

### III. Neural Networks and Control Problems in a Robotic Field

As stated earlier, the neural network can be applied to control fields. We can apply neural networks to the following robotic fields:

1. Position/trajectory control
2. Force control
3. Sensing and perception
4. Planning (path, trajectory, graph, task, etc.)

The neural network can give more intelligent control than conventional theories, that cannot be applied to nonlinear dynamics, unknown dynamics and parameters of the environment, and others. The neural network approach tackles these difficult problems of system identification and control using its self-organizing capabilities.

The valuable characteristics of the neural network for robotics are as follows:

1. It can learn training data which can then be used system identification.
2. It can execute nonlinear mapping.
3. It exhibits parallel processing capability.
4. It can filter noise.
5. It has generalization capabilities for data interpolation.
6. The neural network can be used as a black box without a priori knowledge of the system.

## Part II. Formal Overview of Artificial Neural Networks.

### Biological neural networks

In order to fully understand the concepts and structure of artificial neural networks, it is necessary to review the structure and mechanism of the biological neural system.

A *neuron* (nerve cell) is a special biological cell that processes information. Figure 4 portrays schematically the structure of a neuron.

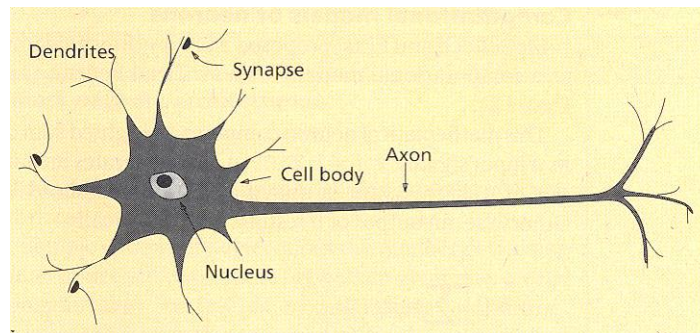


Figure 4. Schematic representation of a biological neuron

It consists of a cell body, or *soma*, and two types of out-reaching tree-like branches: the *axon* and the *dendrites*. The cell body has a nucleus that contains information about hereditary traits and a plasma that holds the molecular equipment for producing material needed by the neuron. A neuron receives signals (impulses) from other neurons through its dendrites (receivers) and transmits signals generated by its cell body along the axon (transmitters), which eventually branches into strands and sub-strands. At the terminals of these strands are the *synapses*. A synapse is an elementary structure and functional unit between two neurons (an axon strand of one neuron and a dendrite of another). When the impulse reaches the synapse's terminal, certain chemicals called neurotransmitters are released. The neurotransmitters diffuse across the synaptic gap. To enhance or inhibit, depending on the type of the synapse, the receptor neuron's own tendency to emit electrical impulses. The synapse's effectiveness can be adjusted by the signals passing through it so that the synapses can learn from the activities in which they participate. This dependence on history acts as a memory, which is possibly responsible for human memory.

The cerebral cortex in humans is a large flat sheet of neurons about 2 to 3 millimeters thick with a surface area of approximately 2,200  $\text{cm}^2$ , about twice the area of a standard computer keyboard. The cerebral cortex contains approximately  $10^{11}$  neurons, which is approximately the number of stars in the Milky Way [2]. Neurons are massively connected, much more complex and dense than telephone networks. It is estimated that each neuron is connected to  $10^3$  to  $10^4$  other neurons. In total, the human brain contains approximately  $10^{14}$  to  $10^{15}$  interconnections.

Neurons communicate through a very short train of pulses, typically milliseconds in duration. The *message* is modulated on the pulse-transmission frequency. This frequency can vary from a few several hundred hertz. Complex perceptual decisions such as face recognition are typically made by humans within a few hundred milliseconds. These decisions are made by a network of neurons whose operational speed is only a few milliseconds. It follows that the computations cannot take more than approximately 100 serial stages. Thus, the brain runs parallel programs that are approximately 100 steps long for such perceptual tasks. This is known as the *hundred step rule* [3]. The same timing considerations show that the amount of information sent from one neuron to another must be very small (a few bits). This implies that critical information is not transmitted directly, but captured and distributed in the interconnections – hence the name, *connectionist* model, used to describe ANNs.

### Computational Models of Neurons

McCulloch and Pitts [1] proposed a binary threshold unit as a computational model for an artificial neuron as shown in Figure 5.

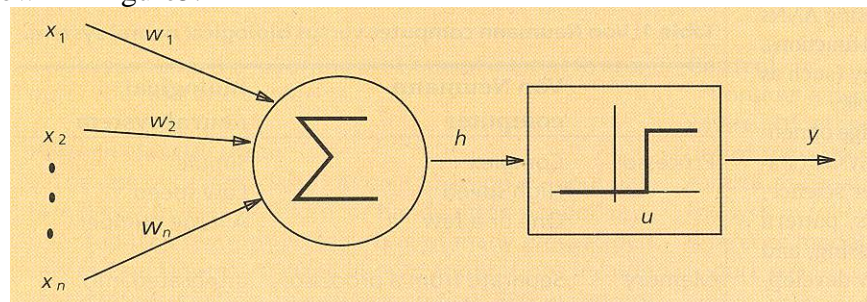


Figure 5. McCulloch-Pitts model of a neuron

This mathematical neuron computes a weighted sum of its  $n$  input signals  $x_j, j = 1, 2, \dots, n$ , and generates an output of 1 if this sum is above a certain threshold  $u$ . Otherwise, an output of 0 results. Mathematically,

$$y = \theta \left( \sum_{j=1}^n w_j x_j - u \right) \quad (1)$$

where  $\theta(\cdot)$  is a unit step function at 0, and  $w_j$  is the synapse weight associated with the  $j$ th input. For simplicity of notation, the threshold  $u$  is considered as another weight  $w_0 = -u$  attached to the neuron with a constant input  $x_0 = 1$ . Positive weights correspond to *excitatory* synapses, while negative weights model *inhibitory* ones. The McCulloch-Pitt neuron has been generalized in many ways. An obvious one is to use activation functions other than the threshold function, such as piecewise linear, sigmoid, or Gaussian, as shown in Figure 6.

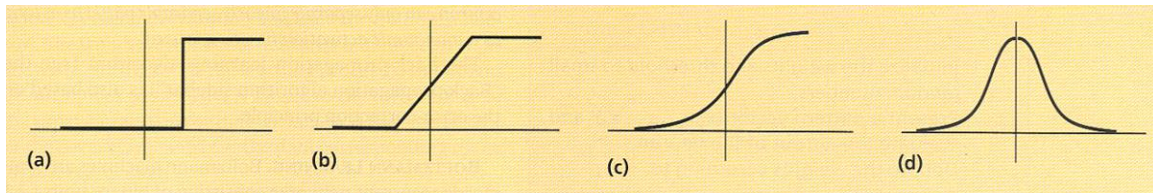


Figure 6. Different types of activation functions: (a) threshold, (b) piecewise linear, (c) sigmoid, and (d) Gaussian.

The sigmoid function is by far the most frequently used in ANNs. It is a strictly increasing function that exhibits smoothness and the desired asymptotic properties. The standard sigmoid function is the logistic function, defined by

$$g(x) = \frac{1}{(1 + \exp \{-\beta x\})} \quad (2)$$

where  $\beta$  is the slope parameter.

### Network Architecture

Essentially, ANNs are weighted directed graphs in which artificial neurons are nodes and directed edges (with weights) are connections between neuron outputs and neuron inputs. Based on the connection pattern (architecture), ANNs can be grouped into two categories (see Figure 6.):

1. *feed-forward networks*, in which graphs have no loops, and
2. *recurrent (or feedback) networks*, in which loops occur because of feedback connections.

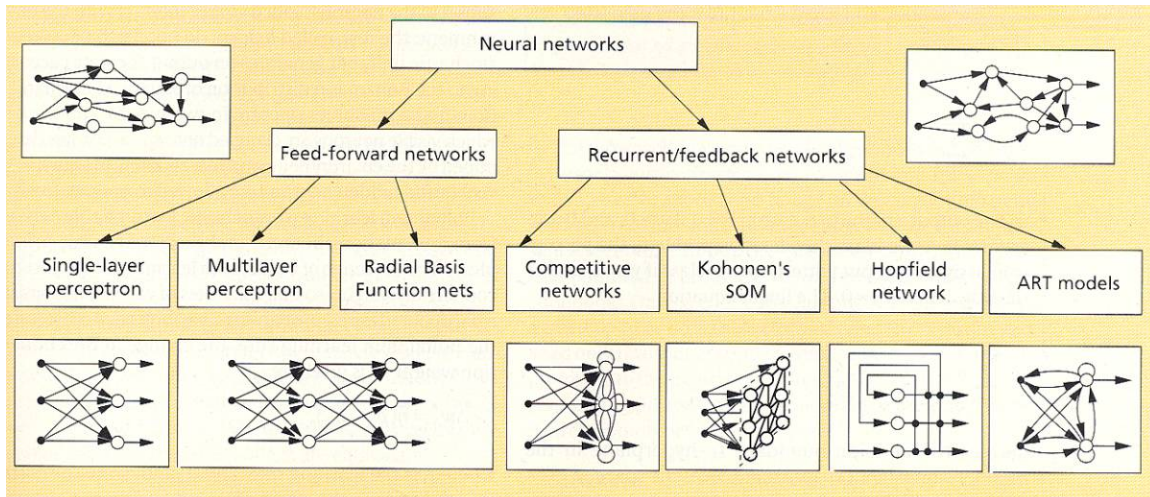


Figure 6. A taxonomy of feed-forward and recurrent/feedback network architectures.

In the most common families of feed-forward networks, called multilayer perceptron, neurons are organized into layers that have unidirectional connections between them. Different connectivities yield different network behaviors. Feed-forward networks are *static*, that is, they produce only one set of output values rather than a sequence of values from a given input. Feed-forward networks are memory-less in the sense that their response to an input is independent of the previous network state. On the other hand, recurrent or feedback networks are dynamic systems. When a new input pattern is presented, the neuron outputs are computed. Because of the feedback paths, the inputs to each neuron are modified, which leads the network to enter a new state.

Different network architectures require appropriate learning algorithms. These will be outlined in the next section.

## Learning

A fundamental trait of intelligence is the ability to learn. Although a precise definition of learning is difficult to formulate, a learning process in the ANN context can be viewed as the problem of updating network architecture and connection weights so that a network can efficiently program a specific task.

The network usually must learn the connection weights from available training patterns. Performance is improved over time by iteratively updating the weights in the network. ANNs' ability to automatically *learn from examples* makes them attractive for the solution of many numerical problems. ANNs appear to learn underlying *rules* (like input-output relationships) from the given collection of representative examples. This is one of the major advantages of neural networks over traditional expert systems. The design a learning process for a neural network includes the following components:

1. A model of the environment in which a neural network operates, that is, it is necessary to know what information is available to the network. We refer to this model as a *learning paradigm*.
2. An understanding of how the network weights are updated, that is, which *learning rules* govern the updating process. A *learning algorithm* refers to a procedure in which learning rules are used for adjusting the weights.

There are the following learning paradigms available to ANNs:

- Supervised learning- This is learning with a “teacher” process. The network is provided with a correct answer (output) for every input pattern. Weights are determined to allow the network to produce answers as close as possible to the known correct answers.
- Unsupervised learning – Learning without a “teacher”. Here, the network does not require a correct answer associated with each input pattern in the training data set. It explores the underlying structure of the data, or correlations between patterns in the data, and organizes patterns into categories from these correlations.
- Hybrid learning – Combines supervised and unsupervised learning. Part of the weights are usually determined through supervised learning , while the others are obtained through unsupervised learning.

*Learning theory* must address three fundamental and practical issues associated with learning from samples:

1. Capacity – It is concerned with the number of patterns that can be stored, and with the type of functions and decision boundaries a network can form.
2. Sample complexity – This determines the number of training patterns needed to train the network to guarantee a valid generalization. Too few patterns may result in “over-fitting” , that is, the network performs well on the training data set, but poorly on independent test patterns drawn from the same distribution as the training patterns, as shown in Figure 7.

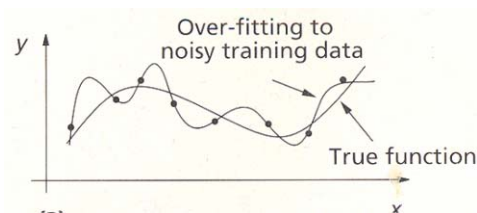


Figure 7. Over-fitt

3. Computational complexity – The time required for a learning algorithm to estimate a solution from training patterns. Many existing learning algorithms have high computational complexity. Designing efficient algorithms is a very active research topic.

Table I summarizes the properties and characteristics of the most well known learning algorithms.

Table I. Well-known learning paradigms and their characteristics

Paradigm	Learning rule	Architecture	Learning algorithm	Task
Supervised	Error-correction	Single- or multilayer perceptron	Perceptron learning algorithms	Pattern classification
			Back-propagation	Function approximation
	Boltzmann	Recurrent	Adaline and Madaline	Prediction, control
			Boltzmann learning algorithm	Pattern classification
Hebbian	Multilayer feed-forward	Linear discriminant analysis	Data analysis	
Unsupervised	Competitive	Competitive	Learning vector quantization	Pattern classification
			ART network	ARTMap
	Error-correction	Multilayer feed-forward	Sammon's projection	Within-class categorization
			Hebbian	Feed-forward or competitive
Competitive	Competitive	Hopfield Network	Associative memory learning	Data compression
		Kohonen's SOM	Kohonen's SOM	Associative memory
		ART networks	ART1, ART2	Categorization
Hybrid	Error-correction and competitive	RBF network	RBF learning algorithm	Data compression
				Categorization
				Pattern classification
				Function approximation
				Prediction, control

There are four basic types of learning rules: (1) error correction, (2) Boltzmann, (3) Hebbian, and (4) competitive learning. (*Continued in the July bulletin*).

