v , the neuron works as a linear combiner (an ADALINE) with the gain (slope) α .
- For large activation potential, v , the neuron saturates and generates the output signal either 0 or 1.
- For large gains $\alpha \rightarrow \infty$, the piecewise-linear function is reduced to a step function.

Sigmoidal functions

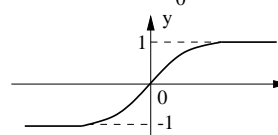
unipolar:

$$\sigma(v) = \frac{1}{1 + e^{-v}} = \frac{1}{2}(\tanh(v/2) + 1)$$



bipolar:

$$\sigma(v) = \tanh(\beta v) = \frac{2}{1 + e^{-2\beta v}} - 1$$



The parameter β controls the slope of the function.

The hyperbolic tangent (bipolar sigmoidal) function is perhaps the most popular choice of the activation function specifically in problems related to function mapping and approximation.

Radial-Basis Functions

- Radial-basis functions arise as optimal solutions to problems of interpolation, approximation and regularisation of functions. The optimal solutions to the above problems are specified by some integro-differential equations which are satisfied by a wide range of nonlinear differentiable functions (S. Haykin, *Neural Networks – a Comprehensive Foundation*, Ch. 5).
- Typically, Radial-Basis Functions $\varphi(\mathbf{x}; \mathbf{t}_i)$ form a family of functions of a p -dimensional vector \mathbf{x} , each function being centered at point \mathbf{t}_i .
- A popular simple example of a Radial-Basis Function is a symmetrical multivariate Gaussian function which depends only on the distance between the current point, \mathbf{x} , and the center point, \mathbf{t}_i , and the variance parameter σ_i :

$$\varphi(\mathbf{x}; \mathbf{t}_i) = G(\|\mathbf{x} - \mathbf{t}_i\|) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{t}_i\|^2}{2\sigma_i^2}\right)$$

where $\|\mathbf{x} - \mathbf{t}_i\|$ is the norm of the distance vector between the current vector \mathbf{x} and the centre, \mathbf{t}_i , of the symmetrical multidimensional Gaussian surface.

- The spread of the Gaussian surface is controlled by the variance parameter σ_i .

Two concluding remarks:

- In general, the smooth activation functions, like sigmoidal, or Gaussian, for which a continuous derivative exists, are typically used in networks performing a function approximation task, whereas the step functions are used as parts of pattern classification networks.
- Many learning algorithms require calculation of the derivative of the activation function (see the assignments/practical 1).

1.6 A layer of neurons

- Neurons as in sec. 1.4 can be arrange into a **layer** of neurons.
- A **single layer neural network** consists of m neurons each with the same p input signals.
- Similarly to a single neuron, the neural network can be represented in all **three** basic forms: **dendritic**, **signal-flow**, and **block-diagram** form
- From the **dendritic representation** of the neural network it is readily seen that a layer of neurons is described by a $m \times p$ matrix W of synaptic weights.
- Each row of the **weight matrix** is associated with one neuron.

Operations performed by the network can be described as follows:

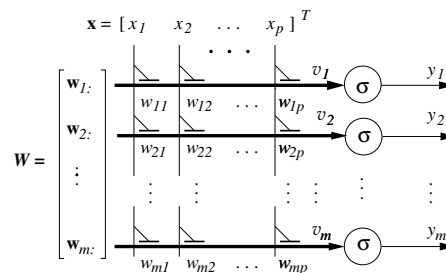
$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} = \begin{bmatrix} w_{11} & \cdots & w_{1p} \\ w_{21} & \cdots & w_{2p} \\ \vdots & \cdots & \vdots \\ w_{m1} & \cdots & w_{mp} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix} ; \quad \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} \sigma(v_1) \\ \sigma(v_2) \\ \vdots \\ \sigma(v_m) \end{bmatrix}$$

A.P. Papliński

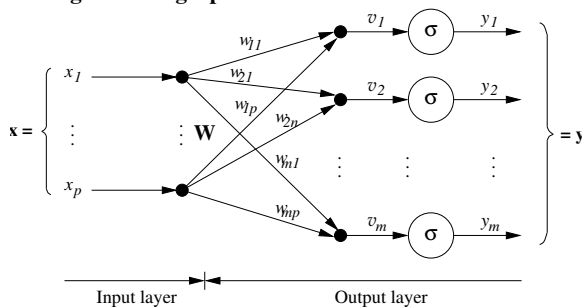
$$\mathbf{v} = W \cdot \mathbf{x} ; \quad \mathbf{y} = \sigma(W \cdot \mathbf{x}) = \sigma(\mathbf{v}) ; \quad \mathbf{v} \text{ is a vector of activation potentials.}$$

1–23

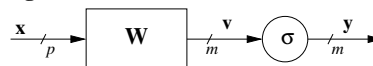
a. Dendritic graph



b. Signal-flow graph



c. Block-diagram



From the **signal-flow graph** it is visible that each weight parameter w_{ij} (synaptic strength) is now related to a connection between nodes of the input layer and the output layer.

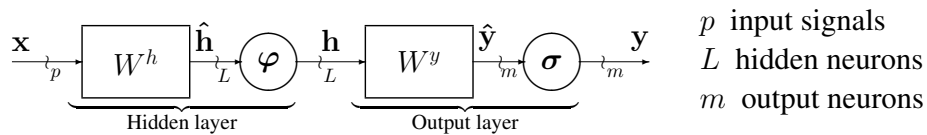
Therefore, the name **connection strengths** for the weights is also justifiable.

The **block-diagram** representation of the single layer neural network is the most compact one, hence most convenient to use.

1.7 Multi-layer feedforward neural networks

Connecting in a serial way layers of neurons presented in sec. 1.6 we can build multi-layer feedforward neural networks also known as **Multilayer Perceptrons (MLP)**.

The most popular neural network seems to be the one consisting of **two layers of neurons** as in the following block-diagram



In order to avoid a problem of counting an input layer, the two-layer architecture is referred to as a **single hidden layer** neural network.

Input signals, \mathbf{x} , are passed through synapses of the hidden layer with connection strengths described by the **hidden weight matrix**, W^h , and the L **hidden activation signals**, $\hat{\mathbf{h}}$, are generated.

The hidden activation signals are then normalised by the functions ψ into the L **hidden signals**, \mathbf{h} .

Similarly, the hidden signals, \mathbf{h} , are first, converted into m **output activation signals**, $\hat{\mathbf{y}}$, by means of the output weight matrix, W^y , and subsequently, into m **output signals**, \mathbf{y} , by means of the functions σ . Hence

$$\mathbf{h} = \varphi(W^h \cdot \mathbf{x}), \quad \mathbf{y} = \sigma(W^y \cdot \mathbf{h})$$

If needed, one of the input signals and one of the hidden signals can be constant to form a biasing signals.

Functions φ and σ can be identical.

1.8 Static and Dynamic Systems — General Concepts

Static systems

Neural networks considered in previous sections belong to the class of **static** systems which can be fully described by a set of m -functions of p -variables as in Figure 1–5.

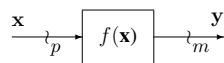


Figure 1–5: A static system: $\mathbf{y} = f(\mathbf{x})$

The defining feature of the static systems is that they are **time-independent** — current outputs depends only on the current inputs in the way specified by the mapping function, f .

Such a function can be very complex.

Dynamic systems — Recurrent Neural Networks

In the dynamic systems, the current output signals depend, in general, on current and past input signals.

There are two equivalent classes of dynamic systems: continuous-time and discrete-time systems.

The dynamic neural networks are referred to as **recurrent neural networks**.

1.9 Continuous-time dynamic systems

- Continuous-time dynamic systems operate with signals which are functions of a continuous variable, t , interpreted typically as **time**. A **spatial variable** can be also used.
- Continuous-time dynamic systems are described by means of **differential equations**. The most convenient yet general description uses only **first-order** differential equations in the following form:

$$\dot{\mathbf{y}}(t) = f(\mathbf{x}(t), \mathbf{y}(t)) \quad (1.3)$$

where

$$\dot{\mathbf{y}}(t) \stackrel{\text{df}}{=} \frac{d\mathbf{y}(t)}{dt}$$

is a vector of time derivatives of output signals.

- In order to model a dynamic system, or to obtain the output signals, the **integration** operation is required. The dynamic system of eqn (1.3) is illustrated in Figure 1–6.

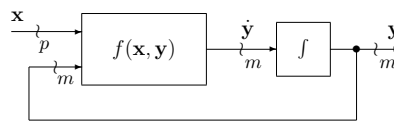


Figure 1–6: A continuous-time dynamic system: $\dot{\mathbf{y}}(t) = f(\mathbf{x}(t), \mathbf{y}(t))$

- It is evident that **feedback** is inherent to dynamic systems.

1.9.1 Discrete-time dynamic systems

- Discrete-time dynamic systems operate with signals which are functions of a discrete variable, n , interpreted typically as time, but a discrete spatial variable can be also used.
- Typically, the discrete variable can be thought of as a **sampled** version of a continuous variable:

$$t = n \cdot t_s ; \quad t \in \mathcal{R}, \quad n \in \mathcal{N}$$

and t_s is the **sampling time**

- Analogously, discrete-time dynamic systems are described by means of **difference equations**.
- The most convenient yet general description uses only **first-order** difference equations in the following form:

$$\mathbf{y}(n + 1) = f(\mathbf{x}(n), \mathbf{y}(n)) \quad (1.4)$$

where $\mathbf{y}(n + 1)$ and $\mathbf{y}(n)$ are the predicted (future) value and the current value of the vector \mathbf{y} , respectively.

- In order to model a discrete-time dynamic system, or to obtain the output signals, we use the **unit delay** operator, $D = z^{-1}$ which originates from the z-transform used to obtain analytical solutions to the difference equations.
- Using the delay operator, we can re-write the first order difference equation into the following operator form:

$$z^{-1}\mathbf{y}(n + 1) = \mathbf{y}(n)$$

which leads to the structure as in Figure 1-7.

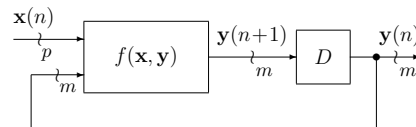


Figure 1-7: A discrete-time dynamic system: $\mathbf{y}(n+1) = \mathbf{f}(\mathbf{x}(n), \mathbf{y}(n))$

- Notice that **feedback** is also present in the discrete dynamic systems.

1.9.2 Example: A continuous-time generator of a sinusoid

- As a simple example of a continuous-time dynamic system let us consider a linear system which generates a sinusoidal signal.
- Eqn (1.3) takes on the following form:

$$\dot{\mathbf{y}}(t) = A \cdot \mathbf{y}(t) + B \cdot \mathbf{x}(t) \quad (1.5)$$

where $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$; $A = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix}$; $B = \begin{bmatrix} 0 \\ b \end{bmatrix}$; $\mathbf{x} = \delta(t)$

$\delta(t)$ is the unit impulse which is non-zero only for $t = 0$ and is used to describe the initial condition.

- In order to show that eqn (1.5) really describes the sinusoidal generator we re-write this equation for individual components. This yields:

$$\begin{aligned} \dot{y}_1 &= \omega y_2 \\ \dot{y}_2 &= -\omega y_1 + b \delta(t) \end{aligned} \quad (1.6)$$

- Differentiation of the first equation and substitution of the second one gives the second-order linear differential equation for the output signal y_1 :

$$\ddot{y}_1 + \omega^2 y_1 = \omega b \delta(t)$$

- Taking the Laplace transform and remembering that $\mathcal{L}\delta(t) = 1$, we have:

$$y_1(s) = b \frac{\omega}{s^2 + \omega^2}$$

- Taking the inverse Laplace transform we finally have

$$y_1(t) = b \sin(\omega t)$$

- The internal structure of the generator can be obtained from eqns (1.7) and illustrated using the dendritic representation as in Figure 1–8.

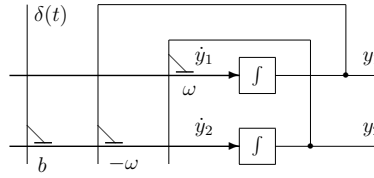


Figure 1–8: A continuous-time sinusoidal generator

- The generator can be thought of as a simple example of a **linear recurrent neural network** with the fixed weight matrix of the form:

$$W = [B \ A] = \begin{bmatrix} 0 & 0 & \omega \\ b & -\omega & 0 \end{bmatrix}$$

- The weights were designed appropriately rather than “worked out” during the learning procedure.

1.9.3 Example: A discrete-time generator of a sinusoid

- It is possible to build a discrete-time version of the sinusoidal generator using difference equations of the general form as in eqn (1.4):

$$\mathbf{y}(n+1) = A \cdot \mathbf{y}(n) + B \cdot \mathbf{x}(n) \quad (1.7)$$

where

$$A = \begin{bmatrix} \cos \Omega & \sin \Omega \\ -\sin \Omega & \cos \Omega \end{bmatrix}; \quad B = \begin{bmatrix} 0 \\ b \end{bmatrix}; \quad \mathbf{x}(n) = \delta(n)$$

- This time we take the z-transform directly of eqn (1.7), which gives:

$$(zI - A)\mathbf{y}(z) = B; \quad \text{where } I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \text{and } \mathcal{Z}\delta(n) = 1$$

Hence

$$\mathbf{y}(z) = (zI - A)^{-1}B$$

and subsequently

$$\mathbf{y}(z) = \left(\begin{bmatrix} z - \cos \Omega & \sin \Omega \\ -\sin \Omega & z - \cos \Omega \end{bmatrix} \begin{bmatrix} 0 \\ b \end{bmatrix} \right) / (z^2 - 2z \cos \Omega + 1)$$

Extracting the first component, we have

$$y_1(z) = \frac{b \sin \Omega}{z^2 - 2z \cos \Omega + 1}$$

Taking the inverse z-transform finally yields

$$y_1(n + 1) = b \sin(\Omega n)$$

which means that the discrete-time dynamic system described by eqn (1.7) generates a sinusoidal signal.

- The structure of the generator is similar to the previous one and is presented in Figure 1–9.

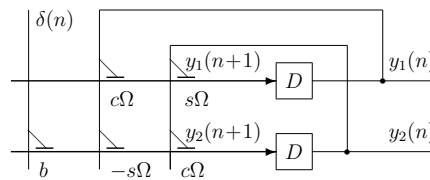


Figure 1–9: A discrete-time sinusoidal generator

- This time the weight matrix is:

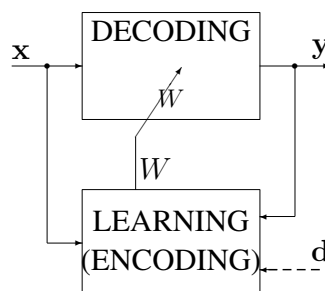
$$W = [B \ A] = \begin{bmatrix} 0 & c\Omega & s\Omega \\ b & -s\Omega & c\Omega \end{bmatrix}$$

where

$$s\Omega = \sin \Omega, \quad \text{and} \quad c\Omega = \cos \Omega$$

1.10 Introduction to learning

- In the previous sections we concentrated on the **decoding** part of a neural network assuming that the **weight matrix**, W , is given.
- If the weight matrix is satisfactory, during the decoding process the network performs some useful task it has been design to do.
- In simple or specialised cases the weight matrix can be pre-computed, but more commonly it is obtained through the **learning** process.
- Learning is a dynamic process which modifies the weights of the network in some desirable way. As any dynamic process learning can be described either in the continuous-time or in the discrete-time framework.
- A neural network with learning has the following structure:



- The learning can be described either by differential equations (continuous-time)

$$\dot{W}(t) = L(W(t), \mathbf{x}(t), \mathbf{y}(t), \mathbf{d}(t)) \quad (1.8)$$

or by the difference equations (discrete-time)

$$W(n+1) = L(W(n), \mathbf{x}(n), \mathbf{y}(n), \mathbf{d}(n)) \quad (1.9)$$

where \mathbf{d} is an external teaching/supervising signal used in **supervised learning**.

- This signal is not present in networks employing **unsupervised learning**.
- The discrete-time learning law is often used in a form of a **weight update** equation:

$$\begin{aligned} W(n+1) &= W(n) + \Delta W(n) \\ \Delta W(n) &= L(W(n), \mathbf{x}(n), \mathbf{y}(n), \mathbf{d}(n)) \end{aligned} \quad (1.10)$$