ARTIFICIAL NEURAL NETWORKS

Since the invention of the digital computer, the human being has attempted to create machines which directly interact with the real world without his intervention. In this sense the Artificial Intelligence, in general, and particularly the Artificial Neural Networks (ANN’s) represent an alternative for endowing to the computers one of the characteristics that makes the difference between humans and other alive beings, the intelligence.

An artificial neural network is an abstract simulation of a real nervous system and its study corresponds to a growing interdisciplinary field which considers the systems as adaptive, distributed and mostly nonlinear, three of the elements found in the real applications.

The ANN’s are used in many important engineering and scientific applications, some of these are, signal enhancement, noise cancellation, pattern classification, system identification, prediction, and control. Besides, they are used in many commercial products, such as modems, image processing and recognition systems, speech recognition, and bio-medical instrumentation, among others.

This is an e- learning course supported by the ALPHA program, edu.GL.LA project, and it was designed by professors and students of the Instituto Tecnológico de Toluca in México.

Contact

Professor Dr. Eduardo Gasca A. egasca@ittoluca.edu.mx

Goals

You will learn how to design and how to train supervised and unsupervised artificial neural networks, particularly, you will use the Multilayer Perceptron (Backpropagation) and the Self-Organizing Map (SOM) networks for making classification and clustering tasks, respectively.

Contents

1. Introduction to Artificial Neural Networks
   1.1. Introduction
   1.2. General characteristics of the human brain
   1.3. What an artificial neural network is.
   1.4. Benefits of the artificial neural networks
   1.5. Applications of the artificial neural networks
   1.6. Computational model of the neuron
   1.7. Structure of a neural net (topology)
2. Learning Methods
2.1. Introduction
2.2. Supervised learning
   2.2.1. Error correction learning
   2.2.2. Reinforcement of learning
   2.2.3. Stochastic learning
2.3. Unsupervised learning
   2.3.1. Hebbian learning
   2.3.2. Competitive learning
3. Perceptron
   3.1. Introduction
   3.2. Convergence Theorem of the Perceptron
   3.3. Virtues and limitations
   3.4. Adaline and Madaline
4. Multilayer Perceptron
   4.1. Introduction
   4.2. Algorithm of Backpropagation
   4.3. Learning rate and momentum
   4.4. Algorithms of Second order
   4.5. Pruning
5. Self-Organizing Map (SOM)
   5.1. Introduction
   5.2. Topology
   5.3. Learning rule
   5.4. Operation stage of SOM network
   5.5. Geometrical interpretation
   5.6. Bibliography

The Introduction will contain historical topics and applications of the nets. The supervised and unsupervised, learning methods, are going to be explained in the second unit. For historical reasons, the Perceptron is introduced in the third unit; there the computational neurons are showed too in the same unit. The Backpropagation and SOM algorithms are presented in the last two units. In those parts, commercial software will be used for creating the networks. In the course, some classification tasks will be done.

Style

✓ Self study based on online material and recommended books
✓ Homework at the end of the units 4, 5, and 6.
✓ An exam at the end of the course.
✓ Access to the teacher via e-mail.
✓ Students workload: 60 h (2x30 h), equivalent to 2 credit points (ECTS)
Syllabus

Participants

✓ Students at IFGI (Münster), ISEGI (Portugal), and UJI (Spain)
✓ Basic knowledge about Differential Calculus and Analytical Geometry are necessary.

Organization

✓ Start and end: May 1st. - June 15th. 2006.
✓ Exam: June 15th. 2006.
✓ Contact: via Internet (course page)

Successful participations

✓ Done homework
✓ Final exam passed.

Literature

Suggested books


Kung, S. Y., *Digital Neural Networks*, Ed. Prentice Hall.

Credits

General Coordinator

Dr. Eduardo Gasca A.

Technical Assistant

M. in C. María Antonia Olmos S.
M. in C. Vicente García J.
Edgar Villafaña G.

Flash

Carlos A. Alcántar S.

Systems programmer

Fernando Pérez I.

Translation

Anton Iñaki Cuevas Ramos.
Introduction to Artificial Neural Networks

Table of contents:

1. Introduction.
2. General characteristics of the human brain.
3. What an artificial neural network is
5. Applications of the artificial neural networks.
7. Structure of a neural net (topology).

Objectives:

At the end of his unit the student:
- Will describe the basic behaviour of neurons.
- Will be aware of the ideological basis of the artificial neural networks.
- Will learn the origins of the artificial neural networks.
- Will recognize the benefits of the artificial neural networks.
- Will know some applications of the artificial neural networks.
- Will identify different structures of the artificial neural networks.

Readings:

- Haykin, S., Neural Networks: A comprehensive foundation, pages 1 – 23.
- Kung, S. Y., Digital Neural Networks, pages 1-4
- http://www.faqs.org/faqs/ai-faq/neural-nets, check part one particularly the questions: What is a neural network (NN)? What can you do with an NN and what not? Who is concerned with NNs?

1. Introduction

Nowadays there is a new field of computational science that integrates the different methods of problem solving that cannot be so easily described without an algorithmic traditional focus. These methods, in one way or another, have their origin in the emulation, more or less intelligent, of the behaviour of the biological systems.

It is a new way of computing denominated Artificial Intelligence, which through different methods is capable of managing the imprecisions and uncertainties that appear when trying to solve problems related to the real world, offering strong solution and of easy implementation. One of those techniques is known as Artificial Neural Networks (ANN).
Inspired, in their origin, in the functioning of the human brain, and entitled with some intelligence. These are the combination of a great amount of elements of process - artificial neurons – interconnected that operating in a parallel way get to solve problems related to aspects of classification. Before going forward with the description of the artificial neural networks, for its relation with the human brain, some of the characteristics of the brain will described.

2. General characteristics of the human brain.

The human brain is formed by a great amount of units of process denominated neurons, which differing from other cells which have a great capability to communicate. This is fundamental, because such elements group among them to save or process information. Figure 1 shows the three main components of a typical nervous cell, which are:

- Dendrites, branches, through which neuron gets information.
- Body of the cell, processes the gotten signals and with base in them emits a signal.
- Axon, The only structure with ramifications, through which the information is transmitted from the cell body to the dendrite of another neuron.

It is important to mention that between the terminal from the axon and the dendrite there is no physical contact, the transference of information is produced using a union known as synapse.

The communication between the neurons is made through impulses, which nature is of two kinds: electrical and chemical. The signal that is generated by the neuron and carried through the axon corresponds to an electrical impulse; on the other hand the signal between the terminals of the axon and the dendrites has a chemical origin. This is carried by molecules called neurotransmitters which flow through the cell membrane in the synapse region. The cell membrane is permeable for some ionic species (chlorine -, sodium+, potassium +), and acts in such a way to keep a difference of potential between the intracellular fluid and the extra-cellular fluid (rest potential). This effect is gotten firstly by the variation of the sodium and potassium concentration in the opposite sides of the cell membrane, see fig. 2. When there is an exchange of information from one cell to another (synapse) there is a variation of the potential value (potential of action). This potential of action produces a variation in the permeability of the membrane, which allows the exchange of neurotransmitters substances. There are two kinds of synapse:

- Exciting synapse, which neurotransmitters provoke diminishing in the potential, facilitating the generation of impulses of greater speed.
- Inhibiting synapse, then neurotransmitters tend to stabilize the potential, dificulting the emission of impulses.

This is shown in every of the synapse of the neuron, so the total entrance is equal to the pondered addition of each one of the coming signals from the other neurons. Depending
on the reached amount, if it overcomes the one of the limit the neuron is activated (generating an exit), in the opposite case it is not activated.

3. What is an artificial neural network?

Before giving an answer to this question, it is important to mention that the objective of the Artificial Neural Networks is not building systems to compete against human beings, but to do some tasks of some intellectual rank, to help them. From this point of view its antecedents are:

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>427 - 322 B. C.</td>
<td>Plato and Aristotle: Who conceive theories about the brain and the thinking. Aristotle gives reality to the ideas understanding them as the essence of real things.</td>
</tr>
<tr>
<td>1596-1650</td>
<td>Descartes: Influenced by the platonic ideas proposes that mind and matter are two different kinds of substance, of opposite nature, able to exist in an independent way, which have the possibility to interact.</td>
</tr>
<tr>
<td>1936</td>
<td>Alan Turing: studied the brain as a form to see the world of computing.</td>
</tr>
<tr>
<td>1943</td>
<td>Warren McCulloch and Walter Pitts: Create a theory about the functioning of neurons.</td>
</tr>
<tr>
<td>1949</td>
<td>Donald Hebb: Established a connection between psychology and physiology.</td>
</tr>
<tr>
<td>1957</td>
<td>Frank Rosenblatt: Developed the Perceptron, the first Artificial Neural Network.</td>
</tr>
<tr>
<td>1959</td>
<td>Bernard Widrow and Marcial Hoff: Created the Adaline Model.</td>
</tr>
<tr>
<td>1982</td>
<td>James Anderson, Kunihiro Fukushima, Teuvo Kohonen, and John Hopfield: Made important works that allowed the rebirth of the interest for the artificial neural networks. Reunion U.S. – JAPAN: Begin the development of what was known as thinking computers for their application in robotics.</td>
</tr>
</tbody>
</table>
There are different ways of defining what the ANN are, from short and generic definitions to the ones that try to explain in a detailed way what means a neural network or neural computing. For this situation, the definition that was proposed by Teuvo Kohonen, appears below:

*Artificial Neural Networks are massively interconnected networks in parallel of simple elements (usually adaptable), with hierarchic organization, which try to interact with the objects of the real world in the same way that the biological nervous system does.*

As a simple element we understand the artificial equivalent of a neuron that is known as computational neuron or node. These are organized hierarchically by layers and are interconnected between them just as in the biological nervous systems. Upon the presence of an external stimulus the artificial neural network generates an answer, which is confronted with the reality to determine the degree of adjustment that is required in the internal network parameters. This adjustment is known as learning network or training, after which the network is ready to answer to the external stimulus in an optimum way.

### 4. Benefits of artificial neural networks

It is evident that the ANN obtain their efficacy from:

1. Its structure massively distributed in parallel. The information processing takes place through the iteration of a great amount of computational neurons, each one of them send exciting or inhibiting signals to other nodes in the network. Differing from other classic Artificial Intelligence methods where the information processing can be considered sequential – this is step by step even when there is not a predetermined order - , in the Artificial Neural Networks this process is essentially in parallel, which is the origin of its flexibility. Because the calculations are divided in many nodes, if any of them gets astray from the expected behaviour it does not affect the behaviour of the network.

2. Its ability to learn and generalize. The ANN have the capability to acquire knowledge from its surroundings by the adaptation of its internal parameters, which is produced as a response to the presence of an external stimulus. The network learns from the examples which are presented to it, and generalizes knowledge from them. The generalization can be interpreted as the property of artificial neural networks to produce an adequate response to unknown stimulus which are related to the acquired knowledge.

These two characteristics for information processing make an ANN able to give solution to complex problems normally difficult to manage by the traditional ways of approximation. Additionally, using them gives the following benefits:

**No linearity**, the answer from the computational neuron can be linear or not. A neural network formed by the interconnection of non-linear neurons, is in itself non-linear, a trait which is distributed to the entire network. No linearity is important over all in the cases
where the task to develop presents a behaviour removed from linearity, which is presented in most of real situations.

**Adaptive learning**, the ANN is capable of determine the relationship between the different examples which are presented to it, or to identify the kind to which belong, without requiring a previous model.

**Self – organization**, this property allows the ANN to distribute the knowledge in the entire network structure, there is no element with specific stored information.

**Fault tolerance**, This characteristics is shown in two senses: The first is related to the samples shown to the network, in which case it answers correctly even when the examples exhibit variability or noise; the second, appears when in any of the elements of the network occurs a failure, which does not impossibilitate its functioning due to the way in which it stores information.

5. Applications of the Artificial Neural Networks

From the applications perspective we can comment that the strength of the artificial neural networks is in the management of the no lineal, adaptive and parallel processes. The ANN have found diverse successful applications in computers vision, images/signal processing, speech/characters recognition, expert systems, medical images analysis, remote sensing, industrial inspection and scientific exploration. In a superficial way, the domain of the applications of the artificial neural networks can be divided into the following categories:

- Pattern recognition: Or supervised classification, an entry given represented by a vector, it assigns a class label existing in the predefined structure of classes.
- Grouping: Also denominated non supervised classification, because there is no a predefined structure of classes. The networks explore the objects presented and generate groups of elements that follow certain similarity criteria.
- Approximation of functions: with base in a group of pairs (ordered pairs of entry/exit) generated by an unknown function of the network adjusts its internal parameters to produce exits that implicitly correspond to the approximation of the function.
- Prediction: Predicting the behaviour of an event which depends from the time, with base in a group of values that are obtained from different moments.
- Optimization: A great variety of problems in Math, Science, Medicine and Engineering can be focused as problems where is required to determine a solution that accomplishes with a group of restrictions and diminishes or maximizes an objective function.
- Association: There are made two kinds of associative formulations: the self-association and the hetero-association. In the problems of self-association problems from partial information complete information is recovered. The
hetero-association consists in recovering an element from a group $B$, given an
element from a group $A$.

- Control: From a defined system by the pair $(u(t), y(t))$, where $u(t)$ is the control
entrance and $y(t)$ is the exit of the system to time $t$, in a model of adaptive
control. The objective is to generate a control entrance $u(t)$ so that the system
keeps the expected behaviour, which is determined by the reference model.

For such applications are successful solution form a classic perspective, however in most
of the cases they are only valid in restricted environments and present little flexibility out
of its domain. The ANN give alternatives which give flexible solutions in a great domain.

6. Computing model of a neuron

Just as the neurons from biological nervous systems the task of the artificial neuron – also
known as computational neuron, unity or element of process, or node – is simple and
unique, this consists in receiving the entrance of the neighbour nodes and calculate an
output value which destination correspond to, in most of the cases, a great number of
computational neurons.

In the construction of any given ANN we can identify, depending on the location
in the network, three kinds of computational neurons: input, output and hidden.
The input nodes as their name indicates are the entrance door to the network and
get information from its surroundings; these can have as origin any sensor or come
from other system sectors. The units of output have the function of transmitting
the answer of the artificial neural network to the exterior (output network), and
can be used to directly control a system. Finally, the hidden units are those which
entrances and exits are inside the net, they do not have any contact with the
exterior.

One of the most used models to represent the computational neuron is presented in Figure
3, mathematically is modeled the body of the neuron and the axon by a net function (or
basic function) and an activation function. The selection of these functions frequently
depends on the kind of application for which the computational neuron will be employed.
Most of the times the net function is represented by the lineal combination of the outputs
of the anteceding neurons

$$net_j = \sum_i w_{ji}O_i$$

(1.1)

where $w_{ji}$ is an amount denominated weight assigned to the communication link between
the neuron $j$ and the node $i$, and $O_i$ is the exit of the computational neuron $i$. Each
connection between nodes has a weight that represents the strength of the influence
between them, this is modified during the training (learning) of the network, due to which
we can consider that the knowledge resides in the weight of the communication bonds.
The activation function is applied on the $net_j$, and represents the output of the computational neuron. Being the main objective to emulate the possible states -activated or non activated- of the neurons, normally are employed expressions of the kind of threshold, hyperbolic tangent, sigmoidal, in Figure 4 some examples of activation functions are presented.

It is important to mention that the input nodes are the only ones that are removed from the mentioned behaviour because these only transmit information towards the superior layer of the network. They do not execute the calculation of the net because there are not anteceding nodes, or employ the activation function.

The computational neurons are organized in layers, understanding as a layer or level to a group of neurons which entrances come from the same source, this can be another layer of neurons, and their exits are directed to the same destination, which can be located inside or outside the network.

7. Structure of a neural network. (Topology)

With the expressions structure, architecture or topology of an artificial neural network we talk about the way in which computational neurons are organized in the network. Particularly, these terms are focused in the description of how the nodes are connected and in how the information is transmitted through the network.

Number of levels or layers: As it has been mentioned, the distribution of computational neurons in the neural network is done forming levels or layers of a determined number of nodes each one. As there are input, output and hidden neurons, we can talk about an input layer, an output layer and one or several hidden layers. By the peculiarity of the behaviour of the input nodes some authors consider just two kinds of layers in the ANN, the hidden and the output, in this course we will consider the input layer as part of the neural network, Figure 5 shows an artificial neural network formed by three layers, this can be briefly described as a 4-4-3- network, because it contains 4 nodes in the input layer, 4 nodes in a hidden layer and 3 computational neurons in an output layer.

Connection patterns: Depending on the links between the elements of the different layers the ANN can be classified as: totally connected, when all the outputs from a level get to all and each one of the nodes in the following level, in this case, there will be more connections than nodes (Figure 5); if some of the links in the network are lost, then we say that the network is partially connected.

Information flow: Another classification of the ANN is obtained considering the direction of the flow of the information through the layers, the connectivity among the nodes of a neural network is related with the way in which the exit of neurons are directed to become into entrances for other neurons. The output signal of a node can be one of the entrances to another element of process, or even be an entrance to it (auto-recurring connection).
When any output of the neurons is input of neurons of the same level or preceding levels, the network is described as feedforward. In counter position if there is at least one connected exit as entrance of neurons of previous levels or of the same level, including themselves, the network is denominated of feedback. The feedback networks that have at least a closed loop of back propagation are called *recurrent*, see Figure 6.

![Figure 1](image1.png)

![Figure 2](image2.png)
Introduction to Artificial Neural Networks

\[ S_j = f(\text{net}_j) \]

\[ \text{net}_j = \sum w_{ji} s_i \]

Figure 3

exit layer

Hidden layer

Entrance layer

Figure 4

Figure 5

Eduardo Gasca Alvarez
nodo con propagación hacia atrás sobre sí misma

Red con propagación hacia atrás o nodos de niveles anteriores

Salida

Figure 6
Learning Processes

Table of contents:

1 Introduction
2 Supervised learning
   2.1 Error Correction learning
   2.2 Reinforcement learning
   2.3 Stochastic learning
3 Unsupervised learning
   3.1 Hebbian learning
   3.2 Competitive learning

Objectives:

At the end of this unit the student:

➢ Will describe the difference between supervised and unsupervised learning.
➢ Will explain Error Correction learning.
➢ Will understand Reinforcement learning.
➢ Will describe Stochastic learning.
➢ Will explain Hebbian learning.
➢ Will understand Competitive learning.

Readings:

✓ Kulkarni, D. Arun, *Artificial Neural Networks for Image Understanding*, pages 109 - 113

1. Introduction

The main property in all the Artificial Neural Networks models (ANN) is the ability to learn from its surroundings, which is shown with the improvement of their performance through learning. This improvement takes place in the time according with some prewritten rules, which through an interactive process modify the networks upon the presence of external stimulus. Ideally, the ANN acquires more knowledge, about their environment, during the iteration of the learning process.
Even when giving a definition for the term *learning* could be adventurous, given the amount of areas of human knowledge, which participate in its study and to which this document does not pretend to deepen, in general we can understand *learning* as a modification of the behaviour as a result of experience. From this point of view Mendel and McClaren (Haykin) defined the utterance *learning* in the context of the Artificial Neural Networks in this way:

*Learning is a process by which, the free parameters from a neural network are adapted, through a stimulation process, by the environment in which a network is contained. The kind of learning is determined by the way in which the change of the parameter has place.*

This definition implies the following sequence of events:

1. The network is stimulated by its environment.
2. As a result of the stimulation the neural network suffers some changes in its internal parameters. These changes are regulated by a group of pre–written rules.
3. The neural network responds, to its surroundings, in a different way by the changes that happen in its internal structure.

It is a good moment to mention that to the prewritten group of rules, defined to solve a learning process, is denominated as algorithm or learning rule. Besides, the expression internal parameter is used to design the group of communication links between the computational neurons, this is the weight. In other words, learning or the training of an Artificial Neural Network basically consists in the modification of its weight through the application of a learning algorithm when a group of patterns is presented.

As it can be anticipated, there is not a unique learning algorithm for the design of the Neural Networks. In a general way, they are usually considered of two kinds: supervised learning and unsupervised learning. Depending on which paradigm is used we can classify the networks in:

- Artificial Neural Networks with supervised learning.
- Artificial Neural Networks with unsupervised learning.

The fundamental difference between both kinds is in the existence of an external agent (supervisor) that controls the learning process in the net. Before making a more detailed description of the supervised learning and the unsupervised learning, we will comment that other classification criteria of the ANN consists in determining if the network learns during its normal functioning (on line) or if the learning assumes the unplugging of the network (off line).

When the learning is *off line*, it is distinguished between learning or training phase and an operation or functioning phase. In the first one a group of data is used –training sample– which main objective is to contain the example from where the network will take the
knowledge. Thus, a group of proof data should be used –control sample– which will be used to determine the efficiency of the training. In the networks using off line learning, the weight of the connections remain still after the training stage finishes in the network. In the networks that have on line learning there is not distinction between training phase and operation phase in this way the weight vary dynamically always that there is shown new information to the system.

2. Supervised Learning

In the supervised knowledge the training is controlled by an external agent (supervisor, teacher), which watches the answer that the network is supposed to generate from a determined entrance. The supervisor compares the output of the network with the expected one and determines the amount of the modification to be made in the weight. The objective is to decrease the difference between the answer of the network and the desired value. This is included in the training sample and is part of each one of the examples.

The supervised learning can be done through three paradigms, which are denominated:

- Error Correction learning.
- Reinforcement learning.
- Stochastic learning.

Through them the supervised training takes place and requires some correctly identified examples.

**Error correction learning**

During the artificial neural network training by Error Correction, the weight of the communication links are adjusted trying to minimize a function of cost depending on the difference between the desired values and the obtained from the output network. Here is to say, it is required to determine the modification of the weight with a base in the committed error in the output, which can be done through different ways. For instance, a rule in the error correction learning calculates the variation of the weight exploring the expression.

\[ \Delta w_{kj} = \eta_j (d_k - y_k) \]  \hspace{1cm} (2.1)

Where: \( \Delta w_{kj} \) is the weight variation of the connection between the neuron \( j \) from the anterior layer and the node \( k \) of the output layer, \( d_k \) value of desired output for the neuron \( k \), \( y_j \) and \( y_k \) are the output values produced in the neuron \( j \) and \( k \) respectively, and \( \eta \) a positive factor denominated learning rate (0 < \( \eta \) < 1) that regulates the learning speed. In other words the new weight is proportional to the committed error produced by the network and the answer of the node \( j \) in the previous layer and is deducted by:

\[ w_{kj}^{(\text{actual})} = \Delta w_{kj} + w_{kj}^{(\text{ant})} \]  \hspace{1cm} (2.2)
An example of criterion function is found in the known as mean square error or Widrow – Hoff rule, which is expressed by:

\[
Error_{\text{global}} = \frac{1}{2P} \sum_{p=1}^{P} \sum_{k=1}^{N} (y_k^{(p)} - d_k^{(p)})^2
\]  \hspace{1cm} (2.3)

Being \( N \) the amount of neurons in the output layer, and \( P \) the number of examples in the training sample. Normally to minimize this criterion function is employed a learning rule given by the descending gradient.

\[
\Delta w_{kj} = -\eta \nabla_{w_{kj}} (Error_{\text{global}}) \hspace{1cm} (2.4)
\]

The difficulty to realize the learning evaluation rule, equation (4), is in the lack of knowledge of the explicit expression of error, equation (3), in function of the weight, which prevents calculation. We should remember that the exit of a node is determined applying the function of activation of the net, defined in unit 1, and this depends on the weight. However, even this allows us to find the dependence of the error in function of the weight; the result is an approach to reality because the obtained expression is valid for the example given in that moment. The last mentioned plus the non linear behavior of the output nodes brings as a consequence, that the graphic of error in function of the weight is not a monotonous decreasing with one minimum only, but can exist one or many local minimums added to the global, see figure 2.1.

The presence of local minimums generates an increasing of the probability that in the network does not follow to the minimum global and can be trapped in one of them. Under this situation the learning rate acquires importance because this controls not only the speed of convergence of the learning but also the network does it. If \( \eta \) takes a little value the learning process is made smoothly, which gives as a result an increment in the convergence time to a stable solution; on the other hand if \( \eta \) is big the speed of learning gets increased, but there is the risk that the process has divergence and for this it could be unstable. Later this topic will be enhanced for some ANN models.

**Reinforcement learning**

Reinforcement learning is a supervised learning, slower than the one mentioned before, which basic idea is not to exhibit to the network a complete example of the desired behaviour; we mean, during the training it is not exactly indicated the output that we want the network to give in a determined input.

In the reinforcement learning the function of the supervisor is reduced to indicate through a signal if the obtained output by the network is what we expect (success = +1 or failure = -1), and in function of this weight are adjusted through a mechanism of probabilities. It can be said that in this type of learning the function of the supervisor is similar to a critic (that has an opinion about the answer of the network) more than a teacher function (who indicates to the network the concrete answer to generate).
Stochastic learning

Basically this kind of learning consists in selecting the weight randomly, making changes in their value and following a distribution of pre-established probability, and evaluate its effect from the expected objective.

In the Stochastic learning there are usually analogies in thermodynamic terms, associating the artificial neural network with a physical system that has certain energetic state.

The energy from the network would symbolize the stability degree, in such a way that the state of minimum energy would correspond to a situation in which the values from weight produce the functioning that is perfectly adjusted to the expected objective.

According, to what was just mentioned, learning has to do with choosing randomly a weight, make the change of its value and determine the energy of the network (generally the energy function is a function denominated of Lyapunov). If the energy is lower after the change; it means if the behaviour of the network is close to the expected one, the change is accepted. If, on the other hand, the energy is not lower, the change would be accepted in function of a determined and pre – established probability distribution.

3. Unsupervised learning

The Artificial Neural Networks with unsupervised learning (also known as self-supervised) do not require any external element to adjust the weight of the communication links to their neurons. They do not receive the information from the environment that indicates if the generated output, in response to a determined input is or not correct: that is why, it is said that artificial neural networks that are not supervised are capable of self-organization.

The main problem in the unsupervised classification is to divide the space where the objects are (space of characteristics) in groups or categories. For which intuitively closeness criteria is used, an object belongs to a group if it is similar to the elements that integrate that group.

Since, there is no supervisor to indicate to the network the answer that should be generated in a concrete input. This kind of networks should find by themselves the characteristics, regularities, correlations or categories in which can establish between the data that was presented in their input.

There are many different interpretations of the output of the unsupervised networks, which depend on their structure and the learning algorithm used. In some cases, the output represents a degree of familiarity or similarity between the signal that is being introduced in the network and the displayed information until then. Under other circumstances can make the grouping of the information (clustering), generating a category structure, the network detects the categories from the correlations between the presented information. In such situation, the output of the network permits realizing a
codification of the input data, keeping the relevant information. Finally, some networks with unsupervised learning, make a mapping of characteristics which is called feature mapping, which generates in the output neurons a geometric disposition that represents a topographic map of the characteristics of the input data, presenting to the network similar information that will always affect output neurons close to each other, in the same mapping zone.

In general there only two kinds of unsupervised learning:

- Hebbian Learning.
- Competitive learning.

**Hebbian learning**

This kind of learning is based in the following statement made by Donal O. Hebb in 1949:

> “When the axon of an A cell is close enough to get a B cell to get excited, and persistently takes part in its activation, a process of growing or metabolic change has place in one or both cells, in such a way that the efficiency of A, when the cell to activate is B increases.”

As a cell Hebb understands a group of neurons strongly attached by a complex structure. The efficiency could be identified with the intensity or magnitude of a connection, the weight. Thus, Hebbian learning basically consists of the adjustment of the weights of the links of communication according to the correlation of the values of activation (outputs) of two neurons connected, when the two neurons are activated at the same time the weight is reinforced through this expression.

$$\Delta w_{ik}(t) = \eta \cdot y_k(t) \cdot x_i(t)$$  \hspace{1cm} (2.5)

Where: $\eta$ is a positive constant denominated learning rate, $y_k$ is the state of activation of the $k$th computational neuron in observation, $x_i$ corresponds to the state of activation of the $i$th node of the preceding layer. This expression answers to the idea of Hebb, because if the two units are activated (positive), a reinforcement of the connection is produced. On the other hand, if one is activated and the other is not (negative), a weakening of the value of the connection occurs. It is a rule of unsupervised learning, because the modification of the weight is done in function to the states (outputs) of the obtained nodes without asking if is expected to obtain or not this states of activation.

A Hebbian process is characterized by these four key properties:

1. Depending on the time. This refers that the link reinforce of communication is made when it is presented the activation of the two computational neurons.
2. Local. For the effect described by Hebb to take place, the nodes have to be continuous to the space. The modification that is produced only affect locally.
3. Interactive. The modification is done when is proved that both units are activated, because it is not possible, to predict the activation.

4. Correlated. Given the co-occurrence of the activations that have to be produced in very short periods of time, the effect described by Hebb is also known as compound synapse. On the other hand, for the activation of one of the computational neurons to take place, it has to be related to the activation of one or many previous nodes for which is called correlated synapse.

A disadvantage in the expression (2.5) is related to the exponential growing of which can be shown, which provokes a saturation of weight. To avoid that situation Kohonen proposed a modification that includes the incorporation in a term that modulates growing,

$$\Delta w_k(t) = \eta \cdot y_k(t) \cdot x(t) - \alpha \cdot y_k(t) \cdot w_k(t)$$  \hspace{1cm} (2.6)

obtaining the following expression.

Another version of the learning rule is the denominated Hebbian differential, it uses the correlation of the derivative, in the time, of the functions in the neurons of the computational neurons.

**Competitive learning**

In the networks with competitive and cooperative learning, it can be said that the neurons compete and cooperate ones with the others to give a specific task. Differing from the net of Hebbian learning basis, where many output nodes can be activated simultaneously, in the case of competitive learning only one can be activated at a time. In other words, competitive learning pretends to obtain a winner take all unit, or a by the group of nodes, that gets their value of maximum response introducing some information. The rest of the nodes are forced to take minimum value answer keys. The competition between units is made in all the layers of the network, having been able to appear in neighboring nodes (of a same layer) recurrent connections of excitation or inhibition. If the learning is cooperative, these connections with the neighbors they will be of excitation. Since one is no supervised learning, the objective of the competitive learning is to group the data that are introduced in the network. In this way, the similar patrons are organized constituting a same category, and therefore they must activate the same exit neuron. The network creates the groups by the detection of correlations between the input data. Consequently, the individual units of the network learn to specialize in a set of similar elements, and for that reason, they become detectors of characteristics. The simplest artificial neural network that uses a competitive learning rule is formed by an input layer and an exit layer totally connected, besides the exit nodes include lateral connections (figure 2.2). For example, the connections between layers can be exciting, and the lateral ones between nodes of the inhibiting exit layer. In this type of networks, each neuron in the exit layer has assigned a gross weight, sum of all the weights of the connections that it has to his entrance. The learning affects only to the connections with the
Learning Processes

winning neuron. For this, it redistributes its gross weight between its connections, removing a portion the weights of all the connections that get at the winning neuron and distributing this amount between all the connections coming from active units. Therefore, if \( j \) is the winning node the variation of the weight between unit \( i \) and \( j \) is null if neuron \( j \) does not receive excitation on the part of neuron \( i \) (it does not win in the presence of a stimulus on the part of \( i \)), and it will be modified (it will be reinforced) if it is excited by this neuron \( i \). In the end, each one of the units of winning exit has discovered a group.
Table of contents:

1. Introduction
2. Convergence Theorem of the Perceptron
3. Virtues and limitations
4. Adaline and Madaline

Objectives:

At the end of this unit students will be able to:

- Describe the Perceptron functioning
- Make the Perceptron training
- Use the Perceptron to do classification tasks

Readings:


1. INTRODUCTION

The Perceptron was the first supervised model of an artificial neural network. Developed by Rosenblatt in 1958, it created a great interest during the 60’s due to its capability to learn and recognize simple patterns. It is formed by two layers, an input layer and an output layer. The first one gets information from the exterior and sends it to the output layer, where it is accumulated in the only computational neuron that constitutes this layer. (Figure 3.1)

As it has been said, one of the objectives of the training phase, in the supervised networks, is to learn to discriminate the class of the patterns which are contained in the training sample and lately generalize this knowledge to objects which were not seen during the learning. On the other hand, the training can also be interpreted as the process through which the classifier rehearse to divide the space of observation in regions where there are only one class elements. In other words, it determines the discriminating functions that approximate the border between classes. The figure 3.2 shows examples of decision frontiers or borders adjusted in lineal and not lineal ways, those classes that can be separated with a line or plan and are denominated lineally separable. Under this point of view and with the topology given we can say that the Perceptron was designed to deal
with tasks of two kinds using a discriminating lineal function to simulate the decision frontier.

We should remember that the nodes in the input layer are the only ones in which the input is equal to their output. The input to any other node is determined calculating the net function equation 1.1; \( net_j = \sum_i w_{ji} O_i \) and the corresponding output is obtained applying the activation function to the net. In the particular case of the Perceptron the activation function is a discriminating linear function, for which the answer of the computational neuron in the output layer is:

\[
y = \sum_{j=1}^{N} w_j x_j + \theta
\]  

(3.1)

The term \( \theta \) it is known as threshold, if this reduces the entrance to the function (negative), or bias when \( \theta \) increases the entrance (positive). From the geometric point of view, the member on the right of the equation (3.1.) represents a hyperplane that if not being because of the threshold (bias) would always pass through the origin, which would be unpractical in a great amount of tasks. For convenience the threshold value \( \theta \) is introduced in the vector of weight, obtaining:

\[
w = [w_1, w_2, ..., w_N, \theta]^T
\]  

(3.2)

If we define the pattern increasing \( X \) as:

\[
z = [x_1, x_2, ..., x_N, 1]^T
\]  

(3.3)

The threshold will be represented though a virtual node which output is always one. The linear discriminating function can be written as:

\[
y = w^T z
\]  

(3.4)

Because the Perceptron deals with two kinds of problems, so the decision rule should be expressed in a binary way, this is:

\[
d = \begin{cases} 
1 & y > 0 \\
0 & y \leq 0 
\end{cases}
\]  

(3.5)

Using this relation an \( X \) pattern is classified as belonging to \( A \) class when \( d=1 \); otherwise it belongs to class \( B \). Recurring the supervisor criterion it is determined if the pattern is correctly classified. When and only when a bad classification occurs the weight of the net are adjusted supporting in the learning rule. The learning rule of the Perceptron is the following:

Algorithm: If when presenting the \( n-th \) training pattern \( z^{(n)} \) it is erroneously classified, the weight vector will be actualized applying the expression:

\[
w^{(n-1)} = w^{(n)} + \eta (t^{(n)} - d^{(n)}) z^{(n)}
\]  

(3.6)
Perceptron

Where $\eta$ is the learning rate, positive quantity less than 1.
In a more precise way, the learning rule of Perceptron can be seen from two perspectives.

- Reinforcement: If a pattern belongs to the $A$ class but is not selected as such, then
  the vector of weights has to be reinforced adding to this a proportion of the
  training pattern $z^n$. As $t^{(n)} - d^{(n)} = 1$, the actualization takes the value $\eta z^{(n)}$.

- Anti – reinforcement: If a pattern does not belong to the $A$ class but was selected as
  an element from $A$, the vector of weight should be modified subtracting from it a
  proportion from the training pattern $z^n$. Because $t^{(n)} - d^{(n)} = -1$, then the
  actualization is $-\eta z^{(n)}$.

Being the training an iterative process, during which all the elements of the training
sample are presented to the network in each iteration, it should be counted with one or
various criteria that allow to stop the learning. Among the most used ones we find:
marking the maximum number of iterations, or establishing the percentage of maximum
and minimum error. Normally, in the Perceptron are combined a maximum number of
iterations, and not committing erroneous classifications during the lapse of a complete
iteration what happens firstly interrupts the learning.

2. Convergence Theorem of the Perceptron

Differing from the most of the models of artificial neural networks, the Perceptron has the
advantage of counting with a theorem of convergence which ensures to discover the
optimum answer when the task includes lineally separable classes. In a succinct way the
convergence theorem of the Perceptron states that:

**Theorem:** being $C1$ and $C2$ two lineally separable classes and $TS$ a
training group with elements of these two classes. If the Perceptron
is trained with $TS$ then the algorithm of learning has convergence
to a solution in a finite number of iterations.

It is important to mention that the expression 3.4 does not reveal a unique solution;
therefore the convergence theorem only ensures to reach one of the solutions. Which is
strongly influenced by the learning reason and the initial values of the weight $w(0)$, this is
reflected by the increasing or reduction of the amount of iterations to reach the solution.

**Vector of Initial Weight:** There is not a uniform criterion to select the group of initial
values for the communication links between the nodes, normally they are chosen
randomly with values in the interval. $(-1, 1)$. This can be justified if we interpret the weight
as the coefficient of the hyperplane used to divide the classes, because huge values in the
coefficients provoke saturation and reduce the mobility of the hyperplane.

**Constant learning rate:** the speed of convergence in the Perceptron depends strongly on
the election of the learning rate. If its value is small it converges slowly. On the other hand,
if \( \eta \) is big can cause oscillations around the solution then what is the ideal value for the learning rate? To answer this we will analyze the following situation:

**Ideal learning rate:** Assuming that one of the solutions is known to the vector of weight, represented by \( w \), then the value of the optimal learning rate can be obtained minimizing the square error, from where:

\[
\eta_{opt} = \frac{|w^T z^{(n)}| + |w^{(n+1)T} z^{(n)}|}{\|z^{(n)}\|^2}
\]  

(3.7)

The convergence is guaranteed in a finite number of iterations, because the total square error always decreases with each one of the actualizations. When an erroneous classification takes place, the rule of learning from Perceptron becomes:

\[
w^{(n+1)} = w^{(n)} + \frac{|w^T z^{(n)}| + |w^{(n+1)T} z^{(n)}|}{\|z^{(n)}\|^2}(d^{(n)} - d^{(n)}) z^{(n)}
\]  

(3.8)

From where it is deduced that the right size is given by:

\[
\eta_{opt} = \frac{(w^* - w^{(n)})^T z^{(n)}}{\|z^{(n)}\|^2}
\]  

(3.9)

By simple geometry, it is possible to realize that the expression (3.9) corresponds to the projection of \( (w^* - w^{(n)})^T \) on \( z^{(n)} \). Therefore, the optimal learning rate approaches the vector of weight to the ideal value \( w^* \).

The expression (3.9) is not practical, because \( w^* \) is unknown. To solve this it is proposed to work with the normalized Perceptron. Under this schema the task is to preserve the normalization of \( w^{(n)} \) in each of the training iterations. Therefore, the term \( |w^T z^{(n)}| \) in the equation (3.9) is substituted by \( |w^{(n)T} z^{(n)}| \), so:

\[
\eta_{norm} = \frac{2|w^{(n)T} z^{(n)}|}{\|z^{(n)}\|^2}
\]  

(3.10)

To select the learning rate through the expression (3.10), guarantees that the total square error decreases strictly. Therefore the convergence is guaranteed in a finite number of iterations for the case for the normalized Perceptron.
3. Virtues and Limitations

It is clear that the simplicity of the Perceptron is one of the advantages that it has. Besides, its performance is good when the problem is lineally separable and there are two classes.

However, the disadvantages can be more significant, because most of the problems are not lineal, and have more than two classes.

4. Adaline and Madaline

The Adaline networks (ADAdaptive LINear Element) and Madaline (Multiple Adaline) were developed by Bernie Widrow just after Rosenblatt developed the Perceptron. The architectures for Adaline and Madaline are essentially the same that in Perceptron. Both structures use neurons with step function of transference. The Adaline net is limited to an only output neuron, while Madaline can have many. The fundamental difference with the Perceptron is related to the learning mechanism, Adaline and Madaline use the denominated delta rule of Widrow–Hoff or least-mean-square-error rule (LMS, equation (2.3)). This makes the searching for the minimum of the error function, among the desired output and the obtained lineal output, before applying an activation function. Due to this new form of evaluating errors, these networks can process analogical information, input as well as output, using a lineal activation function or sigmoidal.

Adaline

The structure of the Adaline network includes an element which is denominated Adaptive Lineal Combiner (ALC) that obtains a linear response which can be applied to other element of bipolar commutation. So if the output of the ALC is positive, the response of the Adaline network is +1, if ALC is negative, then the result of the Adaline Network is -1 (figure 3.3). The linear output that the ALC generates is given by:

\[
y(t + 1) = \begin{cases} +1 & s > 0 \\ y(t) & s = 0 \\ -1 & s < 0 \end{cases}
\]

The binary answer corresponding to the Adaline network is:

\[
s = \sum_{j=0}^{N} w_j x_j = w^T X
\]
The Adaline and Madaline Networks use a supervised learning rule, off–line, denominated Least Mean Squared (LMS), through which the vector of weight, \( \mathbf{w} \), should associate with success each input pattern with its corresponding value of desired output, \( d_k \). Particularly, the network training consists in modifying the weights when the training patterns and desired outputs are presented. With each combination input–output an automatic process is made of little adjustments in the weights values until the correct outputs are obtained.

Concretely, the learning rule LMS minimizes the mean squared error, defined as:

\[
E = \frac{1}{2P} \sum_{k=1}^{P} \varepsilon_k^2
\]  

(3.13)

Where \( P \) is the quantity of input vectors (patterns) that form the training group, and \( \varepsilon_k \) is the difference between the desired output and the obtained when is introduced the \( k \)-th pattern, in the case of the Adaline network, it expressed as \( \varepsilon_k = (d_k - s_k) \), where \( s_k \) corresponds to the exit of the ALC (expression 3.11), that is to say:

\[
s_k = \sum_{j=0}^{N} w_{jk} x_j = \mathbf{w}^T \mathbf{x}_k
\]  

(3.14)

Even when the error surface is unknown, equation (3.13), the gradient descent method gets to obtain the local information of it. With this information it is decided what direction to take to arrive to the global minimum. Therefore, based in the gradient descent method, the rule known as delta rule or LMS rule is obtained. With this the weight are modified so that the new point, in the space of weights, is closer to the minimum point. In other words, the modification of the weights is proportional to the gradient descent of the error function

\[
\Delta w_j = -\eta (\partial E_k / \partial w_j).
\]

Using the chain rule for the calculus of the derivative of expression (3.13), we obtain:

\[
w_j(t + 1) = w_j(t) + \eta \varepsilon_k x_{kj}
\]  

(3.15)

Based in this, the learning algorithm of an Adaline network contains the following steps:

1. A pattern is introduced \( \mathbf{X}_k = (x_{k1}, x_{k2}, \ldots, x_{kn}) \) in the entrances of Adaline

2. The linear output is obtained \( s_k = \sum_{j=0}^{N} w_{jk} x_j = \mathbf{w}^T \mathbf{x}_k \) and it is calculated the difference with respect from what is expected \( \varepsilon_k = (d_k - s_k) \).

3. The weights are actualized.

\[
\mathbf{w}(t + 1) = \mathbf{w}(t) + \eta \varepsilon_k \mathbf{X}_k
\]

\[
w_j(t + 1) = w_j(t) + \eta \varepsilon_k x_{kj}
\]

4. Steps from 1 to 3 are repeated, with all the entrance vectors.

5. If the mean squared error

\[
E = \frac{1}{2P} \sum_{k=1}^{P} \varepsilon_k^2
\]

Eduardo Gasca Alvarez
Perceptron

Has a reduced value, the learning process ends; if not, the process is repeated from step one with all the patterns.

Madaline network

The Madaline network (Multiple Adaline) is formed as a combination of Adaline modules, which are structured in layers (figure 3.4). Madaline overcomes some of the limitations of Adaline networks.

Given the differences between Madaline and Adaline, the training of these networks is not the same. The LMS algorithm can be applied to the output layer because its exit is already known for each one of the entrance patterns. However, the expected exit is unknown for the nodes of the hidden layer. Besides, the algorithm LMS works for the linear outputs (analogical), of the adaptive combiner and not for the digital of the Adaline.

To employ the algorithm LMS in the Madaline training is required to modify the activation function for a derivable continuous function (the step function is discontinuous in zero and therefore not derivable in this point).

Because of its similarities with the Multilayer Perceptron, the description of the Madaline learning rule will not be described, this will be handled lately.

Fig. 3.3
Multilayer Perceptron

Table of Contents:

5. Introduction.
7. Learning Rate and momentum.
8. Second order algorithms.
9. Pruning

Objectives:

At the end of the study of this unit the student will be able to:

➢ Describe the functioning of Multilayer Perceptron
➢ Make the training of Multilayer Perceptron.
➢ Use Multilayer Perceptron in classification tasks.
➢ Identify different rules of learning for the Multilayer Perceptron.
➢ Choose the topology of Multilayer Perceptron.
➢ Find different ways to obtain the learning rate.

Readings:

✔ Kung, S. Y., *Digital Neural Networks*, páginas 152 - 168.

1. Introduction.

One of the most important models in the artificial neural networks is the called Multilayer Perceptron (or multilevel). This is of the supervised kind, feedforward. Probably it is one of the most studied artificial neural networks, with a wide rank of successful applications. It is constituted by one or several layers of hidden neurons, between the input and the output elements (fig. 4.1).

The Multilayer Perceptron (MP) allows establishing decision regions which are much more complex than the two semi-planes generated by the Perceptron. It has been said in the last unit that the two layers Perceptron (the input with lineal neurons and the output with activation function like step type) can only establish two regions which are separated by a linear frontier in the observation space, this in comparison with a Multilayer Perceptron with three layers of neurons, which can limit any convex region in the observation space. The convex regions are formed by the intersection between the zones generated by each neuron of the hidden layer. Each one of these elements behaves like a Simple Perceptron.
activating its output for the patterns of a side of the hyperplane. The decision region that results from the intersection will be the convex regions with a number of sides almost the same number of neurons of the second layer.

Because of the importance that the hidden layer nodes represent when forming the decision regions it should be asked How many nodes and hidden layers does the Multilayer Perceptron need to do a given task? This question does not have a widely accepted theoretical answer by the scientific community, heuristic processes have been made to generate some golden rules which success is relative and depends on the task to be done.

In general, the number of hidden neurons should be big enough to generate a region, complex enough to give solution to a task. On the other hand, it is not convenient to have a great amount of nodes that generates an estimation of the weights that is not trustworthy, if the group of patterns of input that is available is not enough. Kanellopoulus et al [Kanellopoulus, 1997], suggest that the number of nodes in the first hidden layer should be exactly the same as the maximum value that results when estimating between two and four times the amount of nodes in the input layer, or two or three times the number of nodes of the output layer.

About the number of hidden layers, it has been said that a Multilayer Perceptron (MP) with three levels can delimitate a convex region in the space of observation, which is improved by one with four layers that can form regions of decision arbitrarily complex. From the geometric point of view, the delimitation of the observation space in classes consists in dividing the desired region in little hyper-cubes (squares for two inputs in the network). Each hyper-cube requires $2N$ neurons in the second layer –one for each side, being $N$ the number of entrances to the network–, and other in the third that carries the unification of the outputs of nodes in the previous level. The response of the elements of the 3rd level will be activated just for the patterns of each hyper-cube.

From the last analysis we can infer that it is not required to have more than four levels in a network of the Multilayer Perceptron type, because as it has been seen, a network of four levels can generate decision regions arbitrarily complex. In contrast the criterion of Kanellopoulus et al [Kanellopoulus, 1997], is in increasing a second hidden layer when the amount of classes is bigger than 20.

2. Backpropagation Algorithm

Once defined the topology of the MP it is required to establish the learning algorithm that allow an adequation of the inner parameters of the network (weights). In 1986, Rumelhart, Hinton and Williams, formalized a method that allows the association among the patterns of input and its corresponding class, for this they used a neural network with more levels of neurons than the ones that Rosenblatt used to develop the Perceptron. This method, known as Backpropagation is based in the generalization of the delta rule and, even its
own limitations, has enhanced in a considerable way the rank of application of the Multilayer Perceptron.

During the learning of the MP a sample of the training is used and the algorithm Backpropagation (BP). Basically, consists in the modification of the values of the weights, to the presentation of the patterns contained in the training sample. That change is made considering the minimization of the mean squared error, which quantifies the difference between the correct and the assigned class though the network to the input pattern, equation (4.1) where: \( e_i^p = (d_i^p - s_i^p) \) is the error committed between the desired value \((d_i^p)\) and the output produced by the network \((s_i^p)\) of the \(p\)-th pattern in the \(i\)-th node of the output layer, and \(N_S\) is the amount of nodes of the output layer.

Calculating the mean squared error, the algorithm BP uses the gradient descent to optimize the values of the weights that minimize the error using the expression

\[
\Delta w(t) = -\eta \nabla_w E_p + \alpha \Delta w(t - 1)
\]

(4.2)

Where: \(\eta\) is the learning rate, \(\nabla_w E_p\) is the gradient of the error function with respect to the weights, \(\alpha\) the momentum, and \(t\) the number or iterations, the number of times that the training sample is presented to the network.

Identifying the output nodes with the sub index \(k\), the one(s) of the hidden layer(s) by \(j\), and the nodes of the input layer with \(i\), we have that \(w_{kj}\) identifies the link weight between the \(k\)-th node of the output layer and the \(j\)-th node of the last hidden layer and \(w_{ji}\) represents the weights between nodes of the input layer and the corresponding to the hidden layer. Substituting the equation (4.2) in (4.1) and applying the chain rule we obtain the explicit equations for the actualization of the weights between layers, equation 4.3

\[
\Delta w_{kj}(t) = \eta(d_k - s_k) f'_k (net_k) s_j + \alpha \Delta w_{kj}(t - 1) = \eta \delta_k s_j + \alpha \Delta w_{kj}(t - 1)
\]

\[
\Delta w_{ji}(t) = \eta f'_j (net_j) s_j \sum_k \delta_k w_{kj} + \alpha \Delta w_{ji}(t - 1)
\]

(4.3)

With: \(\delta_k = (d_k - s_k) f'_k (net_k)\) and \(f'_k (net_k)\) the derivative of the activation function. The expressions of the equation (4.3) differ because the error function is only evaluated in the exit layer. In the practice the modification of the weights is done until it reaches a certain number of iterations or a threshold value of the mean squared error, both established by the user.

Once that the MP training has been done it is in possibility to classify not–container patterns in the training simple, the success that is obtained when doing the classification is known as generalization, and it will be high if classifies correctly or low on the contrary. If it is wished to evaluate how efficient was the learning, training, is a requirement to estimate the generalization of the network.

In summary, the functioning of a Backpropagation network consists in the learning of a predefined group of inputs-outputs pairs given as examples, using a cycle propagation-
adaptation of two phases: first an input pattern is applied as stimulus to the neurons of the first layer of the network, and starts propagating through all the superior layers until it generates a response, the obtained result is compared in the output neurons with the desired value and the error value is calculated per each output neuron. Then, these errors are transmitted backwards, starting on the final layer, to all the neuron of the intermediate layer that directly contribute to the result, receiving the error percentage approximate to the participation of the intermediate neuron in the output value. This process is repeated, layer by layer until all the network neurons have received an error that describes its relative contribution to the total error. Based in the value of the received error, the connection weights of each neuron are adjusted, so that the next time that the same pattern is presented, the response is closer to the desired one, and it means that the error decreases.

The modification of the weights can be done to the presentation of each one of the patterns (adaptation on line or per event), or after the pass of a group of elements (adaptation in block or batch). Both procedures present differences in the yield of the numeric approximation. The actualization in block is considered wider because an average is made over all the training patterns. If it is not required a training in real time is preferable the actualization by block because it moderates some sensitivity problems. On the other hand, the modification per event even though quicker is more sensitive to the noise in the patterns.

For the adaptation in block we employ the error function

\[
E = \frac{1}{2P} \sum_{p=1}^{P} \sum_{i=1}^{N_i} (e_i^p)^2 \tag{4.4}
\]

where \( P \) is the amount of patterns in the training sample. Given the above-mentioned the learning rule takes the aspect

\[
\Delta w(t) = -\eta \nabla_w E_p + \alpha \Delta w(t-1) \tag{4.5}
\]

that is the version for the gradient descent of the actualization of the weights per block, in the iteration \( t \).

3. Learning rate and momentum.

Even though the Multilayer Perceptron method is one of most studied and used neural network models, there are some theoretical–practical aspects of its functioning that are not defined in a satisfactory way, which limits its application. Among these we find some parameters in the expressions used during the training or learning process, such as the learning rate and the momentum factor. There is a wide list of published works dedicated to study this question and to propose alternatives. However, there is not an adequate comparison to evaluate the best proposal, because each author is limited to expose the
good points of his technique and to demonstrate, if so, the results of a few practical applications. However there is not homogeneity in the conditions under which the applications of the different methods are applied, the specialists who want to use the model do not have a trustworthy guide.

The learning rate indicates the size of the pass to make in the searching direction. Theoretically, if it is larger, the convergence to the minimum will be faster, because it can accelerate the learning when a flat region is crossed. However, this is not always convenient a large learning rate can provoke undesirable oscillations, which in some cases can be amortiguated by the corresponding term to the momentum. On the other hand, if the learning rate is little, the convergence speed can be too slow, having the risk of the network to be stuck in a local minimum. If estimating the magnitude order of the step size is uncertain it is more difficult to determine the adequate value. Traditionally, it is used to use a near constant positive value to one, which has produced more or less acceptable results, but there remains the doubt if this is the best.

During an experiment Dai and MacBeth [Dai, 1997] showed –for a problem in particular– that the learning rate and the momentum mainly affect the convergence of learning, and determined that the combinations of \( \eta = 0.7 \) and \( \alpha \in [0.8, 0.9] \) or \( \eta = 0.6 \) and \( \alpha = 0.9 \) exhibit the best convergence. Such combinations are similar to \( \eta = 0.7 \) and \( \alpha = 0.9 \) suggested in [McClelland, 1988; Pao, 1989; Demuth, 1993]

On the other hand, there have been developed different techniques to determine dynamically, during the training the values of the learning rate and the momentum. In general, the techniques of self–dynamic adaptation have as an objective to determine iteratively the values \( \eta \gamma \alpha \) that generate a decreasing in the error function. For this, Salomon and Van Hemmen [Salomon, 1996], propose to calculate the learning rate from its estimation in a step behind. Increase and diminish this value, for a fixed quantity, evaluate the error function and select that that produces the highest diminishing of the error.

Magoulas et al [Magoulas, 1997] use a modified version for the equation (4.2) that allows the usage of a variable learning rate. The optimum value of this is obtained considering that the new weights converge to a value that guarantees the diminishing of the error function.

Because of the change of weights vector in two successive iterations follows the behaviour of the curvature of the error function, Hsin et al [Hsin, 1995] formulate that the modification of the learning rate has to be done by the weigh addition of the director cosines of the increment of the weights vectors.

So far, we have only mentioned the works which central objective is the learning rate, independently from the behaviour of the momentum. Estimating that both parameters are important, Yu and Chen [Yu, 1997], pretend to make the Backpropagation learning efficient with the simultaneous optimization of the learning rate and the momentum. For
this, they make parametrization of the error function taking as parameters $\eta$ and $\alpha$, and through recursive formulas determine the optimum values of learning rate and momentum. These are used to determine the weights in which the subsequent iteration and minimize the error function.

None of the options has achieved to demonstrate, in a general way, its supremacy in the improvement of the performance of the behaviour of the Multilayer Perceptron training with the Backpropagation algorithm. An answer to this situation, in certain way, is given by the second order algorithms.

4. Second Order Algorithms

The effectiveness of any learning algorithms is in the optimum selection of the direction of the weights actualization. The methods of first order do not always have strength enough in the determination of the numeric approximation of the gradient. For this, there have been developed methods of second order that in general present a better development, and are susceptible of being calculated numerically. These frequently, involve in a direct or indirect way the calculus of the Hessian matrix and its inverse. Among the main techniques of second order we can mention: The Newton method, the Constrained Newton method, the Conjugated Gradient method, the Modified Newton method, the Quasi-Newton method. From these the most popular correspond to the Conjugated Gradient and the Quasi–Newton.

Conjugated Gradient Method

As all the second order methods the Conjugated Gradient Algorithm is a method of training per epoch, which can reach the minimum of a multi – varied function faster than the gradient descent procedure. Each one of the steps of the Conjugated Gradient is at least as good as the descending method by steps starting from the same point.

The formula is simple and the use of memory is of the same order as the number of weights. The method is based in the minimization of the error function given by.

$$E(n+1) = E(n) + E'_w(n)^T \Delta w_n + \frac{1}{2} \Delta w_n^T E''_w(n) \Delta w_n$$

(4.6)

An optimum heuristic condition that minimizes $E$, is to choose $\Delta w$ so that $\frac{\partial E(n+1)}{\partial \Delta w} = 0$. From where we obtain

$$E'_w(n) + E''_w(n) \Delta w_n = 0$$

(4.7)

The problem is to minimize the equation (4.6) so that

$$\Delta w_n = \eta_n d_n$$

where
Multilayer Perceptron

\[ d_n = -E'_w(n) + \beta_{n-1}d_{n-1} \]

In other words, the direction of weights actualization is a lineal combination of the actual gradient and the direction of previous actualization. If besides it accomplishes that the error is given by \( E(n) = w^TE'_w \) \( w - w^Tb \) then we have the Conjugated Gradient Method.

For this it is necessary to exploit the orthogonality between \( E_w \) and \( d_{n-1} \):

\[ d_n^TE'_w(n) + \beta_{n-1}d_{n-1} = 0 \]

For this orthogonality the equation can be simplified to:

\[ \beta_{n-1} = \frac{E'_w(n)^TE'_w(n)d_{n-1}}{d_{n-1}^TE'_w(n)d_{n-1}} \]

Additionally, it can be proved that:

\[ \beta_{n-1} = \frac{E'_w(n)^TE'_w(n)}{E'_w(n-1)^TE'_w(n-1)} \]

This last relation will be used in the following procedure.

Conjugated Gradient procedure: It starts from an initial point \( w^0 \) and calculates \( d_0 = -E'_w(0) \). The algorithm of Conjugated Gradient starts from \( n=1 \), for each increase of \( n \) we do the following:

1. Calculate \( E'_w(n) \).
2. Actualize the direction vector \( d_n = -E'_w(n) + \beta_{n-1}d_{n-1} \) where \( \beta_{n-1} \) can be calculated with one of the following relations

\[ \beta_{n-1} = \frac{E'_w(n)^TE'_w(n)}{E'_w(n-1)^TE'_w(n-1)} \]

or

\[ \beta_{n-1} = \frac{E'_w^T(E'_w - E'_{w-1})}{\|E'_w - E'_{w-1}\|^2} \]

3. Actualize the vector \( w: w_{n+1} = w_n + \eta_n d_n \)

A lineal search algorithm is used to find the learning rate. Under the supposition that the cost function is square the lineal search produces

\[ \eta_n = \frac{E'_w(n)^TE'_w(n)}{d_n^TE'_w(n)d_n} \]
Multilayer Perceptron

In the Conjugated Gradient method the last expressions are used to $\beta_{n-1}$ and an algorithm of lineal search to determine $\eta_n$. That avoids, to determine the Hessian matrix explicitly, $E^\prime\prime_{w}(n)$.

Quasi–Newton Method

An example of the methods that are known as Quasi-Newton is the Broyden – Fletcher – Goldfarb – Shannon (BFGS), which pretend to approximate the equality given by the expression (4.7) through the formula

$$H_{n+1}p_i = q_i, \quad 0 \leq i \leq n$$

(4.8)

where $p_n = \Delta w_n = w_{n+1} - w_n$ y $q_n = \Delta E'_{wn} = E'_{w_{n+1}} - E'_{wn}$

The matrix $H_{n+1}$ is an approximation of the Hessian matrix and is calculated in an iterative way with

$$H_{n+1} = H_n + \frac{q_n^Tq_n}{q_n^Tp_n} - \frac{H_np_n q_n^T H_n}{p_n^T H_n p_n}$$

It can be demonstrated that $H_{n+1}$ satisfies the equation (4.8). Supposing that $H_k$ is a good approximation of the Hessian matrix, then by the formula of inversion of a matrix we can calculate in an iterative way the inverse way of the Hessian matrix. ($S_{n+1}$):

$$S_{n+1} = S_n + \frac{1 + q_n^T S_n q_n}{q_n^T p_n} p_n^T q_n - \frac{p_n q_n^T S_n q_n}{q_n^T p_n} q_n^T p_n$$

7. Pruning

A way to estimate the future behaviour of the MP, later the training, is to present a control sample (CS) to its classification. This should be formed with not seen patterns, which will allow examining the behaviour of the generalization. There have been made some works [Gasca, 1999; Schittenkopf, 1997] where the behaviour and the error percentage are observe in the TS and CS in function of the number of iteration. The error in CS decreases to a minimum point, from where its value increases. In the case of the TS such situation is not presented, because even over-passing this point the error keeps diminishing. This phenomenon is known as over–training (over adjustment).

Reed [Reed, 1993] explains such situation as the result of a higher number of liberty degree in the network than the number of patterns in the training sample. If the amount of weights determines the degree of liberty, then reducing the first we avoid the difficulty. This is the equivalent to reducing the number of nodes in the network, and can be done through the application of a pruning technique.
Multilayer Perceptron

There are pruning methods as simple as for example; making 0 any weight and evaluating the change in the error. If it increases to restore the weight, it not definitively eliminates it.

Considering the used procedure to make the elimination the traditional methods of pruning are divided in two groups: those which evaluate the sensibility of the error function to remove the elements with less influence. The methods of sensibility modify the network once that the training is done, this is with the trained network the sensibility is calculated, and with base in its value the weights and nodes are eliminated [Mozer, 1989; Karnin, 1990; Le Cun, 1990; Segee, 1991]. The other group makes the elimination adding a term of penalization to the function to minimize, which praises the network for choosing efficient solutions.

The methods with penalization term modify the function of cost in such a way that the unnecessary weights are equaled to zero in the training [Chauvin, 1989, 1990a, 1990b; Weigend, 1990. 1991a, 1991b; Ji, 1990; Ishikawa, 1990; Matsuyama, 1994; Schittenkopf, 1997].

Additionally to the mentioned processes, there have been generated some techniques with particular methodologies, Sietsma and Dow [Sietsma, 1991], describe an interactive method in which a designer inspects a trained network and decides which node to remove. Many heuristics are used, for instance: a) if a node has constant output for the training patterns, then it does not participate in the solution and can be removed; b) If a number of units have highly correlated answers,(for example, identical or opposite) in all the patterns, they are redundant and can be combined in one unit. Heuristics of this kind are applied during the training to achieve the reduction of a network size.

Bibliography


Multilayer Perceptron


Multilayer Perceptron

capa de salida

\[ W_{kj} \]

\[ W_{ji} \]

capa oculta

capa de entrada

Fig 4.1

Eduardo Gasca Álvarez
Self-Organization Map (SOM)

Table of contents:

10. Introduction
11. Topology
12. Learning Rule
13. Operation stage of network SOM
14. Geometric interpretation
15. Bibliography

Objectives:

When finalizing the study of this unit the student will be able to:

- Describe the operation of the artificial neural network SOM
- Do training with network SOM
- Use SOM to make group tasks (clustering)

Readings:

- Haykin, S., *Neural Networks: A comprehensive foundation*, páginas 397 - 427
- Kung, S. Y., *Digital Neural Networks*, páginas 85 - 90

1. Introduction

The competition by resources is a way to diversify and to optimize the function of the elements of a distributed system, because this takes us to a level of local optimization without the necessity of a global control to assign resources, which is important in distributed systems.

As it was commented in unit 2, one of the types of unsupervised learning of the artificial neural networks is the competitive learning. In it the decision units receive the same information of the input layer and compete (by the lateral connections in the topology or by the learning rate) among them remaining with the entrance element. The objective of this learning is to make categories (clustering) the data that are introduced in the network, in this way the similar information is classified forming part of the same category and therefore must activate the same node. The categories must be created by the own network, because it is an unsupervised learning, through the correlations between the data. The units of the output layer specialize in different areas from the entrance space; their exit can be used to represent in a way the structure of this space.

The competition is by itself a nonlinear operation, and can be divided basically in two types: hard and smooth. The strong competition means that only a node gains the input...
element. On the other hand, the smooth competition alludes to that there is a clear winner, but its vicinity also shares a small percentage of the success.

On the other hand, there are evidences that demonstrate that in the brain are neurons that are organized in many zones so that the caught information of the surroundings through the sensorial organs is represented internally in form of maps of two dimensions. For example; in the visual system, maps of the visual space in zones of the cortex (external part of the brain) have been detected, also in the auditory system is detected an organization according to the frequency to which each neuron reaches greater answer (tonotopical organization).

Although to a great extent this neuronal organization is predetermined genetically, it is probable that part of it originates by the learning, this suggests the brain could have the inherent capacity to form topological maps from the received information of the outside, in fact this theory could explain its power to operate with semantic elements: some areas of the brain simply could create and to order specialized neurons or groups with characteristics of high level and their combinations, definitively special maps for attributes and characteristics would be constructed.

From these ideas Teuvo Kohonen introduced a system with a similar behavior in 1982, it is a model of an artificial neural network with the capacity to form maps of characteristics as it happens in the brain. This network is called Self-Organizing Map (SOM) and makes use of the competitive learning.

2. Topology

Network SOM is composed by two layers of nodes. The input layer (formed by \( N_E \) nodes, one per each characteristic of the pattern) is in charge of receiving and of transmitting to the output layer the information coming from the surroundings. The output layer (integrated by \( N_S \) elements) is the one in charge of processing the information and of forming the map of characteristics. Normally, the neurons of the output layer are organized in form of a two-dimensional map as it is in figure 5.1, although sometimes also layers of a single dimension (linear chain of neurons) or of three dimensions (parallelepiped) are used.

The connections between the two layers that form the network are always feed-forward, it means that the information propagates from the input layer towards the output layer. Each node of entrance \( i \) is connected with each one of the nodes of exit \( j \) by means of a weight \( w_{ij} \). In this way, the elements of the output layer have associated a vector of weight \( W_j \) called vector of reference or codebook because it constitutes the prototype vector of the category represented by \( j \)-th output node.

In addition, between the nodes of the output layer lateral connections of implicit excitation or inhibition exist. Even though such connection physically does not exist, each one of the computational neurons is going to have certain influence on its neighbors. This is obtained
through the process of competition between the nodes and the application of a function called “of vicinity”, which can be, for example, of the Gaussian or Mexican hat types.

3. Learning Rule

The learning in network SOM is of the type off-line, this is why a stage of learning and another of operation are distinguished. In the learning stage the values of the connections between the elements of the output and input layers are set. For this an unsupervised learning of competitive type is used. The nodes of the output layer compete to be activated and only one of them remains active before the presence of a stimulus of entrance to the network, the weight of the connections adjust based on the exit node that has been the winner. In the operation stage, a pattern is given to the network and this associates a winning node to it, which is related to a category.

During the training phase a set of objects appears to the network (described through its characteristics or attributes) so that this establishes, based on the similarity between the data, the different categories. Each output node corresponds to a category, and will serve during the stage of operation to make the categorization of new patterns that appear to the network. The final values of the weight will correspond with the values of the components of the input vector that activates the corresponding neuron. In the case of existing more patterns of training than output neurons, more than one will have to be associated with the same neuron, it means that they will belong to the same category.

For model SOM the learning does not conclude after presenting a single time all the patterns of entrance, but it will be necessary to repeat all the process several times, which refines the topological map of output. The more times the data appear, more reduced will be the zones of neurons that must be activated in the presence of seemed entrances, getting that the network makes a more selective grouping.

A very important concept in the network of Kohonen is the zone of vicinity around the winning neuron \( j \), the weight of the neurons that are in this zone, denominated \( Zone_j(t) \), will be updated with the vector of weight of the winning neuron, it is important to mention that the zone of vicinity changes with the number of iteration \( t \).

The procedure of learning of network SOM is the following one:

1. The weight is initialized, \( w_{ji} \), with small random values and the initial zone of vicinity between the exit neurons is given.

2. A pattern \( p \), is introduced to the network, each node in the input layer corresponds to a characteristic of the entrance vector.

3. As this is about a competitive learning the winning node of the output layer is determined, this will be, that \( j \) which the vector of weight \( w_j \) be the most similar to the entrance pattern \( p \). For each of the exit neurons, the distances or the range of
differences are computed between both vectors, the Euclidean distance
\[ d_j^2 = |p - w_j|^2 = \sum_{i=0}^{N_k} (p_i - w_{ji})^2 \]
or the inner (dot) product \( w_j^T \cdot p = \sum_{i=0}^{N_k} p_i \cdot w_{ji} \) usually
is used. The expression (5.1) shows the criteria to determine the winning node
\[ \text{winner} = \max_j (w_j^T \cdot p) \quad \text{or} \quad \text{winner} = \min_j |p - w_j|^2 \]  
\[ (5.1) \]
\( p_i \) is the \( i \)-th component of the input pattern \( p \); \( w_{ji} \) is the weight of the connection
between the \( i \)-th node of the input layer and the \( j \)-th node of the output layer.

It is possible to mention that when the inner product is used as criterion the vectors
that represent the input patterns and the vector of weight must be normalized. The
previous thing is because the internal product is sensible to the direction and length of
the vectors, which can generate errors. So it is the case shown in figure 5.2, the node
that wins the competition does not correspond to the node 1 (N1), which is the closest
to the input vector \( p \), but the node 2 (N2) that has the biggest value of the inner
product due to an unusual high value.

4. Once located the winning node, \( j^* \), the linking weight between the input units and the
winning one, as well as those of the neighboring nodes are updated. What is wanted
with this is to associate the information of input with a certain zone of the output
layer. The update is made with the equation (5.2)
\[ w_{ji}(t) = w_{ji}(t-1) + \Lambda_{j,j^*}(t) \eta(t) (p_i - w_{ji}(t)) \quad \text{for all } j \in \text{Zone}_{j^*}(t) \]  
\[ (5.2) \]
\( \Lambda_{j,j^*} \) is a function of density focused on the winning node. Typically as the vicinity as
the size of step change with the number of iterations, of such form that they gradually
decrease. For example, if the vicinity function is a Gaussian one with a variance that
decreases with the iterations can be represented with the expression (5.3), it is also
possible to use the function called Mexican hat, figure 5.3.
\[ \Lambda_{j,j^*}(t) = \exp \left( -\frac{d_{j,j^*}^2}{2\sigma^2(t)} \right) \]  
\[ (5.3) \]
The selection of parameters is definitive to reach a map that preserves the topology of
the input space to the discreet entrance. There are two phases in the SOM learning.
First stage tries to order the weight from the topological point of view, this is
implicitly made when defining the vicinity. The duration of this phase is of \( T_0 \)
iterations. In addition, the vicinity function must remain big, covering the total of the
output space, which allows to identify the nodes that respond to similar entrances to
be considered together. Nevertheless, as the vicinity must be reduced to only an
element or an element and its nearer immediacy. For this, normally a linear decrement
of the radius of vicinity is used by means of the expression
\[ \sigma(t) = \sigma_0 (1 - t/T_0) \]  
\[ (5.4) \]
During this period the learning rate must be high (above of 0.1) to allow to the network the self-organization (and eventually obtaining the map). The modification of $\eta$ is also linear and it is possible to be considered through the equation (5.5).

$$\Delta \eta(n) = \eta_0(1 - n/(N + K))$$ (5.5)

Where $\eta_0$ is the reason of initial learning and K helps to specify the final learning rate. The second phase of the learning is denominated phase of convergence. With this one the fine adjustment of the map is performed, so that the vectors of weight adjust more to the training patterns. The process is similar to the previous one although usually it is longer, taking the value from the learning rate constant and small (0.01), and a constant radius of vicinity also equal to 1.

5. The process must be repeated, from point 2, for each element of training, which is called iteration, until fulfilling a certain amount of iterations.

Traditionally the adjustment of the weight is made after presenting a training pattern. Nevertheless, there are authors [Masters, 1993] who recommend to make the update by iteration, that is to say, to accumulate the increases of each element of training and to modify the weight from the average of these increases. By means of this procedure it is avoided that the direction of the vector of weight oscillates from a pattern to another and accelerates the convergence of the network.

An objective criterion does not exist about the total amount of iterations necessary to make a good training of SOM. Nevertheless, the total of iterations must be proportional to the amount of nodes of the map independent to the number of entrance variables. Although 500 iterations by exit node is a suitable number, from 50 to 100 are usually enough for most of the problems [Kohonen, 1990].

4. Operation stage of network SOM

Once finalized the learning stage all the linking weight values between the nodes of the input and output layers they remain constant. Therefore when presenting a pattern $p$ in the layer of entrance of the network SOM, this is directly transmitted towards the exit layer, where each neuron calculates the similarity between the input vector and its own vector of weight ($w_j$). The previous thing using one of the criteria contained in the expression (5.1) and with the objective to identify the winning node. The components of the output vector of network SOM can be determined by means of the following mathematical expression

$$y_{pj} = \begin{cases} 
1 & \text{if } \min_j \left| p - w_j \right|^2 \\
0 & \text{in the other cases}
\end{cases}$$ (5.6)

Where $y_{pj}$ represents the answer of the $j-th$ node of the output layer before the presence of the pattern $p$, when the Euclidean norm is used as a measure of similarity (it is also
possible to use as criterion the inner (dot) product \( \max_j (w_j^T \cdot p) \), being careful of standardizing the vectors.

The node \( j \) that he has as exit the value of 1 is the winning node, therefore the pattern \( p \) will be labeled like pertaining to the category represented by that element. As in the presence of similar patterns the same neuron of exit is activated, or another near to the previous one, then the network is especially useful to establish relations, previously unknown, between data sets [Hilera and Martinez, 1995].

5. Geometric interpretation

The following geometric interpretation [Masters, 1993] of the learning process can turn out interesting to understand the operation of network SOM. The effect of the learning rule is to approach the vector of weight of the node with greater activity (winning) to the input vector, all this in an iterative way. Thus, in each one of the iterations, the vector of weight of the winning node turns towards the entrance, and it gets closer to it in an amount that depends on the learning rate.

In figure 5.4 the way the learning rule operates is shown for the case of several patterns pertaining to a space of entrance of two dimensions, represented in the figure by the vectors in black. Let us suppose that the vectors of the entrance space are grouped in three categories, and that the number of elements of the output layer is also three. At the beginning of the training the vector of weights of the three nodes (represented by vectors in red) are random and are distributed by the circumference. In the way the learning advances, these are progressively approached to the patterns, until staying stabilized as centroids of the three categories.

When finalizing the learning, the vector of weight of each node of exit will correspond with the input vector that is able to activate the corresponding neuron. In the case of existing more patterns of training than exit elements, like in the exposed example, more than a pattern will have to be associated with the same neuron, it means that they will belong to the same category. In such case, the weight is obtained as an average (centroid) of these patterns.

6. Bibliography

Figura 5.1

Eduardo Gasca Alvarez

Figura 5.2

FIGURE 7-4  Wrong choice of winner for the inner (dot) product
Figura 5.3

Figura 5.4