Universidade da Coruña
FACULTAD DE INFORMÁTICA
Departamento de Computación

MODELO DE UN SISTEMA PARA LA SELECCIÓN AUTOMÁTICA EN DOMINIOS COMPLEJOS, CON UNA ESTRATEGIA COOPERATIVA, DE CONJUNTOS DE ENTRENAMIENTO Y ARQUITECTURAS LINEALES DE REDES DE NEURONAS ARTIFICIALES UTILIZANDO ALGORITMOS GENÉTICOS

DOCTORANDO: JULIÁN DORADO DE LA CALLE
DIRECTOR: ALEJANDRO CRESPO SIERRA

A Coruña, Marzo 1999

Universidade da Coruña
FACULTAD DE INFORMÁTICA
Departamento de Computación

MODELO DE UN SISTEMA PARA LA SELECCIÓN AUTOMÁTICA EN DOMINIOS COMPLEJOS, CON UNA ESTRATEGIA COOPERATIVA, DE CONJUNTOS DE ENTRENAMIENTO Y ARQUITECTURAS IDEALES DE REDES DE NEURONAS ARTIFICIALES UTILIZANDO ALGORITMOS GENÉTICOS

DOCTORANDO: **JULIÁN DORADO DE LA CALLE**
DIRECTOR: **ALEJANDRO C. PAZOS SIERRA**

A Coruña, Marzo 1999

UNIVERSIDADE DA CORUÑA
FACULTAD DE INFORMÁTICA
DEPARTAMENTO DE COMPUTACIÓN

TESIS DOCTORAL

MODELO DE UN SISTEMA PARA LA SELECCIÓN AUTOMÁTICA EN DOMINIOS COMPLEJOS, CON UNA ESTRATEGIA COOPERATIVA, DE CONJUNTOS DE ENTRENAMIENTO Y ARQUITECTURAS IDEALES DE REDES DE NEURONAS ARTIFICIALES UTILIZANDO ALGORITMOS GENÉTICOS

DOCTORANDO: **JULIÁN DORADO DE LA CALLE**
DIRECTOR: **ALEJANDRO C. PAZOS SIERRA**
A Coruña, Marzo 1999

D. ALEJANDRO PAZOS SIERRA, Profesor Titular de la Universidade da Coruña.

HACE CONSTAR QUE: La memoria titulada "MODELO DE UN SISTEMA PARA LA SELECCIÓN AUTOMÁTICA EN DOMINIOS COMPLEJOS, CON UNA ESTRATEGIA COOPERATIVA, DE CONJUNTOS DE ENTRENAMIENTO Y ARQUITECTURAS IDEALES DE REDES DE NEURONAS ARTIFICIALES UTILIZANDO ALGORITMOS GENÉTICOS" ha sido realizada por D. JULIÁN DORADO DE LA CALLE, bajo mi dirección, en el Departamento de Computación de la Universidade da Coruña, y constituye la Tesis que presenta para optar al Grado de DOCTOR en Inteligencia Artificial y Computación de la UDC.

A Coruña, 5 de Marzo de 1999

Fdo.: Prof. Dr. D. Alejandro C. Pazos Sierra

AGRADECIMIENTOS

Deseo expresar mi agradecimiento a mi director de Tesis, profesor D. Alejandro Pazos Sierra, por su confianza y apoyo durante estos años de trabajo.

A AntoNino Santos por la difícil tarea de abrir camino y por su generoso apoyo en las sustituciones docentes y de trabajo.

A Nieves por la paciencia, por sus atentas revisiones y, por su presencia.

A Juan Ramón Rabuñal por su colaboración inestimable en el desarrollo del sistema. A Jesús Arias por sus trabajos en el ámbito de las técnicas evolutivas. A Javier Pereira por las horas de pruebas y por los papeleos realizados. A JJ Romero, Delia Alvarez, Ana Belén Porto por los trabajos que realizaron en el ámbito de nuestro laboratorio.

Al resto de integrantes del Laboratorio RNASA por el buen ambiente de trabajo.

A Bernardino Arcay y José Carlos Dafonte por el apoyo dentro y fuera del trabajo.

Al Instituto de Ingeniería de Sistemas y Comunicaciones (INESC) de Portugal por la colaboración mantenida durante estos años y al School of Computing de la Universidad de Sunderland por la posibilidad de poner en común el conocimiento de sus investigaciones.

A mis padres,

familia y

amigos

INDICE

ANEXOS

Capítulo 1: Introducción

Chapter 1: Introduction

"When the only tool you own is a hammer,

every problem begins to look like a nail"

Ph. D. Abraham H. Maslow

1. Introduction

Computing, from the start of Von Neumann's architecture in the 40's [NEUM-56][NEUM-58], has been marked by sequentiality in the processing of information, both in the command execution and in the data reading. Although this kind of architecture has been practically the only one in decades, the sequential execution of instructions is an obstacle that, on the one hand, hinders the growth of the equipment's calculating potential and, on the other, does not seem to be the best option to face certain kinds of complex or not well defined problems.

There has been an attempt at overcoming these limitations in the recent years using parallel processing of information techniques. In the field of hardware architectures, all developers stress the importance of implementing multiprocessor machines, either by integrating several processors in the same machine or by offering communication solutions of different machines, memories and processors with a great bandwidth. All these implementations are already available, even for PCs. In the present situation of continuous increase in the processing capacity and the communication possibilities of the equipment, the problem is now in the field of operating systems and the programming of applications that can make the most of the new developments in hardware and communications.

Obviously, the use of several processors working at the same task entails an increase in the processing potential with respect to an only processor. But, from the point of view of software, the programming philosophy must adapt itself to make the

most of this potential increase in the processing capacity. Von Neumann's architecture set the pace for the development of computers as we know them, establishing the use of binary codification and the inner storing of programmes and data. Apart from these characteristics, it also imposed the sequential access of commands to the central processing unit. Although most of the ideas that support this architecture are still valid, the arrival of parallelization to hardware allows to change the way of executing commands. For this reason, parallelization is more readily admitted in hardware than in software.

Nowadays, it is quite common to find machines with 2 or 4 processors whose programmes do not actually use parallelism, but share the load among the different processors (programmes to be executed by the operative system at a given time). Although this entails a considerable improvement in the machine's performance, obviously it is not the same to execute a programme in a machine using its "n" processors simultaneously in a transparent way for the user, than to share "n" tasks to be executed subsequently by the operative system in "n" different processors.

The arrival of parallelism in computer hardware brings about a real change in the world of computing. However, we should not consider that this step is accomplished until there are real parallel programmes which can use a variable "n" number of processors for their execution.

Returning to the subject of traditional programming, it cannot be assumed that this paradigm will be adapted in a trivial way to parallelization, being designed for sequential use. The attempts carried out up to the present time add process control to the programmers specific tasks. This is a new task that the programmer must codify and include in the programmes. Even when we work in this way, which is not the most logical one, due to the programmers' need of continuous training, a series of important problems exist. On the one hand, there is a possibility that, if a machine cluster is used for parallelizing, not all of them might have the same calculation potential or the same velocity of response. Even if we consider this problem to be solved, it seems impossible to adjust the load to be sent to one of these machines so that it does not overtake or lag behind the rest members of the cluster. On the other hand, one of the machines or processors may have to coordinate the parallelized tasks and manage the process communications. This part of the program is totally different from the rest.

As one can easily imagine, the opposite case also exists. Some problems are orientated to be solved in a parallel way, but they could not be solved that way until now, due to the high price of parallel machines and to the difficulty posed by their programming. Besides, the solution to these problems in monoprocessor machines with traditional programmes requires a considerable time, which limits the use of this approach. Among this type of cases we can find problems liable to be treated with Artificial Neural Networks (ANN from now on). This part of Artificial Intelligence, whose origins can be traced back to 1943 [MCCU-43], is thriving nowadays, after a recession period from the early 70s to the early 90s. An ANN is composed of a set of very simple process elements and a certain number of connections among them. This structure integrates a parallel information processing system. On the one hand, the inner organization orientated to parallelization has always been difficult to adapt to sequential processing. On the other hand, ANNs are capable of working correctly even when some process element fails, i.e. they are failure-tolerant systems. Besides, the ANN training process also implies a parallel information processing system, given that the values associated to the connections of the process elements are also modified in order to adjust the ANN output, which serves as a response to the possible inputs. As it can be seen, ANNs are systems that require a type of parallel programming, both in the construction-training and in the performance phase. ANNs are developed to respond to actual problems, using natural knowledge which can be uncertain, imprecise, inconsistent and incomplete. Here the development of a traditional programme that would cater for every possibility would be unthinkable or, in any case, very expensive and laborious.

A new field of Artificial Intelligence, Evolutionary Programming (EP from now on) is developing fast in the recent years. This field is composed of a set of techniques whose common characteristic is to be based in the mechanism of natural evolution. Although the first steps in EP were taken by John Holland in the 70s with Genetic Algorithms (GA from now on) [HOLL-75], John Koza introduces Genetic Programming (GP from now on) in the 80s [KOZA-92]. At this point in time, we witness an expansion in the research and use of EP techniques.

EP is based on the development of populations of individuals in which each individual represents a solution to a given problem. For instance, in a function optimization problem, the individuals could be the function's quotient sets. In order to

use EP techniques, it is necessary to have an evaluation function that allows to measure each individual's adequacy to solve each problem. From this point, by using the dynamics of reproduction and natural selection, the different solution-individuals are made to evolve into those which best solve the specific problem.

It could be said, in a more formal way, that EP consists of a set of search methods that rapidly converge into the best solution areas, even in complex dominions. The reason for this good performance is that these techniques are based on parallelism. On the one hand, there is an explicit parallelism when an "n" number of solutions is developed and used simultaneously. On the other hand, there is an implicit parallelism, given that each individual samples different areas of the searching space simultaneously, evaluating each of them. This double use of parallelism is made in a natural way in every EP technique.

It is even possible to deal with the use of microprocessors in various ways. The most usual one is the development of separate populations that evolve more or less independently from the rest. As in the case of ANNs, this characteristic means that the speed at which an EP technique works depends on the number of available processors. Performance improves when the number of processors increases, and the failure in one of them simply causes a decline in performance, not a halt in the system, i.e., it is failure-tolerant.

Both ANNs and EP are techniques orientated to the parallel processing of information, with the characteristic that they have not been adapted from a previous stage of sequential processing, but they are intrinsically parallel. Obviously, this kind of techniques are much more efficient in the use of parallel computation resources, offering new ways of solving actual and optimization problems. The future of parallel computation is also the future of these techniques, which present themselves as the right complement for the new multiprocessor environments, which are becoming more and more abundant, thanks to the increasingly lower price of microprocessors.

1.1 Present Situation

Although ANNs constitute a paradigm that has been used successfully for many years in pattern recognition and characteristic extraction tasks, functionally it is still a relatively young technique which, therefore, poses the typical problems of immature techniques. In the first place, there is a great number of ANNs and it is not clear which one is the most suitable for solving each type of problem. Even though several studies have been carried out in order to calculate the performance of various architectures, learning logarithms, etc, applied to the solution of different problems, the fact that only a small number of problems have been proved, together with the fact that most of these problems are lab ones, has the consequence that the results obtained cannot be efficiently applied to actual problems.

In order to tackle these problems, different ANN design methodologies are being currently developed, integrated with various Artificial Intelligence and general computing techniques. In this respect we can mention the methodology developed in Antonino Santos's thesis [SANT-98] for the development of hybrid ANN, ES and DDBB systems. This methodology has been successfully applied to complex problems in the field of Environmental Impact Evaluation. Another remarkable work is the one carried out by the Intelligent Hybrid Systems group of Sunderland University (UK): the HyM methodology [WERM-98] developed for guiding the integration of the different subsystems that are part of hybrid information systems of great complexity.

If the most suitable kind of architecture for solving a given problem has already been chosen, the next step is facing the configuration of the different parameters of each architecture before contemplating the training stage. The first step in this stage is the specification of the number of layers and the number of process elements per layer. As in the previous case, there is no theoretic ground to support the decisions taken in this field of ANN design. There are only empirical studies which offer generic guidelines based on the experience with a series of thoroughly studied problems. However, this part of design is very important, since it has a significant influence in the performance of the final ANN. If the number of layers and process elements is insufficient for the degree of complexity of the problem, the ANN might be incapable of adjusting all of its weights in order to generalize all the possible input-output patterns. In the opposite case,

if the number of layers and process elements is excessive, the convergence might be too slow, with the consequence of wasting time.

The selection of the number of layers and process elements per layer is the first part of the parameter configuration for the training of ANNs. The specification of parameters is also necessary in process elements such as the input, activation and output functions that are going to be used. With regard to the progress of the training process, there are parameters in almost every learning algorithms such as the learning rate or the temperature. Both indicate the magnitude in which the value of the network weights can change. The first one has a constant value, and the second one has a dynamic value that decreases throughout the training process. These parameters influence the speed at which the ANN will converge into its final weight configuration. Together with these basic parameters, there are others which are specific to each architecture that the designer must configure without having any reliable guideline to establish their values, apart from his/her own experience and the trial and error method.

Another issue that has been studied for some time is the selection of the necessary connections inside an ANN in order to solve a given problem in the best possible way. This is called pruning, and it deals with the already mentioned subject of adjusting as far as possible the ANN architecture, number of layers and process elements (PE from now on) so as to obtain the best ANN learning. Although, as it has been already said, this subject is being studied, its practical application poses difficulties, due to the supplementary computational load it entails. Pruning can be different in each layer configuration that is tested in order to solve the same problem. It can even be different for each training process of the same ANN. As it can be seen, the use of pruning for the development of an ANN is not at the same level of the already mentioned parameter configuration. The simultaneous use of pruning techniques with the development of an ANN improves performance but, presently, it overloads the ANN learning process.

Most of the studies and articles published about ANN development optimization focus on the issues already dealt with in this chapter. The spotlight is especially on the part of the ANN architecture that is going to be used, while less time is devoted to the available data of the problem for the training and the test. However, in order to guarantee the ANN's good performance, the selection process of the training set is a key

point. It is necessary to have a good distribution, a balance among the kinds of cases presented to the network. In complex dominions, such as in the case of problems with a time or diagnose component, it is not possible to classify or categorize completely the cases which are presented to the ANN.

Another problem is the development of new architectures, which is more related to the field of ANN research than to its practical application. This field is limited by the adaptation or creation of learning algorithms that allow to train those networks. Any modification in the ANN architecture or the structure and functioning of the PE requires the development of an algorithm that allows to adjust the network weights and the new characteristics implemented in the new architecture. The need to develop these algorithms limits the improvement of ANN models with parameters that enlarge their functions.

Finally, with the development of recurrent ANNs (RANN from now on), all these problems have increased. In this kind of networks, the presence of connections towards any EP of the network, even with elements of the same layer, of previous layers or with the element itself, can complicate and prolong the training process. The complexity of RANN increases with respect to non-recurrent ones, making difficult its training in sequential machines with traditional learning algorithms. The problems of choosing the architecture and number of EP have also increased, while the number of parameters which influence the training process to which the designer must pay attention have multiplied. This type of networks do not allow the application of those techniques developed for the selection of the training set, given that each example is composed of one or several temporal series that require a different treatment.

At the end of the 80s, EP was sufficiently developed to start offering results with techniques such as GA and GP, which were already being applied to different scientific fields. The combination of these techniques with ANN started to be experimented at this time. The aim was to find alternative methods for the training of ANNs, that, until then, had been dominated by gradient descent methods. In this respect, significant improvements were made in the adjustment of ANN weights with GA, proving its adequacy as ANN training technique. Thus, a very generic method for weight measurement was obtained, which could be applied to many different types of networks. These results validated the use of EP in the field of ANNs. From that moment on, the

already mentioned problems posed by ANN's rules of implementation and training have tried to be solved. From the start of the 90s, work has proceeded at a higher level, trying to use EP for architecture adjustment, for learning algorithms and for all the ANN implementation parameters.

As progress is made in the development of more complex ANN models, such as RANN, the range of possibilities for applying ANNs to the solution of problems is enlarged, and the automatization of the ANN design process is starting to be seen as a problem in itself.

1.2 Aims

The researches carried out in this thesis are orientated to the automatization and optimization of the construction of ANNs applied to complex dominions. The different ANN configurations will be evaluated by using evolutionary techniques. The researches presented in this paper are focused on four areas:

1. The design of a method for the selection of an optimal training set from temporal series, used as a complement to the already existing methods for discriminating input variables.

2. The identification of optimal layer, EP and EP parameter configurations for solving specific problems, given that, as it has been noted in the previous point, the construction of an ANN is based on the designer's or knowledge engineer's decisions, who does not have a theoretical ground to base or justify the resulting design as the best possible.

3. The suggestion of an alternative method for the training process in the different types of ANNs. Until now, the training was usually based on gradient descent techniques, which are more vulnerable to the problem of local minima the more complicated the searching space becomes. With the application of ANN to complex problems, such as prediction in the time dominion, gradient methods present this kind of problems.

4. Finally, the attempt at harmonizing the different ANN development stages: training set design, architecture parameter adjustment and training process. The aim is to reach a simultaneous progress in these three stages, so that the advancement in one of them favours the advancement of the others and, therefore, the acceleration of the ultimate ANN implementation.

In order to reach these goals, an integrated system is going to be developed for the study of the following aspects of ANN development:

a) On the one hand, a GA will be used to make a good selection of the training set data, so that the ANN that results from the training can generalize the problem in an optimal way, and the training process is accelerated.

b) On the other hand, the adjustment of the architecture's parameters is going to be carried out by another GA. In this step, the best layer and elements per layer configuration for solving the problem will be achieved. At the same time, the pruning of each solution takes place, obtaining as this module's output the optimal layer and PE configuration network, with the smallest number of possible connections. With this we manage to automatize the development of the ANN as much as possible, saving the designer or knowledge engineer the need to search for the best architecture by means of the trial and error method, which can be rather laborious and tedious.

Both for selecting the training set and for choosing the parameters of the best architecture, we will need to train a certain number of ANNs that will guide the GA in charge of these tasks. The concrete ANN training will also be carried out with the aid of GA. In this step, the GA developed in the present research automatizes the task of configuring the learning parameters, for instance the learning speed and the EP parameters, such as the kind of functions used in their operation. This GA's output consists of a weight set and the activation functions to be used by each EP in the ANN.

For the implementation of ANNs, the system will use a parallel and co-operative approach. In this approach, two simultaneous simulation processes will work each at one aspect: training set selection and architecture design. The progress made in each of them will immediately support the other's improvement.

EP techniques will try to solve ANN design problems in two senses: on the one hand, they will try to automatize the training set's selection process, on the other hand, they will try to optimize the architecture's parameters. An advantage of this automatization in the development of ANNs is that it will be possible to implement and train easily any variation of standard ANN and EP architectures, due to the fact that it will not be necessary to develop a new learning algorithm or to adapt an old one. It will only be necessary to vary the input/output function of the network's EP, or to implement the necessary changes in the relationships among the EP in order to modify the architecture. If these modifications can be explained in a mathematical way, it is possible to define the new value function that any kind of evolutionary technique needs so as to evaluate the functioning of an ANN.

Time processing will be applied in order to verify the system's functionality, within the complex dominions of ANN application. In this field of application, data possess certain characteristics that require a special treatment. As opposed to what happens in other types of problems where the input variables must be independent, so as not to introduce superfluous information in the training, in time prediction problems data are intrinsically dependent. A variable's values in the future will be predicted from what happened in the recent past. Until fairly recently, traditional ANN with time-windows have been used for these tasks. Nevertheless, RANN have been specifically developed to deal with this kind of problems. As these networks have been recently developed, the computation of their training process is not well optimized, which results in very long learning periods with convergence problems. This could be due to the fact that the learning algorithms used with RANN are usually adaptations of already developed algorithms for feedforward networks. Due to the complexity of this field, there is a vast literature on RANN applications to these problems, but it is full of very specific developments which usually cannot be applied to similar problems.

The solution of time prediction problems places a double challenge. On the one hand, the prediction task, a complex problem in itself, apart from the method used to face it (either in the temporal series analysis theory or with ANN). The difficulty arises from the many types of temporal series and from the great variability that most types of real-temporal series exhibit. On the other hand, we have the problems of the ANN technique that is going to be used in order to solve them. As it has already been noted, the existence of a huge number of options of ANN design is multiplied when we work with RANN. In this environment, the use of automatic design and adjustment optimization techniques is very necessary so as to guarantee the network's convergence and the acquisition of designs of manageable size that carry out predictions with a fairly low error margin.

1.3 Benefits

In this thesis we present an ANN development model for complex environments which offers, above all, ANN generalization, the optimization of their structure for each problem and the automatization of great part of their development. Therefore, the aim is to relieve the development problems of ANNs, which have been present almost from the beginning, such as the selection of architectures, network structure and design optimization. The aim is also to ease the development of RANN, automatizing their design and accelerating their learning process.

One of the advantages is the unification of different types of architectures in a more general model based on a EP open to the definition of new parameters and new behaviours. We define a model which includes feedforward ANN and RANN, and which presents a single training method. The development of ANN via evolutionary techniques makes transparent for the final user the underlying ANN models that EP selects and optimizes. A self-configurable generic ANN is being used to solve different kinds of problems in the presence of the designer and the user.

With this kind of systems we are trying to develop much more potent ANN. Thus, using the same calculation potential for training, we are going to obtain finer ANNs than those with architectures developed via the traditional trial and error method. Networks will be more compact, having the most suitable number of connections and EP to solve a certain problem, while these elements are configured in the best possible way for the desired task. Even in the case of the most complex networks, the recurrent ones, a new path could be opened to their general use, that had been hindered until now by the difficulty in their design and training.

This kind of systems also make possible an easier development of the ANN, due to the automatization of the designing stage. A deep knowledge of all the ANN techniques will not be necessary in order to configure all the ANN parameters so that we can make the most of their possibilities. The evolutionary techniques used for the development will carry out the task of searching for the optimal values, exploring in a parallel way many more configurations than those which the designer could analyze efficiently through the traditional trial and error method.

The task of searching for new architectures is also going to be eased by the possibility of adapting quickly the ANN training process. It is only necessary to modify the functions that define the behaviour of the ANN avoiding the task of developing new learning algorithms. Thus we are trying to open a new path for a generic way of ANN training.

Besides, in the field of parallelization, the set of on-line computers is going to be used for the development of ANN in a transparent way for the final user. As it has already been noted, with this philosophy the speed of training is defined by the number of computers, whose natural characteristic is to be failure-tolerant. The use of parallel EP techniques seems to be a very valid solution for multiprocessor environments, given that, intrinsically, EP techniques work in a parallel way. For this reason, accepting that techniques are parallel in a natural way, it is just enough to develop the best methods for implementing the EP parallelization.

With regard to the nearest future, multiprocessor environments, particularly those with independent machines, are going to be more and more common, both in the research field and in the entrepreneurial environment. It is obvious that, among the various parallelization options, a machine with a huge number of processors working in parallel produces a higher cost than a set of machines of similar potential connected to the network. Due to this fact, the option of several machines working on-line will be widespread in the following years. Thus, the new programming techniques will have to make the most of this processing potential in a natural way. Both EP and ANN are capable of doing this. It is enough with orientating their development towards parallelization which leads to an increasingly complex architecture and an easier usage.

1.4 Summary

Once the introduction, aims and benefits of this thesis have been commented, Chapter 2 will deal with the basis of the techniques involved. In the first place, a general view of the connectionist branch of Artificial Intelligence is given. This point will show briefly the progress made in ANN technology from its beginning, and how each model's limitations have been overcome. Emphasis will be made on the problems posed by the design stages of the different types of ANN. The architectures for time-processing will be analyzed in detail, with careful descriptions of their functioning, their design problems and the limitations for their development.

Chapter 2 continues with an explanation of Evolutionary Programming techniques as optimization method, commenting on the points used for the thesis' development and the available options for the application of GA to problem solving.

Chapter 2 finishes with a presentation of the mathematical techniques traditionally used for the modelization and prediction of temporal series. Here the functioning of the ARIMA method is exposed, stating the restrictions needed for the application of temporal series and describing the steps of the modelization and series prediction processes.

Chapter 3 exposes the Antecedents of the proposed method of ANN development using EP techniques. Research made for the prediction of temporal series with ANN is also included. Finally, a reference list of the articles and papers which deal with the application of ANN developed in an evolutive way to complex problems with a time component is given.

In Chapter 4, the basis for the ANN development system are explained via an incremental approach. It starts by explaining how ANN are developed from GA. Each point in the chapter expands the concepts applied to ANN development, incorporating the modifications in ANN models and the automatic adjustment of architectures and training sets. Finally, all the concepts in the system are integrated in the distributed co-operative application intended to be developed.

Chapter 5 contains the Results obtained from applying the various development stages of the system introduced in the previous chapter to several problems typical of time-series prediction.

Chapter 6 presents the thesis Conclusions and the Future Research on the subject.

Finally, Bibliography is included in Chapter 7.

The paper finishes with an Addenda that offers additional information to the Basis Chapter on relevant points, together with a user's manual of the application developed to check the applicability of the ideas presented in the thesis.

Capítulo 2: Fundamentos

2. Fundamentos

2.1 Redes de Neuronas

2.1.1 Introducción

La generación actual de computadoras (de tipo secuencial y basadas en la arquitectura Von Neuman [NEUM-58]) no es capaz, en general, de reconocer, comprender y actuar, de forma eficiente, ante los problemas del mundo real. Sin embargo, hay muchas tareas que resultan adecuadas para ser resueltas mediante computadoras convencionales: resolución de problemas matemáticos y científicos; manipulación de bases de datos; comunicaciones electrónicas; procesamiento de textos y gráficos; autoedición; tareas más o menos sencillas de control industrial; etc.

Por contra, hay otras muchas aplicaciones que se desearía automatizar (control inteligente, visión artificial, etc.), pero que no se ha hecho completamente como consecuencia de las complejidades que implica la programación de una computadora para llevar a cabo esas tareas. En un elevado porcentaje de estos casos, los problemas no son irresolubles utilizando la tecnología actual; lo que sucede, más bien, es que son costosos de resolver, tanto en potencia computacional como en tiempo, empleando computadoras de tipo secuencial. El problema se centra en que si la única herramienta ampliamente extendida es la computadora secuencial, entonces, de forma natural, se intentará resolver todos los problemas en términos de algoritmos secuenciales. Esta forma de trabajar aplicada a los problemas que no es adecuado resolver de esta manera ocasiona que se inviertan grandes cantidades de esfuerzo y dinero para desarrollar sofisticados algoritmos con los que difícilmente se alcanza una solución eficiente, tanto en recursos como en tiempo.

Dado que las computadoras convencionales son evidentemente poco adecuadas para resolver este tipo de problemas, se han buscado nuevos modelos computacionales que permitan hacer frente, eficientemente, a los retos que plantean este tipo de problemas complejos. Así, surge la computación paralela. Una de las líneas de investigación más prometedoras en el desarrollo de la computación paralela son las

Redes de Neuronas Artificiales. En su concepción original, se han tomado ciertas características básicas de la fisiología del cerebro y de las principales células para el proceso de la información que lo componen: las neuronas.

Desde un punto de vista cibernético, en una primera aproximación, se puede ver una RNA como una colección de procesadores muy simples trabajando en paralelo conectados entre sí en forma de grafo dirigido, organizado de tal modo que la estructura de la red sea la adecuada para el problema que se intenta resolver. Como se puede ver en la Figura 2.1, cada elemento de procesamiento de la red se puede representar como un nodo, indicando las conexiones entre nodos mediante arcos o flechas. La dirección del flujo de información dentro de la red se representa mediante el uso de puntas de flecha en las conexiones.

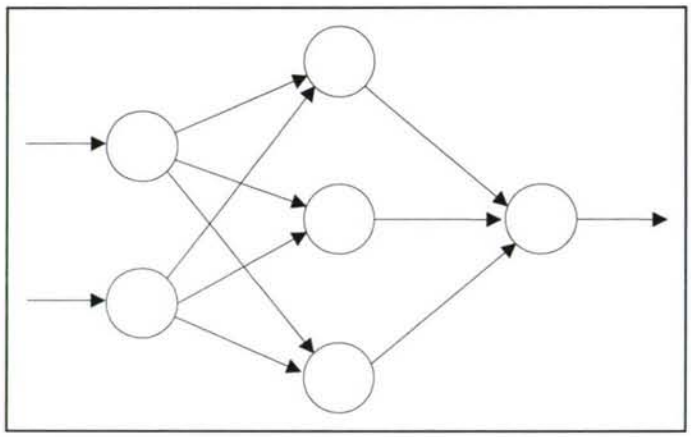


Figura 2.1.- Esquema de una Red de Neuronas Artificiales

Siguiendo el gráfico de la Figura 2.1, se puede ver que la red se compone de un conjunto de unidades o nodos de entrada que reciben las entradas del problema codificadas, un conjunto de unidades de salida que ofrecen la solución que genera la RNA para esas entradas y un cierto número de unidades de procesamiento ocultas que conectan las unidades de entrada y de salida empleando conexiones ponderadas. El valor de estas conexiones se calcula en la fase de entrenamiento en la que se le presentan a la RNA pares de entrada-salida reales para que "aprenda" a partir de ejemplos resueltos del problema. Esta es la ventaja más importante de las RR.NN.AA., ya que no es necesario tener un proceso bien definido para transformar algorítmicamente una entrada en una salida. Más bien, lo que se necesita es una colección de ejemplos representativos del problema que se quiere resolver. La red se

adapta entonces para reproducir las salidas deseadas cuando se le presentan las entradas dadas como ejemplo o unas entradas similares.

La RNA resultante es un tipo de programa robusto, en el sentido de que responderá con alguna salida incluso en el caso de que se averíen o inutilicen algunos de sus elementos o que se le presenten entradas que no haya visto nunca, e incluso entradas que contengan ruido. Si el ruido introducido no altera una parte importante de la información contenida en las entradas, la red producirá una buena sustitución de la información perdida y, empleando la información válida de la entrada, producirá una salida adecuada. La capacidad innata de enfrentarse a tramas ruidosas o distorsionadas es una significativa ventaja de las RR.NN.AA. con respecto a las soluciones algorítmicas tradicionales, sobre todo en problemas del mundo real.

Una vez que la red está entrenada adecuadamente, la información de entrada se propaga a través de los elementos y las capas de la red generando la activación de los elementos de la capa de salida adecuados.

Existen dos tipos distintos de funcionamiento de la red en los modelos con aprendizaje supervisado y alimentación hacia delante: modo de entrenamiento y modo de producción. La naturaleza disjunta de estos dos modos de funcionamiento es otra característica útil de la tecnología de este tipo de RR.NN.AA. El proceso de entrenamiento de la red es la forma de codificar información acerca del problema que hay que resolver y, después, la red invierte la mayor parte de su funcionamiento siendo ejecutada una vez que ha concluido el proceso de entrenamiento. Considérese un sistema que utiliza una simulación software de una red como parte de su programación. En este caso, la red se modela en la computadora de desarrollo como un conjunto de estructuras de datos que representa el estado actual de la red. El proceso de entrenar la red consiste en modificar los pesos de conexión sistemáticamente, de acuerdo a una regla predeterminada de aprendizaje, para codificar las relaciones de los pares entrada-salida deseada. Si se codifica el simulador de la RNA de tal modo que las estructuras de datos empleadas por la red reciban los valores iniciales leyendo los pesos de conexión a partir de un archivo de disco, también se puede crear un simulador de redes de estructura similar en alguna otra computadora de producción. Cuando sea preciso modificar el sistema de desarrollo para que satisfaga nuevos requisitos de funcionamiento, se pueden desarrollar los nuevos pesos de conexión, entrenando el

simulador de redes en el sistema remoto. Posteriormente, se puede actualizar el sistema operacional sin más que cambiar el archivo de iniciación de pesos de conexión, pasando de la versión anterior a la versión posterior que ha sido producida por el sistema de desarrollo.

En el capítulo de Anexos se expone la estructura y funcionamiento de las neuronas naturales, en la que se basa el diseño de las RR.NN.AA. Sin embargo, hay que insistir en el hecho de que los modelos de RR.NN.AA. sólo van a tener un parecido superficial con sus contrapartidas biológicas, cuyo funcionamiento sólo es comprendido parcialmente en la actualidad. También hay que tener claro que el objetivo de esta tecnología no es simular artificialmente la anatomía o estructura de la neurona natural, sino emular sus cualidades de aprender, reconocer y aplicar relaciones entre objetos propios del mundo real. En este sentido, ofrecen todo un nuevo conjunto de herramientas que podrán utilizarse para resolver problemas "difíciles", problemas que se ajusten a una o varias de las categorías de la "maldición de la cuádruple I" que padecen los conocimientos naturales o del mundo real: Inciertos, Incompletos, Inconsistentes o Imprecisos.

2.1.1.1 Circuitos Neuronales y Computación

El primer intento significativo de unir estos dos conceptos se hizo en 1943, a través del trabajo de McCulloch y Pitts [MCCU-43]. Este trabajo es importante por muchas razones, y no es la de mayor peso el hecho consistente en que fueron los primeros en tratar el cerebro como a un organismo computacional.

La teoría de McCulloch-Pitts se basa en cinco suposiciones:

1. La actividad de una neurona es un proceso todo-nada; es decir, no hay respuesta en una activación parcial en una neurona.

2. Es preciso que un número fijo y prefijado de sinapsis positivas sean excitadas dentro de un período de adición latente para que se excite una neurona.

3. El único retraso significativo dentro del sistema nervioso es el retardo sináptico.

4. La actividad de cualquier sinapsis inhibitoria impide por completo la excitación de la neurona en ese momento.

5. La estructura de la red de interconexiones no cambia con el transcurso del tiempo.

La primera suposición indica que las neuronas son binarias: o bien están activadas o bien están desactivadas. Por tanto, se puede definir un predicado Ni(t), que denota la afirmación consistente en que la i-ésima neurona dispara en el instante t. La notación ¬Ni(t) denota la afirmación consistente en que la i-ésima neurona no ha disparado en el instante t. Empleando esta notación, se puede describir la acción de ciertas redes empleando la lógica de proposiciones.

Aunque la teoría de McCulloch-Pitts ha resultado no ser un modelo preciso de la actividad cerebral, la importancia de ese trabajo es muy grande ya que, la teoría ha ayudado a dar forma a los pensamientos de muchas personas que han tenido importancia en el desarrollo de las ciencias de la computación en la actualidad. Así, hay una idea fundamental que se pone de manifiesto en el artículo de McCulloch-Pitts: aunque las neuronas son dispositivos sencillos, se puede obtener una gran potencia de cálculo cuando se interconectan adecuadamente estas neuronas.

2.1.2 Desarrollo de Redes de Neuronas Artificiales

2.1.2.1 El Elemento Formal de Procesamiento

Los elementos individuales de cálculo, el equivalente simplificado de las neuronas naturales, que forman los modelos de RR.NN.AA. suelen llamarse unidades o elementos de procesamiento (en adelante EP). La Figura 2.2 muestra un modelo general de EP, donde el EP tiene varias entradas pero tiene una única salida que se puede aplicar, a su vez, como entrada a muchos otros EP de la red siguiendo los procesos de convergencia-divergencia. La entrada que recibe el i-ésimo EP procedente del j-ésimo EP se indica en la forma x_j (siendo este valor también la salida del j-ésimo nodo, del mismo modo que la salida generada por el i-ésimo nodo se denota x_i). Cada conexión con el i-ésimo EP tiene asociada una magnitud llamada peso o intensidad de conexión. El peso de la conexión procedente del j-ésimo nodo y que llega al i-ésimo nodo se

denota mediante w_{ij}. Con los valores de los pesos, w_{ij}, se intenta emular la intensidad de las conexiones sinápticas entre las neuronas naturales. En estos modelos las cantidades se representan mediante números reales.

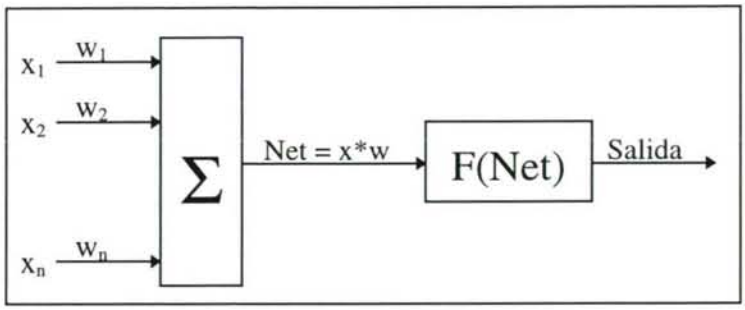


Figura 2.2.- Esquema de Neurona Formal o Elemento de Procesamiento

Las entradas que llegan a un EP son, generalmente, de dos tipos: excitatorias o inhibitorias. Las conexiones excitatorias tienen pesos positivos y las conexiones inhibitorias tienen pesos negativos. Cada EP determina un valor de entrada neto basándose en todas las conexiones de entrada que recibe, siendo normalmente la suma de todas ellas ponderadas mediante sus pesos correspondientes. En otras palabras, la entrada neta de la i-ésima unidad se puede escribir en la forma:

$$neta_i = \sum_j x_j w_{ij} \qquad \text{Ecuación 2-1}$$

donde el índice j recorre todas las conexiones de entrada que posea el EP. La excitación y la inhibición se tienen en cuenta automáticamente mediante el signo de sus pesos. Dado que es frecuente que haya un número de interconexiones muy elevado en las redes, la velocidad con la que se puede llevar a cabo este cálculo suele ser determinante para el rendimiento de la simulación de cualquier red dada.

Una vez que la entrada neta ha sido calculada, se transforma en el valor de activación para ese EP. Se puede escribir ese valor de activación de la forma:

$$a_i(t) = F_i(a_i(t-1), neta_i(t)) \qquad \text{Ecuación 2-2}$$

para denotar que la activación es una función explícita de la entrada neta. La activación actual puede depender del valor anterior de la activación, a(t-1). En la mayoría de los casos, la activación y la entrada neta son idénticas, y los términos suelen emplearse de manera intercambiable. Sin embargo, no siempre tiene porque ser así y, por ello, se hace preciso prestar atención a la diferencia.

La elección de la función de activación es importante ya que puede establecer el rango de la salida del EP. Actualmente, existen varias funciones diferentes y no existe una expresión general, siendo las de uso más habitual: lineal, paso, sigmoide y rampa. En el capítulo de Anexos se puede encontrar una descripción más detallada de estas funciones.

Una vez que se ha calculado la activación del EP, se puede determinar el valor de salida aplicando la función de salida:

$$x_i = f_i(a_i)$$ Ecuación 2-3

Dado que, normalmente, es $a_i = neta_i$, esta función suele escribirse en la forma:

$$x_i = f_i(neta_i)$$ Ecuación 2-4

Una de las razones por las cuales se estudia cuidadosamente el tema de la activación frente a la entrada neta, es que el término función de activación se utiliza en algunas ocasiones para aludir a la función f_i, que transforma el valor de la entrada neta, $neta_i$, en el valor de salida del EP, x_i. Aquí se empleará el término función de salida para aludir a la $f_i()$ de la Ecuación 2-3 y la Ecuación 2-4.

El proceso de aprendizaje consiste en hallar los pesos que codifican el conocimiento que se desea que aprenda el sistema. Para la mayor parte de los sistemas reales, no es fácil determinar una solución en forma cerrada para este sistema de ecuaciones. Existen técnicas, sin embargo, que dan lugar a una aproximación razonable de la solución. Probar la existencia de soluciones estables para estos tipos de sistemas es objeto de multitud de investigaciones en la actualidad, y es probable que siga siéndolo durante algún tiempo.

2.1.2.2 Arquitectura - Topología

La topología de una RNA se refiere a cómo se organizan espacialmente los EP dentro de la red. Normalmente se agrupan en capas, de forma que los EP de cada una de ellas se interconectan con los de las capas anterior y posterior. Las capas de entrada y de salida son casos especiales dentro de cada RNA. Estas capas pueden estar compuestas de EP sin capacidad de procesado que actúen sólo como transmisores de las entradas de la RNA al resto de las capas y de las salidas al exterior de la red. Por este motivo, muchos autores no las contabilizan como capas de la red.

Al tener dividida la RNA en capas, el cálculo de las activaciones de los EP fluye de la capa de entrada al resto de las capas hasta llegar a la capa de salida. Las activaciones no se realizan simultáneamente sino en fases, tantas como capas tenga la red. En el caso de los modelos básicos de RR.NN.AA., las redes no poseen conexiones entre EP de la misma capa, ni de un EP a EP de capas anteriores. Así, las entradas de un EP sólo pueden proceder de salidas de elementos de las capas anteriores de la red. Como se puede ver, en este tipo de redes el flujo de la información siempre es hacia delante, por lo que estas RR.NN.AA. se conocen en la literatura como redes alimentadas hacia delante (feed-forward).

2.1.2.3 Diseño de la Arquitectura

Otra de las labores a realizar antes de entrenar la red y una vez que se ha escogido la arquitectura de la RNA, es escoger los parámetros de esta arquitectura. Los parámetros más comunes son el número de capas ocultas y el número de EP por capa. El número de EP en las capas de entrada y de salida vienen dados por los ejemplos del conjunto de entrenamiento y dependen de la codificación escogida para estos datos por lo que, en este punto, ya están fijados.

Para escoger el número de EP en las capas ocultas y el número de capas no existen reglas fijas, sólo existen heurísticas que indican unos valores que normalmente son aceptables. La más común de estas heurísticas aconseja únicamente una capa oculta y que el número de EP en esta capa sea la mitad de la suma de los EP de la capa de entrada y de la de salida. Realmente no existen muchos estudios que generalicen este tipo de afirmaciones y es la experiencia en el trabajo con RR.NN.AA. la que respalda estas indicaciones. Siguiendo estas directrices, para la mayor parte de los problemas una única capa oculta es suficiente para que el aprendizaje converja y en muy contados casos se observa una mejoría en el proceso de aprendizaje al aumentar el número de capas ocultas de la red, aunque sí es cierto que cuanto mayor sea la diferencia entre los datos de entrada y salida de la RNA, se precisará un mayor número de capas ocultas. El número de EP antes indicado para esta capa no es un valor que se adapte bien a todos los problemas y se plantea como un punto de inicio del entrenamiento, que debe ser estudiado en fases posteriores, para la mejora del rendimiento de las RNA que se desarrollen.

Trabajando sobre estos problemas, P. Yee [YEE-92] realiza un estudio sobre el entrenamiento de redes perceptrón multicapa con una capa oculta aplicadas al reconocimiento de patrones. Para entrenar estas redes utiliza el algoritmo de retropropagación y se centra el estudio en el número óptimo de EP en la capa oculta y en el valor óptimo de la velocidad de aprendizaje. En las conclusiones de estas pruebas cabe resaltar que no se utiliza el Error Cuadrático Medio (ECM) como el único indicador válido, sino que define una probabilidad de que las respuestas de la red sean correctas. Esto es debido a que no sólo es necesario que la red responda adecuadamente a los ejemplos del conjunto de entrenamiento, sino que es un punto fundamental que la red generalice el problema. Como consecuencia de esto es mejor lograr una mayor capacidad de generalización que un mejor acierto (ECM) sobre el conjunto de entrenamiento. Sobre el propio estudio de arquitecturas, y basándose en los parámetros citados de estimación de error, se trabaja con un problema de clasificación de puntos generados con una distribución Gausiana para dos categorías. Para este problema en concreto, la mejor arquitectura que se encuentra tiene dos EP en la capa oculta y una velocidad de aprendizaje de 0.1, habiendo soluciones que ofrecen un menor ECM esta solución maximiza la capacidad de generalización de la red manteniendo bajo el ECM.

Las conclusiones conseguidas no dejan de ser una guía sobre cómo afrontar inicialmente un problema con una arquitectura de RR.NN.AA. Aun con la existencia de estas guías no se evita, por un lado, la necesidad de probar variaciones de la arquitectura para buscar mejoras en el rendimiento y, por otro, no existe ninguna base matemática que ratifique las correspondencias arquitectura-tipo de problema. Además, es difícil, en algunos casos, la inclusión de un problema concreto en las categorías definidas por los estudios por lo que el número de opciones sigue siendo bastante amplio.

Para solucionar estos problemas, se están desarrollando, desde principios de los 90, algoritmos incrementales de RR.NN.AA. que ajustan automáticamente el número de EP. Sobre este tema se habla en un punto posterior de este mismo capítulo.

2.1.2.4 Aprendizaje

La principal característica de la tecnología de las RR.NN.AA. es que no se construyen con el mecanismo de solución de un problema, sino que se diseña su arquitectura y después se entrenan para que puedan resolver un problema. Este

entrenamiento consiste en el ajuste de los pesos de las conexiones entre los EP como consecuencia de la presentación a la red de estímulos a su entrada y, normalmente, respuestas deseadas a su salida, para que la red las asocie y mejore su comportamiento. Aunque idealmente el aprendizaje podría y debería variar la arquitectura de la red en cuanto al número de capas y de EP, las características de conectividad o los parámetros de los EP, ésta no es una práctica habitual y existen muy pocos algoritmos de aprendizaje que varíen cualquiera de estos puntos de la red.

Dentro de estas definiciones generales, se pueden distinguir tres tipos de entrenamiento dependiendo de cómo se planteen al sistema los datos a aprender:

1. No Supervisado: En este tipo de entrenamiento sólo se presentan valores de entrada a la RNA, por lo que no existe información sobre la evolución que sufre el sistema. Con este tipo de entrenamiento se intenta agrupar los datos de entrada en categorías, aún sin tener una idea exacta de su composición ni de su número. Los algoritmos más representativos son los de "aprendizaje competitivo" como el de Kohonen. Normalmente precisan, para un adecuado aprendizaje, de un entrenamiento más amplio que en el aprendizaje supervisado.

2. Supervisado: La característica distintiva de este tipo de entrenamiento es que el algoritmo utiliza la salida correcta que debería producir la RNA y la producida realmente para corregir el valor de las conexiones. Este tipo de algoritmos son los más utilizados y se basan en la disminución del error cometido por la red. La función del algoritmo es minimizar la magnitud del error utilizando métodos clásicos de minimización como el descenso de gradiente en donde el ajuste de cada peso es proporcional a su contribución al error total del sistema. Los algoritmos más utilizados son la Regla Delta y la Regla Delta Generalizada o "Backpropagation".

3. Con refuerzo: Se sitúa en un punto intermedio entre los dos tipos de aprendizaje ya vistos. En este caso, ante unos ciertos valores de entrada, se le indica a la red el grado de acierto de sus salidas ante estas entradas pero sin presentarle los valores concretos necesarios en la salida. Normalmente, la red tiene una entrada adicional de refuerzo para orientar el entrenamiento. Un ejemplo muy común de este tipo es el aprendizaje por búsqueda asociativa.

En el capítulo de Anexos, se ofrece una explicación de los distintos algoritmos de aprendizaje mencionados en los puntos anteriores.

2.1.2.5 Selección del Conjunto de Entrenamiento

En general, aunque se pueden utilizar todos los datos disponibles de un problema para entrenar una RNA, no es conveniente actuar de esta manera. Es mucho más aconsejable trabajar con un subconjunto, bien escogido, para conseguir una convergencia rápida con una buena generalización de la red. El resto de los valores se pueden utilizar en la fase de test para comprobar el adecuado funcionamiento de la RNA.

Es importante diseñar el conjunto de entrenamiento de forma que contemple la mayor cantidad de tipos de entradas y de salidas posibles, para que la RNA tenga la posibilidad de estudiar de la forma más amplia posible el problema que se intenta solucionar [DORA-96]. Actuando de esta forma, la RNA va a poder generalizar la información que le llega en forma de pares entrada/salida. La RNA desarrolla categorías para las entradas comportándose adecuadamente ante estímulos ligeramente distintos a los aprendidos.

En este sentido, existen algunos trabajos teóricos como el de Baum y Haussler [BAUM-89] que estudian cuál sería el tamaño adecuado del conjunto de entrenamiento para conseguir una buena generalización en el aprendizaje de la RNA. En este artículo, se trabaja con una RNA con una única capa oculta que se utiliza como clasificador binario. Según Baum y Haussler si el número de ejemplos escogidos aleatoriamente N que van a formar parte del conjunto de entrenamiento es:

$$N \geq \frac{32\,W}{\varepsilon} \ln\left(\frac{32\,M}{\varepsilon}\right) \qquad \text{Ecuación 2-5}$$

Donde ln representa el logaritmo neperiano, W es el número total de pesos sinápticos de la RNA, M es el número total de EP ocultos y ε va a ser el porcentaje de errores permitidos en la etapa de test. Si, además, el porcentaje de error conseguido en el entrenamiento es inferior a ε/2, la RNA generalizará con éxito el problema. La Ecuación 2-5 da el número mínimo necesario de ejemplos para conseguir una buena generalización. Sin tener en cuenta los valores concretos de la fórmula, se tiene que:

$$N > \frac{W}{\varepsilon}$$

Es decir, que el número de ejemplos necesarios para conseguir una buena generalización es directamente proporcional al número de conexiones e inversamente proporcional al nivel de error que se quiera conseguir en la fase de red. Esta conclusión, que puede parecer hasta cierto punto obvia, se consigue matemáticamente para un número reducido de arquitecturas. Como para muchos otros parámetros del desarrollo de la RNA, viene a refrendar los valores y heurísticas que ya se venían utilizando habitualmente.

Por otro lado, siguiendo con el tema de las características del conjunto de entrenamiento, es importante trabajar con un cierto nivel de ruido en las entradas ya que permite aumentar la generalidad del aprendizaje. Evita la memorización, por parte del sistema, garantizando una mejor adaptabilidad a entradas nuevas no estudiadas durante la fase de aprendizaje.

Por último, siempre dependiendo del tipo de problema, hay que balancear el número de ejemplos por categorías. Si existen "n" categorías y se escogen muchos más ejemplos de una de ellas que de las otras "n-1", obviamente las salidas de la red van a tender hacia esa categoría debido al sobre-entrenamiento que se ha realizado. Este problema aumenta en los casos en que no se tiene constancia del número de categorías presentes en la entrada o en la salida y no se sabe, a ciencia cierta, a qué categoría pertenece cada entrada [DORA-96]. Este problema se plantea normalmente en los sistemas de diagnóstico médico. En este entorno el conocimiento no es explícito, es decir, los médicos no son capaces de explicar de una forma completa la forma en que llevan a cabo su proceso de razonamiento y tampoco son capaces de concretar el conjunto de entradas que utilizan para obtener un diagnóstico. Por tanto, los sistemas que se desarrollen en este entorno van a tener como entrada información redundante y, probablemente, alguna información irrelevante al problema.

Una vez se escoge el conjunto de entrenamiento, también es importante el orden con que los ejemplos se le presentan a la red durante el proceso de entrenamiento. Por ejemplo, no es conveniente presentar secuencialmente todos los ejemplos de cada categoría antes de pasar a la siguiente ya que, de esta forma, los ejemplos de un tipo podrían debilitar el aprendizaje de la categoría anterior.

2.1.3 Modelos Básicos de Redes de Neuronas Artificiales

2.1.3.1 El Perceptrón

El dispositivo conocido con el nombre de "Perceptrón" fue inventado por el psicólogo Frank Rosenblatt a finales de los años cincuenta. Era su intento de "ilustrar algunas propiedades fundamentales de los sistemas inteligentes en general, sin entrar en demasía en ciertas condiciones especiales, y muchas veces desconocidas, que son válidas para organismos concretos" [ROSE-58]. Según Rosenblatt, la conectividad que se desarrolla en las redes biológicas tiene un elevado porcentaje de aleatoriedad. Por tanto, desaprobaba los análisis anteriores, tales como el modelo de McCulloch-Pitts, en los cuales se empleaba lógica simbólica para analizar unas estructuras bastante idealizadas. En realidad, Rosenblatt opinaba que la herramienta de análisis más apropiada era la teoría de probabilidades. Desarrolló una teoría de separabilidad estadística que utilizaba para caracterizar las propiedades más visibles de estas redes de interconexión ligeramente aleatoria. En el capítulo de Anexos se expone en detalle el funcionamiento del "fotoperceptrón" desarrollado para plasmar esta teoría.

Desafortunadamente, los perceptrones dieron lugar a una considerable controversia poco tiempo después de su presentación. No cabe duda de que unas esperanzas poco realistas y unas afirmaciones exageradas tuvieron su influencia en la controversia. En 1969 apareció un libro que, para algunas personas, significaba el final definitivo de las RR.NN.AA. El libro se titulaba, muy apropiadamente, "Perceptrons: An Introduction to Computational Geometry" [MINS-69], y había sido escrito por Marvin Minsky y Seymour Papert, ambos del MIT (Massachusetts Intitute of Technology). En él se presentaba un detallado análisis del perceptrón, en términos de sus capacidades y limitaciones. El que su intención fuera o no la de terminar con el apoyo para la investigación acerca de las RR.NN.AA. sigue siendo una cuestión opinable. Sin embargo, el análisis sigue siendo tan válido en la actualidad como lo era entonces, y muchas de las conclusiones y problemas que plantea también siguen siendo válidos.

En particular, hay ciertas restricciones para las clases de problemas en las cuales es adecuado utilizar un perceptrón, tema que es tratado en detalle en el libro de Minsky y Papert. Los perceptrones de capa única sólo pueden distinguir "tramas" si éstas son

linealmente separables. Dado que hay muchos problemas de clasificación que no son linealmente separables, esta condición impone unos límites bastante restrictivos a la aplicabilidad del perceptrón de capa única.

En el apartado de Anexos se describen las limitaciones del Perceptrón según Minsky y Papert sobre el ejemplo típico de la función XOR y cómo la inclusión de EP en una capa oculta podría solucionar en parte estas limitaciones. Esto ya fue apuntado por Minsky y Papert, pero su gran error fue pronosticar que no se podría encontrar un algoritmo de aprendizaje capaz de entrenar adecuadamente las capas ocultas.

Esto no pretende implicar que todas las criticas del perceptrón podrían tener respuesta añadiendo capas ocultas a la estructura. Lo que se pretende sugerir es que las técnicas siguen avanzando hasta sistemas cuyas capacidades de cálculo son cada vez mayores y que puedan, por tanto, dar soporte a arquitecturas de RR.NN.AA. más complejas que eviten los problemas de las redes ya desarrolladas. Sin embargo, muchas objeciones de las que planteaban Minsky y Papert no deberían tomarse demasiado a la ligera [MINS-88].

De todas ellas la más importante es el problema del escalado. Hay muchas demostraciones de RR.NN.AA. que se basan en la resolución de lo que Minsky y Papert denominan problemas de juguete, esto es, problemas que son solamente simulacros de las situaciones del mundo real. El paso desde estos problemas de juguete a problemas del mundo real se suele considerar sólo como una cuestión de tiempo; únicamente limitado a esperar a que se puedan construir redes más grandes y más rápidas. Minsky y Papert afirman que muchas redes sufren efectos secundarios indeseables cuando se amplían hasta alcanzar un gran tamaño. Se sospecha que existen los problemas de escalado, pero también que parece que se podría encontrar una solución basada en la propia arquitectura del cerebro.

Parece plausible, según la opinión de Minsky y Papert, que el cerebro esté compuesto por muchos sistemas diferentes paralelos, distribuidos, que llevan a cabo funciones bien definidas, pero bajo el control de un sistema secuencial de procesamiento con uno o más niveles. Para resolver el problema del escalado, quizá se necesite aprender a combinar redes pequeñas, poniéndolas bajo el control de otras más grandes.

Por supuesto, una "pequeña" red del cerebro sobrepasa las capacidades actuales de nuestras simulaciones, así que no se sabe exactamente cuáles son las limitaciones. La tecnología de las RR.NN.AA., aunque ya tiene 55 años en este momento, sigue permaneciendo en estado emergente y merece un detenido estudio de las posibilidades que todavía puede aportar.

2.1.3.2 Perceptrón Multicapa

Este modelo de red fue inicialmente propuesto por Paul Werbos [WERB-74] y posteriormente por Parker [PARK-85], Rumelhart y McClelland [MCCL-86] y está diseñada como una red multicapa, con alimentación hacia delante y emplea un modo supervisado de aprendizaje. Este tipo de red surge como solución a los problemas de separabilidad lineal del perceptrón inicial monocapa. La presencia de capas ocultas elimina esta limitación permitiendo afrontar problemas no lineales. Sin embargo, fue necesario el desarrollo de un nuevo algoritmo de aprendizaje que permitiera ajustar los pesos de las conexiones. El método que se utiliza en el entrenamiento del perceptrón multicapa es la regla delta generalizada que, como ya se ha visto, propaga hacia atrás los valores del error de la capa de salida modificando los pesos de las conexiones dependiendo de la contribución de cada peso al error final. Este proceso se repite para cada capa hasta que todos los EP hayan recibido su parte proporcional de la señal de error.

Los pasos del entrenamiento de un perceptrón multicapa son los siguientes: en primer lugar se aplica a la entrada de la red el vector de entradas. A continuación, se calculan los valores de activación de los EP de la capa oculta y, con estos, los valores de sus salidas. Las salidas de los EP de la capa oculta se pasan a las entradas de los EP de la capa de salida para calcular su activación y la salida de estos EP. El paso siguiente es comparar las salidas de la red con las salidas deseadas para calcular el error. Con el valor de error global se calculan los términos del error de la capa de salida y, después, con estos términos se calculan los de la capa oculta. Por último, se actualizan los pesos de ambas capas para volver a alimentar la red con entradas y repetir el ciclo. El proceso de entrenamiento parará cuando el error descienda de un cierto valor predeterminado.

Debido a la naturaleza genérica de su proceso de aprendizaje y a su facilidad de implementación, el perceptrón multicapa es una herramienta muy utilizada. Además, es capaz de entrenar redes con éxito en una amplia variedad de problemas.

2.1.3.3 Consideraciones sobre los Modelos Básicos de RR.NN.AA.

En este punto, se comentan los temas en los que la tecnología de RR.NN.AA. expuesta hasta ahora presenta problemas. Estos problemas, aunque se plantean a partir de los modelos simples de RR.NN.AA., han sido heredados en su mayoría por los modelos actuales, con lo que siguen estando vigentes. Además, la complejidad de los modelos de redes temporales, comentados en un punto posterior, agrava la herencia de este tipo de problemas.

2.1.3.3.1 Emulación del Modelo Biológico

El algoritmo de "retropropagación del error" es un buen ejemplo del paradigma de algoritmo de aprendizaje conexionista ya que se basa en computaciones locales para desarrollar capacidades de procesado de información en RR.NN.AA. El uso de computación local en el diseño de RR.NN.AA. viene dado por tres razones principalmente:

1. Las RR.NN.AA. que realizan este tipo de computación local son una simulación simplificada de las redes de neuronas biológicas.

2. La utilización de computación local permite disponer de la tolerancia a fallos que independiza, en cierta medida, de errores "hardware" que se puedan producir.

3. La computación local favorece el uso de arquitecturas paralelas como un método eficiente de acelerar el procesamiento de la información en las RR.NN.AA.

Los puntos 2 y 3 no plantean dudas en cuanto a que favorecen el funcionamiento de las RR.NN.AA. El punto 2 ha sido validado, hasta cierto punto, con la "inyección" en RR.NN.AA. de valores erróneos transitorios para comprobar que su efecto era mínimo en los resultados [BOLT-92]. En relación con el punto 3, el algoritmo de retropropagación ha sido implementado sin problemas en ordenadores paralelos [PARA-90][SANC-94].

Sin embargo, con respecto al punto 1 existen serias discrepancias [CRIC-89][STOR-89][SHEP-90]. En primer lugar, en cuanto a las conexiones, en un perceptrón multicapa las conexiones sinápticas entre los EP tienen unos pesos que pueden ser excitatorios o inhibitorios, mientras que en el sistema nervioso natural, las neuronas son las que parecen representar estas funciones. Por otro lado, el algoritmo de retropropagación implica que el cambio del valor de las conexiones requiere la transmisión de la señal de error hacia atrás en la RNA. Este comportamiento no es factible en una neurona natural que es incapaz, siguiendo la teoría de la "polarización dinámica" [CAJA-09], de transmitir eficientemente de forma inversa a través del axón hasta el soma celular. También respecto al tema del aprendizaje, si es supervisado, implica la existencia de un instructor. Esto, traducido al contexto del cerebro, implicaría la existencia de un conjunto de neuronas que actuarían de forma distinta al resto para orientar su aprendizaje. La existencia de este tipo de neuronas es biológicamente indemostrable en la actualidad. Por último, en los modelos de RR.NN.AA. no se tienen en cuenta los sistemas periféricos al neuronal, como pueden ser el hormonal, el glial y otros que se sabe que son críticos en el sistema neuronal natural en comportamientos como la atención o el aprendizaje.

La existencia de estas diferencias entre los modelos con retropropagación y el modelo natural no son demasiado importantes en sí mismas. En su diseño no se buscaba tanto una completa similitud con el modelo natural sino que mostrara comportamientos parecidos en su funcionamiento final. Pero, también es verdad, que la aproximación de las RR.NN.AA. al modelo natural es una necesidad en la que se confía para conseguir tanto mejoras de rendimiento como el incremento de complejidad, con el objetivo final de la aparición de comportamientos realmente "inteligentes".

2.1.3.3.2 Arquitectura, Modelos y Escalado

En un punto de este capítulo ya se han comentado los estudios y estado actual del tema de diseño de la arquitectura de RR.NN.AA. El problema reside en que el cálculo del número de capas y de neuronas de una RNA no sigue reglas fijas que se puedan utilizar a la hora de enfrentarse a distintos problemas.

Tampoco es un tema sencillo la ampliación de los modelos existentes de RR.NN.AA. para conseguir el desarrollo de nuevas arquitecturas. La necesidad de

integrar los nuevos parámetros en los algoritmos de aprendizaje para que ajusten sus valores limita la innovación en este campo. Es, por tanto, más común, la aparición de distintas optimizaciones de los algoritmos más utilizados que afrontan, sobre todo, el rendimiento de los cálculos, trabajando la parte computacional del algoritmo. Es mucho menos frecuente la aportación de nuevos parámetros a los modelos de EP que supongan nuevas funcionalidades a las RR.NN.AA.

Por último, en la mayor parte de las investigaciones realizadas en la época del desarrollo de los perceptrones, los problemas a los que se aplicaban las RR.NN.AA. eran problemas de laboratorio, también llamados "juguete". Los éxitos conseguidos en la resolución de estos problemas no garantiza que la aplicación de estas arquitecturas a problemas complejos del mundo real consiga los mismos resultados y, aunque los resultados conseguidos sean buenos, la experiencia en el diseño de RR.NN.AA. para los problemas de laboratorio no supone una reducción del tiempo de pruebas de arquitecturas para los problemas reales, ya que no se ha conseguido una asociación entre arquitecturas y tipos de problemas. Se detecta, aún, una falta de generalización que permita acelerar la velocidad de desarrollo de las RR.NN.AA., una forma de automatizar la labor de diseño y entrenamiento de las RR.NN.AA.

2.1.3.3.3 Convergencia

Suponiendo solucionados los problemas planteados en el punto anterior, se puede decir que ya se dispone de una estructura de RNA con la que realizar el entrenamiento. En este proceso, con los datos de entrada, se intenta conseguir una configuración adecuada de pesos en las conexiones de la red. Sin embargo, un buen diseño de la estructura de la red no garantiza un buen entrenamiento.

En este punto entran en juego los parámetros propios de la fase de aprendizaje, momento y tasa de aprendizaje, que modifican el comportamiento del proceso de descenso de gradiente. La configuración de estos parámetros no sólo influye en la velocidad de aprendizaje sino también en la probabilidad de caer en mínimos locales. Obviamente, los valores óptimos de estos parámetros no son conocidos "a priori" y cambian en cada problema. El ajuste de estos parámetros sería interesante tanto para acelerar el proceso de aprendizaje como para aumentar las posibilidades de convergencia de la red. Incluso es razonable pensar que, dependiendo de la fase del

proceso de aprendizaje, fase inicial de ajuste grueso o fase final de ajuste fino, los valores óptimos de estos parámetros fuesen distintos; con lo que, lo ideal, sería desarrollar una función para cada uno de los parámetros mencionados.

2.1.3.3.4 Mínimos Locales

El descenso del error es un método que degrada su rendimiento con la presencia de mínimos locales. En los espacios de soluciones de problemas complejos del mundo real, la presencia de mínimos locales y de no linealidad es más frecuente y, por tanto, los métodos de descenso de gradiente encuentran más dificultades para alcanzar una eficacia adecuada. En el campo de las RR.NN.AA., estos métodos han sido los más utilizados para los procesos de entrenamiento.

Otro problema asociado es que, al entrenar una RNA con estos métodos, no se tiene la seguridad de encontrar el mínimo global; sin embargo, esto se suple con las decisiones del diseñador de prolongar el aprendizaje hasta conseguir un valor de error aceptable. Una vez marcado un nivel de validez, el entrenamiento que consiga un error inferior se aceptará como bueno, suponiendo que no es necesario alcanzar el mínimo de la función. El problema radica en la necesidad de realizar varios entrenamientos para garantizar la bondad de los parámetros de la RNA escogidos.

2.1.4 Modelos avanzados de RR.NN.AA. para Procesamiento Temporal

Como se ha visto en los puntos anteriores, a través de una evolución, el algoritmo de retropropagación ha superado las limitaciones iniciales para poder resolver problemas no lineales. Sin embargo, hasta ahora sólo se ha hablado de problemas de reconocimiento de patrones estáticos ya que el algoritmo de retropropagación no está preparado para afrontar de una forma natural la resolución de problemas con una componente temporal.

Pese a las limitaciones de su concepción, el algoritmo de retropropagación puede realizar predicciones en series temporales siempre que éstas sean estacionarias. Es decir, que sus parámetros fundamentales no cambian en el tiempo. Para este tipo de problemas, se puede diseñar una red que simule retardos de forma que el vector de

entradas x de la red en cada momento esté compuesto por los "n-1" a "n-p" instantes de tiempo anteriores y se consiga la predicción del instante "n". Siendo "p" el orden de predicción.

$$x = [x(n-1), x(n-2), \ldots, x(n-p)]$$

Esta aproximación es sólo una metodología básica para predicción de fenómenos temporales que utiliza el algoritmo de retropropagación estándar. Es válida sólo para fenómenos dinámicos estacionarios y no es lo suficientemente flexible como para adaptarse a cambios en el comportamiento de las series temporales. El problema es, pues, como extender el diseño del perceptrón multicapa para que asuma de una forma natural el tratamiento de señales temporales. Esto significa añadir propiedades dinámicas para que pueda responder ante fenómenos temporales.

En este sentido, la solución más obvia es la incorporación de algún tipo de memoria en el funcionamiento de la RNA [ELMA-90]. La forma más directa de incorporar una memoria es implementar un conjunto de retardos entre los EP de las distintas capas de la RNA y ajustar el valor de estos retardos durante la fase de entrenamiento. Esta idea se basa en el conocimiento de cómo funcionan las neuronas naturales y en el papel que juegan los retardos en el procesado neurobiológico de la información en el cerebro [BRAI-77].

Sobre esta idea de la implementación de memorias mediante retardos se desarrollaron un conjunto de algoritmos que generalizaban de una u otra forma el funcionamiento del perceptrón multicapa. El primero de ellos es el Time Delay Neural Network (TDNN) que fue descrito por primera vez por Lang [LANG-88] y Waibel et al. [WAIB-89]. Un modelo mucho más eficiente es el algoritmo Temporal Back Propagation [WAN-90a] [WAN-90b] que simplifica la computación y mejora la aproximación temporal. Ambos son descritos en el capítulo de Anexos.

2.1.4.1 Consideraciones sobre los Modelos Temporales basados en Retardos

Todos los modelos expuestos en este punto se basan en la existencia de retardos dentro de la estructura de la red de forma que, ante activaciones pasadas de las

neuronas, parte del valor de estas activaciones permanezca disponible para el procesado futuro de la red y complemente el tratamiento de los datos del instante actual.

Aunque este tipo de arquitectura no es desacertado, esta primera aproximación al procesado temporal sufre una serie de problemas relacionados con el hecho en sí de la utilización de retardos. Su número ha de ser cuantificable y debería ser la propia red la que especificara su necesidad y su valor. Además, no existe información a la entrada de que acciones se tomaron en el pasado; es decir, cuáles fueron las salidas. No existe, por tanto, realimentación.

Como respuesta a estos problemas, a principios de los años 90 se empezaron a desarrollar algoritmos de aprendizaje de redes realmente recurrentes en los que la existencia de retardos se sustituye por la conexión de las neuronas hacia capas anteriores e, incluso, con ellas mismas, como método para mantener niveles de activación dentro de la red para complementar la respuesta a futuros estímulos. Este tipo de redes son la motivación del siguiente punto.

2.1.4.2 Redes de Neuronas Artificiales Recurrentes

2.1.4.2.1 Topología

Una red se considera recurrente cuando existen realimentaciones entre EP de una capa hacia EP de la misma capa o de las anteriores. La existencia de realimentaciones hace que estas RR.NN.AA. sean mucho más potentes. Por un lado, al aumentar el número de conexiones aumenta la capacidad de la red para almacenar información y que, por otro lado, estas conexiones recurrentes le permiten a la red tener una "memoria" de activaciones anteriores de los EP de la red. De hecho, está demostrado que una RNA recurrente trabajando sobre un problema temporal puede aprender a emular a un autómata determinístico de número de estados finito [SIEG-91] y, en general, a cualquier máquina de Turing [MINS-67][ALON-91].

2.1.4.2.2 Algoritmos y Modelos

Las RR.NN.AA. basadas en retardos no tienen en cuenta la existencia de recurrencias entre los EP de la red y utilizan retardos en la estructura de la red para conseguir una respuesta en el tiempo. Una red con un comportamiento realmente

temporal debe incluir conexiones recurrentes entre los EP de forma que se pueda mantener un cierto nivel de activación en el instante "n" dependiente de los sucesos (activaciones) ocurridos en los "p" instantes anteriores. Las dos aproximaciones más utilizadas para entrenar redes recurrentes son los algoritmos Back-Propagation Through Time y Real-Time Recurrent Learning. El primero de ellos es una extensión del algoritmo estándar de retropropagación. Se obtiene desdoblando el procesado temporal de la red recurrente para conseguir una red multicapa alimentada hacia delante. Esto implica que la topología de la red obtenida crece una capa en cada ciclo temporal de procesado. Este algoritmo es original de Werbos [WERB-74] y fue redescubierto independientemente en 1986 por Rumelhart et al. [RUME-86].

El algoritmo Real-Time Recurrent Learning realiza el entrenamiento de las redes recurrentes sin necesidad de transformar su arquitectura a la hora de ajustar sus pesos. Lo que hace superior a este algoritmo con respecto al resto de los vistos hasta ahora es la forma de tratar las entradas para conseguir unas salidas en el tiempo a través de una auto-organización interna, sin ningún artificio como puede ser el desdoblamiento de la red en el tiempo. Fue presentado por Williams y Zipser en 1989 [WILL-89] y se basa en trabajos anteriores de McBride y Nardendra [MCBR-65] sobre el ajuste de parámetros en sistemas dinámicos.

Dentro de las RR.NN.AA.RR. existe un tipo especial que se caracteriza por la existencia de un conjunto de neuronas ocultas que mantienen un nivel de activación que se realimenta en el instante siguiente. Este conjunto de neuronas ocultas, también llamadas de contexto, contiene las neuronas que realmente tienen conexiones recurrentes. Debido a que no todas las neuronas tienen este tipo de conexiones, estas redes se denominan parcialmente recurrentes.

En el capítulo de Anexos se comenta en detalle el funcionamiento de cada tipo de red y cada algoritmo de aprendizaje, comentados en este punto.

2.1.4.2.3 Consideraciones sobre los Modelos Temporales Recurrentes

Las redes recurrentes parecen la solución idónea para afrontar problemas que presenten una componente temporal. La existencia de recurrencias entre las conexiones permite mantener un estado de activación que contextualiza la llegada de nuevos datos en el tiempo de una forma natural. Esto evita tener que utilizar artificios como las

ventanas temporales de las redes alimentadas hacia delante o los retardos de los modelos temporales ya comentados.

Sin embargo, aunque su adecuación a este tipo de problemas es alta, este tipo de redes presentan también problemas. Entre los más graves están la potencia de computación necesaria para su entrenamiento, debido a la necesidad de mantener y utilizar una gran cantidad de datos en el tiempo para proceder al ajuste de sus pesos. Por otro lado, la mayor complejidad de estas redes provoca que la convergencia del entrenamiento sea más lenta y dificultosa, pudiendo caer en mínimos locales debido a la complejidad de los espacios de búsqueda de los problemas sobre los que trabajan.

2.1.4.3 Problemática de las RR.NN.AA. para Procesamiento Temporal

En los puntos anteriores de este capítulo se han comentado dos aproximaciones, ambas válidas, para tratar problemas temporales mediante RR.NN.AA. La primera de ellas se basa en el desarrollo de un modelo espacio-temporal de neurona que sea capaz de mantener una especie de memoria de las activaciones pasadas. Por otro lado, existe otro enfoque en el que se utilizan neuronas estándar y se desarrolla la capacidad del procesado de información temporal por medio de la inclusión de conexiones recurrentes. Este otro enfoque da lugar a las redes recurrentes y sus variantes.

Cualquiera de las soluciones o algoritmos de entrenamiento son adecuadas, en principio, para el proceso de datos temporales. Hay que tener en cuenta la especificidad de cada tipo de entrenamiento, comentada en el propio capítulo, lo que hace que, dependiendo de las características del problema a resolver unos entrenamientos sean más apropiados que otros. Así, por ejemplo, los últimos algoritmos sobre redes recurrentes demandan muchos más recursos, tanto de memoria como de capacidad de proceso, mientras que, los basados en retardos, demandan menos capacidad de máquina pero no son los más adecuados para el procesado en tiempo real. Aunque, "a priori", se puedan dar estas indicaciones de cuando usar un determinado tipo de red para un procesado temporal, como en el caso de los tipos de redes no temporales vistas anteriormente, no existe ningún resultado matemáticamente probado que ayude a escoger el tipo de red.

Tampoco se dispone de indicaciones sobre el número de neuronas a utilizar o sobre el número de conexiones que, en el caso de redes de procesado temporal, son mucho más numerosas. La gran proliferación del número de conexiones provoca la necesidad de minimizar en lo posible su número para poder acelerar el proceso de entrenamiento. Sin embargo, tampoco se ha avanzado mucho, todavía, en el podaje de conexiones en redes recurrentes.

Por último, los algoritmos que se han visto son de computación intensiva, unos (redes con retardos) porque desdoblan la red multiplicando en número de conexiones con respecto a los instantes en que se computa la red y los otros (redes recurrentes) porque necesitan del almacenamiento en el tiempo de variables para la modificación de los pesos. Debido a esto, cualquier tipo de reducción en la complejidad del entrenamiento de redes de procesado temporal parece ser bien venido para poder realizar el proceso de entrenamiento a estaciones de bajo coste en un tiempo razonable.

2.1.5 Modelos de Redes Incrementales

2.1.5.1 Introducción

En varios puntos de la explicación sobre RR.NN.AA. se pone de manifiesto un problema común, no hay reglas claras a la hora de escoger una u otra arquitectura. Este es uno de los problemas principales a la hora de aplicar RR.NN.AA. a un cierto problema. Desde el punto de vista matemático sólo existen algunos resultados [LAPE-88][CYBE-89] que indican el número mínimo y el máximo de neuronas necesarias para casos muy específicos que, obviamente, son difíciles de generalizar para otro tipo de casos y, por tanto, difíciles de usar en problemas reales.

Muy recientemente han surgido un tipo nuevo de algoritmos de aprendizaje que son capaces de modificar la arquitectura de la red añadiendo nodos según va siendo necesario para la resolución del problema. Estas redes se denominan redes neuronales incrementales. También se les denomina evolutivas debido a que su estructura no es fija y va adaptándose y evolucionando para poder resolver el problema que se le plantea. También existe la modalidad de redes decrementales que funcionan de forma inversa, eliminando nodos a partir de una topología de red inicial.

Además de la ventaja obvia de conseguir una red con una topología con el número "probablemente" óptimo de neuronas, las redes incrementales aportan ventajas en tres campos:

La primera se deriva de contener el número exacto de neuronas. Si la red que se quiere entrenar no tiene un número innecesariamente amplio de neuronas el proceso de entrenamiento es más corto, debido al menor número de conexiones. Además, se evita caer en un sobreentrenamiento que impediría conseguir la generalización necesaria para el adecuado entrenamiento de una RNA.

La segunda ventaja parte de que si el tamaño de la red es más reducido, al ajustar el número de neuronas, el número de conexiones también será menor, por lo que el proceso de aprendizaje consumirá menos tiempo. Con cada agregación de neuronas al sistema se ofrece al sistema mayor capacidad para resolver problemas complejos, por lo que en cada paso se está convergiendo hacia la solución del problema.

Por último, algunos tipos de redes incrementales permiten que, ante la llegada de nuevos datos al conjunto de entrenamiento, se puedan añadir nuevas neuronas que se encarguen del procesado de los nuevos datos, sin necesidad de realizar un re-entrenamiento de la red completa.

Debido a su forma de funcionamiento, que se verá a continuación, las redes incrementales se pueden aplicar fundamentalmente a problemas de clasificación, que tengan que ver con la teoría de la decisión. Son problemas clásicos para este tipo de redes: el reconocimiento en imágenes y sonido, la codificación, los sistemas expertos y las memorias asociativas.

2.1.5.2 Fundamentos

Un sistema clasificador, ya esté basado en una RNA o en cualquier otro método, se reduce a un sistema que es capaz de agrupar adecuadamente patrones. Esto quiere decir que, a partir de un conjunto de características o propiedades que se tienen de un objeto, a través de un procesado de estos datos, se obtiene la clase a la que pertenece el objeto de entrada que se está estudiando.

Tradicionalmente, un clasificador óptimo, se puede basar en la teoría estadística Bayesiana, obteniendo las probabilidades "a priori" y "a posteriori" de que cada patrón pertenezca o no a una determinada clase. Lo que se va a obtener con esto es una función discriminante que identifique si un determinado punto o patrón pertenece a una clase o a otra. Lo que se intenta con cualquier otro método utilizado, como las RR.NN.AA., es aproximar el clasificador de Bayes encontrando discriminantes que separen las clases, minimizando el número de patrones mal clasificados.

Para realizar esta aproximación se puede partir de distintos puntos de vista. En primer lugar se pueden utilizar redes que se basan en la determinación de uno o varios discriminantes conformando clasificadores de los denominados "clasificador de separación lineal a tramos" (*Piecewise Linear Separation* - PLS). En este tipo de redes se obtienen discriminantes mediante la utilización de secciones de varias funciones lineales para separar los patrones de las distintas clases.

Una segunda orientación es la creación de regiones dentro del espacio de patrones, estando cada región asociada a una de las clases. De esta forma, se sustituye la definición de un discriminante por la especificación de varias regiones. A este segundo tipo de red se le denomina "red de regiones de influencia" (*Region Of Influence* – ROI).

En los dos métodos comentados se evita el cálculo de la densidad de probabilidad de la teoría Bayesiana y se trabaja con la probabilidad "a posteriori" que se calcula a través de las muestras. Esto plantea que surjan problemas de sobreentrenamiento al intentar adaptar la definición de discriminantes o regiones a muestras que pueden no ser representativas. Esto se puede evitar con el tercer enfoque de redes incrementales, las denominadas RR.NN.AA. probabilísticas, que sí utilizan la densidad de probabilidad en la resolución de los problemas de clasificación.

2.1.5.3 Consideraciones sobre las RR.NN.AA. Incrementales

Como se ha podido observar en la exposición de este capítulo, las redes incrementales surgen como respuesta al problema de la especificación de la topología en las RR.NN.AA. clásicas. Con este tipo de redes, y partiendo de una configuración mínima de una neurona o una clase, es posible llegar a utilizar sólo el número necesario de neuronas para resolver eficientemente un problema determinado.

En los modelos que se han repasado en este punto, no se llega a solucionar del todo la dependencia que muestran de la topología inicial con que se inicia el proceso de entrenamiento. Este problema hace que el diseñador tenga que recurrir, de nuevo, al proceso de "prueba y error" para, a partir de varias inicializaciones con distintos parámetros escoger la configuración inicial adecuada para su problema.

Otro de los puntos débiles de las redes incrementales es que están orientadas, sobre todo, a procesos de clasificación, por lo que pueden no ser adecuadas en otro tipo de problemas como predicción o producción, no habiendo ningún algoritmo incremental orientado a estas otras situaciones.

Por último, tampoco se han desarrollado hasta la fecha algoritmos incrementales para el desarrollo de redes recurrentes, las cuales, como se ha visto, debido a su grado de complejidad y número de conexiones, necesitarían de algoritmos que ajusten y, sobre todo, minimicen su tamaño con vistas a un proceso de entrenamiento lo más breve posible.

Debido a esto, se puede decir que, aunque el campo de las redes incrementales tiene un futuro muy prometedor con respecto al desarrollo de redes bien dimensionadas, de momento, su uso está restringido al área de la clasificación y no se contempla la solución de problemas temporales. Existe, sin embargo, una gran actividad investigadora en este tipo de redes, con lo que hay que esperar interesantes avances para estos próximos años.

2.2 Algoritmos Genéticos

2.2.1 Computación Evolutiva

2.2.1.1 Introducción

El campo de la Computación Evolutiva (CE) es muy joven. De hecho, el propio nombre del área se acuñó en el año 1991 en la celebración del Workshop on Parallel Problem Solving form Nature [SCHW-91], y representa un esfuerzo por acercar a un conjunto de investigadores que habían seguido distintas aproximaciones para simular distintos aspectos de la evolución. Estas técnicas como los Algoritmos Genéticos (AA.GG.), las Estrategias Evolutivas (EE) y la Programación Genética (PG) tienen todas en común que utilizan la reproducción, el azar, la competición y la selección de los individuos de una población. Así, en cualquiera de estas técnicas evolutivas están presentes la esencia de la evolución y al menos uno de estos cuatro procesos presuponiendo que, tanto en la naturaleza como en la informática, la evolución es la mejor herramienta de optimización [ATMA-76]. Desde un primer momento, las ideas de la CE se orientan hacia cuatro objetivos fundamentales.

- Optimización. La evolución en sí es un proceso de optimización [MAYR-88] en el que se busca adaptar lo más posible a los individuos al entorno en el que habitan. Darwin [DARW-59] estudió sorprendido la extrema perfección que ciertos órganos habían alcanzado mediante la evolución, como por ejemplo los ojos. Sin embargo, optimización no tiene que significar perfección. La evolución descubre soluciones funcionales altamente precisas para problemas concretos que se dan en el medio ambiente de un determinado individuo. Debido a esto, ya desde un primer momento, se pensó en su utilización para resolver problemas de optimización en ingeniería. Las técnicas clásicas utilizadas en esta área, como el "descenso de gradiente", el "hill climbing" o las puramente aleatorias habían sido claramente inadecuadas al enfrentarse a problemas de optimización no lineales, especialmente con aquellos que contenían una componente estocástica, temporal y de caos. Pero, este es el tipo de problemas que se van a encuentran en la naturaleza, y ésta parece resolverlos de una manera eficaz. Estas ideas fueron claves para el desarrollo de las Estrategias Evolutivas [RECH-65][RECH-94] [SCHW-65] [SCHW-95].

- Sistemas Robustos. Los problemas del mundo real casi nunca son estáticos y los problemas de optimización temporal son cada vez más comunes. Estas circunstancias requieren un cambio en la estrategia que se aplica para resolver el problema. En estos casos, la realimentación sobre el éxito o el fracaso de la estrategia actual es fundamental. Holland [HOLL-75], utilizando Algoritmos Genéticos, describe un procedimiento que puede evolucionar estrategias, tanto como cadenas codificadas de números o como bases de reglas llamadas "Sistemas Clasificadores". El resultado es un procedimiento robusto que tiene el potencial de ajustar el rendimiento basándose en la realimentación de la salida del sistema.

- Inteligencia Artificial. Una de las múltiples definiciones de Inteligencia puede ser: la capacidad de un sistema para adaptar su comportamiento para alcanzar sus metas en distintos ambientes [FOGE-95]. Entonces, de un cierto comportamiento, para poder decir que es inteligente, se requiere que pueda predecir. Para que pueda adaptarse a circunstancias futuras tiene que predecir que se pueden dar estas circunstancias y tomar las acciones apropiadas. Como se puede comprobar, la evolución ha creado especies con una inteligencia cada vez más desarrollada. Por tanto, la forma de conseguir Inteligencia Artificial, además de las alternativas clásicas que persiguen la emulación de la inteligencia humana, ya sea mediante la emulación de las estructuras neuronales o de otros sistemas, tiene una nueva posibilidad, simular la evolución para construir algoritmos predictivos. Esta es la base de las investigaciones en Computación Evolutiva [FOGE-62][FOGE-66].

- Vida Artificial. En el campo de la biología (ciencia que se ocupa del estudio de la vida), más que utilizar la evolución como una herramienta, se intenta capturar la esencia de la propia evolución en una simulación por computador y usar entonces esta simulación para conseguir nueva información sobre la física de los procesos de la evolución natural [RAY-91]. El éxito en este campo abre la posibilidad de estudiar sistemas biológicos alternativos de cómo puede ser la vida en otros lugares y averiguar qué características puede tener en común con la evolucionada en la Tierra [LANG-87]. Aunque cada modelo es incompleto y las características que tiene la vida en otros lugares del universo es pura especulación, las simulaciones de vida por ordenador, denominadas genéricamente "Vida Artificial", ya han

conseguido generar algunos patrones que se corresponden con fenómenos que normalmente ocurren en la vida de la Tierra.

Una vez vistas las distintas ramas de aplicación de la Computación Evolutiva, la cuestión fundamental que se plantea ahora es por qué simular los procesos de la evolución. Una respuesta convincente es que en estos momentos no se dispone de mejores alternativas. Normalmente, no se pueden utilizar eficientemente las técnicas clásicas de optimización para encontrar un máximo global en una función si está rodeado de máximos locales. Las técnicas desarrolladas hasta ahora para emular la inteligencia humana, como los Sistemas Expertos, son bastante frágiles y no son suficientemente adaptables a los dominios cambiantes ya que no son suficientemente capaces de predecir ni de anticipar sus acciones. Tampoco se puede visitar un nuevo mundo, enviarle los compuestos químicos primigenios que dan lugar a la vida y esperar millones de años para ver qué tipo de vida se desarrolla allí. En contraste, los métodos de computación evolutiva ofrecen posibles soluciones a estos problemas basándose en la utilización dirigida del azar y la incertidumbre. Tan sólo hay que dejar que la computación evolutiva descubra soluciones a los problemas de formas, a priori, impredecibles.

2.2.1.2 Aplicabilidad

De acuerdo con el teorema "no-free-lunch" (en adelante, NFL) [WOLP-96], no puede existir ningún algoritmo que resuelva todos los problemas (por ejemplo de optimización) que sea en general (es decir, de media) superior a cualquier otro competidor. Por tanto, la cuestión de si los algoritmos evolutivos (en adelante, AE) son superiores o inferiores a cualquier otra aproximación no tiene sentido. Sin embargo, es posible afirmar que los AE se comportan mejor que otros métodos con respecto a problemas específicos y, como consecuencia, se comportarán peor con respecto a otras clases de problemas.

El teorema NFL puede corroborar que, en el caso de los AE frente a otros métodos clásicos de optimización, es más eficiente resolviendo problemas lineales, cuadráticos, fuertemente convexos, unimodales, separables y otras clases de problemas específicos. Por otro lado, los AE también se enfrentan con éxito a problemas discontinuos, no diferenciables, multimodales, con ruido y cualquier tipo de superficies

de búsqueda no convencionales. Por el contrario, su efectividad disminuye al afrontar problemas más simples para los que se han desarrollado algoritmos específicos en su resolución. Obviamente, la conclusión que se puede extraer es que, si ya existe un método tradicional, no se deberían usar los AE, ya que no lo van a hacer, ni mejor, ni con menos esfuerzo temporal o de potencia computacional.

En el caso de que no exista ningún algoritmo específico, encontrar este algoritmo puede ser, desde el punto de vista de la investigación, un reto, pero desde el punto de vista de la aplicación, el tiempo necesario para formalizar este algoritmo ha de sumarse al tiempo computacional en la resolución de problemas. Con respecto a esto, parecen claras las ventajas de la utilización de algoritmos no especializados y robustos como los AE.

2.2.1.3 Fundamentos

La más comúnmente aceptada base de teorías sobre la evolución es el paradigma "neo-Darwiniano". En esta teoría se afirma que la mayor parte de la historia de la vida puede ser completamente justificada mediante procesos físicos y mediante poblaciones y especies [HOFF-89]. Estos procesos son: la reproducción, la mutación, la competición y la selección. La reproducción es una propiedad común de las especies. Hay que tener en cuenta que, normalmente, la capacidad reproductiva de las especies es enorme y el número de individuos se incrementaría exponencialmente si todos sus individuos se pudieran reproducir con éxito simultáneamente [MALT-26][MAYR-82]. La reproducción supone la transferencia del código genético de cada individuo a su descendencia. La mutación sucede debido a que los errores en la replicación durante el proceso de transferencia de información genética son inevitables además de ser necesario incluir variabilidad en las especies. La competición es una consecuencia del crecimiento de la población dentro de un espacio físico finito donde no hay espacio o energía para todos. La selección es el resultado de la reproducción de las especies en un medio con competencia, gracias a ella sobrevivirán los mejores individuos; esto es, los más adaptados al medio. La evolución surge inevitablemente al interactuar estadísticamente todos estos procesos físicos [HUXL-63][WOOL-68][ATMA-79].

Los individuos y las especies pueden ser vistos como una dualidad de su código genético, el genotipo, y su forma de plasmarse con respecto al mundo, el fenotipo. El

genotipo ofrece la posibilidad de almacenar la experiencia acumulada, adquirida de la información histórica. Desafortunadamente, el resultado de la variación genética es impredecible debido a los efectos combinados de la "pleitropía" y la "poligénesis" [MAYR-63] [MAYR-88] [WRIG-31] [WRIG-60] [SIMP-49] [DOBZ-70] [STAN-75] [DAWK-86]. La "pleitropía" implica que un único gen afecta simultáneamente a varias características del fenotipo. Mientras que la "poligénesis" significa que cada característica del fenotipo viene determinada por la interacción de varios genes. No se conoce ninguna relación uno-a-uno entre genes y características en los sistemas evolutivos naturales. Debido a esto, el fenotipo cambia siguiendo una función compleja, no lineal, de la interacción entre las estructuras genéticas y las condiciones medioambientales. De esta forma, códigos genéticos muy distintos pueden tener comportamientos equivalentes igual que programas diferentes pueden conseguir resultados similares.

Dentro del proceso de evolución, hay que tener en cuenta que la selección actúa sólo sobre los comportamientos externos de los individuos y las especies [MAYR-88] y no sobre el genotipo que representan. La selección viene dada por la adaptación de los individuos al entorno y esta adaptación se puede ver como un espacio con una topografía adaptativa dependiente de las condiciones del entorno [WRIG-32]. Una población de genotipos genera sus respectivos fenotipos [LEWO-74] que, por sus características de adaptación al medio, están situados en un cierto lugar del mapa de adaptación. Cada pico de este mapa se corresponde con una zona de fenotipos optimizados. De una forma probabilística, la evolución acerca a los individuos a estos picos, mientras que la selección elimina las variantes fenotípicamente peores.

Visto de esta forma, la evolución es claramente un proceso de optimización en resolución de problemas. La selección conduce los fenotipos tan cerca como es posible del óptimo con respecto a unas condiciones iniciales y unas restricciones del entorno. Sin embargo, el entorno está continuamente cambiando y las diferentes especies constantemente evolucionan hacia nuevos valores óptimos. Debido a esto, no se puede decir que ningún organismo esté perfectamente adaptado a su entorno.

Para terminar este punto, se resumen las características más importantes del paradigma del neo-Darwinismo [MAYR-88]:

1. El individuo es el principal objetivo de la evolución.

2. La variación genética es un fenómeno de probabilidad. Los fenómenos estocásticos juegan un importante papel en la evolución.

3. La variabilidad en el genotipo se basa, en su mayor parte, en un proceso de recombinación y, sólo en menor medida, en la mutación.

4. La evolución gradual puede incluir discontinuidades en el fenotipo.

5. No todos los cambios en el fenotipo tienen que ser consecuencia de la selección natural.

6. La evolución es un cambio en la adaptación y la diversidad, no solamente un cambio de la frecuencia de los genes.

7. La selección es probabilística, no determinística.

Todas estas características, en mayor o menor medida, constituyen el marco de desarrollo de las técnicas de la Computación Evolutiva.

En el capítulo de Anexos se incluyen, como referencia, un punto sobre Genética que sirve de base y explicación a conceptos que se van a utilizar en este capítulo y otro punto sobre la evolución histórica de la CE.

2.2.1.4 Funcionamiento

Todas las técnicas de CE se basan en el modelo de la evolución natural, por lo que todas comparten un conjunto de características comunes, que son las siguientes:

- Los AE utilizan el proceso de "aprendizaje colaborativo" de un conjunto de individuos. Normalmente, cada individuo representa o codifica un punto del espacio de búsqueda de soluciones en un problema dado. Además, los individuos pueden incorporar información adicional acerca del problema.

- La descendencia de los individuos de la población se genera aleatoriamente mediante procesos de cruce y recombinación. La mutación se corresponde con una auto-replicación errónea de los individuos (normalmente cambios pequeños),

mientras que la recombinación intercambia información entre dos o más individuos ya existentes.

- Mediante la evaluación de los individuos en su entorno, se consigue una medida del ajuste que se puede asignar a cada individuo. De acuerdo con esta medida de ajuste, el proceso de selección favorece más a los individuos mejor adaptados, que serán los que se reproduzcan más frecuentemente.

Estas son las características generales que comparten todas las ramas de la CE, sin embargo, las diferencias en la implementación de estos principios es lo que caracteriza a los AG, las EE y la PG. Fogel [FOGE-95] y Bäck [BACK-96] proporcionan una referencia detallada de las similitudes y diferencias entre las distintas técnicas de las técnicas de CE que, resumida, se puede ver como:

- Los AA.GG. se basan en la recombinación (cruce) considerándola el operador de búsqueda más importante. Aplican el operador de mutación con una probabilidad muy pequeña y no se considera un operador fundamental. Se usa la selección probabilística y frecuentemente se codifican los individuos en cadenas binarias.

- Las EE usan normalmente la mutación para modificar vectores de números reales y utilizan la recombinación y el cruce como operadores principales de búsqueda. El operador de selección es determinístico y, normalmente, el tamaño de la población de los padres y el de la descendencia es distinto.

- La PG enfatiza el uso de la mutación y no incorpora la recombinación de individuos. Normalmente se trabaja con mutaciones con distribución normal y el proceso evolutivo se extiende a los parámetros de la estrategia evolutiva. El operador de selección es probabilístico y, actualmente, la mayoría de las aplicaciones están relacionadas con problemas que evolucionan soluciones codificadas como vectores de números reales, aunque originalmente el algoritmo fue desarrollado para evolucionar máquinas de estado finito.

Se expone a continuación un marco general de funcionamiento de las técnicas de CE. Con I se denota un espacio de búsqueda arbitrario de individuos $a \in I$, y con $F : I \rightarrow \Re$ se denota la función de ajuste de los individuos de la población. Con μ se referencia el tamaño de la población padre y con λ el tamaño de la población de la descendencia.

Se van a utilizar los operadores de selección, mutación y recombinación con sus correspondientes ratios θ_s, θ_m y θ_r. También es necesario un procedimiento de inicialización que produzca la generación inicial de individuos. Esta inicialización suele ser aleatoria pero también es habitual que se sugieran algunos puntos del espacio de búsqueda como soluciones iniciales. A continuación, una función de evaluación determina el nivel de ajuste de cada individuo y, por último, se especifica un criterio de terminación f para poder decidir cuándo acaba la simulación.

Con todos estos parámetros un algoritmo básico de CE se reduce a un bucle de selección-recombinación-mutación donde se tiene como entradas: μ, λ, θ_f, θ_s, θ_m y θ_r. En la salida puede estar a^*, siendo el mejor individuo de la población o P^*, siendo la mejor población de la simulación.

1 $t \leftarrow 0$;

2 $P(t) \leftarrow$ inicializar(μ);

3 $F(t) \leftarrow$ evaluar($P(t)$, μ);

4 While ($f(P(t), \theta_f) \neq$ true) do

5 $P'(t) \leftarrow$ recombinar ($P(t)$, θ_r);

6 $P''(t) \leftarrow$ mutar ($P'(t)$, θ_m);

7 $F(t) \leftarrow$ evaluar ($P''(t)$, λ);

8 $P(t+1) \leftarrow$ seleccionar ($P''(t)$, $F(t)$, μ, θ_s);

9 $t \leftarrow t+1$;

10 EndWhile

Después de la inicialización del tiempo t (línea 1) y de una población P(t) de tamaño μ (línea 2), se realiza la evaluación de la población (línea 3) y se entra en el bucle de simulación. El criterio de terminación f puede depender de una serie de parámetros que vienen representados por θ_f. De forma similar, la recombinación (línea 5), la mutación (línea 6) y la selección (línea 8) dependen de una serie de parámetros o

criterios. Mientras que de P(t) se supone que tiene un tamaño μ, P'(t) y P''(t) tienen tamaños k y λ respectivamente. Estos parámetros no tienen por que ser distintos, de hecho, en el caso de los AA.GG. $\mu=k=\lambda$. Este algoritmo es una base de funcionamiento con multitud de variantes para cada una de las tres técnicas básicas de CE, de forma que cada variante puede tener parámetros adicionales y evitar la utilización de alguno de los descritos en este punto.

2.2.2 AA.GG.

2.2.2.1 Introducción

Los Algoritmos Genéticos (AA.GG.) son una clase de algoritmos evolutivos que se usan para resolver problemas de búsqueda y optimización. Se distinguen del resto de las técnicas de CE por la utilización, principalmente, del operador de cruce como método evolutivo. Los AA.GG. son capaces de evolucionar soluciones de problemas del mundo real, siempre que se puedan codificar adecuadamente. Por ejemplo, los AA.GG. pueden usarse para maximizar ratios como puede ser el de resistencia/peso de la estructura de un puente, o para minimizar gastos como determinar la menor pérdida de tela al cortar patrones de ropa. También se pueden usar para procesos de control industrial, para balancear la carga en ordenadores multiprocesadores, etc.

Los AA.GG. simulan dentro de poblaciones los procesos que son esenciales para la evolución. En la naturaleza, los individuos de una población compiten entre ellos por toda clase de recursos como agua o comida y los miembros de una misma especie compiten para aparearse. De todos ellos, los que tienen más éxito en sobrevivir y aparearse conseguirán tener un mayor número de descendientes. Los individuos con peores aptitudes producirán muy poca o ninguna descendencia. Esto significa que los genes de los individuos de mayor adaptación al medio serán heredados por un número creciente de individuos con cada generación. La combinación de buenas características de diferentes antepasados puede generar una descendencia de alta adaptación al medio, de forma que estén mucho mejor adaptados que sus padres individualmente. Esta es la forma en que las especies evolucionan para estar cada vez más adaptadas al medio en que viven.

Los AA.GG. presentan una analogía directa a este comportamiento natural. Trabajan con una población de individuos, representando cada uno a una posible solución del problema que se trata. A cada individuo se le asigna una puntuación dependiendo de lo buena que sea la solución que aporta al problema. A los individuos con mayor adaptación se les da la oportunidad de "reproducirse", cruzándolos con otros individuos de la población. Así se produce la descendencia cuyos individuos comparten características de cada uno de los padres. Por otro lado, los individuos menos adaptados de la población tienen menos probabilidades de ser escogidos para reproducirse y más de ser sustituidos por las nuevas generaciones. Se produce entonces una nueva generación de soluciones a partir de los mejores individuos de la generación actual. Esta nueva generación contiene una proporción mayor de individuos con las mejores características de los individuos de las generaciones anteriores. De esta forma, con el paso de las generaciones, las mejores características se generalizan en la población y se mezclan dentro de los nuevos individuos. Al favorecer el cruce de los individuos mejor adaptados, se exploran las áreas más prometedoras del espacio de soluciones del problema. Si el AG ha sido bien diseñado, la población convergerá a la solución óptima del problema.

La potencia de los AA.GG. viene de que la técnica que usan es robusta y puede tratar con éxito un gran número de tipos de problemas, incluyendo y destacando aquellos que son difícilmente solucionables utilizando otros métodos. Los AA.GG. no garantizan la localización de la solución óptima al problema, pero generalmente funcionan muy bien buscando una solución aceptablemente buena en un tiempo razonablemente corto; son, por tanto, procedimientos heurísticos. Si existe una técnica especializada para resolver un tipo particular de problema, es probable que esta sobrepase a un AG tanto en velocidad como en exactitud. El campo principal de aplicación de los AA.GG. es, entonces, la resolución de problemas complejos donde todavía no se ha desarrollado una técnica específica. Incluso si existe una técnica que funciona bien, probablemente se pueda mejorar en algún punto combinándola con un AG.

En el capítulo de Anexos, se incluye un punto sobre la Historia de los AA.GG. que aporta una perspectiva de la evolución de esta técnica.

2.2.2.2 Comparación con otras Técnicas de Búsqueda

Existe un conjunto de técnicas de propósito general que se relacionan con problemas de búsqueda y optimización. Como los AA.GG., asumen que el problema está definido mediante una función de evaluación que puede ser maximizada.

Hay un gran número de técnicas, algunas de las cuales son sólo aplicables a dominios limitados, por ejemplo, la "programación dinámica" [BELL-57]. Este es un método para resolver problemas de control multi-paso que es sólo aplicable cuando el ajuste global es la suma de las funciones de ajuste de cada paso del problema, y no hay interacción entre las fases. Algunas de las técnicas más generales son la búsqueda aleatoria, los métodos de gradiente, la búsqueda iterativa y el enfriamiento simulado, que se describen en el apartado de Anexos.

2.2.2.3 Funcionamiento Genérico

El funcionamiento del AG comienza con una población inicial de individuos generados normalmente de forma aleatoria, denominada P(0). Cada individuo de esta población se evalúa para conocer su adaptación al problema. Después, se seleccionan algunos individuos de la población para realizar los cruces. En el AG original de Holland [HOLL-75] los individuos se escogían probabilísticamente, asignando a cada individuo una probabilidad proporcional a su rendimiento. De esta forma, se les da a los mejores individuos más probabilidades de producir descendencia. Después de la selección, se aplican los operadores genéticos (normalmente cruce y mutación) sobre los individuos seleccionados produciendo la descendencia de esa generación. Los ratios de cruce y mutación son una decisión de configuración del AG. Otros detalles de la implementación son escoger cuántos descendientes se generan de cada cruce (uno o dos) y cuántos individuos se seleccionan y emparejan para el cruce. En la implementación original de Holland sólo un par de individuos se seleccionaban para el cruce en cada ciclo del algoritmo. El pseudo-código de un AG básico es el que se muestra en la Figura 2.3.

Después de que se ha creado la descendencia a través de los operadores genéticos, se mezclan en una única población padres e hijos. La mayoría de las implementaciones de AA.GG. mantienen una población de tamaño fijo, de forma que

un número M de individuos tienen que seleccionarse desde la población de padres y de hijos para crear la nueva población. Una posibilidad es utilizar todos los hijos generados (si el número no es mayor que M) y seleccionar aleatoriamente individuos de la generación anterior para conseguir una nueva población de tamaño M. En el caso de que se produzca muy poca descendencia se pueden incluir en la población antigua sustituyendo a algunos de sus individuos. Esta estrategia es la seguida por Holland en sus primeros trabajos. Por otro lado, si el número de individuos de la descendencia es igual a M, estos individuos reemplazarán por completo a la generación precedente.

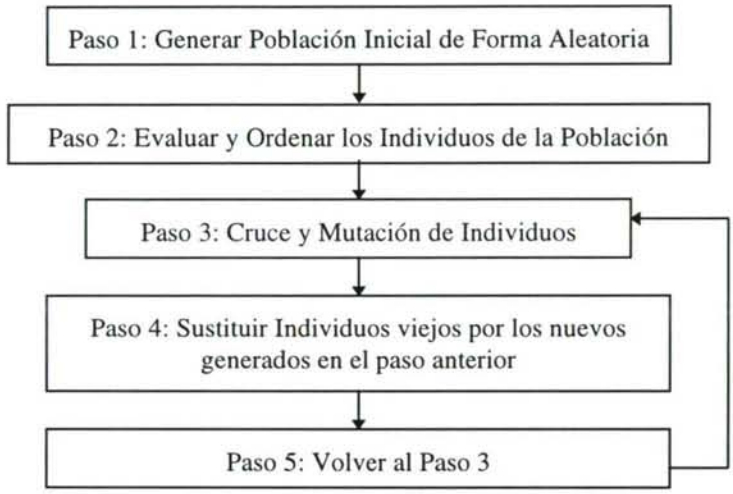


Figura 2.3. Pseudo-código de un Algoritmo Genético Simple

Para aplicar un AG a la resolución de un problema concreto se necesita la configuración de distintas fases. En primer lugar, se debe diseñar una codificación o representación del problema. Después, hay que pensar cómo evaluar los individuos con respecto al problema que se quiere resolver, cómo seleccionar individuos para la reproducción, cómo aplicar los operadores genéticos para generar la descendencia y cómo realizar la sustitución de individuos de generaciones anteriores. Las distintas opciones disponibles para realizar estas tareas se muestran en los siguientes puntos.

2.2.2.4 Codificación del problema

La solución a cualquier problema que quiera ser tratado utilizando AA.GG. se tiene que poder representar como un conjunto de parámetros. Estos parámetros, conocidos como genes, se unen para formar una cadena de valores denominada "cromosoma". Según las investigaciones de Holland [HOLL-75] lo ideal es usar una codificación binaria para esta cadena, siendo la opción más utilizada, aunque existen

también otras posibilidades. Así, se pueden utilizar parámetros con números reales [DAVI-91][JANI-91][WRIT-91], permutaciones [DAVI-85a][GOLD-85][GREF-85b] e, incluso, árboles jerárquicos [ANTO-87]. La Programación Genética de Koza también está basada en los AA.GG., pero se utiliza para evolucionar programas. Aquí, los individuos son estructuras en LISP y el operador de cruce genera nuevas estructuras LISP al intercambiar subárboles de la estructura de los padres.

Una vez se escoja la codificación binaria, es necesario escoger el tipo de codificación binaria que se va a utilizar para los parámetros numéricos. Algunos trabajos experimentales indican que la codificación Gray supera a la codificación en sistema binario estándar [CARU-88], al menos para los problemas de test usados comúnmente. La razón de esta superioridad, es que en la codificación Gray, dos números adyacentes sólo pueden cambiar su representación en un sólo bit. Es decir, un pequeño cambio en el valor del número codificado repercute en un pequeño cambio en la codificación. Por ejemplo, codificando un parámetro de 5 bits con un rango de 0-31, en sistema binario la codificación del 15 sería 01111, mientras que la del 16 sería 10000, cambiando el valor de los 5 bits. Si se utiliza la codificación Gray, el 15 se codificaría como 01000 y el 16 como 11000, cambiando únicamente el valor de un bit. Al escoger una codificación en concreto, es crítico que el operador de cruce que se emplee sea adecuado a la codificación de los parámetros.

Dada la importancia de la representación o codificación de los individuos, un cierto número de investigadores han sugerido distintos métodos para permitir a un AG adaptar su codificación. Desde sus primeros trabajos Holland propuso el operador de inversión para reacomodar la localización de los genes en las cadenas de los individuos. Otra aproximación, es el denominado sistema ARGOT de Shaefer [SHAE-87]. En el sistema ARGOT existe una traducción adaptativa de los parámetros de números reales a cadenas binarias. La resolución con que la cadena binaria representa al número real se puede incrementar o decrementar dependiendo de un cierto número de heurísticas sobre el problema. Una idea similar la emplea Schraudolph [SCHR-92] utilizando una heurística para incrementar la resolución cuando la población empieza a converger. Mathias [MATH-94] propone una codificación denominada "delta coding" en la que, cuando la población empieza a converger, la representación de los parámetros se modifica de forma que su rango esté centrado en el mejor valor conseguido por el AG

hasta el momento, y entonces se reinicializa el algoritmo. También se han propuesto diversas heurísticas para la ampliación y estrechamiento del rango de los parámetros.

Como ya se ha comentado, en términos de genética, el conjunto de parámetros representados por un cromosoma particular se denomina genotipo. El genotipo contiene la información necesaria para construir el organismo, que se denomina fenotipo. La misma nomenclatura se utiliza en AA.GG. Por ejemplo, en el diseño de un puente, el genotipo es el conjunto de parámetros que especifican un diseño en particular, mientras que la construcción terminada es el fenotipo. El nivel de adaptación de cada individuo depende del rendimiento de su fenotipo. En un AG este valor se infiere del genotipo, calculándolo desde el cromosoma por medio de la función de evaluación.

2.2.2.4.1 Codificaciones no Binarias

Como se ha visto en el punto anterior, el tipo de codificación más comúnmente utilizada es la binaria; sin embargo, también han sido usados alfabetos de mayor cardinalidad en algunas investigaciones mostrando ciertas ventajas. Goldberg [GOLD-89a][GOLD-89b] defiende que la codificación binaria es la que permite generar el número más grande de plantillas y que, por tanto, tiene el más alto grado de paralelismo implícito. Sin embargo, Antonisse [ANTO-89] interpreta el concepto de plantillas de forma diferente y concluye que, por el contrario, los alfabetos de cardinalidad alta contienen más plantillas que los alfabetos binarios. Este tema siguió discutiéndose por otros autores llegando a conclusiones similares [ANGE-92][ANTO-92]. En esa misma época, Goldberg desarrolló su teoría de los alfabetos virtuales [GOLD-90b] para explicar por qué los alfabetos de cardinalidad alta pueden comportarse de forma correcta. Según Goldberg, en las primeras generaciones del AG cada símbolo converge, dejando un conjunto muy pequeño de valores posibles para cada símbolo. De esta forma, con el paso de las generaciones, cada símbolo acaba trabajando con un alfabeto de cardinalidad baja.

Los estudios empíricos con alfabetos de alta cardinalidad utilizan cromosomas cuyos símbolos están representados por enteros [BRAM-91] o por números reales [JANI-91][MICH-91]. Como apunta Davis [DAVI-91], los parámetros en los problemas son frecuentemente numéricos así que, representarlos directamente con números, en vez de con códigos binarios, tiene claras ventajas. Una de ellas es la fácil definición del

significado de operadores específicos de cruce y mutación. Así, por ejemplo, en el caso de números reales, se podría definir el cruce como la media aritmética de los valores de los genes de los padres en cada posición o la media geométrica tomando para cada gen la raíz cuadrada del producto de los genes de los padres o la extensión, siendo el valor del hijo la diferencia de los valores de los genes de los padres más el menor de ellos. Para el caso de la mutación, se podría definir como un reemplazo aleatorio generando un nuevo valor dentro del rango válido, añadiendo una pequeña cantidad aleatoria al valor del gen o multiplicando el valor del gen por una cantidad próxima a uno. Para estas dos últimas variaciones de la mutación, el número generado puede tener distintas distribuciones del tipo uniforme, exponencial, Gaussiano, etc.

Janikow y Michalewicz [JANI-91] realizaron una comparación entre las codificaciones binaria y de números reales y descubrieron que la utilización de números reales era más rápida, consistente y ofrecía unos resultados más exactos. Sin embargo, si los parámetros del problema no son numéricos las ventajas de un alfabeto de cardinalidad alta pueden ser difíciles de explotar.

En el número 32 del volumen 6 del GA-Digest (Septiembre de 1992), su editor, Alan C. Schultz, recopila varios trabajos de investigación que utilizan codificaciones no-binarias. Uno de ellos es el trabajo de Grefenstette [SCHU-90][GREF-91] que utiliza una representación que codifica reglas en un sistema que aprende estrategias reactivas para conformar el comportamiento de agentes autónomos. Koza utiliza un proceso conocido como Programación Genética para desarrollar programas en LISP [KOZA-92]. Las codificaciones con números reales han sido ampliamente utilizadas [WHIT-89][JANI-91][MICH-91][ESHE-93] e, incluso, Michalewicz ha trabajado con matrices como estructuras de datos [MICH-92].

2.2.2.5 Función de Evaluación

La función de evaluación tiene que ser diseñada para cada problema que se quiere resolver. Dado un cromosoma concreto, la función de evaluación devuelve un valor de ajuste, que es proporcional a la habilidad del individuo al que representa el cromosoma. Para la mayoría de problemas, en especial optimización de funciones (Figura 2.4), es obvio qué es lo que tiene que medir la función de evaluación; basta con observar el valor de la función. Sin embargo, esto no siempre es así. Si se plantea un

problema con optimización combinatoria, como por ejemplo el diseño de un puente, existen gran cantidad de variables que se pueden querer optimizar: resistencia/peso, anchura, máxima carga, coste, tiempo de construcción, etc., siendo lo más probable que lo que se quiera optimizar sea una combinación de todas ellas.

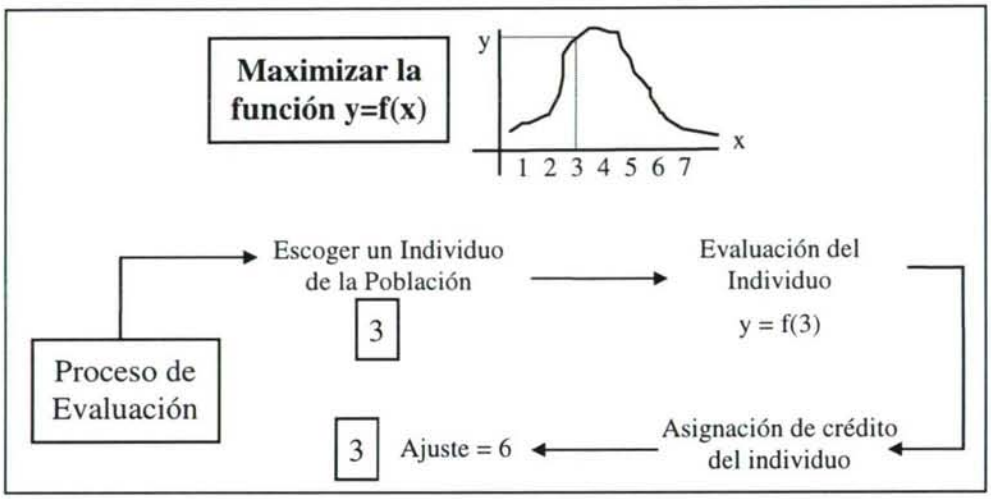


Figura 2.4.- Proceso de Evaluación de un Individuo Nuevo

2.2.2.6 Selección

La selección de los progenitores es la tarea de repartir las oportunidades de reproducción a cada individuo. En principio, los individuos de la población que se van a cruzar se copian a un espacio separado denominado "espacio de cruce" (*mating pool*), usualmente de igual tamaño que la población. Los individuos con un alto nivel de ajuste tienen más probabilidades de ser copiados varias veces y los de menos ajuste es probable que no se copien ninguna. Después de hacer esto, se escogen del "espacio de cruce" parejas de individuos al azar y se cruzan. Esto se repite hasta que el "espacio de cruce" se agota.

Como ya se ha indicado, el comportamiento de los AA.GG. depende mucho de cómo se escojan los individuos del "espacio de cruce". Las formas de hacer esta selección se pueden dividir en dos tipos. En primer lugar, se puede tomar el ajuste de cada individuo como un estimador del número de copias que de él van a ir al "espacio de cruce". Otro método, que alcanza un resultado similar, es no utilizar un estimador del valor de ajuste como criterio principal. Estos métodos se denominan, respectivamente de "ponderación explícita de ajuste" (*explicit fitness remapping*) y de "ponderación

implícita de ajuste" (*implicit fitness remapping*), y se repasan en profundidad en el apartado de Anexos.

2.2.2.7 Reproducción

La fase de reproducción del AG es posterior a la fase de selección de progenitores y, en ella, se recombinan los individuos padres para producir la descendencia que compondrá la nueva generación. Una vez seleccionados los padres sus cromosomas se recombinan mediante los mecanismos de cruce y mutación.

El cruce toma genéricamente dos individuos y corta sus cadenas de cromosomas en una posición escogida al azar, produciendo cuatro partes, dos partes inicio y dos partes finales. Estas últimas se intercambian para producir dos nuevos individuos completos (Figura 2.5). Cada uno de los dos descendientes hereda algunos genes de cada padre. Este tipo de cruce se conoce como el de único punto de cruce. Hay que tener también en cuenta que existen muchas variantes del operador de cruce diseñadas para problemas específicos (las más comunes se pueden consultar en el apartado de Anexos). Además, en un punto posterior sobre los fundamentos matemáticos de los AA.GG., se desarrolla una comparación entre los distintos operadores de cruce.

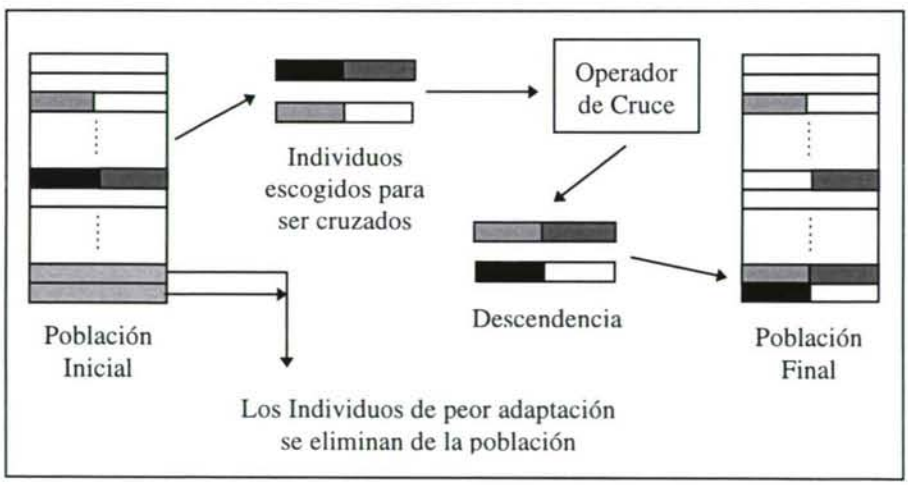


Figura 2.5.- Operador de Cruce

El cruce no se aplica normalmente a todos los individuos escogidos para aparearse. Se realiza una selección al azar en la que se aplica una probabilidad que normalmente está entre 0.6 y 1.0. Si no se aplica el cruce, la descendencia simplemente duplica a los padres. Esto da la posibilidad de que un individuo pase sus genes sin ninguna ruptura en ellos.

Después de la operación de cruce, se aplica a algunos de los descendientes la operación de mutación. Aleatoriamente, se altera algún gen con una probabilidad pequeña (normalmente 0.001). En la Figura 2.6, se muta el quinto gen del cromosoma. La idea más ampliamente aceptada es que el cruce es la más importante de estas dos técnicas para realizar una rápida exploración del espacio de soluciones. La labor de la mutación es proporcionar una pequeña cantidad de búsqueda aleatoria y asegurar que ningún punto del espacio de soluciones tenga probabilidad cero de ser examinado.

Existen muchas variantes del operador de mutación, entre las que se puede destacar la mutación exponencial, utilizada cuando decrece la diversidad de la población [FOGA-89] o la mutación adaptativa, que aplica o no el operador de mutación dependiendo de cómo afecta al rendimiento del individuo.

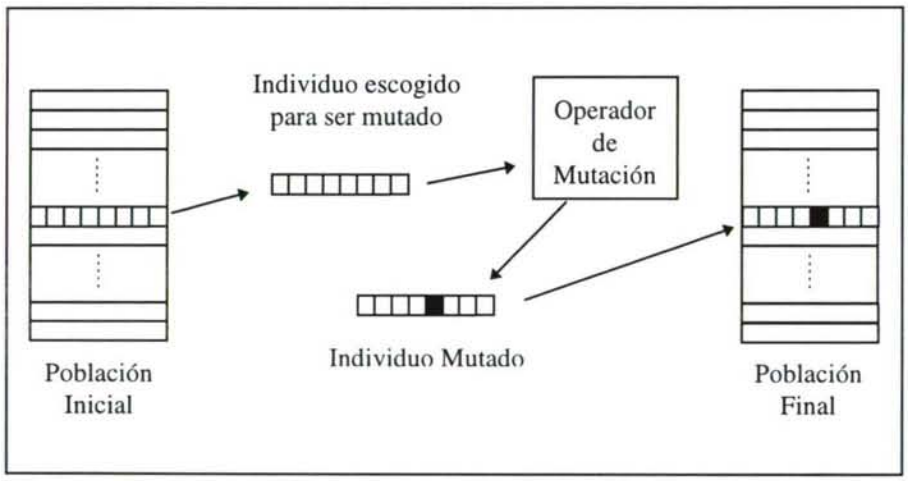


Figura 2.6.- Operador de Mutación

2.2.2.8 Reemplazo de individuos

Una vez realizado el cruce de los individuos que han sido seleccionados de la población, la siguiente etapa es la de generación de la nueva población. Después de generar la descendencia es necesario integrarla en la población para que contribuya a la mejora del nivel de ajuste global. Para realizar esta tarea existen dos aproximaciones radicalmente opuestas. La más antigua consiste en sustituir toda la población en cada generación con sus descendientes "generational replacement". Esta aproximación ha sido estudiada en profundidad por Grefenstette [GREF-86], sin embargo, en investigaciones más recientes [WHIT-89][SYSW-89][DAVI-89], se favorece el reemplazo de individuos, en lugar del reemplazo de la población. Así, en cada

generación solamente se sustituyen unos pocos individuos "steady-state replacement", llegando en algunas implementaciones a sustituir sólo dos.

Esta orientación parece un mejor reflejo de lo que ocurre en la naturaleza. Aunque en las especies de vida corta, como los insectos, los padres ponen y fertilizan huevos para después morir, en las especies de vida más larga como los mamíferos, los padres y la descendencia conviven de forma que los hijos pueden ser criados e instruidos a la vez que también tienen que competir por los recursos del medio.

Si los individuos de la descendencia van a sustituir a unos cuantos progenitores dentro de la población, además de un proceso de selección para escoger los individuos que se van a cruzar, es necesario escoger una estrategia para seleccionar los individuos de la población que tienen que dejar su sitio "morir" para que sea ocupado por la nueva descendencia. La idea más extendida es realizar la sustitución de aquellos individuos que muestren un nivel de ajuste más bajo; es decir, siguiendo las teorías de Charles Darwin, los menos adaptados tienden a desaparecer. En el sistema GENITOR de Whitley [WHIT-89] se sigue exactamente esta filosofía de reemplazo. Sin embargo, en opinión de Goldberg y Deb [GOLD-91b], las ventajas del reemplazo de individuos están relacionadas con una mejora sustancial en el principio de la simulación que también puede ser conseguida utilizando como sistema de selección de padres el "ranking de ajuste exponencial" o el "torneo". A pesar de todos los citados estudios no ha quedado del todo claro que una de las técnicas sea muy superior a la otra.

2.2.2.9 Convergencia

Si el AG ha sido implementado correctamente, la población evolucionará a través de las generaciones de tal forma que el ajuste del mejor y la media del ajuste de todos los individuos en cada generación se incrementa hacia el máximo global. Este proceso de incremento uniforme del ajuste es lo que se denomina convergencia. Se dice que un gen converge cuando el 95% de la población comparte un mismo valor [DEJO-75], y se dice que la población converge cuando todos los genes ya han convergido.

Hay que tener en cuenta que el rendimiento de un AG y, por tanto, su velocidad de convergencia, depende de los parámetros, tamaño de la población, frecuencia de

aplicación de los operadores cruce y selección, además de los mecanismos utilizados para la selección y el cruce.

Al comenzar la simulación, los valores para cada gen de los diferentes miembros de la población están distribuidos aleatoriamente. En consecuencia, existe una gran variedad de valores de ajuste de los individuos. Según avanza la simulación, empiezan a predominar valores concretos para cada gen. Cuando la población converge, el rango de valores de ajuste se reduce. En esta variación del rango de ajustes a lo largo de la simulación, pueden surgir los problemas de convergencia prematura y de lentitud del ajuste final en el caso de no configurar adecuadamente los parámetros del AG.

2.2.2.9.1 Convergencia Prematura

Un problema clásico del funcionamiento de los AA.GG. es que un grupo de individuos con un ajuste relativamente alto, pero no óptimo, pueden dominar rápidamente una población entera, causando la convergencia a un máximo local. Una vez que la población ha convergido, se elimina la habilidad del AG para continuar la búsqueda de mejores soluciones, ya que el cruce de individuos casi idénticos no puede producir nada nuevo. Sólo la mutación es válida en este entorno para explorar áreas completamente nuevas, y esto simplemente realiza una lenta búsqueda aleatoria.

Por la propia naturaleza de los AA.GG. se debería dar la oportunidad de reproducirse a los individuos en relación a su ajuste relativo. Pero, cuando se hace así, aparece el problema de la convergencia prematura, debido a que la población no es infinita. Para que los AA.GG. trabajen adecuadamente con poblaciones finitas hay que modificar la forma de seleccionar los individuos que se van a reproducir. La idea básica es controlar el número de oportunidades que cada individuo tiene para reproducirse de tal forma que no sean ni demasiadas ni demasiado pocas. Así, se previene que cualquier individuo con un ajuste mayor que la media tome el control de la población.

2.2.2.9.2 Lentitud Final del Ajuste

Este es el problema complementario al de la convergencia prematura. Después de un cierto número de generaciones, cuando la población converja, probablemente no se encontrará el máximo con precisión. Aunque la media del ajuste puede ser muy alta y haya poca diferencia entre el mejor y la media de todos los individuos, es probable que

no se localice la solución óptima. En consecuencia, el gradiente de la función de evaluación no es suficiente para conducir al AG hacia el máximo.

La misma técnica que combate la convergencia prematura también combate este problema. Esto se consigue expandiendo el rango de valores de ajuste de la población.

2.2.3 Base Teórica del Funcionamiento de los AA.GG.

La mayor parte de la investigación realizada en AA.GG. se ha concentrado en encontrar reglas empíricas para mejorar su rendimiento. Sin embargo, también se han enunciado varias teorías para explicar, aunque sea parcialmente, por qué funcionan con éxito los AA.GG. En los puntos de los Anexos se expone la teoría sobre las plantillas, la teoría del esquema y la hipótesis de los bloques constructivos. Tener en cuenta estas explicaciones es un factor importante a la hora de implementar mejores AA.GG.

2.2.3.1 Exploración y Explotación

Cualquier algoritmo eficiente de optimización tiene que usar dos técnicas para encontrar un máximo global: "exploración", para investigar nuevas y desconocidas áreas del espacio de soluciones; y "explotación", para hacer uso del conocimiento encontrado en puntos visitados previamente para ayudar a encontrar mejores puntos. Estos dos requerimientos son, en cierta medida, contradictorios y un buen algoritmo de búsqueda debe encontrar un equilibrio entre ellos.

Una búsqueda meramente aleatoria es una buena exploración, pero no realiza una explotación, mientras que un método como el de "hill-climbing" es una buena explotación pero tiene muy poco de exploración. Las combinaciones de estas dos estrategias pueden ser bastante efectivas, pero es difícil conocer dónde está el punto de equilibrio entre ambas. Es decir, ¿cuánta explotación deberíamos realizar antes de volver a realizar exploración?.

Holland [HOLL-75] demostró que un AG combina ambas, exploración y explotación, a la vez y de una forma óptima. Sin embargo, aunque esto puede ser teóricamente correcto, existen inevitablemente problemas al ponerlo en práctica. Esto sucede porque Holland estableció ciertas simplificaciones en su modelo, sin tener en cuenta que, para que el AG funcionara de acuerdo a la teoría, tendría que darse que:

1. la población debería ser infinita,

2. la función de ajuste debería reflejar exactamente la utilidad de una solución, y

3. los genes dentro de un cromosoma no deberían interactuar significativamente.

La aserción 1 no se puede satisfacer en la práctica. Debido a esto, el rendimiento de un AG siempre será objeto de errores estocásticos. Un problema como éste, se puede encontrar en la naturaleza y se denomina "*genetic drift*" [BOOK-87][GOLD-87a].

Incluso en ausencia de una presión de selección, los miembros de la población convergerán a algún punto del espacio de soluciones. Esto ocurre simplemente por la acumulación de errores estocásticos. Si, por ejemplo, un gen llega a ser mayoritario en la población, es muy probable que también lo sea en la generación siguiente. Si un incremento en la presencia de un gen se mantiene en sucesivas generaciones, y la población es finita, entonces ese gen se puede extender a todos los miembros de la población. Una vez un gen ha convergido de esta manera, se fija, ya que el cruce no puede introducir nuevos valores de gen.

El ratio de "*genetic drift*" proporciona entonces un límite inferior en la posibilidad de que un AG pueda converger hacia la solución correcta. Esto es, si el AG explota la información del gradiente de la función de ajuste, la función de ajuste tiene que proveer un alcance suficientemente grande como para compensar cualquier "*genetic drift*". El ratio de "*genetic drift*" puede reducirse al incrementar el ratio de mutación. Sin embargo, si el ratio de mutación es demasiado alto, la búsqueda llega a ser de tipo aleatorio.

Las aserciones 2 y 3 pueden satisfacerse utilizando funciones que hayan sido probadas exhaustivamente, sin embargo son difíciles de satisfacer en problemas del mundo real.

2.2.3.2 Comparación entre los Operadores de Cruce

Aún no está claro cuál de las dos técnicas de cruce presentadas ofrece mejores resultados. Syswerda [SYSW-89] apoya el operador de cruce uniforme. Afirma que, con el cruce uniforme, las plantillas de un cierto orden tienen la misma probabilidad de ser divididas, independientemente de la longitud de su esquema. Con un operador de cruce

con dos puntos de corte es la longitud de la plantilla la que da esta probabilidad y no su orden. Esto significa que con el cruce uniforme, es más probable romper las plantillas de longitud corta mientras que es menos probable que se rompan las de longitud larga. Syswerda concluye entonces que la probabilidad total de ruptura es, por tanto, más pequeña con el cruce uniforme. La ventaja del cruce uniforme es que el ordenamiento de los genes es irrelevante. Esto significa que las operaciones de reordenamiento, como la inversión, son innecesarias y no hay que preocuparse de colocar los genes de forma que puedan desarrollar buenos bloques constructivos. Debido a esto, el rendimiento de un AG que utilice cruce de dos puntos de corte cae dramáticamente si no se tiene en cuenta la formación de los bloques constructivos [BEAS-93a]. En el caso del cruce uniforme, sin tener en cuenta el orden de los genes, funciona casi igual de bien que en el caso del cruce de dos puntos teniéndolo en cuenta. Por tanto, el cruce uniforme se presenta como una técnica de cruce mucho más robusta.

Eshelman y colaboradores [ESHE-89] realizaron una comparación en profundidad de los distintos tipos de operadores de cruce, incluyendo cruce con punto único, con dos puntos, con multipunto y con cruce uniforme. Estos operadores fueron analizados desde el punto de vista teórico con respecto a órdenes y distancias, y desde el punto de vista práctico aplicándolos en distintos tipos de problemas. De estas pruebas, no se consiguió obtener ningún ganador claro, obteniendo unas diferencias de rendimiento de un 20% como máximo. Debido a esto, las recomendaciones de este artículo son que no se debe tener una excesiva preocupación en el método de cruce utilizado.

Spears y DeJong [SPEA-91] son muy críticos con el operador de cruce uniforme. Sus análisis teóricos dieron como resultado que, en general, el cruce con uno y dos puntos de cruce son los operadores óptimos. Sin embargo, comentan que el cruce con dos puntos de corte funciona peor cuando la población ha convergido casi completamente. En este caso, los cromosomas de los padres van a ser muy similares. Con el cruce con dos puntos de corte las partes de los cromosomas escogidas para cruzar serán probablemente idénticas, generando descendencia que va a ser igual a sus padres. Esta situación es menos probable que se produzca utilizando el cruce uniforme. Para solucionar este caso y seguir usando el cruce con dos puntos de corte, proponen repetir el proceso de generar los puntos de corte en el caso de que la descendencia sea igual que los padres. Una vez probado este operador modificado, demostró ser

ligeramente mejor que el operador de cruce uniforme trabajando en el mismo problema con las mismas circunstancias de convergencia larga. En otro artículo de DeJong y Spears [DEJO-90] concluyen que el cruce con dos puntos de corte es mejor en AA.GG. con poblaciones grandes, pero que la mayor ruptura de bloques constructivos del cruce uniforme es más beneficiosa en AA.GG. con poblaciones pequeñas, consiguiendo además un rendimiento más robusto.

2.2.3.3 Evolución mediante Mutación

La mutación, tradicionalmente, ha sido vista como un operador de fondo o de segundo plano en el proceso de la evolución [BOOK-87][DEJO-85]. La tarea realizada por este operador es la re-introducción de valores perdidos de genes y, por tanto, la prevención de la homogenización y la generación de una cierta aleatoriedad en una población después de un tiempo largo de convergencia. Es una idea ampliamente extendida que el cruce es el operador principal para la localización de soluciones en los espacios de búsqueda.

Sin embargo, los ejemplos que se encuentran en la naturaleza indican que la reproducción asexual es capaz de evolucionar individuos sin utilizar el operador de cruce [MAYN-89]. Los expertos en genética ven la mutación como la principal herramienta para generar material genético para el cambio en la evolución [HART-88]. Schaffer [SCHA-89] realizó un amplio experimento para determinar los parámetros óptimos para el funcionamiento de un AG. En este estudio concluyó que el cruce tenía mucha menos influencia en el rendimiento de lo que se creía hasta ese momento. A partir de este estudio, sugirió que la evolución mediante selección y mutación realiza una búsqueda del estilo del "escalado de colinas" que puede ser suficientemente potente aun sin la utilización del operador de cruce. En estudios posteriores [SCHA-91] se afirma que el operador de cruce consigue una mucho más rápida evolución que la utilización del operador de mutación aunque, al final, el empleo del operador de mutación sin cruce encuentra, normalmente, mejores soluciones que el operador de cruce sin mutación. Esto está en concordancia con Davis [DAVI-91] que afirma que, en el momento en que la población converge, la mutación puede llegar a hacerse más productiva que el cruce.

En consecuencia, a pesar de poder especificar una baja probabilidad de utilización de la mutación, este operador tiene una gran importancia y, además, la elección del ratio de mutación es mucho más crítico que la elección del ratio de cruce [SCHA-89].

Spears [SPEA-93] compara en profundidad los operadores de cruce y mutación, y concluye que existen características importantes de cada operador que no están contempladas en el otro. También plantea que ciertas modificaciones del operador de mutación podrían conseguir que la mutación realizara, además, las funciones del cruce. En el citado artículo se concluye que: *"los operadores de mutación y cruce estándar son dos formas simplificadas de un operador más general de exploración"*.

Otros resultados sobre el buen comportamiento de la evolución sin cruce los presentan Ericson [ERIC-91] y Eshelman [ESHE-91a][ESHE-91b]. Según Eshelman [ESHE-91b]: *"La clave del éxito de la evolución sin cruce (suponiendo una codificación binaria) es la utilización de una codificación Gray, haciendo la búsqueda dependiente de la distancia Hamming. Según mi opinión, la evolución sin cruce es una herramienta mucho más potente de lo que la comunidad de investigadores en AA.GG. está dispuesta a admitir"*.

2.2.3.4 Operadores Dinámicos

A lo largo de una simulación de un AG, el valor óptimo para los parámetros de cada operador puede cambiar. Davis [DAVI-85b] utilizó variaciones lineales en la probabilidad de cruce y de mutación, con el cruce decreciendo y la mutación incrementándose durante la simulación. Syswerda [SYSW-91] también encontró ventajas en este planteamiento. Booker [BOOK-87] utilizó una probabilidad de cruce variable dependiendo de los niveles de ajuste de los individuos. Cuando la población converje, el ratio de cruce se reduce para dar más oportunidades a la mutación de encontrar nuevas variantes. Esto es similar a la técnica lineal de Davis pero, al ser adaptativo, es mucho más flexible.

Davis [DAVI-89][DAVI-91] describe otra técnica adaptativa basada directamente en el éxito de un operador para producir descendencia con un buen ajuste. A cada operador se le asigna un nivel de crédito cuando produce un cromosoma que es

mejor que cualquier otro de la población. Normalmente, se pondera este crédito teniendo en cuenta el rendimiento del operador un cierto número de generaciones en el pasado. A partir de aquí, para cada situación reproductiva se selecciona probabilísticamente el operador que va a ser utilizado de acuerdo con el crédito que llevan acumulado. Con esto se consigue que, a lo largo de la simulación, la aplicación de los operadores se auto-regule de acuerdo con la propia evolución de la búsqueda de la solución. Una ventaja muy interesante de esta aproximación es que permite comparar el funcionamiento de nuevos operadores con respecto a los operadores estándar de forma que el propio problema evalúe su funcionamiento. Esta aproximación parece resolver una gran cantidad de problemas a la hora de escoger tanto los operadores como los valores óptimos de aplicación de cada uno de ellos. Sin embargo, una desventaja que se puede presentar en la utilización de esta técnica es que puede beneficiar a operadores que localicen máximos locales en vez de localizar el máximo global.

Varios investigadores [ACKL-87][BRAM-91][FOGA-89][MICH-91] han trabajado en variar el ratio del operador de mutación durante las simulaciones. Desafortunadamente, este autor no conoce que se hayan realizado análisis o razonamientos de por qué se debería utilizar esta variación, aunque Fogarty [FOGA-89] aporta una evidencia empírica de su buen funcionamiento. El razonamiento que apoya estas teorías es que la probabilidad de mutación es un parámetro análogo a la temperatura en el enfriamiento simulado y, por tanto, debería reducirse a lo largo de la simulación para ayudar al proceso de convergencia.

Por último, existen distintos argumentos sobre si la probabilidad de mutación debería incrementarse o decrementarse, si se debería variar de forma lineal o exponencial que aún no han sido demostrados matemáticamente y que son materia de discusión en la actualidad.

2.2.3.5 Mejoras Mediante la Utilización del Conocimiento sobre los Problemas

Mientras la mayoría de los investigadores trabajan con los operadores de cruce estándar o realizando pequeñas variaciones sobre ellos, en otros trabajos se presentan nuevos operadores diseñados para tareas específicas que utilizan como base el conocimiento sobre el problema "domain knowledge" [DAVI-91]. La utilización de este

tipo de operadores hace que el AG sea más específico y, por tanto, menos robusto, pero la mejora de rendimiento es, habitualmente, sustancial.

Suh y Van Gucht [SUH-87] y Grefenstette [GREF-87] apoyan la idea de que se utilice el conocimiento específico del problema para desarrollar operadores de cruce especialmente adaptados. El conocimiento sobre el problema se puede utilizar para evitar la aparición de individuos claramente inadecuados, lo que ahorra el tiempo que se necesitaría para evaluar estos individuos. Por ejemplo, Davidor [DAVD-91a] utiliza un operador que denomina "*analogous crossover*" para el problema de generación de trayectorias de robots. El conocimiento del problema se utilizaba para decidir qué puntos de corte eran inadecuados, ya que producían individuos de bajo ajuste.

El conocimiento sobre el problema puede usarse también para diseñar operadores de mejora local, que permitan realizar un exploración más eficiente del espacio de búsqueda alrededor de los individuos más prometedores [SUH-87]. Otra posibilidad es realizar una inicialización heurística de los individuos de la población en vez de una inicialización al azar, de forma que ya en las primeras generaciones se parta de unos individuos con un ajuste razonable [GREF-87][SCHU-90].

Goldberg [GOLD-89a] describe técnicas para desarrollar operadores de cruce y mutación específicos para distintos tipos de problemas. También aborda el tema de la hibridación de los AA.GG. con otras técnicas de búsqueda.

2.3 Análisis de Series Temporales

2.3.1 Introducción

Una serie temporal es una secuencia ordenada de observaciones. El orden se refiere, normalmente, al tiempo, en concreto a intervalos más o menos extensos como: minutos, horas, días, semanas, años, etc. Las series temporales se pueden referir a múltiples campos como: la agricultura, para seguir los niveles de producción o de precios; los negocios y la economía, para observar los cierres de los mercados de valores, los niveles de los tipos de interés o las ganancias de las empresas; la ingeniería, para estudiar las señales eléctricas o de voltaje; la medicina, para comprender los electroencefalogramas o electrocardiogramas; la meteorología, para seguir las velocidades de los vientos, las temperaturas o las precipitaciones, etc.

Existen distintos objetivos que se plantean para realizar el estudio de las series temporales. Estos incluyen la comprensión y descripción del mecanismo que genera la serie, la predicción de futuros valores y el control óptimo de un sistema. Lo que caracteriza a una serie temporal es que todas las observaciones son dependientes, es decir, están correlacionadas, y el orden de las observaciones es, por tanto, importante. Esto supone que todas las técnicas estadísticas que se basan en la independencia no son aplicables en este caso y se necesitan desarrollar nuevos métodos. El conjunto de estos nuevos métodos para estudiar series temporales es lo que se denomina "análisis de series temporales".

2.3.2 Predicción en Series Temporales

Uno de los objetivos más importantes del estudio de las series temporales es servir de apoyo a la toma de decisiones. El principal problema del que decide es elegir entre distintas alternativas, teniendo en cuenta la utilidad que van a tener sus decisiones ante cada uno de los sucesos que se puedan dar. Estos sucesos son hechos que se sitúan en el futuro y que, por tanto, el que tiene que decidir no conoce. Es obvio que se podrán tomar mejores decisiones si se logra reducir la incertidumbre sobre los sucesos situados en el futuro. Para reducir esta incertidumbre se pueden utilizar las conocidas como "técnicas de predicción". Desde un punto de vista metodológico, los métodos de

predicción se pueden agrupar en dos grandes bloques: métodos cualitativos y métodos cuantitativos.

Los métodos cualitativos se utilizan en aquellos casos en los que el pasado no proporciona una información directa sobre el fenómeno considerado, como ocurre en el caso de la previsión de ventas de nuevos productos que se sacan al mercado. En la predicción de carácter cualitativo, los métodos estadísticos juegan un papel relativamente secundario. En estos casos, lo más importante es contar con un grupo de expertos que tengan un buen conocimiento acerca del fenómeno cuya proyección en el futuro se trata de analizar. En este contexto, los métodos estadísticos serían utilizables en la organización y sistematización de las opiniones de los expertos.

Entre los múltiples métodos de carácter cualitativo, se pueden destacar, de menor a mayor formalismo matemático: el *"Brainstorming"*, el *"Delphi"* y el *"Cross-impact"*. En el conocido método del *"Brainstorming"* la predicción se efectúa a partir de la discusión entre un grupo de expertos donde se crea el ambiente necesario para que afloren nuevas ideas. Prácticamente no necesita de instrumentos estadísticos. En el método "*Delphi*" la predicción se basa en la utilización sistemática e iterativa de juicios de opinión de un grupo de expertos hasta llegar a un acuerdo. Se trata de evitar las influencias de individuos y de fomentar una realimentación que favorezca el acuerdo final. Fue desarrollado originalmente en la empresa "Rand Corporation" fundamentalmente por Helmer y Rescher [HELM-59]. El *"Cross-impact"* o impacto cruzado es el método que requiere mayor formalismo matemático. Este método se utiliza para evaluar una función de distribución de escenarios de anticipación en los que se tienen en cuenta diversos sucesos. Contrariamente a lo que se hace en *"Delphi"*, donde se evalúan de forma individual, uno a uno, varios sucesos, en el método *"Cross-impact"* el tratamiento es conjunto, calculando las probabilidades de ocurrencia, tanto simples como condicionadas, debidas al impacto cruzado de los eventos que pueden suceder. Una de las primeras aplicaciones del *"Cross-impact"* fue realizada por Gordon y Hayward [GORD-68].

En las previsiones de tipo cuantitativo, se parte del supuesto de que se tiene registrada información sobre el pasado del fenómeno que se quiere estudiar. Generalmente, esta información aparece en forma de series temporales. En los métodos cuantitativos la misión del estadístico consiste en extraer toda la información posible

contenida en los datos, y en base al patrón de conducta seguido en el pasado, realizar conjeturas sobre el futuro. Con este tipo de métodos se pretende conocer los componentes subyacentes de una serie Y_t, ya sean determinísticos D_t o aleatorios N_t, y su forma de integración, normalmente aditiva (ver Ecuación 2-6), con objeto de realizar las previsiones de futuro. Con respecto a los métodos cuantitativos se pueden considerar dos enfoques alternativos, el análisis causal y el análisis univariante.

$$Y_t = D_t + N_t \qquad \text{Ecuación 2-6}$$

En el análisis causal la explicación de la variable o variables objeto de estudio se apoya en factores externos. Aunque el término de causalidad aplicado a este análisis puede ser un poco excesivo para describir la relación de estos factores externos con las variables, esta denominación está ya muy asentada.

En el análisis univariante, a diferencia de los métodos de análisis causal, se trata de hacer predicciones de valores futuros de una variable, utilizando como información únicamente la contenida en los valores pasados de la serie temporal. En el análisis univariante se pueden considerar tres grandes grupos: métodos de descomposición, métodos de alisado exponencial y modelos ARIMA univariantes.

En los métodos de descomposición se parte de que el patrón o esquema de generación de una serie temporal se puede descomponer en varios subesquemas. Generalmente, se distinguen los siguientes: tendencia, factor cíclico, movimiento estacional y movimiento irregular. La tendencia refleja las variaciones a largo plazo. El factor cíclico consiste en variaciones superiores al año que no son estrictamente periódicas. Los movimientos estacionales se caracterizan por tener una periodicidad de naturaleza fija, aunque la amplitud puede ser variable y, finalmente, el movimiento irregular sería el componente de la serie no sujeto a ninguna periodicidad en el tiempo. No debe confundirse el movimiento irregular con la parte aleatoria. El movimiento irregular no está sujeto a ninguna periodicidad pero se basa en circunstancias que podrían hacerlo previsible, mientras que la parte aleatoria en ningún caso es previsible. La serie Y_t en el contexto de los métodos de descomposición se puede obtener de la forma:

$$Y_t = T_t \cdot C_t \cdot E_t \cdot I_t \qquad \text{Ecuación 2-7}$$

Donde T_t es la tendencia, C_t es el factor cíclico, E_t es el movimiento estacional e I_t es el movimiento irregular.

En los métodos de descomposición, se parte de un esquema propio y se trata de aislar cada uno de los componentes. Para aislar la tendencia se puede ajustar una curva a los datos mediante una ecuación matemática y después hacer proyecciones en el futuro mediante esta ecuación. Para aislar la estacionaridad también existen métodos que son más complicados que en el caso de la tendencia pero que están bien estudiados. Uno de ellos, de uso muy extendido, es el "X-11" desarrollado por Shiskin y colaboradores [SHIS-67] para el Bureau of Census del Departamento de Comercio de los EE.UU. Los otros dos tipos de componentes presentan más dificultades para su aislamiento.

Los métodos de alisado exponencial permiten también calcular los valores de la tendencia, pero a diferencia de la tendencia de carácter global, en cada punto se hace una revisión en función de las observaciones más recientes. Por esta razón, se dice que los métodos de alisado exponencial proporcionan una tendencia de carácter local. Las aportaciones más relevantes sobre estos métodos pueden verse en el artículo de Makridakis [MAKR-78] y en la obra más reciente de Abraham [ABRA-83].

Tanto en los métodos de descomposición como en el alisado exponencial, el analista establece un esquema "a priori" y después procede a los cálculos estadísticos correspondientes. En los modelos ARIMA (Auto-Regresive Integrated Moving Average) univariantes se hace un planteamiento inicial de carácter general. Se considera que la serie temporal objeto de estudio ha sido generada por un proceso estocástico. Las técnicas de elaboración de los modelos ARIMA van dirigidas precisamente a identificar el modelo generador de las observaciones, para después, en un proceso iterativo, estimar y verificar el modelo, que una vez aceptado se utiliza para predecir valores futuros de la serie temporal. Los modelos ARIMA se popularizaron a partir de 1970 gracias a la obra de los estadísticos Box y Jenkins [BOX-76]. Para referirse a la elaboración de los modelos ARIMA se utiliza también la expresión de análisis de series temporales, aunque lógicamente esta última expresión tiene una mayor amplitud.

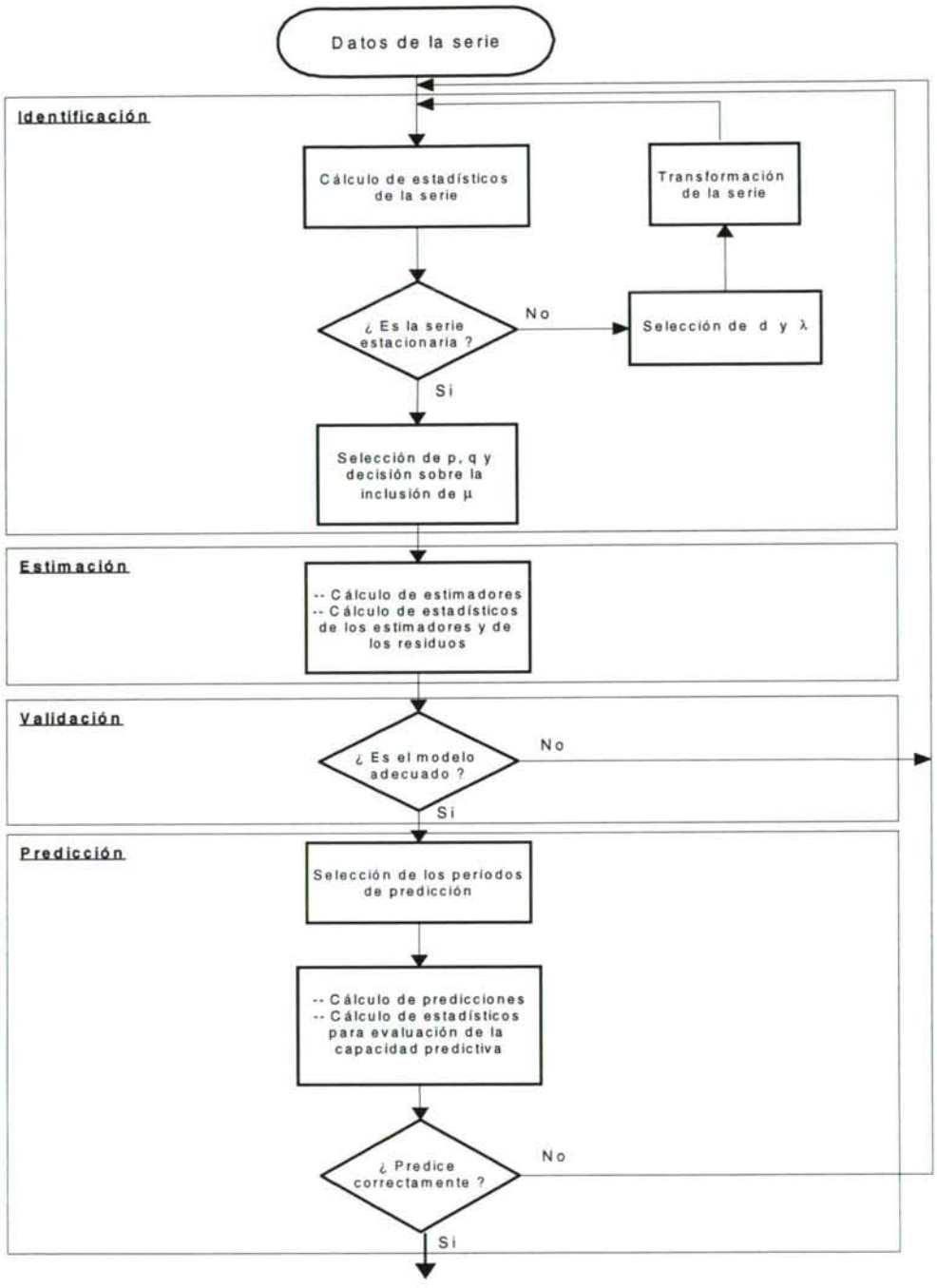


Figura 2.7.- Fases de la Elaboración de un modelo ARIMA

2.3.3 Modelo ARIMA

Los procesos ARIMA(p,d,q) son una clase particular de procesos estocásticos, pero que, sin embargo, pueden ser utilizados para describir el comportamiento de gran parte de las series temporales. En el apartado de Anexos, se incluye información acerca de los procesos estocásticos y de los conceptos de linearidad, estacionariedad y ergodicidad necesarios para la definición de los modelos ARIMA(p,d,q). La elaboración

de un modelo ARIMA consiste en la búsqueda de un proceso ARIMA (p,d,q) que verosímilmente haya podido generar la serie temporal objeto de estudio. Las fases de que consta la elaboración de un modelo ARIMA, se recogen en la Figura 2.7.

En la fase de identificación, en una primera etapa se procede a efectuar un análisis de estacionariedad de la serie. En el caso de que se trate de una serie no estacionaria se aplican las transformaciones adecuadas al objeto de convertirla en estacionaria. En una segunda etapa se procede a determinar el orden de la parte autorregresiva (es decir, p) y el orden de la parte de medias móviles (es decir, q) del proceso ARMA que se considere haya podido generar a la serie estacionaria w_t. En la fase de estimación, se obtienen los valores estimados para los parámetros $\phi_1,...,\phi_p$, $\theta_1,...,\theta_q$, σ y μ del proceso ARMA(p,q).

Una vez concluida esta fase, se tiene conocimiento de un proceso que, hipotéticamente, ha podido generar la serie temporal transformada w_t y a partir de la cual se puede obtener la serie original Y_t.

La fase de validación va dirigida a establecer si se produce o no la adecuación entre datos y modelo y, finalmente, en la fase de predicción se realizan los pronósticos de valores futuros de la variable. La predicción es el verdadero banco de pruebas de un modelo. En el caso de existir discrepancias de carácter sistemático entre los valores pronosticados y los que se observen realmente, se deberá cuestionar la validez del modelo identificado.

2.3.3.1 Análisis de la Estacionariedad

Como ya se ha indicado, para poder aplicar la metodología de los modelos ARIMA es preciso realizar las transformaciones adecuadas. Sin embargo, primero es necesario analizar si la serie temporal es o no estacionaria.

En el caso de la media se utiliza tanto el gráfico de la serie como la "FACE" (función de autocorrelación estimada). El examen visual de la trayectoria de la serie a lo largo del tiempo puede dar una idea de si es o no estacionaria en media. En cualquier caso, es conveniente realizar esta inspección ocular conjuntamente con un examen de la "FACE". Si los coeficientes de la "FACE" no decaen rápidamente, esto sería un indicio claro de que la serie es no estacionaria. Si así ocurriera, se tomaría una diferencia, y se

volvería a analizar la "FACE" de la serie diferenciada. De esta forma, se continuaría hasta obtener una serie diferenciada de orden "d" en la que los coeficientes de la "FACE" decaigan rápidamente a partir de un determinado retardo.

La toma de diferencias de un determinado orden suele ser suficiente en muchos casos para obtener series estacionarias en media y varianza. No obstante, en series que se extienden a lo largo de un período dilatado de tiempo y que están afectadas por una fuerte tendencia, suele ser necesario efectuar además alguna transformación como la de "Box-Cox". Para detectar si la serie es estacionaria en varianza se utiliza también la inspección del gráfico de la serie intentando detectar si se mantiene o no la dispersión de los valores. Además, se utiliza un gráfico de dispersión/media. Para construir este gráfico se divide la serie en intervalos en los que se calcula la media y la varianza. Después, cada par de valores se representan en una gráfica de ejes media y varianza. Si los puntos están más o menos alineados en torno a una línea recta con pendiente ascendente sería indicativo de que los datos no son estacionarios en varianza. Cuando el gráfico no muestre un esquema claro, o bien los puntos estén alineados en torno a una línea paralela al eje de abscisas, entonces no será necesario realizar ningún tipo de transformación.

2.3.3.2 Identificación de un Modelo

Como instrumentos básicos de identificación de los modelos estacionarios también se utiliza la función de autocorrelación estimada "FACE" y, además, la función de autocorrelación parcial "FACPE". El proceso de identificación consiste en comparar el comportamiento de estos dos estadísticos con funciones de "autocorrelación" y "autocorrelación parcial" teóricas correspondientes a distintos modelos teóricos con los que puedan guardar similitud, teniendo en cuenta que nunca cabe esperar una similitud perfecta debido a los errores de muestreo.

Hay que tener en cuenta que el tamaño de la muestra puede influir en el comportamiento de la "FACE" y la "FACPE" de forma que no guarden similitud con el comportamiento de las correspondientes funciones teóricas. Rosa Fernández [FERN-84] ha realizado un estudio sistemático mediante simulación con métodos de Montecarlo para distintos tamaños de muestra. No obstante, como una conclusión general se puede establecer que, en las series cuyo número de observaciones sea pequeño, la

identificación del proceso generador a partir de los instrumentos propuestos es muy difícil. Algunos autores consideran pequeño un tamaño de muestra inferior a 50 ó 60 observaciones, aunque siempre hay que tener en cuenta las características particulares de la serie que se está analizando.

2.3.3.3 Validación

El objetivo perseguido al elaborar un modelo ARIMA es encontrar un modelo que sea lo más adecuado posible para representar el comportamiento de la serie estudiada. Así, un modelo ideal sería el que cumpliera los siguientes requisitos [BOX-76]:

- Los residuos del modelo estimado se aproximan al comportamiento de un ruido blanco o aleatorio.
- El modelo estimado es estacionario e invertible.
- Los coeficientes son estadísticamente significativos y están poco correlacionados entre sí.
- Los coeficientes del modelo son suficientes para representar la serie.
- El grado de ajuste es elevado en comparación al de otros modelos alternativos.

Conviene señalar que es esencial que se cumpla el primer requisito pues, en caso contrario, el modelo debe ser rechazado, ya que ello sería indicativo de que los residuos contienen información relevante para la predicción.

Si después de aplicar los contrastes y análisis de los epígrafes anteriores se llega a la conclusión de que el modelo seleccionado no es adecuado, se debe proceder a reformular el modelo. Para ello, junto a los estadísticos de la fase de identificación, es conveniente tener en cuenta los estadísticos y resultados obtenidos en la fase de validación, puesto que pueden arrojar luz sobre la dirección en que debe reformularse el modelo.

2.3.3.4 Ejemplos Utilizados

2.3.3.4.1 Manchas Solares

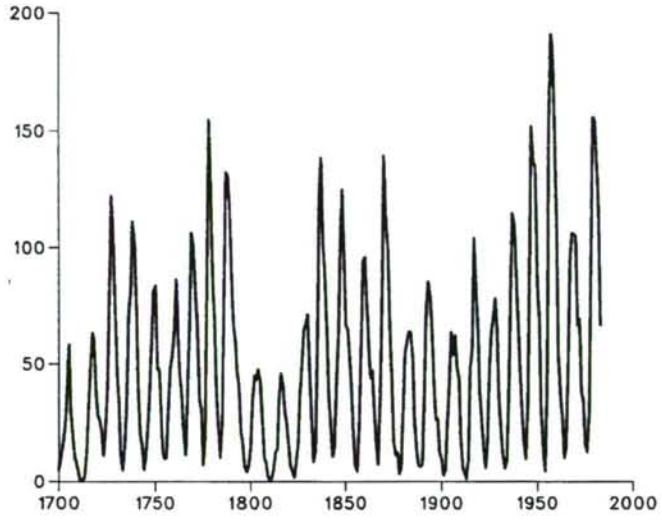


Figura 2.8.- Serie Temporal del Número Medio Anual de Manchas Solares

Esta es la clásica serie de Wolf que presenta el número de manchas solares entre los años 1700 y 1984 con un total de 285 observaciones. La importancia de esta serie radica en el efecto del número de manchas solares tanto en el tiempo atmosférico en la tierra como en la actividad del "cinturón de Van Hallen" que nos protege de las radiaciones solares. Como consecuencia de esto, la predicción del número de manchas solares repercute directamente en actividades como la agricultura y las telecomunicaciones respectivamente. Esta serie ha sido ampliamente discutida en la literatura de series temporales [YULE-27][BART50][WHIL-54] y se conoce también como las "manchas solares de Wolfer" que fue un estudiante de Wolf. La serie entre 1700 y 1955 fue recopilada por Waldmeirer [WALD-61] y el resto de las observaciones fueron calculadas a partir de la media mensual de manchas solares por Andrews y Herzberg [ANDR-85].

El gráfico de la serie indica que la serie es estacionaria en la media. Sin embargo, el estudio de los residuos de la serie aconseja realizar una transformación en los datos para conseguir la estacionariedad en la varianza, por lo que se suele trabajar con la raíz cuadrada de los datos de la serie.

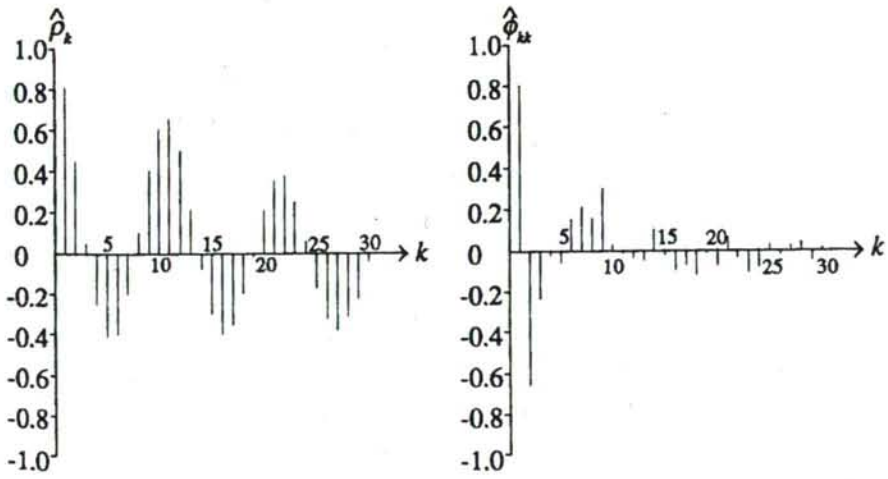


Figura 2.9.- Gráfico de la FACE y la FACPE de la serie de Manchas Solares

Una vez calculada la FACE se observa una forma seno-coseno y en la FACPE aparecen grandes saltos en los puntos 1, 2 y 9. Esto sugiere que los datos transformados se pueden adaptar a un modelo AR(2) o a un modelo AR(9). Ignorando los valores del FACPE más allá del 3, Box y Jenkins [BOX-76] sugieren también la posibilidad de un modelo AR(3), aunque este estudio estaba basado en datos no transformados del intervalo 1770-1869.

2.3.3.4.2 Producción de Tabaco en EEUU

La serie de la Figura 2.10 contiene los datos de la producción anual de tabaco en EE.UU. desde 1871 hasta 1984, y fue publicada en el año 1985 en el Agricultural Statistics del Department of Revenue de los EE.UU.

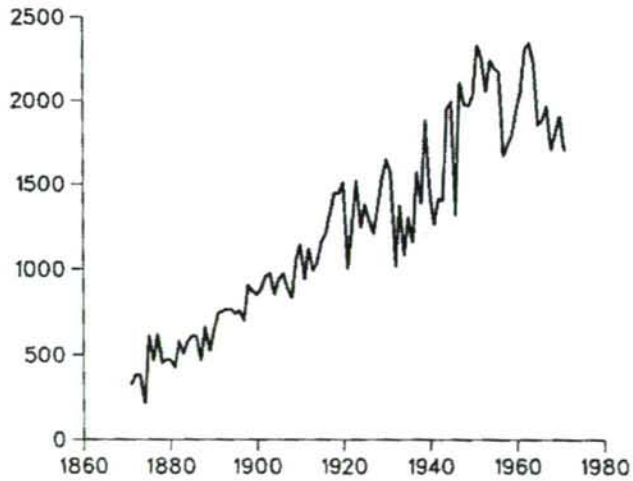


Figura 2.10.- Serie de Producción anual de Tabaco en EEUU

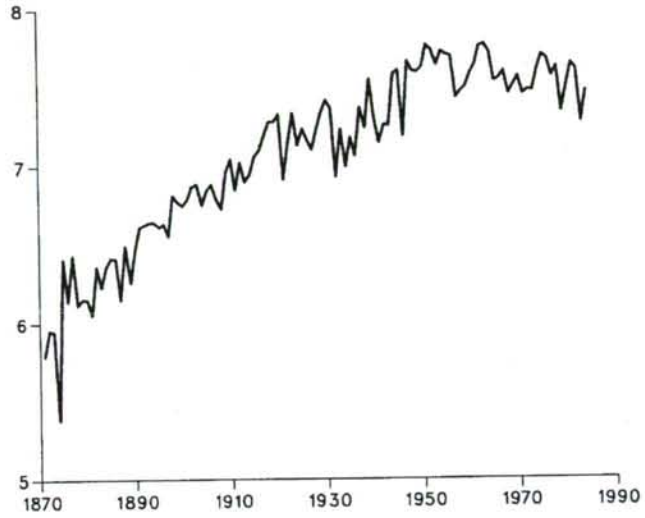


Figura 2.11.- Serie Transformada de Producción de Tabaco en EEUU

El gráfico indica que la serie no es estacionaria ni en la media ni en la varianza. De hecho, la desviación típica es claramente proporcional a la evolución de la serie. Debido a esto se realiza una transformación logarítmica. Esta transformación se puede observar en la Figura 2.11.

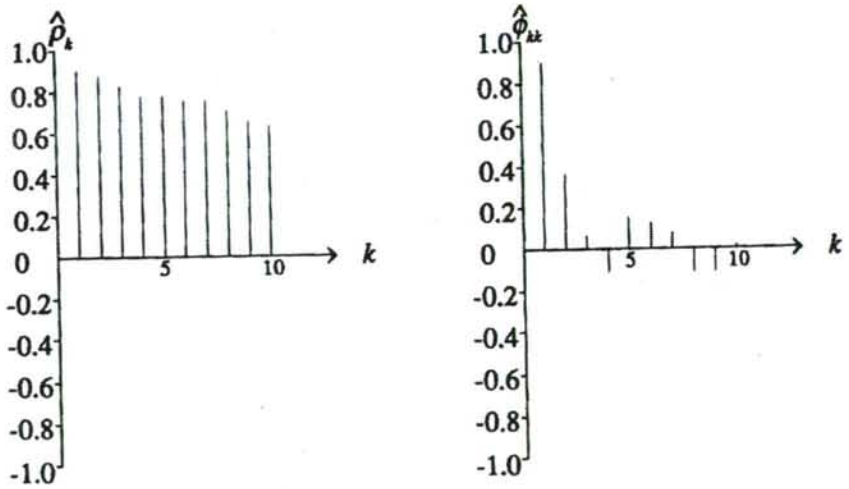


Figura 2.12.- Gráfico de la FACE y la FACPE de la serie de Producción de Tabaco

La lenta disminución de la FACE indica la necesidad de aplicar una diferenciación. Una vez aplicada ésta, se vuelven a calcular la FACE y la FACPE. En ellas el corte en la primera componente de la FACE y la forma exponencial de la FACPE sugieren la posibilidad de un modelo IMA(1,1).

Para los trabajos realizados en esta tesis, se ha implementado este modelo IMA(1,1). A partir de él, se ha realizado la predicción de los últimos 5 valores de la

serie que, como en el caso anterior, no se han tenido en cuenta para desarrollar el modelo, ya que se disponía de datos suficientes.

2.3.4 Consideraciones sobre los Modelos ARIMA

La finalidad última de la construcción de los modelos ARIMA es la predicción y, más concretamente, la predicción a corto plazo, habiéndose obtenido resultados satisfactorios en distintas áreas tales como la macroeconomía, la sociología, los procesos tecnológicos, etc.

No obstante, es importante señalar que la aplicación mecánica de la metodología ARIMA puede conducir a la elaboración de modelos inadecuados que, en muchos casos, carecerán de sentido. Es por tanto necesario que la metodología ARIMA sea aplicada por personas expertas en esta área, para conseguir un modelo fiable y ajustado a las series que se analizan.

En el proceso de modelización son especialmente delicadas las fases de estimación del modelo y análisis de la estacionariedad; ya que, en ellas, se pueden escoger entre distintos parámetros que pueden ofrecer resultados similares pero que influyen notablemente en la bondad del modelo final.

No es, pues, la elaboración de un modelo ARIMA, una labor fácil, ni consiste en una serie de pasos rígidos. Es, más bien, un proceso iterativo de ajuste de parámetros que conduce a la consecución de un modelo ajustado a una serie temporal. La destreza y experiencia del diseñador influyen, de manera notable, tanto en el nivel de ajuste conseguido como en la rapidez de consecución de este ajuste. De esto, se puede concluir que la elaboración de modelos ARIMA no es una labor que se pueda abordar con unos conocimientos superficiales y que, aun realizada por expertos, el proceso de ajuste puede dilatarse en el tiempo para conseguir buenos resultados.

Capítulo 3: Estado del Arte

3. Estado del Arte

En este punto se hace una revisión de los antecedentes de los temas que se tratan en esta tesis: RR.NN.AA., AA.GG. y dominios complejos de aplicación. Por un lado, se revisan los aspectos del diseño de RR.NN.AA. mediante AA.GG., tanto los más tradicionales como los tratados en las investigaciones más recientes. Por otro lado, se revisa la utilización de los distintos tipos de RR.NN.AA. para la categorización y predicción de series temporales, vistas éstas como un dominio complejo de utilización de las RR.NN.AA. En la última parte de este capítulo se presenta una aplicación basada en RR.NN.AA., desarrollada mediante AA.GG., para la predicción de series temporales, como precedente de los trabajos e ideas en que se sustenta esta tesis.

3.1 Diseño de RR.NN.AA. utilizando AA.GG.

3.1.1 Introducción

Uno de los problemas a los que más investigación se está dedicando últimamente en el campo de las Redes de Neuronas Artificiales es encontrar la topología idónea que resuelva una determinada tarea. Es decir, encontrar cuál debe ser en número de capas ocultas, el número de neuronas por capa e, incluso, qué conexiones entre las neuronas son necesarias para el buen funcionamiento de la red. Este problema se plantea, sobre todo, desde que se descubrió que el "perceptrón" no podía resolver problemas que no eran linealmente separables [MINS-69][MINS-88]. Para solucionar este tipo de problemas se hizo necesario colocar capas ocultas entre la capa de entrada y la de salida. De esta forma se consiguió afrontar mediante RR.NN.AA. problemas no linealmente separables, pero se trasladó el problema a decidir el número de capas ocultas y cuántas neuronas eran necesarias en cada capa para resolver un determinado problema. Hasta ahora, la decisión de elegir estos valores se realiza mediante la tradicional técnica de prueba y error, la cual hace que el tiempo necesario para alcanzar una buena solución sea excesivamente largo, teniendo en cuenta que el tiempo de entrenamiento para algún tipo de red puede ser muy largo no es asumible repetir el entrenamiento tantas veces como arquitecturas se quieran probar. Desgraciadamente, no existe una metodología

general que resuelva el problema y el diseñador se basa en su propia experiencia para fijar la topología de la RNA.

Intentando resolver el problema del número óptimo de neuronas en la red se ha desarrollado un conjunto de algoritmos que dinámicamente añaden y eliminan neuronas durante el proceso de entrenamiento. Genéricamente, se conoce a estos algoritmos como de tipo "constructivo/destructivo" o "incrementales" (ver Capítulo de Fundamentos). Para su funcionamiento, empiezan con una posible topología de la red y mediante pequeños ajustes se añaden o eliminan neuronas de las capas ocultas durante el proceso de aprendizaje. Actualmente, estos métodos dependen fuertemente de la topología inicial de la red y presentan ciertas facilidades para caer en mínimos locales, no garantizando una solución satisfactoria o presentando convergencias demasiado lentas o demasiado rápidas. Además, como la estructura de la red está fuertemente ligada a la existencia de un algoritmo de aprendizaje, no se puede garantizar que el proceso de aprendizaje de la red sea el más idóneo para un determinado algoritmo de aprendizaje y, por otra parte, si el proceso de aprendizaje no da buenos resultados, no es posible saber si lo que no era adecuado era la estructura de la red o el algoritmo de aprendizaje.

Tampoco está resuelta completamente la forma de entrenar una RNA a pesar del gran número de algoritmos de aprendizaje desarrollados con sus múltiples variantes. Para algunos problemas en dominios complejos, los algoritmos de aprendizaje pueden tener la tendencia a quedar atrapados en mínimos locales consiguiendo soluciones que no son óptimas. El tipo tradicional de algoritmos de aprendizaje está basado en métodos de gradiente, como el de "retropropagación del error", que son bastante sencillos de implementar y se pueden ajustar para una rápida convergencia. Sin embargo, estos métodos tienen un serio problema en que no garantizan que la solución que alcanzan sea el máximo global. Esto supone que, en aplicaciones reales, los algoritmos de gradiente convergen a soluciones no óptimas de las que no pueden escapar.

Por otro lado, en situaciones complejas, donde están inmersos un número importante de parámetros de entrada, existe el problema de determinar cuáles de estos parámetros son necesarios al desarrollar la RNA. Hasta ahora, se utilizaban métodos estadísticos o metodologías heurísticas que necesitaban de un conocimiento, a priori, del problema a tratar. El empleo de parámetros innecesarios trae consigo una reducción de

la eficiencia del proceso de entrenamiento a la vez que introduce ruido en el proceso, interfiriendo la convergencia del aprendizaje. Por tanto, consiguiendo la reducción de los parámetros de los conjuntos de entrenamiento, no sólo se consigue más velocidad al tratar menos datos sino que los resultados de la RNA van a ser más ajustados a la solución debido a la disminución de ruido a la entrada de la red.

Otro gran problema en el diseño es la falta de escalado que presentan las RR.NN.AA.; la elección de una buena topología de la red para un determinado problema puede no ser buena si se aumenta la complejidad del problema. Por ejemplo, si se elige la topología de la red que resuelva el problema de la paridad de tres bits, no se puede garantizar que para "n-bits", si "n" es un número grande, siga siendo la correcta sin más que aumentar el número de neuronas ocultas. En este sentido, hay que tener en cuenta que la mayoría de las investigaciones se validan mediante pequeños problemas, no siendo extrapolables para problemas de mayor envergadura. Por otra parte, en una RNA existe gran cantidad de información redundante contenida en los pesos; una red óptima debería eliminar dichos pesos redundantes manteniendo la capacidad funcional de la red. Este proceso se conoce como "podaje de la red" [KARN-90].

En otro sentido, la relación entre la topología de la red y el proceso de aprendizaje es generalmente desconocido; sólo existen estudios a nivel microscópico en el espacio de pesos que intentan encontrar la estrategia de aprendizaje más idónea para una determinada red [JOYA-93].

Como ya se ha visto en el capítulo anterior, los AA.GG. son algoritmos de búsqueda para espacios de soluciones complejos inspirados en procesos de la evolución natural. Se aplican principalmente en problemas de optimización y como mecanismos para describir reglas en aplicaciones de aprendizaje de máquinas. Se comportan como una herramienta muy eficaz para tratar problemas que presenten una superficie compleja y ruidosa con múltiples mínimos locales y grandes espacios de búsqueda. Hay que tener en cuenta que la solución que ofrece un AG va a ser siempre una solución aceptable pero, no hay forma de saber si esta solución es la óptima o cuan cerca está de ella. Debido a esto, la utilización de los AG debe verse restringida a las situaciones en que no existe un algoritmo iterativo que consiga una solución aceptable o que sea muy difícil llegar a desarrollar uno. No hay que confundir este hecho con el mencionado sobre los

métodos de gradiente de que pueden conseguir una solución no óptima, ya que aquí, lo que se comenta es que, dentro de la zona de la solución óptima, no hay forma de decir si se está muy cerca o ya se ha llegado a ella. Es por esto que, a veces, se utiliza un aproximador local, como podría ser, en este caso, un método de gradiente, una vez que el AG ha localizado el área de la solución óptima.

Como se puede comprobar por los problemas planteados previamente, el diseño de RR.NN.AA., desde el planteamiento de los datos de entrenamiento hasta la especificación de una arquitectura óptima, parece un campo idóneo para la aplicación de los AA.GG., ya que es un problema de optimización con multitud de variantes y, de momento, no ha sido posible desarrollar una base matemática completa que pueda orientar el diseño de la RNA y sus datos de entrenamiento de acuerdo con el problema que ha de resolver. Por otro lado, los AA.GG. realizan la búsqueda en el espacio de soluciones trabajando de forma paralela, por lo que son muy fáciles de paralelizar implementándolos en máquinas paralelas o redes de ordenadores.

3.1.2 Métodos Básicos para el Diseño de RR.NN.AA. usando AA.GG.

Desde que se inició la aplicación de AA.GG. al diseño de RR.NN.AA. se está trabajando, fundamentalmente, en dos campos de aplicación: primero, en la búsqueda del conjunto óptimo de pesos de conexión para una red en la que ya se ha decidido la arquitectura [MONT-89][WHIT-90a], haciendo aquí, el AG, el papel de regla de aprendizaje; y, segundo, en la búsqueda de la topología óptima (número de neuronas y conectividad) una vez fijada la regla de aprendizaje [HARP-89] [WHIT-90b] [RADC-93] [ROBB-93].

3.1.2.1 AA.GG. para el Ajuste de los Pesos de Conexión

Los AA.GG. han sido aplicados con éxito al entrenamiento de RR.NN.AA. estimando el conjunto óptimo de los pesos de conexión de las redes. Se han estudiado numerosas aproximaciones: desde el simple proceso de evolución del conjunto de pesos mostrado a continuación, hasta algoritmos con sofisticados operadores de recombinación de individuos con intercambios de partes de la topología de las redes.

En el caso del algoritmo más simple, se parte de que la topología de la RNA ya es fija (número de neuronas de entrada y salida, número de capas ocultas, número de neuronas en cada capa oculta y la conectividad entre todas las neuronas) y el algoritmo genético debe encontrar los valores óptimos de pesos; esto es, el AG realiza el entrenamiento de la RNA. Ya se ha aplicado sin problemas a RNA "feed-forward", cambiando el clásico algoritmo de aprendizaje de retropropagación [RUME-86] por el AG.

El funcionamiento del AG para este caso se expone a continuación. Primero se generan todos los individuos de la población, estando compuesto cada individuo por la concatenación de los pesos de esa red "genotipo". Para poder evaluar la adecuación de cada individuo, el genotipo se descodifica asociándole una estructura de RR.NN.AA. "fenotipo" con los valores de los pesos codificados. En este proceso de evaluación de la población, se le presentan los pares de entrenamiento entrada-salida a cada individuo de la población, calculando el error cuadrático medio entre las salidas ideales y las obtenidas al evaluar las redes. El valor de error conseguido supondrá el valor de ajuste o adaptación del individuo. Después de esta etapa de inicialización, se aplican, iterativamente, los operadores genéticos de selección, cruce y mutación para obtener una nueva generación. Estos pasos se repiten hasta que el error cometido por las redes alcance un cierto mínimo prefijado.

La codificación de los pesos de la red se puede realizar de varias maneras, como ya se comentó en el capítulo sobre AG. La manera más común es codificar en binario cada peso de la red y construir cada individuo por la concatenación de estos números binarios. Por ejemplo, en el algoritmo desarrollado por Whitley [WHIT-90a], se utilizan 8 bits para representar cada peso, siendo una magnitud con signo comprendida entre +127 y -127. Este algoritmo se aplicó para la resolución de un XOR de 2 bits, un codificador 4-2-4 y a sumadores de 2 bits con buenos resultados. El principal inconveniente encontrado al utilizar la codificación binaria es la limitación de la precisión por la discretización de los pesos. Si se codifica con muchos bits para aumentar la precisión, los individuos se hacen demasiado grandes con lo que aumenta considerablemente el tiempo de entrenamiento de la red. La solución actual a este problema pasa por hacer un buen balance entre el número de bits de cada peso de

conexión y el intervalo de codificación utilizando técnicas de codificación dinámica [SCHR-92].

Es muy frecuente la utilización de la codificación Gray para asegurar que cambios pequeños en los bits se correspondan con cambios pequeños de los parámetros.

También se ha propuesto una codificación con números reales, pero todavía no está clara cuál de las dos opciones ofrece un mayor rendimiento aunque, probablemente, para individuos de gran tamaño, como la codificación de los pesos de una RNA, ninguna de las dos opciones tenga un redimiento netamente superior. Montana y Davis [MONT-89] diseñaron operadores específicos que incorporan heurísticas para el entrenamiento de la RNA, obteniendo mayor rapidez en el entrenamiento que con la retropropagación tradicional. En este caso, las RR.NN.AA. resultantes se aplicaron a la detección en señales acústicas subacuáticas de entornos ruidosos.

Otro problema tratado con éxito mediante AA.GG. es el de la tolerancia a fallos de las RR.NN.AA. En el trabajo de Sebald y Fogel [SEBA-92] se entrenan los pesos de las conexiones en redes "feedforward" con especial interés en controlar el rendimiento de la red en caso de fallo de algunos nodos. Las redes se entrenan incrementando progresivamente la posibilidad de fallo de nodos. Después del entrenamiento se comprueba la respuesta de la red dados "N" fallos aleatorios durante cada proceso de test. De estas pruebas se obtuvo la conclusión de que las redes tolerantes a fallos tenían un peor rendimiento que las no tolerantes cuando la probabilidad de fallo de nodos era cercana a cero, pero mostraban mucho mejor rendimiento que las no tolerantes cuando esta probabilidad crecía.

3.1.2.2 AA.GG. para el Diseño de la topología

Normalmente, la decisión de elegir la topología de una RNA se basa en las experiencias previas del diseñador, además de en el proceso de prueba y error. Como ya se ha visto, los algoritmos "constructivos/destructivos" dependen fuertemente de la topología inicial y pueden caer en mínimos locales, no garantizando una topología óptima o presentando convergencias demasiado rápidas o demasiado lentas.

Se está trabajando en un espacio de búsqueda muy grande y el número de posibles neuronas y conexiones entre ellas es "a priori" ilimitado. Además, se presenta el problema del escalado, topologías bastante diferentes pueden tener capacidades similares (espacio de búsqueda multimodal) y topologías muy parecidas pueden tener diferentes rendimientos (espacio de búsqueda de apariencia engañosa o no lineal). Todas estas características hacen muy adecuada la aplicación del AG al desarrollo de la arquitectura de RR.NN.AA.

A continuación, se detalla un posible proceso de síntesis de la arquitectura para una RNA. En primer lugar, se generan aleatoriamente todos los individuos de la población, codificando cada uno de ellos a una determinada topología de la red "fenotipo". El "genotipo" de cada individuo se obtiene de la construcción de una RNA "fenotipo" con las características especificadas en la codificación. Una vez se tiene la arquitectura de la red, el entrenamiento se realiza normalmente con algoritmos clásicos, como el "back-propagation", aunque no existiría ningún problema en realizar también el entrenamiento mediante AA.GG. Durante el entrenamiento se presentan los pares de patrones de entrada-salida calculando el error cuadrático medio entre la salida ideal y la obtenida por la red. Hay que tener en cuenta que es necesario definir un límite, ya sea temporal o de ciclos de entrenamiento, para conseguir una estimación determinista del tiempo de evaluación de cada individuo de la población. Es decir, no se puede esperar a ver si la red converge o no con una determinada arquitectura, ya que de esta forma el proceso de selección de la arquitectura óptima se vería bloqueado. Por tanto, es necesario establecer un número de iteraciones del entrenamiento o un cierto tiempo, de forma que todas las arquitecturas se evalúen mediante un criterio homogéneo. El valor de adaptación que habitualmente se toma de cada individuo, el error cuadrático medio, se suele ponderar, en este caso, con otros términos que penalizan la complejidad de la topología de la red o el tiempo de entrenamiento. Una vez se dispone de todos los individuos valorados, se le aplica a la población los operadores genéticos selección, cruce y mutación, para obtener una nueva generación repitiendo estos pasos hasta conseguir el nivel de error deseado.

Como en el punto anterior, uno de los aspectos más discutidos en la utilización de los AA.GG., es la codificación de los individuos. La codificación más habitual para las arquitecturas de RR.NN.AA. es la denominada codificación directa que representa

explícitamente cada conexión de la red mediante dígitos binarios. Miller [MILL-89] utiliza una codificación de este tipo que resuelve el problema de la XOR de dos bits. En general, para una red con N neuronas se define una matriz de conexión W de orden NxN donde sus elementos w_{ij} indican que existe una conexión desde la neurona "i" a la neurona "j". Los elementos w_{ij} toman valores 0 ó 1, representando el 1 que existe dicha conexión y el 0 que no existe. Si la matriz de conexiones es triangular inferior, la red sólo tiene conexiones hacia adelante "feed-forward" y si es triangular superior con elementos distintos de cero en la diagonal principal, se está representando una RNA de tipo recurrente. La ventaja de esta codificación frente a las que se verán a continuación es la fácil aplicación de los operadores genéticos pero, en casos de grandes redes, las matrices de conexión adquieren un tamaño enorme, con lo que el tiempo de proceso se hace casi prohibitivo.

Para reducir el tamaño del individuo "genotipo", se realiza una codificación denominada codificación indirecta en la que sólo aparecen las características más importantes de la conectividad. Los detalles de la matriz de conexión se solucionan con reglas de desarrollo [KITA-90], proyecciones [HARP-89] o grado de densidad [DODD-91] durante la descodificación del individuo "fenotipo". Aquí, el "fenotipo" es distinto al "genotipo". La gran ventaja de la codificación indirecta es la compactación mientras que hace más difícil la aplicación de los operadores genéticos.

En la codificación indirecta, los pesos de la red se codifican en una cadena de dígitos binarios [HARP-90], los cuales son de longitud variable y están constituidos por una concatenación de áreas. Cada área se subdivide en áreas de especificación de los parámetros y área de especificación de la conectividad. Esta codificación requiere utilizar operadores genéticos específicos, ya que individuos funcionalmente correctos pueden producir hijos no funcionales como, por ejemplo, hijos sin conexiones hacia la capa de salida de la red. Su sistema llamado "Neurogenesys" tiene un filtro para desechar individuos con anormalidades o purificar individuos que presenten pequeñas anormalidades. Todo el estudio se ha aplicado para el diseño de redes "feed-forward", en tres tipos de problemas: aproximación de la función seno, el problema de la XOR de dos bits y el reconocimiento de dígitos manuscritos de tamaño 4x8. Los resultados obtenidos fueron sorprendentes: el reconocimientos de dígitos eliminó todas las neuronas ocultas ya que es un problema linealmente separable, y en la función XOR,

permitiendo conexiones directas desde la capa de entrada a la capa de salida se eliminó una neurona de la capa oculta.

Otro método de codificación indirecta [KITA-90] utiliza descripciones recursivas de redes, con producción gramatical para generar matrices de conexión. Esta representación es muy compacta ya que elimina la redundancia permutacional. Por último, el método de codificación indirecta [GRUA-92] empieza con una red de una neurona, la hace crecer y la controla mediante un programa estructurado en árbol. Puede expresar redes recursivas añadiendo bifurcación condicional y permite colecciones de programas etiquetados que pueden ser invocados como una operación. Presenta las importantes propiedades de compactación y escalabilidad. También Mjolsness y colaboradores [MJOL-89] utilizan una codificación compacta donde la matriz de conexiones se construye a partir de un patrón inicial por medio de recursividad de operadores de duplicación, introduciendo en su sistema la propiedad de escalabilidad.

Sobre otras arquitecturas también existen trabajos como los de Polani y Uthmann [POLA-93] que utilizan AA.GG. para mejorar la topología en mapas de características de Kohonen. En los resultados de este trabajo se encontró que el AG era capaz de descubrir topologías no planas que se podían entrenar para conseguir mejor rendimiento que con cualquiera de las topologías planas. Otro trabajo sobre el tema es el de Whitehead y Choate [WHCH-94] en el que tratan de determinar la distribución óptima de redes RBF. El AG se utiliza para calcular el número óptimo de elementos en la primera capa, así como el ajuste de las funciones para conseguir la minimización de los errores residuales en la clasificación. Las conexiones entre la primera capa y la de salida se entrenan con un método estándar de descenso de gradiente. Los resultados obtenidos muestran una marcada superioridad del método evolutivo para calcular los centroides sobre las técnicas estándar de agrupamiento como el K-medias.

Por último, desde hace poco tiempo, se encuentran trabajos como el de Braun [BRAU-93] en los que se evoluciona de forma simultánea la arquitectura y los pesos de conexión, persiguiendo el desarrollo eficiente de grandes RR.NN.AA. que sean capaces de manejar problemas complejos del mundo real.

3.1.3 Métodos Avanzados para el Diseño de RR.NN.AA. usando AA.GG.

Tras el desarrollo de los métodos anteriores para la construcción de RR.NN.AA. más eficientes, que como se puede ver están basados en la arquitectura de las redes, se comienza a trabajar en la optimización de otros aspectos del diseño de una RNA. En este punto, se van a repasar los trabajos realizados en la selección de la regla de aprendizaje óptima una vez que el número de neuronas y la conectividad entre ellas ha sido seleccionada, en la selección de la función de transferencia de las neuronas de la red y en la selección de las características más adecuadas del problema para formar parte del conjunto de entrenamiento de la red.

3.1.3.1 AA.GG. para Seleccionar la Regla de Aprendizaje

Como ya se ha comentado en este mismo capítulo, la estructura de la red está fuertemente ligada a que el algoritmo de aprendizaje funcione de una forma más o menos potente. No se puede garantizar que la estructura de una red sea la más idónea para un determinado algoritmo de aprendizaje y, por otra parte, si el proceso de aprendizaje no da buenos resultados, no es posible saber si ha fallado la red o es el propio algoritmo de aprendizaje el que es incapaz de hacer converger a la red. La relación entre la topología de la red y el proceso de aprendizaje es generalmente desconocida.

Debido a esto, también se han desarrollado AA.GG. que, dada una cierta topología de red, adapten la regla de aprendizaje, de forma que ésta sea óptima para el problema con el que se está tratando y para la arquitectura de RR.NN.AA. que el diseñador cree que es la más adecuada. Como es posible imaginar, se puede haber llegado a una cierta arquitectura a través del clásico método de prueba y error o a través de la aplicación de un AG que escoja la arquitectura adecuada.

El procedimiento tiene los siguientes pasos. Como es habitual, en primer lugar se generan aleatoriamente todos los individuos de la población siendo, en este caso, cada individuo, la codificación de una regla de aprendizaje "fenotipo". En los primeros desarrollos de este tipo de AA.GG. la regla de aprendizaje es la misma para todas las

conexiones de la red. El "genotipo" se construye decodificando la regla de aprendizaje y utilizándola en el entrenamiento de la red. Para este entrenamiento, se representan los pares de patrones de entrada-salida calculando el error cuadrático medio entre la salida ideal y la obtenida al evaluar la red. Como en el caso de la síntesis de la topología, el valor de ajuste de un individuo se toma como la ponderación del error cuadrático medio más otros términos que penalicen la complejidad de la topología de la red o el tiempo de entrenamiento. Por último, de forma iterativa, se aplican los operadores genéticos selección, cruce y mutación para obtener nuevas generaciones hasta conseguir una regla de aprendizaje que haga converger la red de forma aceptable.

Como una regla de aprendizaje, se puede describir como una función lineal de n variables y sus productos:

$$\Delta w(t) = \sum_{k=1}^{n} \sum_{i_1, i_2, \ldots, i_k = 1}^{n} \left[\theta_{i_1, i_2, \ldots, i_k} \prod_{j=1}^{k} x_{i_j}(t-1) \right]$$

donde t es el tiempo, Δw es la variación del peso y x_k son los valores de las activaciones de las neuronas a las que está conectada. Por tanto, la optimización de la regla de aprendizaje consiste en obtener los coeficientes θ óptimos.

Chalmers [CHAL-90] y, posteriormente, Fontaneri y Meir [FONT-91] definen la forma de la regla de aprendizaje como una función lineal de variables locales (sólo necesitan información de esa conexión pero no de las demás conexiones) y términos producto. Cada individuo en la población se codifica como una cadena binaria que contiene 10 coeficientes más un parámetro de escala. Mediante este AG, después de 1000 generaciones, se llega a la conocida regla de aprendizaje delta [WIDR-60] y a algunas de sus variantes [CHAL-90].

3.1.3.2 AA.GG. para Seleccionar la Función de Transferencia

Esta es un área muy reciente de utilización de los AA.GG. en RR.NN.AA. Las funciones que habitualmente se utilizan, sigmoidal, Gaussiana, etc, se seleccionan debido a su diferencialidad, posibilidades de tratamiento matemático y facilidad de implementación. Sin embargo, existe un conjunto prácticamente ilimitado de funciones que se pueden utilizar, desde polinomios a funciones discontinuas, pasando por

funciones no diferenciables. En teoría, la selección de la función de transferencia adecuada, conseguiría soluciones de RR.NN.AA. mucho más robustas. Estos estudios se realizan, normalmente, mediante el desarrollo simultáneo de la arquitectura y de las funciones de transferencia de los nodos de la red.

3.1.3.3 AA.GG. para la Selección de las Variables de Entrada

La Computación Evolutiva, y por ende los AA.GG., son técnicas adecuadas para la selección de las características necesarias en el conjunto de entrenamiento. Esta técnica puede ser una alternativa a los métodos tradicionales como el análisis de las componentes principales y otros métodos estadísticos. Esta búsqueda eficiente y automática en las características del espacio de entrada puede reducir significativamente los requerimientos computacionales de los algoritmos de preprocesado de señal y extracción de características.

Brotherson y Simpson [BROT-95] exponen un algoritmo que, automáticamente, selecciona el subconjunto óptimo de características de entrada y la arquitectura de la RNA, entrenando después la red también mediante técnicas evolutivas. Chang y Lippmann [CHAN-91] estudian el uso de AA.GG. para determinar qué datos de entrada utilizar y para seleccionar las características de entrada más adecuadas para un sistema clasificador empleado en dos tipos de problemas: reconocimiento de voz y visión artificial. Usando esta aproximación, se logró reducir el conjunto de entrada de 153 variables o características a tan solo 33, sin ninguna pérdida de rendimiento en la clasificación. En sus investigaciones en visión artificial constataron la capacidad de los AA.GG. para detectar las características más importantes del conjunto de entrada eliminando los errores de clasificación.

Ultimamente, está creciendo el interés en esta área de la selección de características debido a que multitud de sistemas de RR.NN.AA. se están poniendo en producción. La selección del conjunto óptimo para el entrenamiento de la red puede suponer la convergencia en un tiempo razonable frente a la imposibilidad de realizar un entrenamiento de RR.NN.AA. con un conjunto excesivo de datos de entrada.

3.1.4 Consideraciones sobre el Desarrollo y Optimización de RR.NN.AA. mediante AA.GG.

De los puntos precedentes, se puede obtener la idea de que, en el estado actual de desarrollo de las técnicas de RR.NN.AA., la optimización de las características de la RNA es uno de los puntos clave de las investigaciones que se realizan actualmente. Esta necesidad de optimización es lógica debido a los niveles de complejidad que están alcanzando, tanto las nuevas arquitecturas como los problemas que se intentan resolver con ellas.

En los puntos anteriores se han tratado una serie de aspectos del desarrollo de RR.NN.AA. que ya se pueden abordar actualmente mediante una aproximación evolutiva. Estos aspectos abarcan desde el clásico ajuste de pesos hasta el más novedoso diseño de la regla de aprendizaje. Existen, sin embargo, campos no estudiados en profundidad en las aproximaciones revisadas.

Por un lado, está el estudio y construcción de los conjuntos de entrenamiento de las redes. Se ha comentado sobre la selección de las variables de entrada necesarias para el entrenamiento, pero es posible que no todos los valores de una variable sean necesarios o significativos. En problemas sobre predicción de series temporales la realización de este tipo de estudios es fundamental para limitar el período de entrenamiento de la red.

Otro punto importante es la utilización de AA.GG. para el desarrollo y validación de nuevas arquitecturas de RR.NN.AA. Es posible la utilización de AA.GG. para modificar o ampliar los modelos de neuronas o de conexión para mejorar el funcionamiento de las RR.NN.AA. ante ciertos problemas.

Por último, el AG es una técnica fácilmente paralelizable que permite mejorar el rendimiento en la búsqueda de soluciones a un problema, en este caso el diseño de una RNA, distribuyendo la computación necesaria entre varios procesadores u ordenadores de una red. Este punto es muy interesante en el desarrollo de redes temporales o recurrentes, en las cuales, debido a su complejidad, cualquier algoritmo utilizado en su entrenamiento, ya sea convencional o evolutivo, consumirá un tiempo considerable.

La tendencia de la aplicación de AA.GG. al desarrollo de RR.NN.AA., tanto alimentadas hacia delante como recurrentes, es la integración de las distintas técnicas que se han mencionado en los puntos anteriores. Se busca un desarrollo integral de la RNA en cada una de sus etapas de diseño. Se busca automatizar estas etapas de forma que el diseñador no se preocupe de realizar pruebas repetitivas buscando una mejora de rendimiento o una disminución del error. Se busca, en definitiva, una herramienta que permita desarrollar RR.NN.AA. evolutivas, RR.NN.AA. que se adapten al problema y lo resuelvan con el mínimo de intervención por parte del diseñador.

3.2 RR.NN.AA. para la Predicción de Series Temporales

3.2.1 Introducción

Uno de los campos de la ciencia más común en el estudio de los fenómenos naturales es la predicción; es decir, a partir de los conocimientos sobre un sistema y sobre su comportamiento anterior, suponer qué puede suceder en su evolución futura. Esta tarea se puede realizar con dos tipos de métodos: construyendo modelos de los sistemas sobre los que se quiere realizar las predicciones o mediante una aproximación estadística. En el primer tipo de método, se asume que existe suficiente información "a priori" de forma que se pueda construir un modelo suficientemente exacto del modelo que está generando el proceso observado. En esta aproximación existen dos problemas. Por un lado, no es posible generar un modelo exacto si las leyes que subyacen al proceso no están suficientemente claras, como la predicción de los mercados de valores. Por otro lado, incluso en el caso de que se pueda conseguir un modelo exacto del proceso, la especificación del estado actual del proceso puede requerir mucha más información de la que se puede obtener en la práctica. Para la predicción del tiempo atmosférico, es posible definir un modelo como un conjunto de ecuaciones diferenciales, pero es necesario definir el estado inicial como funciones continuas en tres dimensiones que no es posible conseguir.

La segunda aproximación intenta analizar la secuencia de observaciones producidas directamente sobre el proceso. A partir de las observaciones se espera ser

capaz de inferir los conocimientos necesarios para predecir la evolución futura del proceso. El problema de esta aproximación es que la naturaleza tiende a producir comportamientos complicados, frecuentemente irregulares e incluso caóticos debido a la interacción con sistemas con un gran número de grados de libertad.

Las RR.NN.AA. pueden considerarse como modelos no lineales flexibles que se pueden parametizar y los parámetros de las RR.NN.AA. se adaptan de acuerdo a los datos disponibles. En este sentido, las RR.NN.AA. han sido utilizadas con éxito en la predicción del comportamiento de distintos tipos de series temporales.

3.2.2 RR.NN.AA. con Retropropagación

Desde que se comenzaron a utilizar las RR.NN.AA. en la predicción de fenómenos dinámicos se ha empezado a comparar su rendimiento con otras técnicas tradicionales. Lapedes y Farber [LAPE-87] concluyen en sus trabajos que una RNA es capaz de mejorar los resultados de los métodos tradicionales de predicción en varios órdenes de magnitud. Estos trabajos se basan en dos series temporales sin ruido. En otro trabajo de Sharda y Patil [SHAR-90], se realizó la comparación entre las RR.NN.AA. y el método "Box-Jenkins" en 75 series temporales de distintos tipos. En este estudio, cada método era mejor que el otro en la mitad de los casos. En un estudio más moderno de Tang y colaboradores [TANG-91] se compara también el funcionamiento de las RR.NN.AA. y del método "Box-Jenkins" centrándose más la importancia que tiene en la predicción el ajuste de la configuración de la RNA. Aquí se concluye que, para series con memoria temporal larga, el rendimiento que ofrecen los dos métodos es bueno con alguna ventaja para el método "Box-Jenkins" en predicción a corto plazo. En predicción a largo plazo las RR.NN.AA. son mejores. Para series de memoria corta, las RR.NN.AA. son superiores al método "Box-Jenkins" en todos los casos. Según este estudio, las RR.NN.AA. pueden ser entrenadas para aproximar el funcionamiento de una serie temporal, pero la exactitud alcanzada dependerá de varios factores como la arquitectura de la red, el método de aprendizaje y el tipo de entrenamiento. También concluyen que una RNA sin capa oculta funcionará de forma similar al modelo "Box-Jenkins". Afirman que las arquitecturas de red utilizadas en su estudio pueden no ser las más adecuadas por lo que, los resultados de rendimiento utilizando RR.NN.AA. podrían

mejorar con el ajuste de las RR.NN.AA. a utilizar y con la utilización de RR.NN.AA. del tipo recurrentes y en cascada, para realizar la predicción.

En la línea de buscar la optimización de las arquitecturas de RR.NN.AA. está el artículo de Weigend y colaboradores [WEIG-91]. En estos trabajos, se modifica la función de error del algoritmo de retropropagación para añadir un componente que penaliza la complejidad de la red. Este desarrollo se aplica a la predicción en dos series temporales: la evolución de las manchas solares y la evolución de los ratios de cambio de divisas. Los autores consiguen desarrollar redes más pequeñas que pueden ayudar a prevenir el fenómeno del "sobre-aprendizaje" o memorización de ejemplos. Además, la reducción de unidades de proceso en las capas ocultas permite mejorar la interpretación de las soluciones.

Otro trabajo que refleja las mejoras que aporta la utilización de RR.NN.AA. a la predicción es el de Park y colaboradores [PASH-91]. Utilizan RR.NN.AA. para predecir la demanda eléctrica del día siguiente a partir de los datos de carga actual y temperatura ambiente. Los resultados obtenidos consiguen la mitad de error que los obtenidos con los métodos utilizados habitualmente. La RNA utilizada es un perceptrón multicapa entrenado mediante la regla Delta generalizada. En este estudio se hacen varias pruebas con distintas configuraciones de neuronas en la capa oculta sin utilizar ningún método automático.

Otro estudio interesante es Chackraborty y colaboradores [CHAK-92] donde se trabaja en la predicción de series temporales multivariable en las que se utilizan varias series temporales correlacionadas para apoyar la predicción de otra serie temporal. En el estudio, se compara la predicción para cada serie individual con la predicción con valores de tres series temporales de la evolución de los precios de la harina en tres ciudades norteamericanas. El modelo de RNA utilizado es un "perceptrón multicapa" con el "algoritmo de retropropagación del error". De nuevo, se utilizan varias arquitecturas con distinto número de neuronas en la capa oculta con el objetivo de mejorar el rendimiento de la red, pero sin utilizar ningún método automático. En las conclusiones se resalta que las RR.NN.AA. que utilizan la información conjunta de las tres ciudades ofrecen un rendimiento mucho mejor que las que realizan la predicción con datos independientes y es también mejor que la aproximación realizada mediante un

modelo ARMA. Se menciona en el artículo la posibilidad de utilizar otras arquitecturas de RR.NN.AA., como las RR.NN.AA.RR., pero se justifica su no utilización por la carga computacional excesiva que supone su uso y por la dificultad de ajuste de los parámetros de RR.NN.AA.RR. Por tanto, se indica la diferencia de rendimiento que supondría la utilización de RR.NN.AA. frente a RR.NN.AA.RR.

3.2.3 Arquitecturas Especializadas

En este punto, se describen ejemplos de modificaciones realizadas en distintas arquitecturas para mejorar la predicción de series temporales.

Una de ellas descrita por Sanger [SANG-91] plantea la utilización una RNA con una estructura de árbol de forma que se vayan incorporando RR.NN.AA. en las ramas según se vayan teniendo en cuenta distintas dimensiones de la entrada de la red. Esta técnica ofrece un algoritmo de aprendizaje rápido, exento de mínimos locales una vez que se fija la forma de la red. Primero se entrena una red mediante la regla "delta". Para ciertos casos que la red original no resuelve satisfactoriamente, se utilizan distintas subredes que se encargan de ponderar los pesos de la primera red para obtener un mejor rendimiento. Estas subredes trabajan en distintas dimensiones de los datos de entrada y se van añadiendo a la red principal formando el árbol. Se aplica a tres ejemplos: simular el manejo de un brazo robot con dos grados de libertad, la predicción de la ecuación caótica de Mackey-Glass y la predicción de los valores de pixels en imágenes reales, comportándose aceptablemente en los tres casos de prueba.

Esta arquitectura de red está indicada para la aproximación de funciones en espacios de búsqueda con un número de dimensiones muy grande. En este campo ofrece un método rápido y bastante genérico. También hay que apuntar que, según se indica en el propio trabajo, la estructura de la red se escoge siguiendo ciertas heurísticas que no garantizan la consecución de árboles de tamaño mínimo.

Zaknich y colaboradores [ZAKN-91] utilizan una modificación de las redes probabilísicas PNN para la predicción de series no lineales. Las distribuciones de probabilidad gaussianas de las redes PNN se expresan aquí como un estimador de la función de densidad de probabilidad de Parzen para que estas redes sean capaces de realizar un reconocimiento de patrones no lineal aplicado a series temporales. En este

trabajo, el nuevo tipo de redes PNN se validan en un problema de filtrado de señales. Las señales que utilizan son sinusoidales que, "a posteriori" han sido amplificadas, comprimidas y contaminadas luego con tres niveles distintos de ruido no-gaussiano. Las redes PNN modificadas se comparan con una RNA tradicional alimentada hacia delante. Los resultados obtenidos de "error cuadrático medio" son similares pero el tiempo de entrenamiento de las redes PNN es entre 1000 y 3000 veces más rápido. Como en los trabajos anteriores, se realizan las pruebas utilizando distintas configuraciones en la arquitectura tanto de las redes PNN como en las "alimentadas hacia delante".

Para terminar este punto, se presentan otros dos trabajos que basan la predicción en la serie temporal caótica de Mackey-Glass. Mead y colaboradores [MEAD-91] utilizan una red de tipo "Connectionist Normalized Local Spline" (CNLS) para realizar la predicción de series temporales. Este tipo de red es una optimización de las redes RBF que minimiza el tiempo de entrenamiento. En el trabajo, se realiza manualmente el ajuste de los parámetros óptimos de funcionamiento de la red. Con respecto a los resultados conseguidos por Lapedes [LAPE-87], en este trabajo se mejoran utilizando una máquina de una potencia unas 40 veces inferior (Sun SPARC 1 frente a un computador vectorial Cray XMP) en la mitad de tiempo, una hora frente a dos.

Ensley y Nelson [ENSL-92] plantean la utilización de un tipo de red incremental, las RR.NN.AA. Cascade Correlation, en la predicción de series temporales, buscando que la propia red consiga adaptar su arquitectura. Como problema de validación utilizan la serie temporal caótica de Mackey-Glass. Además de conseguir ajustar el número de neuronas en capas ocultas, en el algoritmo incremental presentado se permite añadir neuronas con distintas funciones de activación con lo que se automatiza también esta faceta del diseño.

3.2.4 RR.NN.AA. Recurrentes

No es fácil encontrar, en revistas, *proceedings* o libros, artículos que traten de la aplicación de RR.NN.AA.RR. a la predicción de series temporales de problemas reales. Los artículos más numerosos se refieren a las dos categorías comentadas anteriormente. Esto es debido a la problemática de entrenamiento de las RR.NN.AA.RR. y a su, todavía, carácter experimental para muchos investigadores.

En este campo, un artículo reciente de Hee Yeal y Sung Yan [HEE-97] trata de la optimización del modelo de redes FIR sustituyendo el algoritmo de "aprendizaje por gradiente" por un algoritmo de aprendizaje de "optimización por capas". Aunque las redes FIR no son redes recurrentes, se incluye esta referencia en este apartado por ser una red de funcionamiento temporal con retardos, de las que tampoco existen demasiados ejemplos sobre problemas reales. En este caso, se busca una mejora en el rendimiento evitando la utilización del algoritmo de gradiente para solucionar los problemas que acarrea este tipo de algoritmos. El nuevo algoritmo de aprendizaje se aplica a la predicción de dos series clásicas: la ecuación de Lorenz y la de Mackey-Glass, consiguiendo una mejora de dos órdenes de magnitud tanto en rapidez como en exactitud.

Siguiendo en este tema de la utilización de RR.NN.AA. de procesado temporal para el tratamiento y reconocimiento de patrones en series temporales, también existen otros artículos recientes sobre RR.NN.AA. FIR [BACK-91][WAN-90a] y sobre redes con retardos temporales [DAY-93], [LANG-88] y [WAIB-89].

En el artículo de Vassilios y colaboradores [VASS-96] se propone un esquema de red recurrente basada en una asignación de crédito para la clasificación de series temporales. La red se compone de un conjunto de módulos predictivos en el nivel de entrada y un módulo de decisión en el nivel de salida. Para cada módulo predictivo, se calcula una función de crédito, siendo el módulo que mejor predice la serie temporal el que más crédito recibe. En el artículo se demuestra que las funciones de crédito convergen a los valores correctos y se prueba su funcionamiento en tres tipos de problemas de clasificación demostrando una clara mejora frente a los métodos Bayesianos de clasificación.

Otro trabajo interesante es el de DeLiang Wang y colaboradores [DELI-96] donde se hace un estudio de las posibilidades de generalización de las redes recurrentes de Elman, que son redes parcialmente recurrentes, frente a problemas de cálculo de trayectorias. Las conclusiones de este estudio, para este tipo de problemas, son que estos modelos simples de redes recurrentes no son capaces de obtener una buena generalización temporal.

Por último, Connor y colaboradores [CONN-94] desarrollan un método de aprendizaje para RR.NN.AA.RR. que se basa en el filtrado de datos de entrada y de salida para lograr una mayor generalización de la red. Las redes se han entrenado utilizando series temporales de prueba y series reales de consumo eléctrico. Según el estudio, estas redes se comportan mucho mejor que las de "alimentación hacia delante", trabajando sobre series temporales con una componente de MA o medias móviles.

3.2.5 Consideraciones de la Aplicación de RR.NN.AA. a la Predicción de Series Temporales

Lo más habitual, por el número de artículos publicados y de referencias bibliográficas existentes, es la utilización de RR.NN.AA. de "alimentación hacia delante". Este tipo de redes exige trabajar con ventanas temporales por lo que es necesario definir el número de entradas de instantes de tiempo pasados que se van a utilizar. Normalmente, no se conoce el valor óptimo de ventana temporal para cada problema. No es posible, por tanto, saber cuantos instantes de tiempo anteriores influyen significativamente en el futuro inmediato de una serie temporal. Nuevamente, el proceso de prueba y error se impone como elemento fundamental de diseño de la RNA.

Otra opción muy usual en la bibliografía es la aparición de arquitecturas especializadas de RR.NN.AA., normalmente variaciones pequeñas de arquitecturas estándar que resuelven problemas muy específicos en el campo de predicción de series temporales. Esto es debido a la dificultad del campo de aplicación para el cual no se adaptan siempre de manera óptima las arquitecturas más conocidas de RR.NN.AA. En estos casos, es habitual la implementación adicional de mecanismos de ajuste automático de ciertos parámetros de arquitectura. La falta de generalización de estas arquitecturas es su desventaja más marcada al no funcionar con el mismo rendimiento ante distintos tipos de problemas.

Por último, las RR.NN.AA. para procesado temporal y las RR.NN.AA.RR. no son las más utilizadas aunque son las que, a priori, presentan la estructura y funcionamiento más adecuados para resolver problemas con una componente temporal. Las razones fundamentales de su poca utilización son la dificultad de aplicación y entrenamiento.

En general, para cualquiera de las categorías de redes mencionadas en este punto, existen problemas generalizados con el diseño y optimización de la arquitectura. Son muy pocos los trabajos en los que se propone una labor de optimización de la arquitectura simultánea con el proceso de entrenamiento. Esto produce que el tiempo de entrenamiento de las redes sea más largo y que se generen redes ineficientes debido a una arquitectura de un tamaño inadecuado.

3.3 RR.NN.AA.RR. desarrolladas con CE para la Predicción de Series Temporales

Aunque existen varios trabajos que utilizan algún tipo de series temporales para la validación del entrenamiento tanto de nuevas arquitecturas de RR.NN.AA. como de RR.NN.AA. entrenadas mediante técnicas evolutivas, es el artículo de McDonnell y Waagen [MCDO-94] uno de los que más se aproxima a la filosofía de lo realizado en el presente trabajo de invetigación. En este artículo se aplica Programación Evolutiva para determinar el orden de la estructura del perceptrón, así como las etapas de retardo necesarias para su funcionamiento. En cuanto a la generación de nuevos individuos, es un sistema híbrido que mezcla la teoría de PE con otros métodos de búsqueda estocástica intentando así acelerar la generación de soluciones óptimas. El funcionamiento de las redes recurrentes se valida con las ecuaciones del movimiento del péndulo, la serie temporal de manchas solares y con dos ecuaciones de la teoría del caos: Logistic Map y Mackey-Glass.

En este trabajo, se intenta realizar el entrenamiento de un tipo de red recurrente adaptando de forma simultánea la estructura de retardos de la red. También se intenta aumentar el rendimiento de la predicción frente a métodos más tradicionales. No trabaja, sin embargo, en aspectos del desarrollo de la RNA como la preparación del conjunto de datos de entrenamiento o ajuste de parámetros en las distintas neuronas. Tampoco se tratan aspectos computacionales de velocidad de aprendizaje ni paralelización del algoritmo, tan necesarios a la hora de llevar a la práctica la teoría desarrollada.

Capítulo 4: Desarrollo Integral de RR.NN.AA. mediante AA.GG.

4. Desarrollo Integral de RR.NN.AA. mediante CE

4.1 Objetivos

Como ya se ha comentado, la aplicación de las RR.NN.AA. a problemas concretos presenta varias dificultades. El primer problema es la selección entre los distintos tipos de RR.NN.AA. debido a que no existen unos criterios claros que asocien cada tipo de red a un tipo determinado de problema. Este problema, por supuesto, también se presenta en el caso de las RR.NN.AA. temporales en las que existen distintas filosofías, como las de retardos o las recurrentes, y múltiples arquitecturas para cada una de ellas. Además, estas arquitecturas no aportan métodos de optimización de la propia estructura de la red durante el entrenamiento de los pesos; por lo que, aunque el proceso de entrenamiento converja, la red resultante puede no ser óptima para el problema que se intenta resolver. Este inconveniente, común a los primeros modelos de redes, sigue estando vigente en las arquitecturas de redes temporales que no soportan, por lo regular, la optimización de la arquitectura, durante o al terminar el proceso de entrenamiento.

Hasta hace muy poco tiempo, no se ha empezado a trabajar en el ajuste del conjunto de entrenamiento, a partir de los datos disponibles, para realizar el entrenamiento de la red. Las aproximaciones existentes de análisis de componentes principales no se pueden adaptar al problema de las series temporales en las que, si se trabaja con una única variable, el análisis sería aplicable a descubrir qué tramos de la serie son representativos, para utilizar únicamente éstos en el proceso de aprendizaje. Las ventajas de una aproximación de este tipo redundarían en el recorte del tiempo necesario en el entrenamiento, a la vez que en la mejora de la generalización de la RNA.

Gracias a la utilización de los AA.GG. también se ha podido implementar una modificación en el funcionamiento de los elementos de proceso para potenciar su comportamiento en fenómenos dinámicos. Mediante la colocación de una memoria local en las conexiones se intenta emular el comportamiento de las sinapsis en los sistemas naturales, reforzando el papel de las conexiones recurrentes. Este desarrollo intenta acercar más los sistemas artificiales a los naturales a la vez que sirve para

demostrar la potencialidad del entrenamiento evolutivo a la hora de evaluar, de una forma fácil, nuevos desarrollos en el campo de las RR.NN.AA.

Por último, la disponibilidad a bajo precio tanto de hardware, en este caso computadoras personales (PC), como de software que permita la comunicación por red entre distintos programas, favorece el desarrollo de programas paralelos o programas que utilicen los recursos de una forma paralela más o menos directa. En este entorno, se ha conseguido aprovechar el paralelismo a tres niveles diferentes, por un lado el paralelismo que utilizan los AA.GG. al realizar la búsqueda de soluciones mediante una población con un cierto número de individuos; a un segundo nivel, se ha acelerado este proceso al implementarlo distribuido en un cierto número de ordenadores; y a un tercer nivel, se han utilizado diferentes AA.GG. de población paralela en el desarrollo de los distintos aspectos de la construcción de las RR.NN.AA.RR. para predicción temporal.

4.2 Generalización en el diseño de RR.NN.AA.

Desde los comienzos del desarrollo de las RR.NN.AA., los investigadores que han trabajado en este campo han desarrollado un número bastante amplio de tipos de RR.NN.AA. Partiendo del concepto de elementos de proceso simples, la replica de la neurona natural y de una interconexión masiva de estos elementos, se han generado muy distintas implementaciones de lo que puede ser una RNA. Aunque todas ellas mantienen la similitud en estos aspectos de neuronas y conexiones, es muy distinta la típica red alimentada hacia delante de, por ejemplo, una RNA de contra-propagación. También ha habido, y sigue habiendo, muchos avances en el campo de los algoritmos de aprendizaje. Unos referidos a nuevos tipos de aprendizaje y, la mayoría, ajustes menores de algoritmos ya existentes que buscan la tan necesaria mejora en velocidad y complejidad computacional.

La presencia de tal cantidad de posibilidades es, por un lado, una ventaja que permite la experimentación de distintas arquitecturas y tipos de entrenamiento a la hora de enfrentar un problema pero, por otro lado, presenta dos dudas. Una es la necesidad de conocer cuál es la mejor opción que se puede tener para resolver un cierto problema. Este problema, que no está resuelto matemáticamente, implica que no se pueda estar seguro de que la elección que se realice al enfrentar un problema sea la correcta. La otra

duda sería si alguna mejora que esté aún por venir va a suponer una mejora substancial a la hora de resolver los problemas a los que se enfrentan las RR.NN.AA.

Hay que decir, sin embargo, que toda esta variedad de posibilidades de diseño, tanto para la arquitectura como para el proceso de entrenamiento de una RNA, están orientadas a minimizar el nivel de error que se produce entre las salidas conseguidas y las salidas deseadas para unas determinadas entradas. Es por tanto el proceso de optimización de un mecanismo, la RNA en este caso, en el que se ha de dar solución a los parámetros de los elementos de la arquitectura y de las conexiones entre dichos elementos.

En problemas simples, como detección de características en un problema lineal de un entorno conocido, las pruebas de diseño de la red pueden no ser muy conflictivas ni consumir excesivo tiempo. Sin embargo, según se van complicando los problemas con los que han de enfrentarse las deseadas (problemas temporales, no lineales o difusos), el tamaño de las redes o su complejidad impiden realizar una búsqueda mediante prueba y error de las configuraciones más apropiadas.

Se propone aquí la utilización de métodos más generales de realizar la optimización de la RNA en el proceso de aprendizaje como pueden ser los métodos evolutivos, que permiten además la búsqueda en paralelo de las mejores opciones en cada uno de los aspectos del diseño de la red. Además, las posibilidades de búsqueda de estos métodos evolutivos permiten la ampliación de los conceptos de neurona, conexión y RNA al permitir cambios en su estructura con una aplicación inmediata a la resolución de problemas.

4.2.1 Modelo de Neurona

En este sentido, el concepto de neurona no ha sido prácticamente modificado desde la propuesta original de McCulloch y Pitts [MCCU-43]. Todavía se sigue trabajando con elementos de proceso que ponderando una serie de entradas, las integran (función de entrada), la procesan (función de activación) y generan una salida (función de salida) en un cierto instante de tiempo. Las mejoras que se pueden detallar sobre este aspecto de las RR.NN.AA. se refieren al desarrollo de los distintos tipos de funciones de

entrada, de activación y de salida, siendo todas estas funciones compatibles con la mayoría de los distintos desarrollos de RR.NN.AA.

Aparte de los tipos de funciones que utilizan los elementos de proceso, que ya estaban implícitos en alguna medida en las ideas de McCulloch y Pitts, ningún cambio estructural extraordinariamente relevante ha sido propuesto al modelo de neurona comúnmente aceptado. Sin embargo, revisando los aspectos básicos del funcionamiento de una neurona natural, de la que se parte como modelo, existen dos puntos en los que existen claras diferencias con el modelo de neurona de las RR.NN.AA., una es la forma en que se produce y mantiene la activación de la neurona natural y el otro es que la transmisión del impulso eléctrico de una neurona a otra se realiza mediante un proceso químico en el que, una neurona emite neurotransmisores en la sinapsis que son recogidos por la siguiente neurona para continuar el impuso. Todo ello dejando de lado la ya conocida simplificación de considerar al elemento natural como un simple dispositivo binario.

El proceso de activación de una neurona natural no se parece en absoluto a su réplica en las neuronas artificiales. Dentro de este mismo capítulo, en el punto en que se explica la topología con conexiones de activación atenuada, se explica en profundidad el mecanismo de activación de las neuronas naturales y como se intenta emular esto en los trabajos realizados en esta tesis. Las diferencias fundamentales son la existencia de unos umbrales de activación variables que dependen de la frecuencia con que se activa la neurona y la atenuación de la salida en el tiempo que impide activaciones sucesivas muy próximas en el tiempo. Estos dos fenómenos no contemplados en el modelo estándar de neurona limitan el funcionamiento de las réplicas artificiales. En este trabajo, se intentará generalizar el modelo de neurona para acercarlo a la neurona natural buscando siempre una mejora de rendimiento a la vez que la posibilidad de que las RR.NN.AA. puedan almacenar más información al disponer de un mayor número de parámetros.

En la transmisión del impulso eléctrico entre la inmensa mayoría de las neuronas se utiliza un proceso químico en el que una neurona, al recibir la activación por el áxon libera unas substancias, denominadas neurotransmisores (en adelante NT), en el espacio sináptico que existe entre neuronas. En la neurona que recibe el estímulo existen unos receptores situados sobre todo en las dendritas "botones sinápticos" en los que se

reciben los NT específicos. Al integrarse el NT con el receptor se produce una despolarización local de la membrana de la neurona que se desplaza hacia el "cono axonal" mediante un proceso de integración con lo que reciben el resto de las dendritas puede provocar una activación de esa neurona que se transmitirá por el axón neuronal en código de frecuencia, como un tren de estímulos iguales, y producirá, como respuesta de esa neurona a todos los estímulos que recibe, la liberación de una determinada cantidad de NT que va a actuar sobre los receptores de las neuronas con las que está conectada, cerrándose así el ciclo. Este proceso de generación-recepción de NT es bastante complejo, sobre todo porque no se produce de una forma lineal. Existe una cierta cantidad de NT disponibles en las terminaciones del axón que una vez que son liberados necesitan un tiempo para reponerse, de ahí la forma de la activación que se ha comentado anteriormente. Por otro lado existen distintos tipos de NT, cada uno con receptores específicos lo que multiplica las posibilidades a la hora de realizar la transmisión de la activación. Esto ya estaba reflejado de alguna manera en el modelo de McCulloch y Pitts en la implementación de dos tipos de conexiones, excitatorias e inhibitorias, no es un tema muy estudiado que se cree que debería explotarse mucho más.

La utilización de AA.GG. en el proceso de entrenamiento de la red permite empezar a utilizar este tipo de mecanismos de la neurona natural que hasta ahora no se habían tenido en cuenta debido a la dificultad que presenta el entrenamiento de RR.NN.AA. con neuronas complejas o mejoradas. Como se verá en un punto posterior, se ha desarrollado una mejora del modelo de neurona que acerca el proceso de activación al de la neurona natural aportando una funcionalidad nueva aplicable, sobre todo, en problemas temporales. Se intenta demostrar que las técnicas evolutivas son adecuadas para la ampliación del modelo de neurona ya que permiten entrenar RR.NN.AA. con nuevos desarrollos sin necesidad de desarrollar un algoritmo de entrenamiento específico o adaptar uno ya existente. Una vez demostrado esto, se espera abrir una vía para el enriquecimiento del modelo de neurona que permita acercarlo lo más posible al funcionamiento de la neurona natural.

4.2.2 Proceso de Entrenamiento Común

La utilización de la CE en el entrenamiento de las RR.NN.AA. ofrece como mejora fundamental la simplificación del proceso. Se reducen todas las opciones o tipos de algoritmos a un único tipo que se encarga de realizar la optimización, liberando al diseñador de la tediosa labor de prueba de un sinfín de opciones.

Por otro lado, el proceso de minimización por gradiente es sensible a la presencia de mínimos locales, multiplicándose el problema a la hora de trabajar en espacios de búsqueda multidimensionales complejos como es el caso del desarrollo de una RNA. Los nuevos tipos de algoritmos de búsqueda, como las técnicas evolutivas, y más concretamente los AA.GG., disponen de recursos para evitar la presencia de mínimos locales incluso en problemas no lineales. Son, por naturaleza, muy adecuados para este tipo de búsqueda al distribuir de forma automática su funcionamiento entre la exploración de soluciones nuevas y la explotación de las zonas más prometedoras del espacio de búsqueda.

Adicionalmente, las modificaciones en la arquitectura de las neuronas o en la forma de conexión de la RNA no afectan al proceso de entrenamiento que es, de esta forma, independiente. Al realizar el entrenamiento mediante un AG, cualquiera de estas modificaciones es posible optimizarla con el mismo tipo de AG que realiza el entrenamiento de un modelo estándar de RR.NN.AA. de propagación hacia delante.

4.2.3 Motivación General

Se busca, con la utilización de las técnicas de CE, la generalización de los modelos de RR.NN.AA. y de neurona artificial. Es necesario desarrollar modelos que se adapten por sí mismos al problema, y no que obliguen al diseñador de la RNA a la búsqueda a través de un número cada vez mayor de tipos de neuronas y arquitecturas. Las técnicas de CE pueden aportar estas dos características, pueden trabajar con distintos tipos de arquitecturas para realizar las pruebas de forma automática y pueden ayudar a la generalización en el campo de las RR.NN.AA., ya que hacen transparente al diseñador los tipos de arquitecturas hoy existentes para trabajar únicamente con RR.NN.AA., sin pensar en cómo realizar su organización interna.

El otro punto importante de la utilización de la CE es que aporta una mayor flexibilidad a la utilización de RR.NN.AA. Como se ha visto en los puntos previos, el entrenamiento con AA.GG. permite incorporar un gran número de parámetros, tanto a las neuronas como a las conexiones o a la estructura de la red, sin afectar al proceso de entrenamiento en sí. Siguiendo las teorías Darwinianas se puede dejar que la evolución configure los parámetros implementados de forma que se consigan las mejores RR.NN.AA.

4.3 Entrenamiento Básico de RR.NN.AA. mediante AA.GG.

4.3.1 Introducción

El primer paso para la construcción del sistema de desarrollo cooperativo de RR.NN.AA. optimizadas mediante AA.GG. es el entrenamiento básico de RR.NN.AA. Para ello, se necesita un AG que calcule los pesos de una RNA; es decir, que realice la fase de entrenamiento de la red. Aunque, como ya se ha comentado, este punto ya está siendo tratado en las investigaciones de RR.NN.AA.-AA.GG. desde hace unos años, es necesario para cualquier sistema complejo partir de este objetivo más básico. La necesidad de este desarrollo se ha aprovechado para integrar características novedosas que permitan automatizar al máximo el diseño de la RNA. Para ello, se ha introducido en el sistema la selección automática de la función de activación para cada neurona de la red, ya que esta selección es normalmente la siguiente preocupación del diseñador, después de la selección de la arquitectura de la RNA.

4.3.2 Objetivo

En este primer paso, se va a desarrollar un sistema que, a partir del fichero de entrenamiento del problema y de una arquitectura fija de RNA que se seleccione en cuanto a número de capas y neuronas por capa, construya una RNA para resolver dicho problema. Este sistema ajustará los pesos de las conexiones y escogerá la función de activación ideal para cada neurona de la red.

4.3.3 Características de las RR.NN.AA.

El sistema desarrollado construirá RR.NN.AA. alimentadas hacia delante y recurrentes. En el caso de redes recurrentes, se podrá especificar el número de entradas relacionadas en el tiempo; es decir, cuántas entradas consecutivas del fichero de entrenamiento están relacionadas. Se busca con esta opción poder entrenar a la red con un conjunto de casos independientes, incluso al trabajar con series temporales.

Con respecto a las neuronas, se especifica el tipo de función de activación entre: lineal, umbral, hiperbólica tangente y exponencial. Si se escoge la activación lineal hay que especificar un valor máximo y la pendiente y; si se escoge la función umbral se necesita especificar el valor de umbral. Para los pesos de conexión también se puede configurar un valor máximo.

El sistema permite configurar manualmente la arquitectura de la RNA. En el caso de redes alimentadas hacia delante, se puede especificar el número de capas ocultas y el número de neuronas por capa. En el caso de RR.NN.AA.RR. únicamente es necesario especificar el número de neuronas ya que las conexiones recurrentes hacen desaparecer el concepto de capas.

Si la RNA es del tipo recurrente, el proceso de evaluación de las RR.NN.AA. generadas por el AG puede realizarse de forma continua, sin re-inicializar los pesos de la red después de cada ejemplo, o discreta haciendo la re-inicialización y asumiendo independencia entre los ejemplos. En el caso de series temporales, por su propia esencia, el modo de entrenamiento ha de ser continuo.

Por último, es posible configurar en las RR.NN.AA.RR., el número de ciclos o iteraciones que la red propagará internamente los estados de activación sin recibir una nueva entrada. El número de iteraciones en redes no recurrentes es 1. Sin embargo, en redes recurrentes es posible distribuir un estado de activación a toda la red permitiendo que la red propague sus activaciones un cierto número de ciclos antes de la llegada de la siguiente entrada del conjunto de entrenamiento.

4.3.4 Diseño del AG

4.3.4.1 Codificación de la RNA

Como ya se ha comentado en la teoría sobre AA.GG., es fundamental una buena codificación de las soluciones para conseguir el buen funcionamiento del AG. El problema que se plantea en este punto es optimizar el conjunto de pesos de una RNA para conseguir que resuelva un cierto problema. Por tanto, los genes de los individuos vendrán representados por un conjunto de pesos de conexión. Como se acaba de comentar, se van a combinar otros parámetros de la RNA en el entrenamiento de los pesos. Estos parámetros son los relacionados con la fase de activación de la neurona, trabajando en el sentido de evitar la selección de la función de activación o los parámetros asociados a ella.

Un punto importante a la hora de realizar la codificación es que el sistema va a poder desarrollar RR.NN.AA. tanto alimentadas hacia delante como recurrentes. Debido a esto, es necesario escoger una representación que sea capaz de soportar los dos tipos de redes para poder aplicar el mismo tipo de operadores genéticos en ambos tipos de redes.

Siguiendo esta idea de la representación común, se representa cada RNA por una matriz NxN donde N es el número total de neuronas de la RNA con independencia de su capa de procedencia. Esta matriz contiene los valores de los pesos de conexión entre las neuronas, de forma que la celda en la posición (i, j) será el peso de conexión entre la salida de la neurona i y la entrada de la neurona j. Si dos neuronas (j, k) no tienen conexión directa, el valor en la matriz de la posición (j, k) es cero.

Nº	1	2	3	4
1		0,2	0,3	
2			0,3	0,6
3		0,1		0,8
4			0,5	

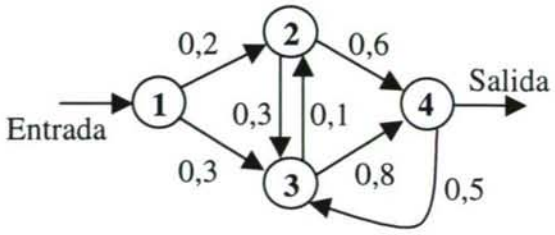


Figura 4.1.- Matriz de Pesos y RNA que representa

En el caso de una RNA alimentada hacia delante, esta matriz de pesos siempre es triangular superior ya que no hay conexiones con neuronas de capas anteriores o de la misma capa. En el caso de RR.NN.AA.RR., como en la Figura 4.1, se utiliza toda la matriz de conexiones para describir las conexiones hacia delante y las recurrencias.

Para las funciones de activación únicamente se hace necesario una matriz con tantas posiciones como neuronas (ver Figura 4.2) para especificar el tipo de función de activación que va a utilizar (lineal L, exponencial E, hiperbólica tangente H y umbral U) y para almacenar los parámetros referidos a las función de activación lineal (la pendiente de la recta) y umbral (el valor del umbral).

	Tipo	Valor
1	L	0,7
2	E	
3	U	0,5
4	H	

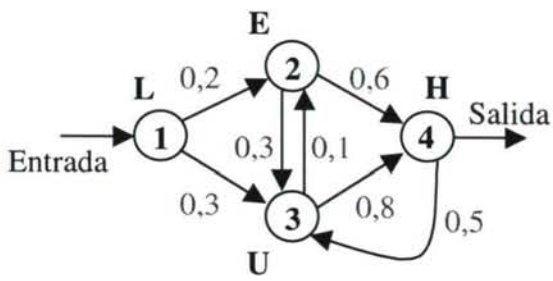


Figura 4.2.- Matriz de Funciones de Activación y RNA que representa

4.3.4.2 Cruce

Como se ha visto en el punto de codificación, cada individuo no se ha podido codificar como una única cadena de características o genes, sino que cada individuo consta de dos matrices. Esto implica modificar este operador para que, en cada operación de cruce, se mezclen todas las características (las dos matrices) de los progenitores en la descendencia.

Debido a esta especial configuración, el operador de cruce se podría dividir en dos operaciones simples, una sobre la matriz de conexiones y otra sobre la matriz de funciones de activación.

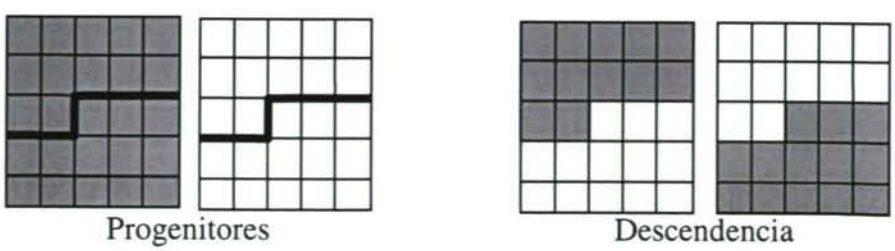


Figura 4.3.- Operador de Cruce para las Matrices de Pesos

El cruce sobre la matriz de conexiones también se puede considerar especial ya que se está tratando con una estructura de dos dimensiones mientras que el operador de cruce estándar se aplica a individuos representados por una matriz unidimensional. En casos como estos hay que adaptar el operador de cruce de forma que realice su tarea sobre una estructura nueva. Una opción posible para aplicar el operador de cruce sobre una matriz sería escoger una celda de la matriz y considerarla el punto de cruce (ver Figura 4.3). Las filas anteriores y las posiciones anteriores de esa fila, desde ese punto, serían para un individuo de la descendencia mientras que, las filas posteriores y las posiciones posteriores dentro de la misma fila serían para el otro individuo de la descendencia. El segundo progenitor pasaría la información de su matriz en orden inverso, la primera parte al segundo descendiente y la segunda parte al primero. También sería posible escoger una fila como punto de corte como caso más general.

Nº	1	2	3	4
1		0,2	0,3	
2				0,6
3				0,8
4				

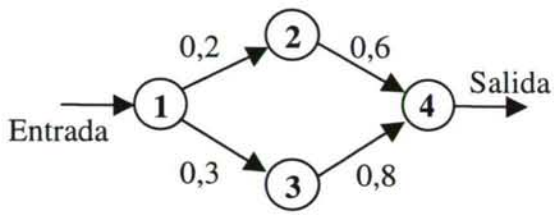


Figura 4.4.- Ejemplo de Matriz de Pesos para Redes no Recurrentes

Las dos opciones anteriores, por filas o por posición, hacen una división horizontal de las matrices a cruzar. Sin embargo, como se puede ver en la Figura 4.4, en el caso de la matriz de pesos de redes no recurrentes, los valores se agrupan en la parte superior de la matriz debido a la falta de conexiones recurrentes y a que las conexiones con la primera capa oculta suelen ser las más numerosas.

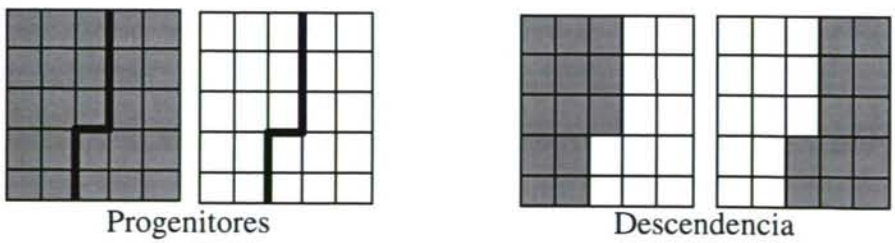


Figura 4.5.- Operador de Cruce Vertical para las Matrices de Pesos

Debido a esto, para realizar un cruce más homogéneo se divide la matriz por columnas (ver Figura 4.5). En primer lugar se selecciona al azar una de las columnas como punto de cruce, esta columna y las anteriores pasarán a formar parte del primer

descendiente mientras que las columnas posteriores pasarán a formar parte del segundo descendiente. Con el segundo progenitor se realiza la misma división pero las columnas se pasan a la descendencia en orden inverso.

El operador de cruce para la matriz de las funciones de activación no presenta tantos problemas ya que se vuelve a trabajar con matrices unidimensionales. En este caso se escoge un punto de cruce que va a ser común para dividir tanto la columna que contiene los tipos de función de activación como la que contiene los valores para su aplicación. Esto es así debido a la relación estrecha de los valores de ambas matrices. Una vez escogido el punto de cruce el operador no presenta modificaciones al cruce estándar.

4.3.4.3 Mutación

El operador de mutación también ha tenido que ser modificado para aplicarlo a la especial configuración del problema del entrenamiento de RR.NN.AA. En este caso, el operador de mutación escoge, en primer lugar, el individuo que se va a mutar. A continuación, se selecciona cual de las características se va a mutar, un peso o un tipo de función de activación. Si se escoge mutar un peso, se selecciona uno de ellos al azar y se genera un nuevo valor de peso, también al azar, dentro de los límites especificados. Si se escoge mutar una función de activación, se selecciona una de las neuronas de la red y una función de activación para ella. Si es necesario se genera el valor asociado (pendiente o umbral).

4.3.5 Consideraciones del Entrenamiento Básico

En este punto se ha desarrollado la primera parte del sistema cooperativo. Esta parte, el entrenamiento de RR.NN.AA. mediante AA.GG., es la que se va a utilizar como evaluadora de individuos de los dos subsistemas principales, el que se encarga de seleccionar el conjunto de entrenamiento y el que se encarga de optimizar la arquitectura de la red. Es, por tanto, este primer sistema muy importante en el funcionamiento del sistema global.

Como se puede ver en el capítulo de resultados, los rendimientos que se obtienen con la selección automática de las funciones de activación no son los deseados. El

rendimiento obtenido es peor que utilizando funciones de activación fijas pero, como se demuestra en las pruebas adicionales realizadas, no se debe al hecho de utilizar distintas funciones de activación sino al comportamiento de las propias funciones de activación en el proceso de entrenamiento de las redes. No por esto se abandona el desarrollo de una técnica de selección automática de funciones de activación, sino que se pasa su implementación al punto posterior en el que se ajustan las arquitecturas de las RR.NN.AA., donde se afronta el mismo problema desde una perspectiva distinta.

A continuación, y antes de seguir con la automatización del desarrollo de las RR.NN.AA.RR. en cuanto a conjuntos de entrenamiento y arquitecturas, se estudian los parámetros del AG más convenientes para la resolución del entrenamiento de RR.NN.AA.

4.4 Estudio de Parámetros del AG

En un primer momento, para desarrollar el sistema básico de entrenamiento de RR.NN.AA. mediante AA.GG., se han utilizado las especificaciones del AG estándar formulado por John Holland [HOLL-75]. No se han utilizado en esta primera aproximación las distintas opciones que existen para los operadores de cruce y mutación, ni para las operaciones selección y reemplazo de individuos en cada generación.

El siguiente paso, antes de comenzar la extensión del modelo de RR.NN.AA., es implementar las distintas opciones de funcionamiento del AG y ajustar los parámetros de estas opciones para obtener los mejores resultados en el entrenamiento de las RR.NN.AA. En este sentido, se han seleccionado para su implementación las opciones que más influencia tienen en la mejora de rendimiento del AG según las tendencias del área reflejadas en la bibliografía. Se busca, de esta forma, evaluar el comportamiento de estas opciones en el entrenamiento de RR.NN.AA.

Las opciones que se han implementado y probado en el simulador de AA.GG. para el entrenamiento de RR.NN.AA. se centran en cinco temas.

1. Experimentar el efecto que el tamaño de la población tiene en el rendimiento del AG.

2. La prueba de distintas aproximaciones para la selección de los individuos que se van a utilizar en la operación de cruce.

3. La utilización de las distintas modalidades de cruce.

4. Las distintas técnicas de selección de los individuos que son sustituidos por los nuevos individuos generados en las operaciones de cruce.

5. Acelerar el proceso de convergencia final del AG mediante la implementación de mecanismos de ajuste fino y de control de la diversidad genética.

Para tener una explicación detallada de los parámetros y opciones empleados en este punto se pueden consultar, en el capítulo de Fundamentos, los apartados que tratan sobre AA.GG.

4.4.1 Población

Uno de los parámetros fundamentales del funcionamiento de un AG es el tamaño de la población. El tamaño de la población influye en el grado de diversidad que poseen los distintos cromosomas al inicio de la simulación del AG. Un tamaño reducido de población provoca que los cromosomas, en su conjunto, no dispongan de la variedad de valores que garantice que la recombinación de individuos pueda encontrar una solución óptima y, por otro lado, si el AG encuentra un mínimo local puede provocar que la población converja a ese mínimo imposibilitando la localización de mejores soluciones. Si el tamaño de la población es excesivamente grande, las necesidades de memoria para la población, así como las tareas de búsqueda, sustitución y reordenación de individuos, harán que el rendimiento del AG se reduzca notablemente.

Debido a la importancia de este parámetro para el funcionamiento del AG se han realizado pruebas con distintos valores de número de individuos en la población. El valor mínimo, teniendo en cuenta que se van a utilizar individuos con codificación de números reales, se establece en 100 individuos. A partir de aquí, se aumenta el número de individuos de forma exponencial.

La limitación superior impuesta en las simulaciones para este parámetro es que el tamaño de la población no afecte de forma apreciable en el rendimiento de los ordenadores utilizados. La carga computacional que se realiza en la búsqueda de

progenitores para realizar el cruce, en la sustitución de individuos por la nueva descendencia y en la recolocación de la descendencia en la población de acuerdo con su ajuste, crece de forma exponencial al aumentar el tamaño de la población de forma lineal. Debido a esto, existe un límite marcado tanto por la potencia del ordenador como por la memoria RAM instalada, que impiden superar un cierto tamaño de población en cada problema concreto para no ralentizar de forma dramática la simulación del AG.

4.4.2 Operadores de Cruce y Mutación

En el simulador básico de entrenamiento de RR.NN.AA. mediante AA.GG. comentado en el punto 4.3, se utilizó el modelo más simple de los operadores de cruce que trabaja con un único punto de corte y mutación. Además, en este caso se realizaba una sola operación de cruce y mutación por generación o paso de la simulación. Con objeto de mejorar el rendimiento del AG, se ha realizado un estudio sobre estos parámetros en dos sentidos, por un lado, viendo cómo afecta el número de cruces y mutaciones al proceso de convergencia y, por otro lado, probando distintos tipos de operadores de cruce buscando el tipo más adecuado al problema de diseño de RR.NN.AA.

Junto con las variaciones en el número de individuos de la población se van a probar distintas combinaciones de número de cruces y número de mutaciones por generación. Estos parámetros están muy relacionados ya que la diversidad inicial depende del tamaño de la población y los operadores de cruce y mutación están trabajando a partir esa diversidad. Es por esto que se necesita realizar la prueba de los tres parámetros de forma conjunta.

Se han utilizado valores de 1 y 5 para el número de cruces y mutaciones por generación, realizando pruebas con combinaciones de estos dos valores para cada valor de tamaño de la población.

En cuanto a los operadores de cruce, se han probado dos tipos diferentes, el operador de cruce de dos puntos, así como el de cruce uniforme. El operador de cruce de dos puntos se implementa para comprobar si la reordenación de genes o inversión, es útil en este caso por las características de no linealidad del problema. Además, la modificación del operador de cruce para realizar el cruce de dos puntos es una forma

más fácil de simular el mecanismo el inversión que la reordenación de los genes dentro de los cromosomas de los individuos. También se han realizado pruebas con el cruce uniforme que escoge de forma aleatoria los genes que transmitirá cada progenitor a su descendencia.

4.4.3 Selección

Otro punto importante en el funcionamiento de un AG es la selección de individuos que van a ser utilizados en las operaciones de cruce. Esta labor es tanto más delicada en cuanto el espacio de soluciones es no-lineal ya que, en estos casos, la valoración de un individuo no es una función lineal de la valoración de sus progenitores. En la práctica esto quiere decir que la selección de los mejores individuos para realizar los cruces entre ellos no generará necesariamente individuos mejores.

El problema que se afronta en esta tesis del entrenamiento de RR.NN.AA. es claramente un problema no-lineal, en el que, además, hay que configurar un número elevado de parámetros. Es obvio que es necesario buscar las ventajas que para la resolución de este tipo de problemas aporten la técnica de los AAGG.

Para afrontar estos problemas se han utilizado tres tipos distintos de selección de individuos, la "puramente aleatoria" entre los individuos de la población; la técnica de "Montecarlo", que basa la probabilidad de selección en la adaptación de los individuos; y la de "torneo entre individuos" que organiza una competición entre individuos seleccionados al azar para ganar un puesto en la operación de cruce.

4.4.4 Reemplazo

En la primera aproximación de sistema se utilizó la opción más habitual, la Darwiniana, en la que los individuos menos adaptados; es decir, aquellos que consiguen una valoración peor sean los candidatos a dejar su sitio a la nueva descendencia. Esta aproximación tiene como problema que puede provocar una rápida homogenización de la población haciendo que el AG caiga en mínimos locales que estanquen el desarrollo de la población.

Para evitar este problema existen distintas estrategias de reemplazo de progenitores que han sido probadas implementándolas en el sistema. A parte de la

sustitución de los menos adaptados, se ha probado la sustitución de los padres, en la que la descendencia sustituye a los padres sólo si su nivel de adaptación es superior y la sustitución de individuos con parecido o igual error donde, una vez valorado un nuevo individuo, se busca en la población individuos con error igual o similar y se escoge uno de ellos al azar para que deje su sitio al nuevo individuo.

4.4.5 Consideraciones al Ajuste de Parámetros del AG

Como se puede ver en los resultados que se plasman en el capítulo siguiente, el mejor rendimiento se obtiene utilizando el tipo de selección aleatoria, el cruce de un solo punto de corte y la sustitución darwiniana o de los menos adaptados. Estos valores de parámetros óptimos son los que se utilizarán en el AG de entrenamiento de RR.NN.AA. en los puntos siguientes, como los más indicados para el proceso de entrenamiento.

Todos los valores de parámetros conseguidos son consecuentes con el problema a resolver. Es decir, el entrenamiento de RR.NN.AA. es un problema complejo y no lineal que se mueve en un espacio de soluciones muy grande.

En cuanto a la selección de individuos, ni la técnica de Montecarlo ni la de torneo aportan mejoras a la habitual técnica de selección aleatoria, lo que se debe a que el problema de entrenamiento de RR.NN.AA.RR. es un problema no lineal en el que la combinación de individuos de adaptación alta no garantiza conseguir individuos mejores.

Los mejores resultados en cuanto a número de cruces y mutaciones, se han obtenido con 5 cruces y 5 mutaciones, no existiendo una diferencia excesiva con otros valores de estos mismos parámetros. Esto sugiere la posibilidad de que la diversidad de la población sea suficiente cuando se trabaja con más de 100 individuos y que el número de cruces y mutaciones no es un parámetro que influya demasiado en el rendimiento.

En el parámetro de tipo de cruce, el cruce uniforme es el que ofrece peores resultados con mucha diferencia, mientras que los resultados de cruce con un punto y con dos son muy similares. Esto parece indicar que existe una fuerte relación entre los genes de los individuos del AG, lo que provoca el mal rendimiento del cruce uniforme.

También parece indicar que debido a la codificación no es necesario aplicar el operador de inversión para variar el orden de los genes en el cromosoma de los individuos ya que el cruce con dos puntos de corte no aporta una mejora significativa en el rendimiento.

Para el operador de sustitución de individuos, la sustitución darwiniana es la que consigue mejores resultados, siendo la siguiente peor la de sustitución de padres y, por último, la de parecidos que además consume más tiempo debido a que es necesario buscar los individuos parecidos por la población.

4.5 Automatización del Desarrollo de la Topología

4.5.1 Introducción

Como se puede ver en la documentación existen precedentes de diseño de arquitecturas de RR.NN.AA. utilizando AA.GG. siendo éste el paso lógico a seguir después de conseguir el entrenamiento de RR.NN.AA. mediante AA.GG. El diseño de la arquitectura es un punto muy importante dentro del desarrollo de una RNA y es un aspecto donde se puede conseguir muy poca ayuda de la teoría de RR.NN.AA.

En el sistema que se propone se optimiza la arquitectura de la red mediante un AG. Es decir, se ajustará el número de capas ocultas y de neuronas por capa en redes alimentadas hacia delante y el número de neuronas ocultas en redes recurrentes. De forma concurrente a la búsqueda de la arquitectura óptima, también se realiza el estudio de la conectividad de las neuronas de la red; es decir, se va a comprobar la utilidad de las conexiones entre las neuronas de forma que sólo permanezcan aquellas conexiones que mejoran el comportamiento de la red.

Además, se recupera en esta fase la selección automática de las funciones de activación para las neuronas de la red. La selección se va a realizar de forma semejante a como se planteó en el punto 4.3 pero separándola del proceso de entrenamiento de la red. Esta independencia es la que permitirá evitar los problemas que se plantearon en la primera aproximación.

En este punto, el sistema va a seguir trabajando tanto con RR.NN.AA. alimentadas hacia delante como con redes recurrentes por lo que se intenta dar una

perspectiva común a la optimización de arquitecturas y conectividad para las dos aproximaciones de RR.NN.AA. Para la búsqueda de estas optimizaciones se va a utilizar un AG que ha de ser modificado de forma apropiada para que realice las tres tareas de forma concurrente.

4.5.2 Modelo de RNA

En este punto de desarrollo del sistema, se introduce como novedad en el sistema el podaje adaptativo o selección de conexiones entre las neuronas. Este podaje se realiza de forma concurrente con el ajuste de la arquitectura de cada problema concreto.

El sistema, para las RR.NN.AA. alimentadas hacia delante va a estudiar distintas configuraciones de capas ocultas y de neuronas en estas capas. El AG que realiza esta tarea utiliza, por un lado, individuos de longitud variable para codificar la estructura de capas y EP y, por otro lado, individuos de longitud fija para codificar las matrices de conexión. Obviamente, las matrices de conexión son triangulares superiores en el caso de RR.NN.AA. de alimentación hacia delante pero se almacena toda la información para mantener la compatibilidad con las RR.NN.AA.RR.

En cuanto a las RR.NN.AA.RR. no existe el concepto de capas por lo que la arquitectura consiste en calcular el número óptimo de neuronas ocultas. Este valor se codifica con individuos de longitud fija de una posición e individuos de longitud fija para la matriz de conexiones.

Para realizar las pruebas de adaptación de cada arquitectura, tanto recurrente como no recurrente, se utilizará el sistema de entrenamiento de RR.NN.AA. mediante AA.GG. comentado en un punto anterior de este capítulo.

Para el funcionamiento de este nuevo esquema se define un elemento adicional en la estructura de la RNA. Además del ajuste del número de capas y de neuronas se necesita una matriz de conexiones que indique qué conexiones entre neuronas son válidas.

Hay que tener en cuenta que la presencia de la matriz de conexión cambia la forma de evaluación de las RR.NN.AA. con lo que es necesario modificar el módulo de

entrenamiento de RR.NN.AA. mediante AA.GG. Se dispone así de una nueva funcionalidad que no era posible configurar sin la utilización de AA.GG. El cambio de este módulo afecta tanto al proceso de cálculo de pesos, ya que es necesario calcular las modifiiones de un número menor de conexiones, como a la evaluación de las RR.NN.AA., ya que el cálculo de las entradas de cada EP tiene que tener en cuenta la matriz de conexiones asociada a cada red.

4.5.3 Diseño del AG

El diseño del AG desarrollado para el ajuste de las arquitecturas de las RR.NN.AA. presenta marcadas diferencias con el AG estándar debido a la especial configuración del problema a resolver. En primer lugar, la información codificada sobre las RR.NN.AA., su arquitectura y conectividad, no es de longitud fija, sino que depende del tipo de red, por lo que es necesario utilizar un AG con individuos de longitud variable. En segundo lugar, hay que tener en cuenta que se intentan solucionar tres problemas de forma simultánea y concurrente, el ajuste de: el número de capas/neuronas, la conectividad y las funciones de activación. Esto provoca modificaciones en los operadores de cruce y mutación para que trabajen con individuos con tres cadenas de características siendo lo habitual una única cadena. En tercer lugar la valoración de los individuos que son, en este caso, arquitecturas de RR.NN.AA., se ha de realizar enfrentando a estas RR.NN.AA. con el problema que tienen que resolver. Pero, para realizar esto, primero es necesario entrenarlas. En el entrenamiento de las RR.NN.AA. se utiliza el módulo de entrenamiento de RR.NN.AA. con AA.GG. referido al principio de este capítulo. El valor de error que genere la red en este simulador será el principal indicador del ajuste de la arquitectura.

4.5.3.1 Codificación

4.5.3.1.1 Arquitecturas

Como se ha venido haciendo en este punto, es necesario distinguir entre RR.NN.AA. alimentadas hacia delante y RR.NN.AA.RR. En el caso de las RR.NN.AA. alimentadas hacia delante es necesario configurar el número de capas ocultas y el número de EP por capa. Para ello, el AG codifica los individuos con una cadena donde cada posición representa el número de neuronas de una capa oculta. Obviamente, estos

individuos serán de longitud variable ya que existirán individuos con distinto número de capas ocultas y se configura un valor máximo de capas ocultas que limite el desarrollo de RR.NN.AA. de tamaño desproporcionado.

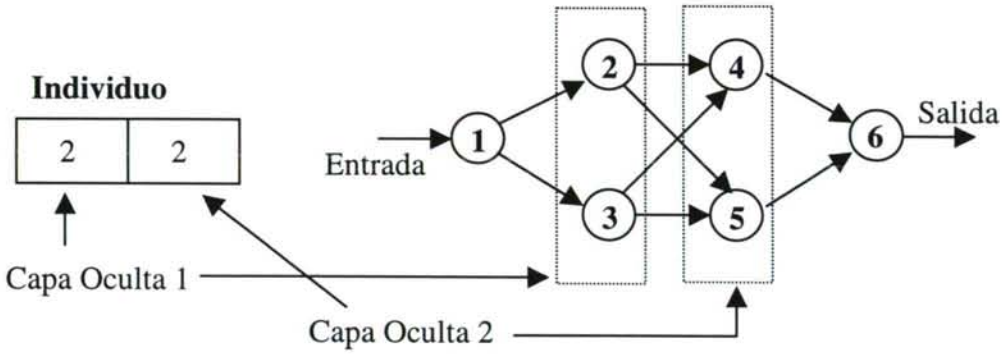


Figura 4.6.- Codificación de la estructura de capas y neuronas en una RNA no recurrente

En el caso de RR.NN.AA.RR. los individuos constan solamente de una posición con el número de neuronas ocultas. Obviamente, no es posible desarrollar un AG que trabaje con individuos de una única posición, esta parte se complementa con la parte de la matriz de conexión. Junto con ella definen individuos de un tamaño mayor sobre los que sí que se pueden aplicar los operadores genéticos.

4.5.3.1.2 Conectividad

Esta parte es común para los dos tipos de RR.NN.AA. Existirá una única matriz de conectividad entre neuronas de cada RNA. Esta matriz indica qué conexiones van a ser ajustadas en la fase de entrenamiento. Debido a la presencia de la matriz de conexiones, en la matriz de pesos se almacenan pesos en todas sus posiciones sabiendo que sólo los que permita la matriz de conexiones serán ajustados. De esta forma, los individuos pasarán a su descendencia información genética que puede ser que no vayan a utilizar, aumentando así la diversidad genética de la población.

En el caso de las RR.NN.AA. alimentadas hacia delante se considera una matriz triangular superior. Para aumentar las posibilidades de este tipo de redes, se aprovecha la existencia de esta matriz para eliminar la restricción de que las conexiones existan sólo entre neuronas de capas consecutivas y se permiten conexiones entre neuronas de distintas capas siempre que las conexiones sean hacia neuronas de capas posteriores.

Como se puede ver en el ejemplo de la Figura 4.7, si en la casilla (1,6) y la (3,6) de la matriz de conexión hubiera un 0 y la (2,5) un 1, se estaría ante la típica

configuración de pesos de una RNA alimentada hacia delante. Al no darse este caso, la red sigue siendo alimentada hacia delante pero hay conexiones entre capas no consecutivas. Las conexiones que están atenuadas en la figura, no son posibles debido a que se está trabajando con una red alimentada hacia delante y supondrían conexiones recurrentes. Por último, de las conexiones que no tienen sus valores atenuados se permiten todas las combinaciones posibles, dejando que el propio AG elimine las arquitecturas inviables que no tengan conexiones con las neuronas de salida o desde las de entrada.

Matriz de Pesos

Nº	1	2	3	4	5	6
1	0,6	0,2	0,3	0,4	0,5	0,4
2	0,5	0,5	0,9	0,8	0,7	0,1
3	0,5	0,3	0,2	0,6	0,3	0,2
4	0,7	0,1	0,5	0,6	0,1	0,5
5	0,2	0,3	0,4	0,9	0,6	0,2
6	0,5	0,9	0,6	0,2	0,8	0,5

Matriz de Conexiones

Nº	1	2	3	4	5	6
1	0	1	1	0	0	1
2	0	0	0	1	0	0
3	0	0	0	1	1	1
4	0	0	0	0	0	1
5	0	0	0	0	0	1
6	0	0	0	0	0	0

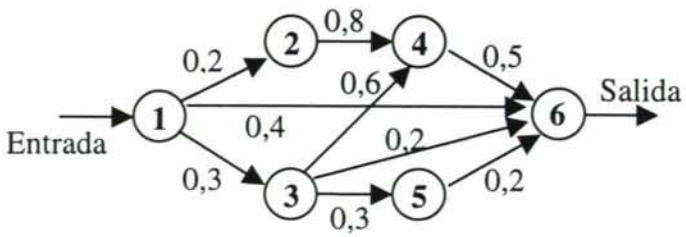


Figura 4.7.- Conectividad en una RNA alimentada hacia delante

En el caso de RR.NN.AA.RR. se utiliza la matriz completa para especificar la conectividad. En este caso no hay restricciones de conectividad por lo que no hay conexiones con valores atenuados. Aquí, la matriz de conexiones permite seleccionar aquellas conexiones significativas, hacia neuronas anteriores, posteriores o hacia si mismas.

Matriz de Pesos

Nº	1	2	3	4
1	0,5	0,2	0,3	0,2
2	0,4	0,7	0,3	0,6
3	0,7	0,1	0,3	0,8
4	0,2	0,9	0,5	0,1

Matriz de Conexiones

Nº	1	2	3	4
1	0	1	1	0
2	0	1	1	1
3	0	1	0	1
4	0	0	1	0

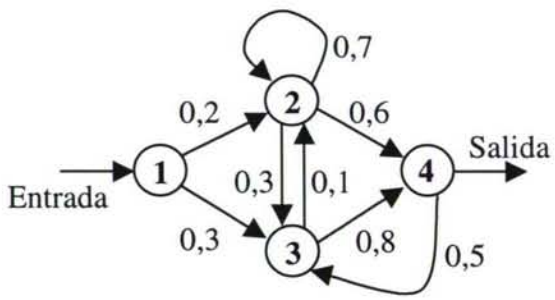


Figura 4.8.- Conectividad en una RNA recurrente

Esta matriz tendrá dimensión NxN siendo N el número máximo de neuronas que se permiten en una red, no el número de neuronas de la arquitectura concreta que se estudia. En el caso de que el número de neuronas de una red en concreto sea M, menor que N, se utilizará en la evaluación de la RNA una submatriz de tamaño MxM con M^2 posiciones. La información que no se utiliza en un cierto caso se mantiene en el individuo y puede volverse a utilizar en el caso de cambiar la configuración de la arquitectura del individuo. De esta forma, cada individuo mantiene un conjunto de información en su código genético que no se está manifestando en su comportamiento pero que puede trasladar a su descendencia emulando, en cierta forma, situaciones ya descritas en la genética tradicional.

4.5.3.1.3 Funciones de Activación

La implementación de la selección automática de las funciones de activación no varía mucho de la comentada en el punto 4.3. Sin embargo, la filosofía de funcionamiento es totalmente distinta. En este caso, cada arquitectura es la que lleva asociada una matriz con las funciones de activación para cada neurona. La matriz es la ya comentada y almacena, para cada neurona, dos parámetros, tipo de función y valor (ver Figura 4.2). Al ser la arquitectura la que lleva asociada las funciones de activación, éstas van a ser fijas para la simulación que se realiza en el módulo de entrenamiento de RR.NN.AA. mediante AA.GG.

En el punto 4.3, el comportamiento conseguido no fue el deseado ya que las funciones de activación eran distintas para cada individuo de la población durante el entrenamiento de una única arquitectura. En este caso, las funciones de activación que se comportaban mejor en el inicio del entrenamiento se extendían por toda la población haciendo caer al AG en un mínimo local.

En este punto, la selección de las funciones de activación se ha elevado un nivel de abstracción, asociándolo a la selección de la arquitectura. De esta forma, una determinada configuración de funciones de activación se va a estudiar en un cierto número de generaciones de simulación aplicada a una RNA. Así, se va a poder conseguir un valor fiable del grado de ajuste de esa determinada configuración de funciones de activación.

4.5.3.2 Evaluación

La evaluación en el AG desarrollado en este punto implica en entrenamiento de una RNA que cumpla las restricciones de arquitectura y conexiones de cada individuo de la población para comprobar el nivel de error que se produce frente a un problema concreto. El AG de este nivel ofrece como soluciones la arquitectura y las conexiones que se utilizan como entrada del sistema RNAAG de entrenamiento de RR.NN.AA. mediante AA.GG. Aquí, se entrenan RR.NN.AA. que ya son capaces de resolver un problema. El nivel de error que se ofrece en la resolución de este problema es lo que se va a utilizar como medida de ajuste en el AG de desarrollo de arquitecturas.

En el sistema RNAAG, además de las entradas de arquitectura y conexiones, se configura un número de iteraciones o generaciones fijo de simulación, para que las evaluaciones de todas las arquitecturas sean comparables. Una vez llegado a ese número de iteraciones, el error cometido por la RNA se considera como representativo del tipo de arquitectura que representa y pasa a ser el nivel de ajuste del individuo que representa esa arquitectura.

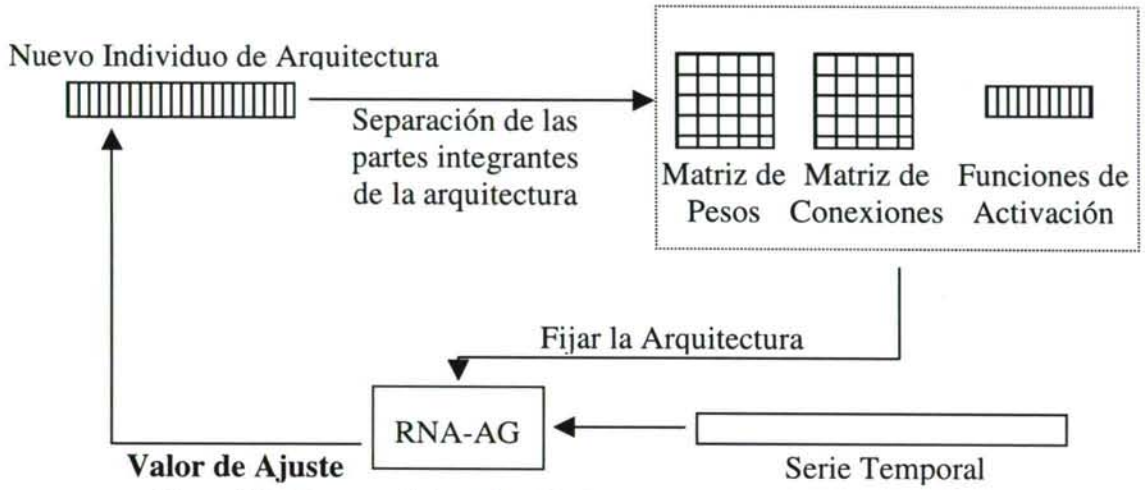


Figura 4.9.- Esquema de la evaluación de una arquitectura concreta de RNA

Hay que tener en cuenta que en la valoración de las arquitecturas no se necesita que las RR.NN.AA. entrenadas alcancen un nivel de error mínimo, el entrenamiento de las RR.NN.AA. se utiliza para dar una idea del ajuste de una determinada arquitectura, por lo que es más importante fijar una meta finita como puede ser el número de ciclos de simulación o un período de tiempo. No sería operativo fijar un nivel de error como meta ya que podría darse el caso de que una arquitectura no lo alcanzase paralizando el proceso de evaluación de individuos.

4.5.3.3 Cruce

En el sistema que se discute en este punto, los individuos están compuestos de tres partes. Estas tres partes se codifican en una cadena de longitud fija de la que, dependiendo del individuo concreto no se utiliza en su totalidad. Además, cada parte se gestiona de forma independiente ya que simbolizan conceptos distintos: número de capas/neuronas, conexiones o funciones de activación.

Debido a esto, la operación de cruce supone que estas tres partes son independientes, por lo que se generan tres puntos de corte para el cromosoma con el

objeto de que cada descendiente herede parte de la información genética de cada progenitor en cada uno de los tres conceptos de la arquitectura.

Hay que tener en cuenta que, como los individuos pueden no utilizar toda la información de su genoma, al realizar un cruce se pueden conseguir individuos idénticos en su comportamiento pero que hereden distinta información que no están utilizando. Su comportamiento en cuanto a su nivel de ajuste puede ser el mismo, pero la descendencia que generen dos individuos de igual comportamiento puede ser distinta.

Matriz de Pesos

Nº	1	2	3	4
1	0,5	0,2	0,3	0,2
2	0,4	0,7	0,3	0,6
3	0,7	0,1	0,3	0,8
4	0,2	0,9	0,5	0,1

Matriz de Conexiones

Nº	1	2	3	4
1	0	1	1	0
2	0	0	0	1
3	0	0	0	1
4	0	0	0	0

Figura 4.10.- Matriz de Pesos y Conexiones de un individuo concreto

Nº	1	2	3	4
1	0	0,2	0,3	0
2	0	0	0	0,6
3	0	0	0	0,8
4	0	0	0	0

Figura 4.11.-Matriz de Pesos Combinada

Matriz de Conexiones

Nº	1	2	3	4
1	0	1	1	1
2	0	0	0	1
3	0	0	0	1
4	0	0	0	0

Matriz de Pesos Combinada

Nº	1	2	3	4
1	0	0,2	0,3	0,2
2	0	0	0	0,6
3	0	0	0	0,8
4	0	0	0	0

Figura 4.12.-Matriz de pesos final por modificación de la matriz de conexiones

Las matrices de la Figura 4.10 denotan la arquitectura de una RNA alimentada hacia delante cuya matriz de pesos se podría reducir, una vez se tuviera la RNA en fase de producción, a la matriz de pesos de la Figura 4.11.

Obviamente, existe una diversidad genética que está latente y que no se manifiesta. Si al realizar la operación de cruce, la matriz de conexiones cambiara, se van a manifestar unos pesos que antes no se estaban utilizando. Por ejemplo, si se considera una nueva matriz de conexiones como la que se puede ver en la Figura 4.12, para utilizar junto a la matriz de pesos original de la Figura 4.10, la matriz de conexiones final a aplicar en la RNA sería la que se puede ver en la siguiente figura.

4.5.3.4 Mutación

El operador de mutación se aplica de la forma habitual modificando un cierto número de individuos de la población en posiciones aleatorias. Sin embargo, en este caso hay que tener en cuenta que las tres partes del cromosoma utilizan distintos rangos de valores. Así, en la primera parte, los genes representan neuronas y existe un número máximo de neuronas por capa, mientras que el mínimo será uno. En la segunda parte, los genes representan pesos de la conexión, para los que se debe especificar un valor máximo y uno mínimo en el programa e indicadores de conectividad que sólo pueden tomar dos valores, cero o uno. Por último, la tercera parte se refiere a las funciones de activación y, aquí, el rango de valores posibles son las cuatro funciones de activación representadas por cuatro número enteros y los parámetros asociados a las funciones lineal y umbral que son números reales.

Debido a estas peculiaridades, el proceso de mutación de un individuo comienza generando un valor aleatorio dentro del rango del número de genes del individuo para saber qué posición del individuo se va a mutar. Después se comprueba el rango de valores que puede tomar ese gen dependiendo de la zona en que haya que realizar la mutación. A continuación, se genera un valor en ese rango y, por último, se sustituye ese valor en el gen correspondiente.

Después se reevalúa el individuo mutado y se recoloca en la población como es habitual en este operador genético.

4.5.4 Consideraciones a la Automatización de la Topología

En este punto, se introduce un sistema de varios niveles de AG que también se va a utilizar en el diseño de conjuntos de entrenamiento. También en este punto, se realiza un ajuste paralelo de varias características de la RNA integrando la mejora de la arquitectura de la red tomando como referencia el módulo de entrenamiento de RR.NN.AA. basado en AA.GG.

Se consigue, con el sistema realizado hasta el momento, el entrenamiento de RR.NN.AA. con una arquitectura adaptada al problema que intentan resolver, no sólo en cuanto a número de capas y número de neuronas sino que también se seleccionan las funciones de activación más adecuadas a la posición de la neurona en la red y se realiza un "podaje adaptativo" de conexiones que permite conseguir RR.NN.AA. optimizadas para su función.

4.6 Diseño del Conjunto de Entrenamiento

4.6.1 Introducción

Como ya se ha comentado, normalmente, el procesado del conjunto de datos que se utiliza tanto en la fase de entrenamiento como en la fase de test de una RNA, no va más allá de un simple filtrado para eliminar valores atípicos y suavizar el ruido de alta frecuencia que puede ralentizar el aprendizaje de la RNA. En un número limitado de casos se realiza un análisis de componentes principales para reducir la redundancia que aportan distintas variables de entrada en entornos complejos. Esta técnica analiza las variables de entrada desde un punto de vista estadístico, buscando dependencias por medio del análisis de las covarianzas entre variables. A partir de este punto, es el diseñador de la RNA el que tiene que buscar un umbral del grado de relación entre variables a partir del cual se va a tener en cuenta o no una determinada variable. Como en el caso del diseño de la arquitectura de una RNA el número de opciones es enorme, el diseñador se debe ceñir a la exploración de un número limitado de posibilidades para escoger el parámetro del grado de covarianza.

Una vez escogidas las variables de entrada con las que va a trabajar la RNA, se entra en el segundo paso del procesado del conjunto de entrenamiento. La cuestión

ahora es dividir el conjunto de datos disponibles en conjunto de entrenamiento y test. La solución habitual es reservar, del conjunto de datos disponibles, un 20 ó 30% para la fase de test y utilizar el resto en la fase de entrenamiento. Estos datos, que no se utilizan en el entrenamiento, sirven entonces para comprobar el funcionamiento y la capacidad de generalización de la RNA.

Con respecto a los datos de test no existirían más problemas; sin embargo, para los datos escogidos para la fase de entrenamiento, restaría todavía un paso de procesado. En este sentido, no está muy estudiada la relación que existe ente la selección de los datos del conjunto de entrenamiento y el rendimiento de la RNA final con respecto al problema que debe resolver. Está claro, sin embargo, que es necesario un conjunto de casos completo y que existe una relación entre una mayor proporción de un tipo de casos y que la red tienda a dar como salida ese tipo de casos. Debido al desarrollo de estas tendencias, se recomienda que, en la construcción del conjunto de entrenamiento, se equilibre el número de casos de cada tipo de salida para evitar sesgos en las respuestas de la RNA entrenada. Sin embargo, en problemas complejos como puede ser el diagnóstico médico [DORA-96], ni siquiera el experto humano es capaz de definir con claridad el número de categorías de entrada y de salida que subsisten en los datos que se proporcionan para el entrenamiento de la RNA. Es por tanto difícil realizar una selección de ejemplos uniforme entre los distintos tipos de entradas y salidas.

Aunque en el caso de unos datos de entrada con múltiples variables existen las complicaciones comentadas, si la entrada es temporal, la selección del conjunto de entrenamiento cambia por completo. Además de que se puedan tener múltiples series temporales como entradas de una misma red, la selección de los datos en cada una de ellas no se puede realizar de la misma forma que con variables que no están relacionadas. En un problema temporal, la salida depende de las entradas pero, además, el valor de la entrada en el instante "t" también depende de los valores de los "n" instantes anteriores en el tiempo, por lo que no se pueden desligar los pares entrada-salida de su contexto temporal.

Para solucionar estos problemas de selección de datos de entrada de la RNA, en este apartado se plantea un AG para la selección del conjunto de entrenamiento de una RNAR que trabaja con datos temporales.

4.6.2 Selección del Conjunto de Entrenamiento

El filtrado del número de variables de entrada es muy necesario para facilitar la convergencia de la red, sin embargo en el caso de una serie temporal donde existen también un número considerable de datos la forma de seleccionar los datos significativos no se puede realizar de la misma forma.

Al trabajar con una serie temporal, no existen distintas variables más o menos independientes sino que se trabaja con una serie de valores que tienen una dependencia en el tiempo y que, por tanto, no tienen sentido de forma individual fuera de la serie. Sin embargo, la serie temporal tiene unas componentes o parámetros que la modelizan y que, en teoría, son suficientes para que, a partir de los datos de un instante, se pueda hacer una predicción en el tiempo. El objetivo de la RNA es generalizar este modelo o estos parámetros de forma que, alimentando a la RNA entrenada con datos de uno o varios instantes consecutivos, sea capaz de realizar las predicciones en los instantes siguientes.

Sin embargo, no es necesario utilizar todos los datos de una serie temporal para realizar el entrenamiento de la RNA, igual que no sería necesario utilizar todos los datos de una función matemática para que una RNA generalice la forma de esa función. Una vez reconocido este punto, el problema radica en escoger los datos de la serie que se van a seleccionar como conjunto de entrenamiento. En el caso de las series temporales existe la particularidad, ya comentada, de que no es posible escoger datos individuales ya que pierden el contexto en el que tienen sentido, por lo que se podrían escoger partes de la serie temporal o subseries, con las que entrenar la RNA. Según esta idea, el conjunto de entrenamiento estaría compuesto por un conjunto de subseries de la serie temporal que sirve como datos de entrada para el entrenamiento de la RNA.

Hay que tener en cuenta que se cambia el proceso de entrenamiento de un tipo continuo en el que se proporcionan a la RNA los datos de la serie consecutivamente en el tiempo y se van ajustando los pesos, para cada dato, a un tipo de entrenamiento continuo solo dentro de la subserie y discreto entre subseries. Aquí, todas las muestras menos la última se utilizan para crear el estado de activación correcto de la red, siendo aquí el entrenamiento en modo continuo. Esto quiere decir que la red no está preparada, durante esta fase, para proporcionar salidas correctas y las salidas de la red no deben ser

tenidas en cuenta. Por esto, el proceso de ajuste de pesos solo se realiza estimando el error que produce la red en la predicción del último dato de la subserie. Una vez acabada de pasar la subserie y ajustados los pesos, los estados de activación de la red deben ser "reseteados" para comenzar el proceso con la subserie siguiente. Es decir, la red tiene que olvidar el estado de activación para adaptarse a la nueva situación de una nueva subserie. En este punto, en el paso de una subserie a otra, el entrenamiento es discreto.

Una vez adoptada esta aproximación, quedan todavía dos puntos por concretar antes de poder aplicar la selección de datos al entrenamiento de la RNA. Por un lado, es necesario definir cuál será el tamaño de las subseries; es decir, el número de valores que van a contener cada una y, por otro lado, también es necesario definir cuál es el número óptimo de subseries para que, a partir de ellas, la RNA pueda generalizar la serie temporal.

El número de datos de cada subserie tiene que ver con la dependencia temporal que exista entre los datos de la serie. Normalmente, la dependencia temporal no abarca muchos instantes de tiempo y además la dependencia es mucho más fuerte con los instantes justamente anteriores y decrece rápidamente en el tiempo. A partir del conocimiento del problema que se quiere resolver podría ser posible definir un tamaño de subserie que funcionara bien a la hora de entrenar la RNA; es decir, que contuviera el suficiente número de datos para que la información de la tendencia estuviera implícita en la subserie y que no tuviera demasiados datos para que no se entrene la RNA con datos superfluos.

El número de subseries que se van a utilizar en el entrenamiento también es importante ya que cada subserie representaría una situación dentro de la serie temporal. Es decir, cada subserie sería un caso particular de comportamiento de la serie temporal y, por tanto, cada caso (en este caso subseries) debe estar representado homogéneamente en el conjunto de entrenamiento. En este aspecto del diseño, se repite el dilema del caso anterior del número de datos por subserie. Si se añaden pocas subseries al conjunto de entrenamiento, no estarán representados todos los casos posibles de la serie y la red no se va a comportar adecuadamente con casos reales. Pero, si se añaden demasiadas subseries, los casos que predominen más establecerán la

tendencia de la red, a la vez que, al existir redundancia, la fase de entrenamiento será más lenta.

El otro punto a tener en cuenta es que, aún teniendo definidos estos parámetros, queda escoger las subseries que serían más adecuadas para el entrenamiento de la RNA. Como es obvio, no es posible, "a priori", indicar qué partes o subseries de una serie temporal contienen los datos más relevantes que deberían proporcionarse a la RNA para que abstraiga la forma de la serie. Sería posible proporcionarle un número elevado de subseries distribuidas uniformemente en el tiempo para garantizar que se cubran todas las posibilidades que se indicaban en el párrafo anterior. Sin embargo, actuar de esta forma conlleva, probablemente, incluir subseries innecesarias que, al hacer el conjunto de entrenamiento más grande, ralentizarán el proceso de entrenamiento. Además, si la distribución de los cambios de tendencia de la serie no es uniforme o es un problema no-lineal, el planteamiento anterior no es válido ya que los datos escogidos deberían concentrarse en las zonas de más interés, de más variación, y no distribuirse de forma homogénea a lo largo de la serie.

Las ventajas de trabajar con un conjunto de entrenamiento reducido son obvias ya que, al ser de menor tamaño, la evaluación de los pesos de la RNA se realiza de una forma más rápida, a la vez que, al no utilizar la información redundante o contradictoria de la serie, se favorece el proceso de convergencia consiguiendo, además, una RNA que generaliza mejor el problema. Sin embargo, los problemas que se acaban de exponer dificultan la aplicación de las ideas de selección de datos para el conjunto de entrenamiento.

Estos problemas son todos de optimización. En primer lugar, habría que escoger el número óptimo de datos para las subseries, después el número óptimo de subseries y, por último, cuáles son las mejores subseries para el entrenamiento de la RNA. Parecen, por tanto, problemas susceptibles de ser resueltos mediante la aplicación de un AG, ya que el espacio de búsqueda es muy amplio y están relacionados tres tipos de parámetros, número de datos por subserie, número de subseries y posición de las subseries en la serie. La aplicación del AG no es, sin embargo, inmediata ya que no se puede utilizar un AG estándar. En el punto siguiente, se explican las modificaciones necesarias para su aplicación en la selección del conjunto de entrenamiento de una RNA.

4.6.3 Diseño del AG

4.6.3.1 Objetivo

Como se ha visto en el punto anterior, se va a desarrollar un AG que seleccione el conjunto de datos óptimo, dentro de la serie temporal, para realizar el entrenamiento de una RNA que realice la predicción de esa serie temporal. En este caso, el conjunto óptimo se refiere al conjunto mínimo de subseries que sean suficiente para que la RNA sea capaz de abstraer la esencia de la serie temporal.

Con la reducción del tamaño del conjunto de entrenamiento, se va conseguir acelerar el proceso de entrenamiento de la RNA a la vez que mejorar el proceso de convergencia, al limitar la información redundante o contradictoria.

Este AG cuenta como entrada una serie temporal completa y sus individuos van a ser conjuntos distintos de subseries. Como se puede suponer, los individuos son de tamaño variable ya que el número de subseries escogidas no va a ser fijo; mientras que, sí se va a fijar el número de datos en cada subserie en esta primera aproximación. Al ser individuos de tamaño variable, es necesario modificar la definición de los operadores genéticos para adaptarlos a esta situación. En lo que respecta a la valoración de cada conjunto de subseries o individuo, se realizará mediante el entrenamiento de una RNA con una arquitectura y conjunto de parámetros fijo. El nivel de acierto de esa RNA en la predicción de la serie temporal completa indicará la bondad del subconjunto de datos escogido. Por último, se va a establecer una función de ponderación de ajuste que favorezca la proliferación de individuos con un número de subseries reducido que aceleren el proceso de aprendizaje de las RR.NN.AA.

4.6.3.2 Codificación de los Individuos

Según el funcionamiento de una RNAR, es necesario proporcionar un cierto número de entradas a la red para que se establezca un estado de activación interno adecuado para empezar a realizar predicciones acertadas. Debido a esto, no es posible establecer simplemente pares de datos de entrada y de salida como en las RR.NN.AA. que tratan problemas sin componente temporal. En el caso de problemas temporales, se va a seleccionar un conjunto de "n" datos consecutivos en el tiempo con los que se va a

alimentar a una RNAR de forma consecutiva en "n-1" instantes de tiempo, haciendo la medición del error en la salida de la red únicamente con el último dato de la serie, el dato "n". Así, los "n-1" datos anteriores sirven para establecer el estado de activación adecuado en la RNAR y se estima el error que se comete en la predicción del último dato.

Los individuos van a estar compuestos, entonces, de un cierto número de subseries que vendrán caracterizadas, cada una, por la posición inicial dentro de la serie temporal. La cadena o "*array*" en que se plasma cada individuo estará compuesto por un número "n" de números enteros que indican la posición inicial de cada subserie. Este número "n" de subseries es variable en cada individuo entre un mínimo de 1 y un máximo definido en el programa.

Si se representa una serie temporal como la siguiente:

Serie temporal	**2**	**4**	**3**	**6**	**3**	**8**	**6**	**5**	**1**	**9**
Posiciones	1	2	3	4	5	6	7	8	9	10

Un ejemplo de individuo que representa 2 subseries sería

1	**5**

A partir de este individuo será necesario generar una serie temporal con la que entrenar la RNA. Con las referencias de las dos subseries del individuo anterior, suponiendo cada subserie de tres posiciones y partiendo de la serie temporal original se genera la serie temporal para el entrenamiento:

Serie temporal	**2**	**4**	**3**	**3**	**8**	**6**
Posiciones	1	2	3	5	6	7
	Subserie 1			Subserie 2		

Como se puede observar, no se utiliza una codificación binaria, sino una codificación de números enteros que, como ya se ha comentado en el apartado dedicado a los AA.GG., es también una alternativa válida refrendada por los trabajos de distintos investigadores.

4.6.3.3 Inicialización

El proceso de inicialización de la población consiste en generar un cierto número de individuos que contengan, cada uno, un número aleatorio de subseries. Para ello, una vez definido el número de individuos que va a tener la población, para cada individuo se genera aleatoriamente el número de subseries que va a contener. Una vez fijado este número, se generan aleatoriamente tantos números enteros como subseries necesite el individuo para representar los puntos de inicio. Una vez generados todos los puntos de inicio, se procede a ordenarlos y a eliminar aquellos repetidos que no aportarían nada al proceso de entrenamiento. El que los puntos de inicio estén ordenados facilitará la operación de cruce de individuos.

Con esto queda inicializada la población, lista para el periodo de simulación.

4.6.3.4 Operadores Genéticos

4.6.3.4.1 Cruce

Es el operador genético que hay que modificar debido a las características especiales de los individuos de longitud variable. El problema que se puede plantear es que, al aplicar el operador de cruce, la posición que se seleccione para dividir los individuos exista en un individuo pero no en el otro, al ser más corto o acabar antes de la posición seleccionada en el otro individuo. Para solucionar este problema, antes de escoger la posición de corte, hay que estudiar la longitud de ambos individuos y definir como máximo la longitud del individuo más corto (ver Figura 4.13).

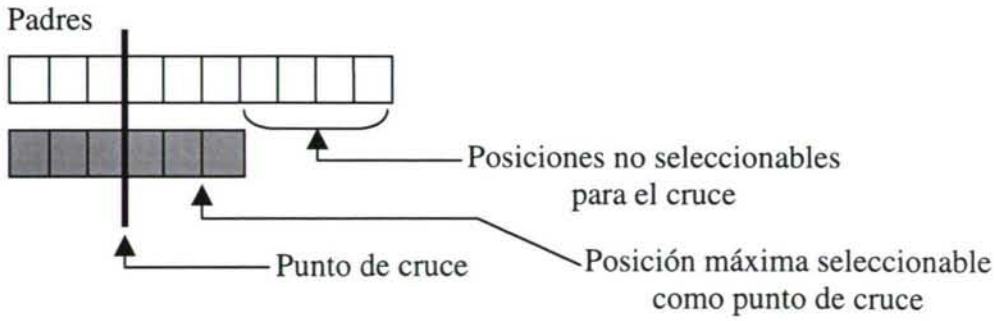


Figura 4.13.- Selección del Punto de Corte para el Cruce de Individuos de Distinta Longitud

Otro de los problemas que se presentan con el operador de cruce es que al intercambiar el material genético de dos individuos puede suceder que acaben teniendo posiciones repetidas. Esto hace necesario realizar un postprocesado de la descendencia

que reordene los genes y que elimine las posiciones repetidas como se puede apreciar en la Figura 4.14.

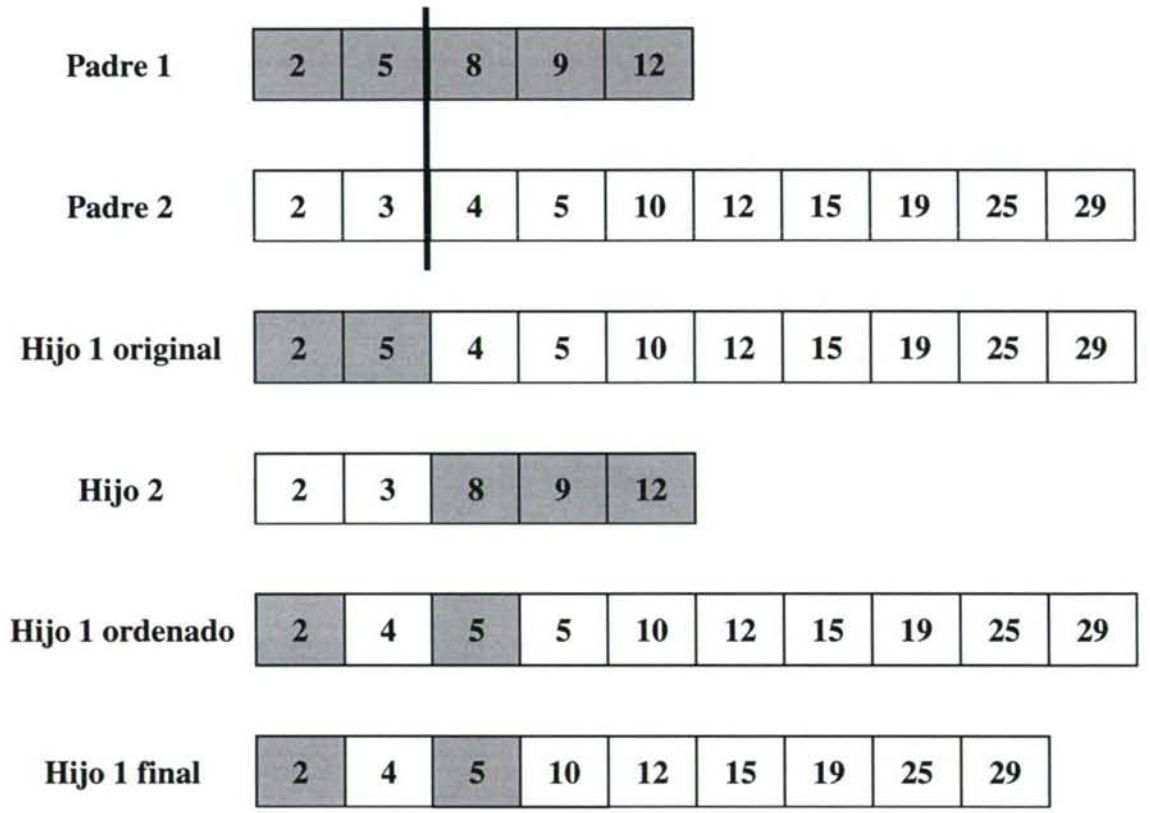


Figura 4.14.- Eliminación de Posiciones Repetidas

La resolución de estos problemas conlleva que la descendencia no tenga la misma estructura que los individuos padres. Normalmente, en el cruce de dos individuos, uno de longitud "n" y otro de longitud "m", generarían también como descendencia dos individuos de longitud "m" y "n". Sin embargo, la eliminación de redundancia actúa aquí como un elemento adicional de diversidad dentro del AG.

4.6.3.4.2 Mutación

En este AG el operador de mutación realiza tareas adicionales en su trabajo de exploración de nuevas áreas del espacio de búsqueda. El trabajo tradicional de este operador se realiza modificando los valores de las posiciones de inicio de las subseries dentro de la población. Según el ratio especificado en el programa, un cierto número de mutaciones se realizan en cada paso de simulación. Para cada una de ellas, se escoge aleatoriamente un individuo de la población y, dentro de él, una posición. En esta posición aleatoria, se sustituye el punto de inicio por un valor entero, generado también aleatoriamente en el rango de posiciones posibles en la serie temporal.

Con respecto a esto, se han añadido dos características nuevas, una para eliminar subseries y otra para añadirlas. Como ya se ha comentado, para sustituir el valor de un punto de inicio, se genera un nuevo valor aleatorio. Para que tenga sentido, este valor tiene que estar entre 1, que es el primer valor de la serie temporal, y el valor que resulta de restar la longitud que puede tener cada subserie al último valor de la serie temporal; es decir, si hay 100 valores en la serie temporal y cada subserie tiene 5 valores el último valor válido para punto de inicio sería el 96. Sin embargo, se ha definido como el primer valor válido el 0 en vez del 1, con el significado de eliminación de subserie. Si como nuevo valor de inicio se genera un 0, esa subserie se elimina y el individuo tiene una subserie menos después de la aplicación de la mutación (ver Figura 4.15).

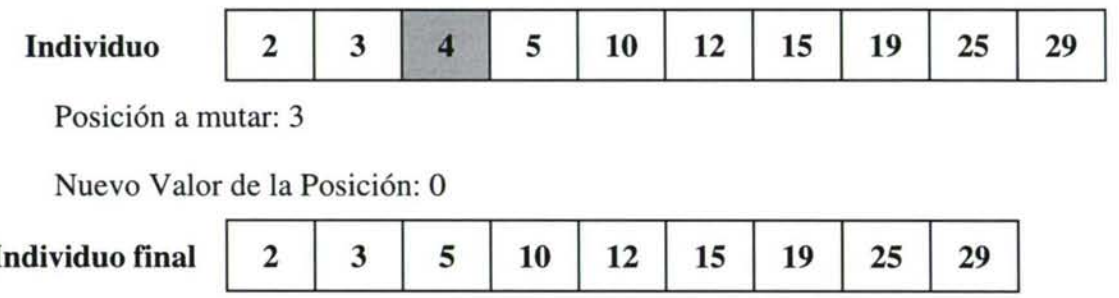


Figura 4.15.- Eliminación de Subseries en la Operación de Mutación

Con respecto a añadir nuevas subseries, cuando se escoge el punto de inicio al que se le va a cambiar el valor, se permite que entre, dentro del rango posible, la posición siguiente a la última. De esta forma, si se escoge esta posición, el individuo contendrá una subserie más después de la aplicación del operador de mutación. Igual que en el caso de los individuos generados en la operación de cruce, es necesario aplicar al nuevo individuo un post-procesado que garantice que se mantiene el orden de las subseries y que no existen valores repetidos.

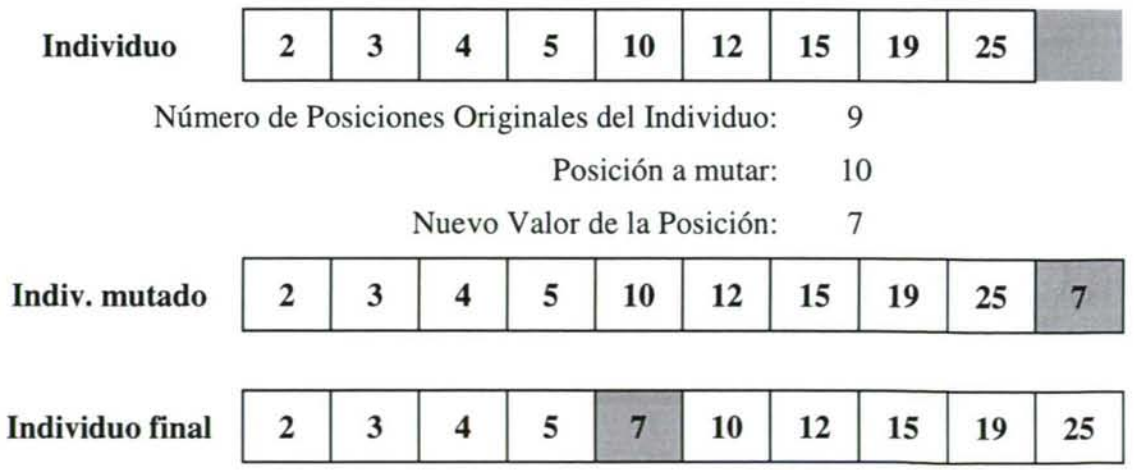


Figura 4.16.- Generación de Subseries en la Operación de Mutación

4.6.3.5 Función de Evaluación

En este caso, lo que se quiere conseguir es un subconjunto del conjunto de datos de entrada que sean equivalentes al conjunto de datos de entrada en sí, a la hora de entrenar una RNAR. En consecuencia, el proceso de evaluación consiste en entrenar una RNAR utilizando el conjunto de subseries que se definen en un individuo. A continuación, esa RNAR ya entrenada es probada mediante la predicción sobre el conjunto completo de la serie temporal. Es necesario realizar el test sobre la serie temporal completa para comprobar si, mediante el entrenamiento de la red con el subconjunto de datos, la RNAR es capaz de generalizar el comportamiento de la serie completa.

Para la evaluación de cada individuo, se utiliza el simulador de RNA-AG comentado en el punto anterior. Antes de comenzar la simulación se le suministran los parámetros de la RNAR, arquitectura y funcionalidad de la neuronas, que tiene que utilizar para evaluar cada individuo. Después, para cada individuo se genera la serie temporal representada por los puntos de inicio del individuo. Esta serie se pasa al simulador para que desarrolle una RNAR entrenada para ese conjunto de datos. A continuación, se le transmite la serie completa al simulador para que realice la fase de test y devuelva el valor de ajuste de la red entrenada (ver Figura 4.17).

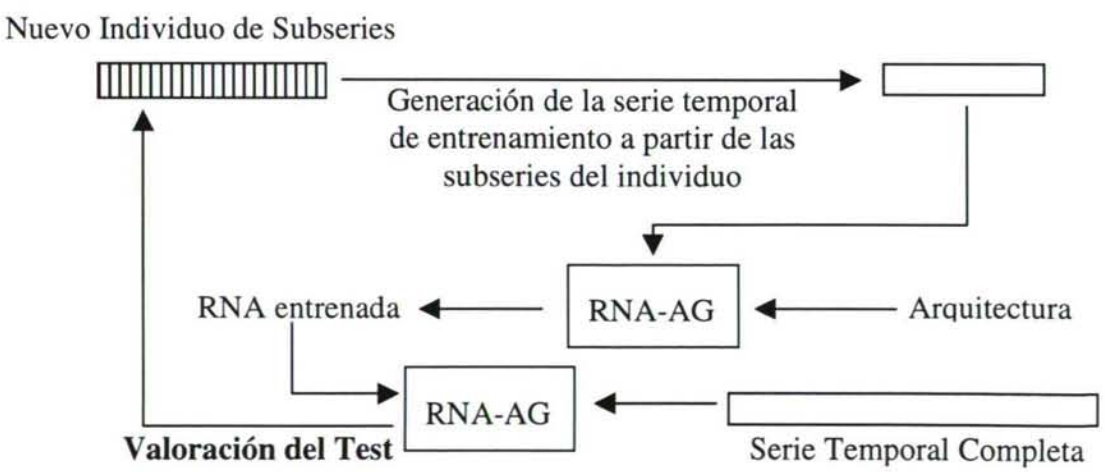


Figura 4.17.- Evaluación de Individuos de la Población de Conjuntos de Entrenamiento

Hay que tener en cuenta que, para que la simulación sea ágil y los datos sobre selección del conjunto de entrenamiento que puede aportar este sistema sean útiles, las RR.NN.AA.RR. que genere este sistema no tienen por qué funcionar con niveles de error extremadamente bajos. Lo que se necesita en este punto es detectar qué zonas de la

serie temporal aportan una información más importante para el entrenamiento de la RNAR. No es, por tanto, necesario entrenar al red de una forma exhaustiva. Por todo esto, los parámetros de ciclos de simulación para usarlos en el entrenamiento de la red, así como la arquitectura seleccionada para la RNAR no deben ser excesivos, para que la labor de evaluación de los conjuntos de entrenamiento se realice en un tiempo razonable.

4.6.4 Consideraciones al Diseño del Conjunto de Entrenamiento

En este punto, se realiza la selección de los valores de una serie temporal que son relevantes para construir el conjunto de entrenamiento de una RNAR. Como se puede ver en el capítulo de resultados, se consigue reducir el tamaño del conjunto de entrenamiento utilizado, acelerando el proceso de entrenamiento de la red. Un punto muy interesante del sistema es la superposición de los valores de las subseries que se produce al escoger subseries contiguas dentro de la serie temporal. Esta circunstancia permite el entrenamiento intensivo de la RNA sobre las zonas más interesantes o que contienen más información dentro de la serie temporal.

La desventaja de este sistema es la necesidad que tiene el diseñador de escoger la arquitectura de la RNA para las pruebas. Este tema, que ya fue tratado en el punto anterior, está resuelto aunque no integrado con la parte de diseño del conjunto de entrenamiento, punto que se tratará en el siguiente apartado de este capítulo.

Otros puntos que pueden ser ampliables dentro de esta parte del sistema son, por un lado, la automatización del número de valores de la subserie, dejando al propio sistema la posibilidad de validar el número de instantes temporales necesarios para realizar predicciones ajustadas sobre la serie temporal. Por otro lado, sería interesante la modificación dinámica del número de ciclos de simulación. Es decir, que el sistema, según avanza la simulación y se producen cambios menos frecuentes en los mejores individuos de la población, realizara una ampliación del número de ciclos con que se entrena una RNA con un determinado conjunto de entrenamiento; de esta forma, se irían consiguiendo resultados más ajustados a lo que será el entrenamiento final de la RNA.

4.7 Desarrollo de Nuevas Arquitecturas

4.7.1 Introducción

Como ya se ha comentado en algunos puntos de este documento, los problemas principales en el desarrollo de RR.NN.AA. son, por un lado, la dificultad para escoger un modelo de RNA y los parámetros dentro de ese modelo, de entre la ingente cantidad de opciones que se le presentan al diseñador y, por otro lado, la dificultad para innovar o añadir nuevas posibilidades a los modelos existentes con el objetivo de hacerlos más potentes.

En dos de los puntos anteriores de este capítulo, entrenamiento de RR.NN.AA. mediante AA.GG. y automatización del desarrollo de la topología, se trabaja sobre el primero de los problemas mencionados, intentando hacer transparente al diseñador el ajuste de los parámetros de la RNA que van a utilizar. En este punto, se presenta una posibilidad de cómo afrontar el segundo tipo de problema. Se va a plantear cómo se puede modificar la arquitectura de las neuronas artificiales o las conexiones entre ellas y entrenar una RNA que explote estas modificaciones mediante AA.GG.

Actualmente, la realización de una modificación en la arquitectura de un tipo de RNA tiene que verse complementada con un algoritmo de aprendizaje que, normalmente, está basado en el descenso de gradiente y es una modificación o una ampliación de alguno ya existente. Esto supone un lastre a la hora de realizar una experimentación amplia sobre las arquitecturas de las RR.NN.AA.

Es, sin embargo, muy deseable, como se menciona en la introducción de este capítulo, acercar el diseño de las neuronas artificiales a los modelos de los que parten las neuronas naturales y sus modelos de organización en el cerebro. Como también se ha comentado, existen puntos bien conocidos del modelo natural que todavía no se han implementado en RR.NN.AA. pese a que, con la incorporación de estos parámetros, sería muy probable conseguir una mejora en el rendimiento de las RR.NN.AA.

La utilización de los AA.GG. se presenta como una muy buena solución a la hora de entrenar modificaciones en los parámetros de las RR.NN.AA. La introducción de parámetros nuevos en el modelo de neurona y de RNA presentadas previamente en el

apartado sobre el entrenamiento mediante AA.GG. no presenta ningún problema. El AG, además de ajustar los pesos de las conexiones y los parámetros de las funciones de activación, es capaz de ajustar cualquier nuevo parámetro siempre que se pueda integrar en su función de evaluación. En este caso, la función de evaluación significa el test de una RNA frente al conjunto de datos de prueba, por lo que esta restricción no supone ningún problema al funcionamiento del AG.

Uno de los puntos en los que se indicaba que las neuronas artificiales no emulan todavía el comportamiento de las neuronas naturales es el proceso de activación. La flexibilidad y complejidad de este proceso en la neurona natural no está todavía reflejado en los modelos de neuronas artificiales, lo cual supone una restricción en el funcionamiento de las RR.NN.AA. En el siguiente punto, se comenta en detalle cómo se realiza este proceso en la neurona natural.

4.7.1.1 Proceso de Activación en la Neurona Natural

La respuesta de activación de la neurona biológica viene dada por las características de conducción de la electricidad en su membrana celular. La membrana celular es selectivamente más permeable para los iones de potasio (K+) que para los iones de sodio (Na+). El gradiente químico del potasio tiende a hacer que los iones de potasio salgan de la célula por difusión, pero la fuerte atracción de los iones orgánicos negativos que están dentro de la célula tiende a mantener dentro el potasio. El resultado de estas fuerzas opuestas es que se alcanza un equilibrio en el cual hay más iones de sodio y cloro (Cl-) fuera de la célula, y más iones orgánicos y de potasio dentro de ella. El equilibrio resultante produce una diferencia de potencial a través de la membrana de la célula de unos 70 a 100 mV (milivoltios), siendo más negativo el fluido intracelular. Este potencial se denomina potencial de reposo de la célula.

Las influencias de las entradas excitatorias que llegan a la célula desde otras neuronas se suman en el montículo axonal y producen una despolarización sucesiva de la membrana. Se invierte entonces el potencial hasta +35 mV en sólo 0'1 milisegundos. La despolarización resultante en el montículo del axón altera la permeabilidad de la membrana celular a efectos de los iones de sodio. Como resultado hay un flujo entrante de iones de sodio positivos, que penetran en la célula, contribuyendo aún más a la

despolarización. Este efecto autogenerado da lugar al potencial de acción (véase Figura 4.18).

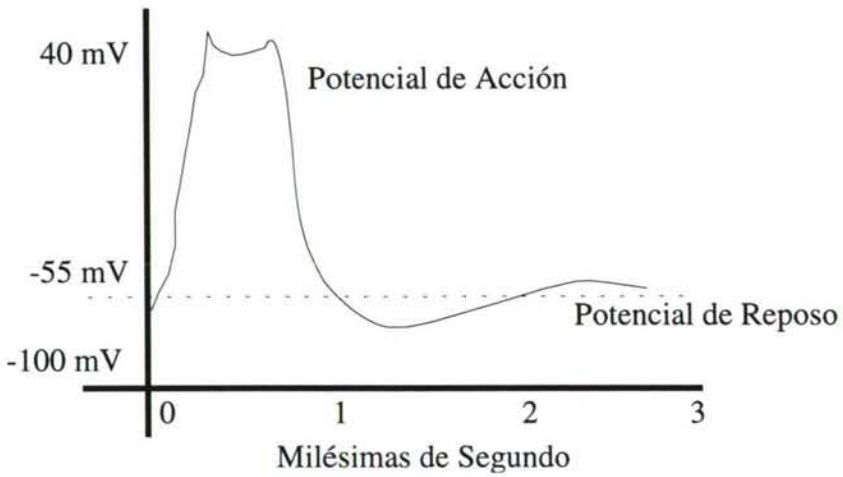


Figura 4.18. Desencadenamiento del Potencial de Acción en la Membrana de la Neurona.

Después de un potencial de acción, el potencial de la membrana se vuelve más negativo que el potencial de reposo durante unos milisegundos hasta que se restablece el equilibrio iónico. Para iniciar el potencial de acción se requiere un incremento súbito de aproximadamente 15-30 mV, considerándose entre –55 y -40 mV el umbral de estimulación. En una fibra excitable no puede producirse una segunda corriente de acción, mientras la membrana se encuentre despolarizada. Al período durante el cual no puede desencadenarse otro potencial de acción, incluso con un estímulo muy poderoso, se llama "período refractario absoluto" y dura aproximadamente 0'5 milisegundos. A continuación le sigue un "período refractario relativo" que dura entre 1 y 1'5 milisegundos, donde se precisa un estímulo mayor que lo normal para producir el potencial de acción al no haber recuperado todavía el equilibrio iónico del potencial de reposo.

Ese punto no puede volver a ser excitado en, aproximadamente, 1 milisegundo, que es el tiempo que tarda en volver a su potencial de reposo. Este período refractario limita la frecuencia de transmisión de los impulsos nerviosos a unos 1.000 por segundo.

4.7.2 Modelo de RNA con Conexiones de Activación Atenuada

Obviamente, el comportamiento de los elementos de proceso de las RNA no es muy parecido, en cuanto a la fase de activación, a lo que se acaba de describir. En las neuronas artificiales tradicionales, al resultado de la función de entrada, que pondera los pesos con las activaciones de las neuronas de la capa anterior, se le aplica una cierta

función de activación. En el caso de que la salida de esta función sea positiva y se produzca, por tanto, la activación, en ese instante y sólo en ese instante, las neuronas a las que les llegue esa activación tendrán una entrada. La activación en el instante t se propagará hasta el instante t+1 por la red.

Esta forma de propagar la activación de las neuronas funciona de un modo muy puntual en el tiempo, sin dar la idea de continuidad que se aprecia en el modelo de neurona natural que se acaba de comentar, donde la activación fluye desde un punto máximo (potencial de acción) hasta llegar progresivamente al nivel mínimo (potencial de reposo) o de equilibrio de la membrana. Es también muy interesante el hecho de que los valores de ciertos parámetros varíen en la fase de activación dependiendo de la cercanía en el tiempo de la activación anterior como, por ejemplo, la diferencia de potencial necesaria para desencadenar el potencial de acción.

En este trabajo, se ha añadido a la arquitectura clásica de elemento de proceso la función del potencial de acción (véase Figura 4.19), aproximándola mediante una recta que va desde el nivel de salida de la función de activación hasta el nivel cero.

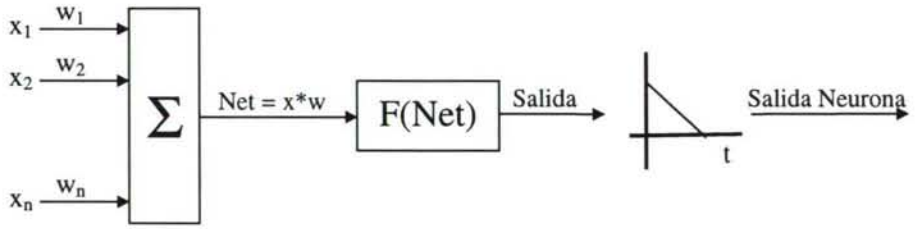


Figura 4.19.- Representación de la Conexión Atenuada en el Tiempo.

Este modelo de RNA sólo cambia en la concepción de las conexiones entre neuronas que pasan de ser fijas (un valor numérico) a ser representadas por una función (una recta con pendiente negativa) caracterizada por el valor de la pendiente.

El funcionamiento de la neurona hay que expresarlo ahora en función del tiempo. Es decir, en este modelo, la activación que produce en una neurona la conjunción de una serie de entradas en un cierto momento del tiempo, no afecta solo a las neuronas conectadas a su salida en ese instante, sino también en "n" instantes posteriores (véase Figura 4.20). El valor de "n" depende de la pendiente de la recta que tenga esa conexión.

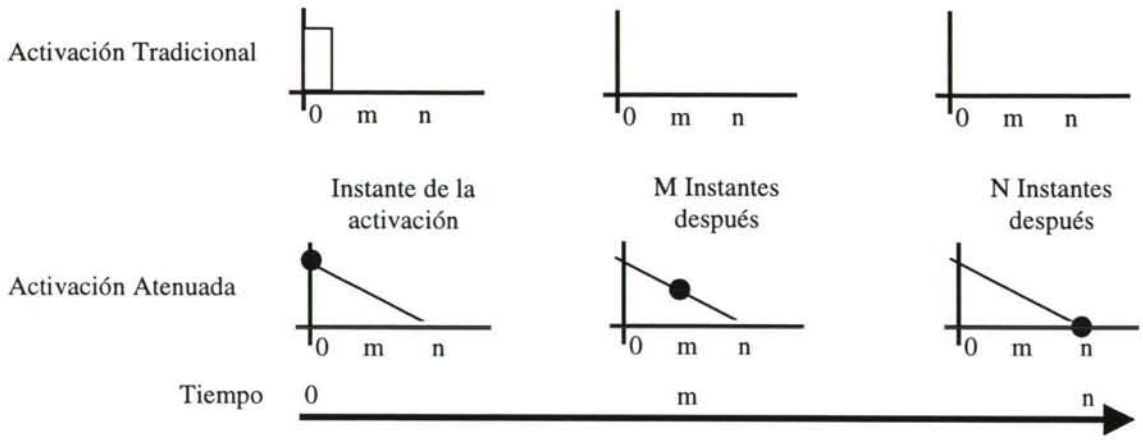


Figura 4.20. Funcionamiento de la Activación de la Neurona.

En la Figura 4.21 se puede observar la fórmula de la recta que emula la activación atenuada de las neuronas naturales. En esta fórmula ***y*** representa el valor de salida de la neurona que reciben las neuronas siguientes. La letra ***b*** representa la salida de la función de activación de la neurona. La letra ***x*** representa el tiempo que pasa desde la activación y la letra ***m*** es la pendiente de la recta que, como ya se ha comentado es siempre negativa. En el momento en que se produce la activación, el tiempo es 0 por tanto la salida ***y = b*** a partir de ahí y dependiendo de la pendiente, la salida **y** decrece hasta 0 en un tiempo ***x = n*** (Figura 4.22).

$$y = mx + b$$

Figura 4.21. Fórmula de la recta genérica que se usa de función de activación atenuada en el tiempo

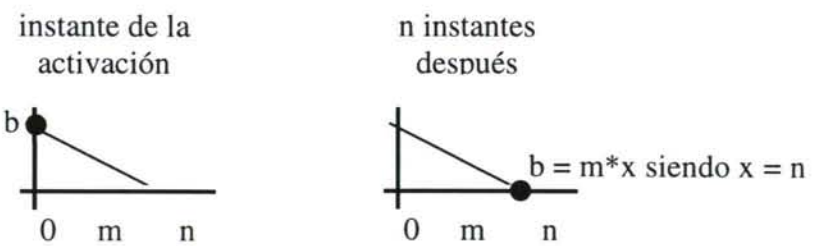


Figura 4.22.- Límites de la Función de Activación Atenuada en el Tiempo

Una vez planteado el funcionamiento de la activación atenuada, para poder realizar su aplicación en los modelos de RNA sólo queda cuantificar el valor de los avances en el eje de las "x" para sincronizar la disminución de la activación con el avance de la simulación de la RNA. La aproximación más sencilla es asumir que cada ciclo de simulación de la RNA supone avanzar una unidad en el eje de las "x", para calcular la activación como el valor correspondiente a la fórmula de la recta en el eje "y". También se puede parametrizar este incremento en el eje "x"; es decir, en el

tiempo, a un valor distinto de la unidad. Esto supone que, para utilizar la activación atenuada en una red, se necesita definir un parámetro común a toda la red, el valor del incremento temporal y los valores de las pendientes de la red que serían tantos como conexiones existan en la red que se quiere entrenar.

Una vez que está definida la nueva arquitectura, para poder aplicar este tipo de RR.NN.AA. a la resolución de problemas, se necesita un método de entrenamiento que ajuste las pendientes de las conexiones de la red. Para realizar este ajuste se va a añadir la funcionalidad de ajuste de pendientes al AG básico de entrenamiento de RR.NN.AA. La forma en que se realiza se expone en el siguiente punto.

4.7.3 Diseño del AG

4.7.3.1 Codificación

La codificación de esta nueva característica tiene dos aspectos. Por un lado, es necesario disponer de una nueva estructura, la matriz de pendientes, que almacene la pendiente calculada para cada conexión. Al ser un valor para cada conexión, la estructura utilizada es una matriz cuadrada de NxN, siendo N el número de neuronas de la RNA. La estructura es idéntica a la matriz de pesos expuesta en el sistema de entrenamiento de RR.NN.AA. mediante AA.GG.

Por otro lado, es necesario modificar el sistema que realiza la simulación de las RR.NN.AA. para que incluya el nuevo modo de funcionamiento de las conexiones. Ahora, es necesario almacenar las activaciones de las neuronas hasta que su activación haya desaparecido del todo, aportando durante este tiempo una parte de la activación inicial a las neuronas de destino de la conexión.

Ya en cuanto a los valores de las pendientes, estos van entre valores próximos a cero que representan una recta casi horizontal, hasta valores negativos tendiendo a infinito. Los valores próximos a cero suponen que, después de la activación, esa conexión continuará activada un tiempo muy largo, mientras que los valores con mucha pendiente negativa permanecerán muy pocos ciclos con valores de activación a su salida.

La matriz de pendientes se incluye en el genoma de los individuos de la misma forma que la matriz de pesos, afectando su introducción más a la evaluación de las RR.NN.AA. que al funcionamiento del AG.

4.7.3.2 Cruce y Mutación

El operador de mutación utilizado no tiene ninguna variación sobre el propuesto por Holland [HOLL-75] ya empleado en el sistema básico de entrenamiento de RR.NN.AA. con AA.GG. Se escoge un individuo al azar y, dentro de éste, se escoge un gen, pudiendo corresponder a una posición de la matriz de pendientes. Si esto es así, se genera un valor nuevo al azar dentro del rango de pendientes, un número menor que cero, y se incorpora al individuo. Este individuo se re-evalúa y se inserta dentro de la población dependiendo de su nuevo nivel de adaptación al problema.

El operador de cruce tampoco se modifica con respecto al del sistema básico de entrenamiento de RR.NN.AA. mediante AA.GG. Dependiendo del tipo de cruce que se aplique, se escoge el punto de corte del cromosoma, se recombina para generar los individuos de la descendencia que se recolocan en la población de acuerdo a su nivel de adaptación.

4.7.4 Consideraciones al Diseño de Arquiteturas

Queda patente, en este punto, la facilidad de inclusión de nuevos parámetros en el modelo de RR.NN.AA., siempre que sea posible plasmar estos parámetros en el algoritmo de evaluación de la red. Realmente, es más problemático ampliar el programa para incluir los parámetros en el sistema que modificar el AG para que sea capaz de ajustar los valores idóneos para ellos.

La facilidad con que se pueden entrenar modelos ampliados de RR.NN.AA. con AA.GG., ofrece la posibilidad de incluir características presentes en la neurona natural que se conocen de forma suficiente como, en este caso, el desencadenamiento de la activación o nuevas ideas que se sospeche que pueden mejorar el funcionamiento o la capacidad de abstracción de las RR.NN.AA.

4.8 Integración de los Módulos para el Desarrollo Cooperativo de RR.NN.AA.

4.8.1 Introducción

El sistema, en este punto, presenta un módulo común de entrenamiento de RR.NN.AA. y dos módulos que seleccionan características de diseño de las RR.NN.AA. previas a la fase de entrenamiento (ver Figura 4.23). Por un lado, se seleccionan características de la arquitectura de la red como son el número de capas, de neuronas y las funciones de activación para las neuronas de la red. Por otro lado, se construye el conjunto de entrenamiento a partir de la serie temporal original con el objetivo de mejorar y acelerar la fase de entrenamiento de la red. Estos dos módulos trabajan, hasta el momento, de forma independiente. Así, en el módulo que diseña la arquitectura, se entrenan las RR.NN.AA. con la serie temporal completa, mientras que, en el que construye el conjunto de entrenamiento, se trabaja con una arquitectura fija de red.

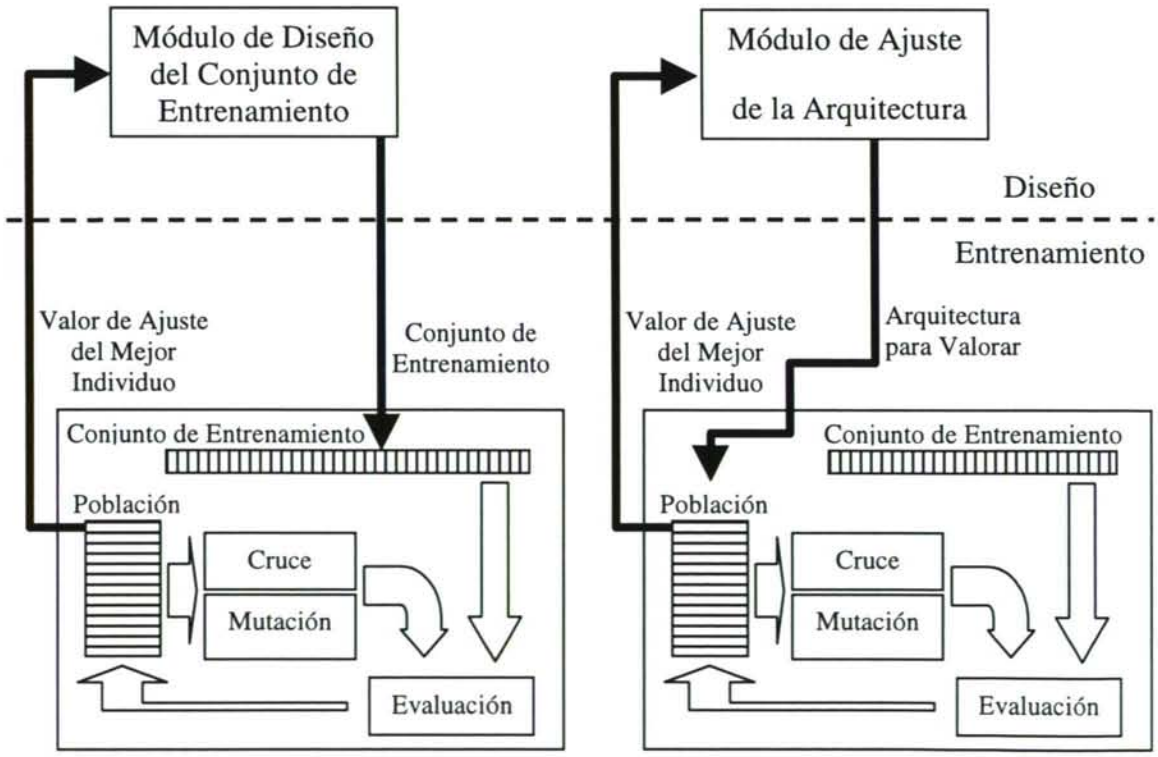


Figura 4.23.- Sistema Actual

Obviamente, las ventajas de integrar estos dos módulos son muchas, debido a que, de esta forma, se necesitará utilizar un único programa para realizar la simulación. Se parte, además, con la ventaja de que ambos módulos utilizan la misma parte de

simulación para el entrenamiento de RR.NN.AA. por lo que utilizan el mismo tipo de estructuras tanto de arquitecturas de red como de conjuntos de entrenamiento.

Al realizar la integración, se van a utilizar dos módulos, uno para el diseño de la arquitectura y otro para la construcción del conjunto de entrenamiento empleando cada uno de ellos un AG distinto. Estos dos AA.GG. emplean como función de validación el módulo de entrenamiento de RR.NN.AA. que se basa en otro AG. Esto implementa un doble paralelismo en dos distintos niveles de abstracción del desarrollo de las RR.NN.AA. Un primer nivel, más básico, de entrenamiento de una arquitectura concreta con un conjunto de entrenamiento concreto y, un segundo nivel, más abstracto, en el que se realizan las tareas de diseño de las RR.NN.AA. que se entrenan en el primer nivel.

4.8.2 Estrategia Cooperativa

Como se puede ver en el capítulo de resultados conseguidos sobre las propuestas de los puntos 4.5 y 4.6 de este capítulo, los módulos de diseño de la arquitectura y de construcción del conjunto de entrenamiento consumen muchos recursos de computación, a la vez que necesitan de bastante tiempo de simulación para explorar todas las posibilidades de diseño que se les presentan.

Es, por tanto, difícil pensar en la integración de estas dos piezas del sistema en una única aplicación que funcione sobre una única computadora. Las necesidades, tanto de memoria para las múltiples poblaciones, como de potencia de procesador para la simulación de los AA.GG., encarecerían demasiado el precio de la máquina necesaria para que el sistema consiguiese resultados en tiempos aceptables.

En esta situación, se ha optado por la utilización de varias computadoras de precio reducido conectadas por una red mediante el protocolo TCP/IP (Transfer Control Protocol/Internet Protocol) en la que los módulos se organizan en una arquitectura Cliente-Servidor. El precio a pagar por el ahorro en ordenador es la implementación de un programa servidor o gestor que organice el trabajo de los dos módulos de diseño de RR.NN.AA., que antes trabajaban por separado y módulos de comunicaciones para el paso de información entre el servidor y los clientes. Cada una de las partes del programa que funcionan como clientes se ejecutan en una computadora de la red y envían sus

resultados a la computadora en la que se ejecuta el programa servidor que permite el intercambio de datos entre los dos módulos (ver Figura 4.24).

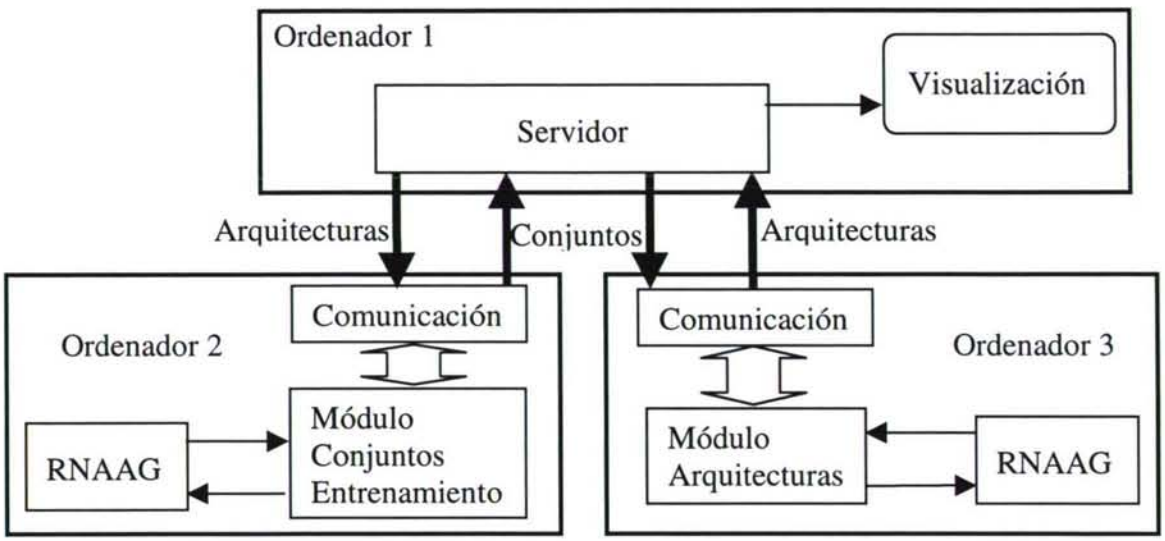


Figura 4.24.- Sistema Distribuido

El funcionamiento general del sistema partirá entonces del programa servidor que inicializará las poblaciones de los dos clientes de forma que cada uno ajuste las características de arquitectura y conjuntos de entrenamiento. Inicialmente, cada uno de los módulos comienza con la misma información. El módulo de arquitecturas utiliza la serie temporal completa para el entrenamiento de RR.NN.AA. y el módulo de diseño de los conjuntos de entrenamiento utiliza una arquitectura inicial fija.

La estrategia cooperativa propuesta en este punto consiste en un intercambio periódico de mejoras entre los dos módulos principales, a través del módulo servidor. Cuando el módulo de diseño de arquitectura encuentra una arquitectura de mejor adaptación se lo comunica al módulo servidor y le envía todos los datos referentes a este individuo. El módulo servidor contacta con el módulo que diseña el conjunto de entrenamiento para que cambie la arquitectura de las RR.NN.AA. que utiliza para la evaluación de los conjuntos de entrenamiento, aprovechándose este módulo del avance realizado por el otro. En el caso de que sea el módulo de diseño de conjuntos de entrenamiento el que encuentre un conjunto de entrenamiento mejor, se comunica con el servidor para enviarle la secuencia de datos de la serie que se deben utilizar para entrenar las RR.NN.AA. El módulo servidor contactará con el módulo de diseño de arquitecturas para enviarle la información del nuevo conjunto de entrenamiento que será el que se utilice desde ese momento en la evaluación de arquitecturas.

Como consecuencia de esta estrategia cooperativa, cuanto más tiempo pase de simulación, las redes generadas por los módulos de entrenamiento de RR.NN.AA. mediante AA.GG. alcanzarán errores menores, al aprovecharse de las mejoras generadas por los dos módulos de diseño.

En esta nueva aproximación, se van a utilizar los tres módulos que se han desarrollado hasta ahora, el de entrenamiento básico, el de ajuste de la arquitectura y el de diseño del conjunto de entrenamiento. Hasta ahora, el módulo de entrenamiento básico es utilizado por los otros dos como función de validación por lo que ya está implementada la comunicación entre ellos. Lo que se necesita ahora es comunicar los otros dos módulos de forma que sean capaces de intercambiar los resultados que vayan consiguiendo.

Se va a utilizar una arquitectura cliente-servidor que se encargue de la comunicación de la información entre los distintos módulos. A cada uno de estos módulos, se le añade un subsistema de comunicaciones que se encarga de la transferencia de datos. Como protocolo de comunicaciones se va utilizar el protocolo TCP/IP ya que su utilización está muy extendida. Este protocolo es el que se utiliza en la red Internet y también es el que utiliza la Universidade da Coruña en su red de investigación para la conexión a Internet. Esto, potencialmente, permitiría la utilización en las simulaciones del sistema de cualquier computadora conectada a Internet.

El sistema consta de un programa Servidor, de control y visualización centralizada de la evolución del entrenamiento de las redes. Este programa se encarga de la asignación de tareas a las computadoras, el paso de información entre módulos y la visualización de los resultados que le envían los distintos módulos.

Para la comunicación entre las computadoras se utilizan dos tipos de protocolos, el protocolo UDP (User Datagram Protocol) para la localización de computadoras en la red, que no realiza confirmación de llegada de mensajes, siendo así más rápido y el protocolo TCP (Transport Control Protocol) que sí realiza confirmación de llegada de datos, utilizado para la transmisión de información entre los módulos.

El sistema Servidor, al comenzar a funcionar, contacta con los programas clientes por medio del protocolo UDP de forma que se les identifica, se les asigna una tarea, ajuste de arquitecturas o diseño del conjunto de entrenamiento, y se establece un

canal de comunicación permanente a través del protocolo TCP para realizar una transmisión segura de los datos de la simulación. Una vez que todos los clientes han empezado a simular sus respectivas poblaciones, la información entre los módulos de ajuste de arquitecturas y de diseño del conjunto de entrenamiento pasa a través del servidor que se encarga de que lleguen al módulo de destino y de visualizar la información de las mejoras que se producen en cada módulo.

El enfoque cooperativo con que se realiza la integración de los módulos de ajuste de arquitecturas y de diseño del conjunto de entrenamiento exige realizar una serie de cambios en cada uno de los módulos, de forma que realicen su tarea dentro de un objetivo común. El primero de estos cambios, comentado en el punto anterior, es que cada módulo tenga la posibilidad de enviar y recibir datos, es decir de comunicarse.

El segundo de los puntos es que cada uno de los módulos pueda aprovechar los logros conseguidos por el otro. Así, si el módulo de ajuste de arquitecturas, en cada iteración, utiliza un conjunto de entrenamiento más compacto y optimizado, el entrenamiento en este módulo será más rápido y conseguirá arquitecturas con mayor ajuste. Por otro lado, si el módulo de diseño del conjunto de entrenamiento utiliza para validar las subseries mejores arquitecturas de red, el ajuste de las RR.NN.AA. que utilizan los conjuntos de entrenamiento también será mayor.

Sin embargo, la utilización de estos datos exige realizar ciertos cambios en el funcionamiento de ambos módulos. Hasta ahora, el módulo de ajuste de arquitecturas entrenaba las RR.NN.AA. con la misma serie temporal, mientras que el módulo de diseño del conjunto de entrenamiento buscaba las mejores subseries entrenando redes con una arquitectura fija.

En el sistema integrado o cooperativo, los dos módulos han de ser más flexibles. Para ello, el módulo de ajuste de arquitecturas comenzará las simulaciones utilizando la serie temporal completa. Cuando el módulo de conjuntos de entrenamiento le remita, a través del servidor, un conjunto de entrenamiento de mejores cualidades, lo sustituirá en el entrenamiento de las redes. El módulo de conjuntos de entrenamiento funciona de forma similar; comienza la evaluación de sus conjuntos de entrenamiento utilizando redes con una arquitectura inicial predefinida y, según el módulo de arquitecturas vaya

consiguiendo mejoras, se las envía a través del servidor, y este módulo sustituye la arquitectura de las redes que utiliza en la simulación.

En este punto, se ha expuesto cómo el sistema final se ha distribuido en una red de ordenadores, utilizando un mínimo de tres equipos, en la tarea de desarrollar, de forma integrada y semi-automática, una RNA optimizada para la resolución de un determinado problema.

Las comunicaciones entre módulos son el punto más importante a afrontar en este momento, dependiendo en gran medida el éxito del sistema de la implementación de estos nuevos módulos y su integración con los módulos de diseño de la RNA y de simulación de los AA.GG.

En el siguiente capítulo, Resultados, se pueden comprobar las pruebas empíricas realizadas con el sistema para validar los conceptos expuestos en este capítulo.

Capítulo 5: Resultados

5. Resultados

En este capítulo, se presentan los resultados de las pruebas realizadas para evaluar el rendimiento de las propuestas explicadas en el capítulo anterior. Se va a seguir la estructura del capítulo anterior en cuanto a puntos para poder ver la evolución de los resultados según se van añadiendo nuevas características al sistema. Se comienza probando el sistema básico de entrenamiento de RR.NN.AA. mediante AA.GG. A continuación, se le añade la potencialidad de selección automática de las funciones de activación para cada neurona de la red. Una vez conseguido el entrenamiento de las redes, en el siguiente punto, se realiza un estudio de los parámetros del AG que son más adecuados para el tipo de problemas que se afronta en esta tesis, el entrenamiento de RR.NN.AA. recurrentes. Después de esto, se realizan las pruebas para los módulos de automatización del desarrollo de la arquitectura de las RR.NN.AA. y de diseño de los conjuntos de entrenamiento. En el siguiente punto, se evalúa el modelo de atenuación temporal como referencia para el entrenamiento de nuevos modelos de RR.NN.AA. utilizando AA.GG. En el último punto, se realizan las pruebas finales en red de la integración de todos los módulos del sistema trabajando de forma cooperativa en la consecución de RR.NN.AA. recurrentes de arquitectura ajustada y conjunto de entrenamiento mínimo.

Para acabar, se presentan dos puntos de comparación con otras técnicas de predicción de series temporales. Por un lado, se comprara el funcionamiento del sistema final con RR.NN.AA.RR. entrenadas con los algoritmos BackPropagation Through Time y Real Time Recurrent Learning. Por otro, se comparan las predicciones obtenidas con el sistema con las conseguidas a partir de los modelos ARIMA de las series temporales utilizadas en las pruebas.

Para la realización de las pruebas, se utilizan, como conjuntos de entrenamiento, las series temporales presentadas en el Capítulo 2: la serie de manchas solares y la serie de producción de tabaco en los EE.UU. Ya que, como ya se ha comentado, son ejemplos típicos utilizados para la predicción de series temporales en la literatura de Análisis de Series Temporales, permitiendo la comparación de los resultados con los de otros autores y técnicas.

Para la cuantificación del error cometido por las RR.NN.AA. que se desarrollan y entrenan en este capítulo, se calcula, en cada prueba, el Error Cuadrático Medio (en adelante ECM) y el Error Medio (en adelante EM) de las salidas de la red para cada una de las series utilizadas.

Todas las pruebas se han realizado sobre ordenadores con procesador Pentium II a una velocidad de reloj de 233MHz, con una capacidad de memoria de 64Mb con chips de tipo DIMM de 100MHz, con una velocidad de acceso de 10ns. Los equipos disponen de 512Kbytes de memoria cache de segundo nivel.

5.1 Entrenamiento de RR.NN.AA.RR. mediante Ajuste de Pesos

En este punto, se muestran los resultados conseguidos con el sistema más simple, en el que se implementa solamente el ajuste automático de pesos. Este apartado va a servir como punto de referencia para comparar los resultados obtenidos con el sistema al ir añadiendo nuevas funcionalidades.

Para conseguir valores fiables, como la inicialización de un AG, se realiza con una población aleatoria. Para cada una de las 18 pruebas con distintos parámetros que se realizarán en este apartado, se van a repetir 3 veces. Realizando un total de 54 pruebas para cada una de las series.

Si se dispone de un sistema como el que se comenta en este punto, que solamente ajusta los pesos de una RNA, ya sea con un método de descenso de gradiente o, como en este caso, con un método evolutivo, sería necesario comprobar los valores de una serie de parámetros para ajustar el diseño de la red.

Los parámetros más comunes serían el número de neuronas (siendo la arquitectura de la red recurrente no es necesario especificar el número de capas ocultas), las funciones de activación para las neuronas y el número de iteraciones que se deja a la red para que emita una salida válida (al ser la red recurrente). Para estas pruebas, se han utilizado los siguientes valores buscando una indicación de la mejor configuración:

Número de Neuronas	2	4	6
Funciones de Activación	Lineal	Hiperbólica Tangente	Sigmoide
Número de Iteraciones	1	5	

De momento, los parámetros de la simulación del AG se han fijado para que no afecten al estudio de los parámetros de diseño de la RNA. Los valores utilizados son los siguientes.

Ciclos de Simulación	Número de Cruces	Número de Mutaciones
25.000	1	1

5.1.1 Manchas Solares

La duración total de las pruebas con esta serie fue de 39.841 segundos de simulación en el Pentium II. La simulación del entrenamiento de la RNA era el único proceso en la computadora durante el tiempo de la prueba. Los errores mínimos se consiguieron con la siguiente configuración:

Neuronas	Función de Activación	Iteraciones	ECM	EM	Segundos
4	Lineal	1	0,0071182	0,062089	231

Sin embargo, para ganar en fiabilidad e independencia sobre las pruebas, como se indicaba en la introducción de este punto, los resultados se van a guiar por los grupos de 3 pruebas utilizando los mismos parámetros. En este caso, la configuración de parámetros que consiguió un menor nivel de error es el que se muestra a continuación:

Neuronas	Función de Activación	Iteraciones	ECM	EM	Segundos
6	Lineal	1	0,0089187	0,066056	361
6	Lineal	1	0,0079474	0,065201	372
6	Lineal	1	0,0086316	0,067245	332
Valores Medios de las 3 tres pruebas			**0,0084992**	**0,0661673**	**355**

Con objeto de poder validar qué valores de esta simulación son los más adecuados para la solución de este tipo de problemas, se han calculado los errores y tiempos medios alcanzados con los valores utilizados en cada uno de los parámetros, independientemente de los valores de los demás parámetros.

Para las funciones de activación, los resultados medios son los siguientes:

Neuronas	Función de Activación	Iteraciones	ECM	EM	Segundos
-	Lineal	-	0,0126313	0,0814079	471,5
-	Hiperbólica	-	0,0181284	0,10143	962,5
-	Exponencial	-	0,018953	0,1050742	779,4

Con respecto al número de neuronas utilizado, los resultados medios son:

Neuronas	Función de Activación	Iteraciones	ECM	EM	Segundos
2	-	-	0,0158828	0,0956448	475,3
4	-	-	0,0159771	0,0929725	742,5
6	-	-	0,0178527	0,0992949	995,5

Y, con respecto al número de iteraciones, los resultados medios son:

Neuronas	Función de Activación	Iteraciones	ECM	EM	Segundos
-	-	1	0,018453	0,0999467	325,3
-	-	5	0,0146887	0,0919948	1150,3

5.1.2 Producción de Tabaco

La duración total de las pruebas con esta serie fue de 18.925 segundos de simulación en el mismo equipo Pentium II. Igual que en el caso anterior, la simulación del entrenamiento de la RNA era el único proceso en la computadora durante ese tiempo.

Los errores mínimos se consiguieron con la siguiente configuración:

Neuronas	Función de Activación	Iteraciones	ECM	EM	Segundos
6	Exponencial	5	0,0069356	0,059954	841

Como en el caso anterior, los resultados se van a guiar por la media de los grupos de 3 pruebas utilizando los mismos parámetros. En este caso, la configuración que consiguió un menor nivel de error es la siguiente.

Neuronas	Función de Activación	Iteraciones	ECM	EM	Segundos
2	Lineal	5	0,0074528	0,061933	200
2	Lineal	5	0,0073314	0,06279	215
2	Lineal	5	0,0073789	0,062634	224
Valores Medios de las 3 tres pruebas ⇨			**0,0073877**	**0,0624523**	**213**

Como en el caso anterior, para poder validar genéricamente los valores de cada parámetro, se han calculado los errores y tiempos medios alcanzados con cada valor utilizado en cada uno de los parámetros independientemente de los valores de los demás parámetros.

Para las funciones de activación, los resultados medios son los siguientes:

Neuronas	Función de Activación	Iteraciones	ECM	EM	Segundos
-	Lineal	-	0,00838343	0,06726208	258,6
-	Hiperbólica	-	0,01482602	0,09217789	454,5
-	Exponencial	-	0,0135	0,0871123	338,2

Con respecto al número de neuronas utilizado, los resultados medios son:

Neuronas	Función de Activación	Iteraciones	ECM	EM	Segundos
2	-	-	0,01075546	0,07843682	226,8
4	-	-	0,0116602	0,07933413	326,2
6	-	-	0,01429378	0,08878133	498,4

Y, con respecto al número de iteraciones, los resultados medios son:

Neuronas	Función de Activación	Iteraciones	ECM	EM	Segundos
-	-	1	0,00908962	0,07160126	140,4
-	-	5	0,01538334	0,09276693	560,5

5.1.3 Consideraciones del Entrenamiento mediante AA.GG.

En estas pruebas iniciales se plantean ciertos aspectos que indican que la tarea de diseño de una RNA no es una tarea fácil. En el caso del entrenamiento para la predicción de la serie de las manchas solares, tras 40.000 segundos de simulación existe unanimidad en que con la función de activación lineal y con una iteración es como se alcanzan los mejores resultados. En cuanto al número de neuronas, en la mejor simulación se utilizan 4 mientras que en el mejor grupo de 3 pruebas se mejora utilizando 6, pudiendo concluir, como mucho, que es necesario utilizar un número de neuronas superior a 2. Sin embargo, en los resúmenes realizados por parámetro individual, los mejores resultados se consigue si se utiliza la función lineal, pocas neuronas y cinco iteraciones, coincidiendo con los datos anteriores sólo en el tipo de función de activación.

En el caso de la serie de producción de tabaco, la simulación tarda 19.000 segundos para conseguir unos resultados igual de desconcertantes. En este caso, ni la función de activación , ni el número de neuronas del mejor caso y del mejor grupo de 3 coinciden. Sólo se consigue uniformidad en el número de iteraciones de la red que en este caso son 5. Estos resultados tampoco están en sintonía con los resúmenes por parámetro que indican como mejores opciones la función lineal, pocas neuronas y una iteración.

Obviamente, existen relaciones no-lineales entre los valores de estos tres parámetros que impiden ajustarlos de forma individual. Es necesario la configuración conjunta de los tres de forma que converjan a una configuración óptima. Es por esto que, cuanto más automático sea el proceso, menos tiempo tiene que emplear el diseñador en comprobar de forma exhaustiva los valores de configuración. En este caso, como se refleja en este apartado, se han empleado más de 16 horas de simulación para conseguir los mejores valores de entre sólo 8 distintos valores, para 3 parámetros de la RNA, para únicamente 2 series temporales. En el siguiente apartado, se intentará automatizar la selección de las funciones de activación reduciendo la labor de diseño y acelerando el proceso de simulación al reducir el número de posibilidades disponibles.

5.2 Ajuste de Pesos y Selección Automática de Funciones de Activación

En este punto, se añade la selección automática de la función de activación con objeto de facilitar el trabajo del diseñador al tener que dedicar los esfuerzos a configurar el resto de los parámetros. Los resultados de este apartado, se corresponden al punto de Entrenamiento Básico de RR.NN.AA. mediante AA.GG., donde se relata la codificación de los individuos y los operadores genéticos que se aplican en esta evolución del modelo.

Al realizar de forma automática la selección de la función de activación, tan solo se efectúan 6 pruebas distintas. Igual que en el caso anterior, cada prueba se ejecuta 3 veces de forma que se consiga una fiabilidad razonable en los resultados. Se realizan, por tanto, un total de 18 pruebas para cada una de las series.

En las tablas de resultados que se exponen a continuación, la columna funciones de activación ha cambiado de significado con respecto al punto anterior. Ahora, representa las funciones de activación seleccionadas por el sistema para cada una de las neuronas de la red. En ella, se presenta una cadena de letras que simbolizan las funciones de activación siendo U la función umbral, L la función lineal, E la función exponencial y H la Hiperbólica Tangente. En esta cadena de letras, la primera función será la de la neurona de entrada, la última función será la de la neurona de salida, y el resto de funciones se corresponden con el resto de neuronas de que consta la red.

Los otros parámetros de funcionamiento, tanto del AG como de la RNA, son los mismos que ya se han comentado en el punto 5.1.

5.2.1 Manchas Solares

En este caso el tiempo total de pruebas se redujo a 18.436 segundos de simulación en las mismas condiciones que el caso anterior.

Los errores mínimos se consiguieron con la siguiente configuración:

Neuronas	Iteraciones	ECM	EM	Segundos	Funciones de Activación
2	1	0,011668	0,07311	226	H E H H

Teniendo en cuenta la media de ajuste de los grupos de 3 pruebas, la configuración que consiguió un menor nivel de error es la siguiente:

Neuronas	Iteraciones	ECM	EM	Segundos	Funciones de Activación
2	1	0,013661	0,08439	219	L H E L
2	1	0,011668	0,07311	226	H E H H
2	1	0,012554	0,08023	243	H L H H
		0,012627	**0,07924**	**229,3**	Valores Medios-3 pruebas

Para validar genéricamente los valores de cada parámetro, se han calculado los errores y tiempos medios alcanzados con cada valor utilizado en cada uno de los parámetros, número de neuronas e iteraciones, independientemente de los valores de los demás parámetros.

Los resultados medios por número de neuronas son los siguientes:

Neuronas	Iteraciones	ECM	EM	Segundos
2	-	0,016675	0,09501167	652,5
4	-	0,02491103	0,1227275	1019,5
6	-	0,030379	0,132585	1400,7

Y, con respecto al número de iteraciones, los resultados medios son:

Neuronas	Iteraciones	ECM	EM	Segundos
-	1	0,0232916	0,11168556	487,5
-	5	0,02468509	0,12186389	1560,9

5.2.2 Producción de Tabaco

En el caso de la serie de producción de tabaco, el tiempo total de pruebas fue de 6.452 segundos en las condiciones de simulación ya descritas. En este caso, los errores mínimos se consiguieron con la siguiente configuración.

Neuronas	Iteraciones	ECM	EM	Segundos	Funciones de Activación
4	1	0,00937	0,07644	140	L L E E H E

Teniendo en cuenta la media de ajuste de los grupos de 3 pruebas, la configuración que consiguió un menor nivel de error es la siguiente:

Neuronas	Iteraciones	ECM	EM	Segundos	Funciones de Activación
6	1	0,02013	0,11439	200	E H E E E E E E
6	1	0,01459	0,09364	189	L E E L E H E L
6	1	0,014311	0,0968	185	E E L E E E L L
	⇒	**0,016343**	**0,10161**	**191,3**	Valores Medios-3 pruebas

Para cada valor de cada parámetro, se han calculado la media de los errores y tiempos alcanzados, número de neuronas e iteraciones, independientemente de los valores de los demás parámetros. Los resultados medios por número de neuronas son los siguientes:

Neuronas	Iteraciones	ECM	EM	Segundos
2	-	0,01787668	0,105645	205,3
4	-	0,01837667	0,10639833	340,2
6	-	0,01782953	0,107225	529,8

Y, con respecto al número de iteraciones, los resultados medios son:

Neuronas	Iteraciones	ECM	EM	Segundos
-	1	0,01664346	0,10258667	149
-	5	0,0194118	0,11025889	567,9

5.2.3 Problema de Ajuste de Funciones de Activación

Obviamente, la evolución propuesta en este punto no alcanza los resultados esperados. Utilizando la selección automática de la función de activación, los resultados de las mejores redes conseguidas son considerablemente peores que los conseguidos especificando los valores de las funciones de activación de forma conjunta para todas las neuronas de la red, manteniendo el resto de los parámetros. En la página siguiente, se puede ver una comparativa de los resultados ya mostrados en los apartados anteriores, indicando en tanto por ciento el aumento del nivel de error.

Es de destacar que, al fijar las funciones de activación, los casos de mejor ajuste eran los que incluían las funciones de activación lineales. Sin embargo, en el caso de la selección automática, las funciones de activación seleccionadas son, mayoritariamente y por este orden: la exponencial y la hiperbólica tangente. Al no alcanzar los valores de error esperados, cabe preguntarse si estas dos situaciones, los niveles de error alcanzados y la no aparición de las funciones de activación lineales, están relacionadas.

Series Temporales	Tipo de Prueba	ECM con funciones fijas	ECM con selección automática de funciones	Diferencia
Manchas Solares	Mejor	0.0071	0.0116	+63%
	Grupo de 3	0.0085	0.0126	+48%
Producción de Tabaco	Mejor	0.0069	0.0093	+34%
	Grupo de 3	0.0074	0.0163	+120%

El problema tiene dos posibles causas: a).- que sea mejor mantener iguales las funciones de activación de la red en todas las neuronas o, b).- que la elección que realiza el sistema de las funciones de activación no sea la adecuada. Por lo comentado en el párrafo anterior, referido a la no aparición de la función lineal, es más plausible la segunda justificación; por lo que, para verificar esta hipótesis se han realizado pruebas adicionales. En estas pruebas se busca, por tanto, verificar que el empeoramiento del nivel de error no se debe a que se utilizan varias funciones de activación en la misma red y que sí se debe a una mala selección de estas funciones. Para comprobar esto, con la serie de producción de tabaco, se van entrenar RR.NN.AA.RR. fijando los valores de las funciones de activación de la red y utilizando la mejor configuración encontrada en el apartado anterior.

Neuronas	Función de Activación	Iteraciones
2	Lineal	5

Esta configuración de 2 neuronas ocultas más una de entrada y otra de salida, consiguió el menor error entre los grupos de 3 intentos. En estas pruebas adicionales, como se puede comprobar en la siguiente tabla, se van a combinar, para cada RNA, tres funciones de activación lineales con una función exponencial o hiperbólica, en las cuatro neuronas de la red, de forma que se pueda evaluar el error conseguido con cada combinación de funciones de activación.

HLLL	ELLL
LHLL	LELL
LLHL	LLEL
LLLH	LLLE

Se han realizado tres pruebas con cada una de las 8 configuraciones de funciones de activación, siendo el mejor individuo:

ECM	EM	Segundos	Funciones de Activación
0,007116	0,06033	235	L L H L

El mejor grupo de 3 pruebas tiene también la configuración anterior, siendo los valores de error conseguidos los siguientes:

Neuronas	Iteraciones	ECM	EM	Segundos	Funciones de Activación
2	1	0,007511	0,063009	210	L L H L
2	1	0,00928	0,07058	252	L L H L
2	1	0,007116	0,06033	235	L L H L
	⇨	0,007969	0,06463	232,3	Valores Medios-3 pruebas

Como se puede comprobar en el siguiente cuadro, los errores conseguidos vuelven a ser comparables a los conseguidos en las pruebas que utilizaban el mismo tipo de función de activación, la lineal, para todas las neuronas de la red.

	Tipo de Prueba	ECM con funciones fijas	ECM selección automática de funciones	ECM con variaciones de Funciones	Funciones de activación resultantes
	Mejor	0,0069	0,0093	0,0071	L L H L
Diferencia			+34%	+ 3%	
	Grupo de 3	0,0074	0,0163	0,0079	L L H L
Diferencia			+120%	+ 7%	

Se desestima entonces la hipótesis de que la bajada de rendimiento se debía a que las funciones de activación dentro de una red tenían que ser iguales, y se confirma que la forma en que el sistema actual escoge las funciones de activación no es la adecuada.

Queda por saber la razón del mal funcionamiento del sistema. La causa se encontró en el funcionamiento de las propias funciones de activación. En las redes que tienen funciones exponenciales e hiperbólicas, se comienza con un nivel de error inferior y el proceso de convergencia es progresivo desde un primer momento aunque, a la vista de las pruebas realizadas, saturan el entrenamiento con un nivel de error peor. Sin embargo, las redes que tienen funciones de activación lineales, comienzan con un nivel de error mayor y tardan un tiempo en empezar a disminuir su error pero, desde que comienza la disminución, ésta se realiza de forma mantenida consiguiendo, como se ha dicho, menores errores.

Debido a estos comportamientos, al mezclar en una RNA distintas funciones de activación, las redes que tienen un mayor número de neuronas con funciones de activación exponenciales o hiperbólicas, al tener un comportamiento inicial mejor, dominan en un primer momento la población, haciendo desaparecer aquellas redes con mayoría de funciones lineales, que darían mejor rendimiento en una fase más avanzada del entrenamiento a la que nunca se llega. Una vez que la población está dominada por redes con mayoría de funciones exponenciales e hiperbólicas ya no es posible que vuelvan a aparecer redes con mayoría de funciones lineales. Esto indica que, con el

sistema actual no es posible realizar el ajuste simultáneo de los parámetros de pesos de las conexiones y funciones de activación.

Otro punto interesante es el caso de la función exponencial. En el siguiente cuadro se presentan los resultados medios obtenidos para las cuatro configuraciones en las que se combinaron funciones lineales con la función exponencial. Como se puede apreciar en una de ellas, la que presenta la función exponencial en la neurona de salida tiene un nivel de error sustancialmente menor que en el resto de los casos, aproximándose al error conseguido al combinar las funciones lineales con la función hiperbólica. Estos resultados indican la importancia de las funciones de activación utilizadas y también de su ubicación dentro de las neuronas de la RNA.

ECM	EM	Segundos	Funciones de Activación
0,02759	0,14094	202	E L L L
0,03594	0,15274	210,3	L E L L
0,02600	0,12841	225,3	L L E L
0,00854	0,07163	205,3	L L L E

5.2.4 Consideraciones sobre la Selección Automática de Funciones de Activación

El problema que ha surgido con el ajuste de las funciones de activación exige replantear el sistema desarrollado. Como se ha comentado en el punto anterior, la búsqueda de las funciones de activación cae en un mínimo local representado por las redes con mayoría de funciones exponenciales o hiperbólicas, en vez de alcanzar el mínimo global que representa una configuración con mayoría de funciones lineales. Esta situación se debe al propio comportamiento de las funciones de activación, por lo que la variación en los parámetros del AG o la codificación de las redes no soluciona el problema.

La solución está en separar el proceso de ajuste de los pesos del proceso de ajuste de las funciones de activación. Es necesario que, para cada configuración de funciones de activación, se realice el ajuste de los pesos la RNA. Es decir, si se construye un AG que escoja una configuración de funciones de activación, la función de valoración de cada configuración de funciones de activación será una simulación del AG que ajusta los pesos de la RNA utilizando esas funciones de activación concretas. Se necesita pues un AG como valoración de los individuos de otro AG.

En el punto posterior en el que trata el ajuste de la arquitectura de la RNA, ya se utiliza esta técnica de valoración de individuos de un AG mediante el empleo de otro AG. Debido a esto, se pospone la inclusión del ajuste de las funciones de activación hasta este punto de desarrollo del sistema en el que se espera conseguir mejoras en el nivel de error de las redes ya desarrolladas.

5.3 Estudio de Parámetros del AG

Hasta este punto, en el sistema se han probado distintas configuraciones de parámetros de las RR.NN.AA. para intentar conseguir el mínimo error en el funcionamiento de la red. Sin embargo, para realizar el entrenamiento de las RR.NN.AA., se utiliza un AG que todavía no ha sido ajustado para que consiga el funcionamiento óptimo en el desarrollo de las RR.NN.AA.

En este punto, se van a realizar pruebas sobre las distintas facetas comentadas en el punto del mismo nombre del capítulo anterior. Para comprobar la efectividad de las pruebas de estos parámetros del AG en minimizar el error de las redes, se van a mantener fijos los parámetros de diseño de las RR.NN.AA. Las configuraciones de RNA que se van a utilizar son los mejores individuos conseguidos en el primer apartado para cada una de las series. Estas configuraciones se muestran a continuación:

Series Temporales	Neuronas	Función de Activación	Iteraciones
Manchas Solares	6	Lineal	1
Producción de Tabaco	2	Lineal	5

Para cada evaluación de la RNA, se mantiene la cifra de 25.000 iteraciones o generaciones del AG, lo que servirá para poder comparar los resultados con las pruebas realizadas anteriormente.

5.3.1 Población, Cruce y Mutación

Como ya se comentó en el capítulo del Sistema, existe una fuerte relación entre el tamaño de la población y los operadores de cruce y mutación, por lo que las pruebas sobre estos tres parámetros se realizan de una forma conjunta. Los valores que se van a utilizar para las pruebas de estos parámetros son los siguientes:

Tamaño de la Población	100	500	1000
Número de Cruces / Generación	1	5	
Número de Mutaciones / Generación	1	5	

5.3.1.1 Manchas Solares

La simulación total de todas las pruebas realizadas abarca 40.798 segundos. Los errores mínimos se consiguieron con la siguiente configuración:

Población	**Cruces**	**Mutaciones**	**ECM**	**EM**	**Segundos**
100	5	5	0,0074499	0,062091	1581

En los resultados conseguidos con los grupos de 3 pruebas, utilizando los mismos parámetros de AG, la configuración que consiguió un menor nivel de error es la siguiente:

Población	Cruces	Mutaciones	ECM	EM	Segundos
100	5	5	0,0074499	0,062091	1581
100	5	5	0,0078453	0,063124	1562
100	5	5	0,0079521	0,066548	1568
Valores Medios de las 3 tres pruebas			**0,0077491**	**0,063921**	**1570,3**

5.3.1.2 Producción de Tabaco

La simulación total de todas las pruebas realizadas abarca 30.793 segundos de tiempo de simulación. Los errores mínimos se consiguieron con la siguiente configuración:

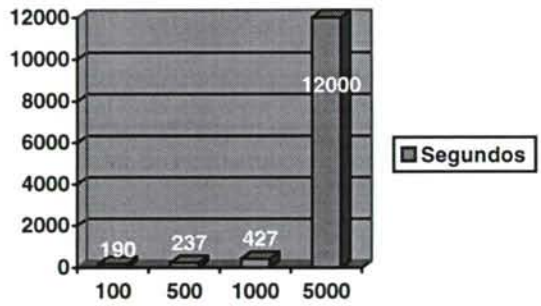
Población	Cruces	Mutaciones	ECM	EM	Segundos
100	1	5	0,0068864	0,0659696	392

Nuevamente, se realizan pruebas con los grupos de 3 pruebas, la configuración que consiguió un menor nivel de error es la siguiente:

Población	Cruces	Mutaciones	ECM	EM	Segundos
500	5	5	0,0070144	0,0637418	1202
500	5	5	0,00719485	0,0618727	1235
500	5	5	0,00710843	0,0629592	1218
Valores Medios de las 3 tres pruebas			**0,00710589**	**0,0628579**	**1218,3**

5.3.1.3 Pruebas Adicionales

Con respecto al tamaño de la población, no se han realizado pruebas con valores más allá de 5000 individuos, ya que el tiempo de simulación se dispara al alcanzar este tamaño. Esto es debido a que, cuanto mayor es la población, mayor es el tiempo que se emplea en realizar las búsquedas y reordenaciones de individuos. El límite para trabajar con comodidad viene fijado por la potencia del ordenador en el que se está trabajando y por el problema que se intenta resolver; sin embargo, el tiempo de simulación crece exponencialmente, como se puede ver en la siguiente gráfica. En el caso más simple, en el que se aplica un cruce y una mutación, el tiempo de simulación para la serie de producción de tabaco tiene la siguiente evolución.



5.3.1.4 Consideraciones sobre Población, Cruce y Mutación

Se han comprobado, aleatoriamente, configuraciones mayores de número de cruces y mutaciones, llegando a utilizar hasta 50 cruces o 50 mutaciones por generación, no consiguiendo mejores resultados experimentales. Además, estas pruebas necesitan de un tiempo de simulación demasiado grande como para realizar una investigación exhaustiva, suponiendo cada prueba individual más de 3 horas de simulación. En la siguiente tabla, se pueden observar los resultados conseguidos en este punto, comparados con los mejores resultados conseguidos hasta ahora en la predicción de las dos series temporales. Es interesante destacar que, aunque todavía no se ha conseguido un mejor rendimiento en las pruebas individuales, el error medio de los grupos de tres pruebas sí que ha disminuido de forma apreciable, indicando una mejora en la estabilidad del proceso de entrenamiento.

Series Temporales	Tipo de Prueba	ECM con funciones fijas	ECM con ajuste de parámetros del AG	Diferencia
Manchas Solares	Mejor	0,0071	0,0074	+ 4%
	Grupo de 3	0,0085	0,0077	- 9%
Producción de Tabaco	Mejor	0,0069	0,0069	0%
	Grupo de 3	0,0074	0,0071	- 4%

Por último, indicar que las pruebas sobre los tipos de cruce y los procesos de selección y sustitución se van a realizar con los mejores valores conseguidos en este apartado. En las pruebas realizadas para las dos series temporales coinciden los mejores resultados en cuanto a número de cruces y de mutaciones a realizar. Sin embargo, difieren en cuanto al tamaño de la población. Sabiendo que la diferencia por este parámetro es pequeña y que sería conveniente un tamaño de población reducido, para acelerar el proceso de entrenamiento se va a utilizar un tamaño de población de 100 individuos en la realización de las pruebas de los restantes parámetros del AG,

5.3.2 Tipos de Cruce

Hasta este momento, se ha utilizado el cruce de un solo punto de corte. En este punto, se muestran los resultados conseguidos con el cruce de dos puntos de corte y el cruce uniforme para las dos series temporales.

5.3.2.1 Manchas Solares

Para esta serie, el mejor resultado conseguido hasta ahora es el del punto 5.3.1.1 donde, con la misma configuración de 100 individuos, 5 cruces y 5 mutaciones, se alcanzó un ECM de 0,0074 en prueba individual y 0,0077 en grupo de 3, con una media de tiempo por prueba individual de 1570 segundos.

En la siguiente tabla, se muestran los resultados para los dos tipos de cruce en pruebas individuales y en grupos de tres.

Tipo de Prueba	Tipo de Cruce	ECM	EM	Segundos
Mejor	2 puntos	0,0075652	0,06368	1611
	Uniforme	0,0099654	0,069564	1617
Grupo de 3	2 puntos	0,0081543	0,0666501	1589
	Uniforme	0,0120828	0,0719683	1615,3

5.3.2.2 Producción de Tabaco

Para esta serie, el mejor resultado conseguido hasta ahora es el del punto 5.3.1.2 donde, con la configuración de 100 individuos, 1 cruce y 5 mutaciones, se alcanzó un ECM de 0,0069 en prueba individual en 392 segundos; mientras que, con una configuración de 500 individuos, 5 cruces y 5 mutaciones un ECM de 0,0071 en grupo de 3, con una media de tiempo individual de 1218 segundos.

Como ya se ha indicado en un punto anterior, debido a las diferencias mínimas existentes, y en aras de mantener una cierta homogeneidad en las pruebas, se van a utilizar, con la serie de producción de tabaco, los mismos parámetros que con la serie de manchas solares. Se parte entonces de los valores de configuración de 100 individuos, 5 cruces y 5 mutaciones; y se comprobará si estos resultado se mejoran al modificar el tipo de operador de cruce. Con esta configuración, los mejores resultados base eran:

Tipo de Cruce	ECM	EM	Segundos
1 punto	0,0073129	0,0625515	790

Y la media de las tres pruebas:

Tipo de Cruce	ECM	EM	Segundos
1 punto	0,0073657	0,0630108	810,6

Sustituyendo el cruce de un punto de corte por el de dos y por el uniforme, los mejores resultados individuales y en grupos de tres son:

Tipo de Prueba	Tipo de Cruce	ECM	EM	Segundos
Mejor	2 puntos	0,0073157	0,0628654	835
	Uniforme	0,0075515	0,063578	841
Grupo de 3	2 puntos	0,0074499	0,0626594	819
	Uniforme	0,0075940	0,06309	838,3

5.3.2.3 Consideraciones sobre el Tipo de Cruce

Como se puede observar, en ninguna de las dos series se mejora el entrenamiento de la red al modificar el operador de cruce. El que se comporta peor es el cruce uniforme, haciendo descender en gran medida el ajuste para la serie de manchas solares, mientras que el cruce de dos puntos consigue resultados comparables.

A la vista de lo conseguido en este punto, se puede concluir que la operación de inversión no es necesaria ya que el operador de cruce de dos puntos de corte no aporta un incremento de rendimiento. Es decir, los bloques constructivos del AG están suficientemente próximos en el genoma como para no verse afectados negativamente por el operador de cruce. De este punto, también se puede concluir, en cierta manera, que la codificación utilizada para las RNA es adecuada en cuanto a la ubicación de bloques constructivos se refiere.

Por otro lado, el empeoramiento del rendimiento, fruto de la utilización del operador de cruce uniforme, indica una fuerte relación entre los valores de los genes. Este comportamiento es totalmente lógico ya que la mayor parte del genoma está constituido por la matriz de pesos, estando clara la fuerte relación que tiene que existir entre los valores de pesos que salen o entran a una determinada neurona. Por tanto, es

normal que las redes se comporten peor si, constantemente, en cada operación de cruce, se mezclan aleatoriamente los pesos de dos redes en cualquier punto de su estructura.

5.3.3 Selección

Hasta ahora, la selección de los individuos que se utilizan en la operación de cruce se realizaba de forma aleatoria en la población. En este punto, se van a estudiar dos estrategias distintas de selección de individuos. La tradicional selección de la "ruleta" o de "Montecarlo" y la selección por "torneo".

Debido a los resultados obtenidos en el punto anterior, el tipo de corte utilizado va a ser el de un punto de corte y se van a mantener el resto de parámetros del AG. Los mejores valores conseguidos hasta ahora siguen siendo los de los puntos 5.3.1.1 y 5.3.1.2.

5.3.3.1 Manchas Solares

En la siguiente tabla, se presentan los mejores resultados para los dos tipos de selección tanto en pruebas individuales como en grupos.

Tipo de Prueba	Selección	ECM	EM	Segundos
Mejor	Montecarlo	0,0079654	0,070345	1626
	Torneo	0,0074688	0,062978	1974
Grupo de 3	Montecarlo	0,0088177	0,0716876	1629,3
	Torneo	0,0084130	0,0693186	1973,3

5.3.3.2 Producción de Tabaco

Para esta otra serie también se presenta una tabla con los mejores resultados de los dos tipos de selección, conseguidos en las pruebas individuales y en grupos de tres.

Tipo de Prueba	Selección	ECM	EM	Segundos
Mejor	Montecarlo	0,0074687	0,061358	879
	Torneo	0,0073687	0,062875	1055
Grupo de 3	Montecarlo	0,0075521	0,06203	877
	Torneo	0,0075391	0,063532	1050

5.3.3.3 Consideraciones sobre el Tipo de Selección

Con los tipos de selección utilizados tampoco se consigue alcanzar el nivel de error ya conseguido con la selección aleatoria. Es de destacar que los resultados obtenidos no son comparables, sino significativamente peores. Además, los dos tipos de operadores de selección estudiados en este punto suponen una sobrecarga computacional del sistema de simulación de AG debido a que, en el caso de la selección de tipo "Montecarlo", es necesario repartir las probabilidades de cada individuo en función de su ajuste y, en el caso de la selección por "Torneo", es necesario seleccionar un conjunto de individuos y compararlos para conseguir la selección final. En el caso de este último tipo, en las pruebas realizadas se observa un incremento de tiempo de simulación de entre el 25 y el 33%.

En este caso, la selección del tipo "Montecarlo" no mejora los resultados de la selección aleatoria porque el problema de entrenamiento de RR.NN.AA. es un problema no lineal. Es decir, la consecución de individuos mejores no depende, en exclusiva, del cruce de los mejores individuos, sino que es necesario un cierto azar en la selección para dar la posibilidad de que se reúnan, en un individuo, genes que, por separado, no generan un nivel alto de ajuste, pero que sí que lo hacen cuando están juntos.

Siguiendo el razonamiento del punto anterior, la selección del tipo "Torneo" no presenta tan malos resultados porque selecciona al azar en la población los individuos que van a intervenir en el torneo pero, por otro lado, realiza la selección de los

individuos del torneo basándose en un mejor ajuste, por lo que su comportamiento es un poco peor que la selección aleatoria.

5.3.4 Sustitución

En este punto, se hace un estudio de los métodos de sustitución de individuos de la población para dejar sitio a la nueva descendencia. En las implementaciones del AG utilizadas hasta ahora se ha empleado la sustitución Darwinista o del peor adaptado, eliminando los peores individuos de la población en cada generación. Ahora, se va a probar la sustitución por "parecido error" y la sustitución de los padres. Como en puntos anteriores, se van a mantener los parámetros del AG utilizando la selección aleatoria.

5.3.4.1 Manchas Solares

Los mejores resultados conseguidos con los distintos tipos de sustitución mencionados en el párrafo anterior, tanto en pruebas individuales, como en grupos de tres, se presentan en la siguiente tabla:

Tipo de Prueba	Sustitución	ECM	EM	Segundos
Mejor	Parecido Error	0,0078354	0,0643455	2117
	Padres	0,0081346	0,070355	1622
Grupo de 3	Parecido Error	0,0079111	0,0676198	2114,6
	Padres	0,0093848	0,0746776	1626,6

5.3.4.2 Producción de Tabaco

Para el caso de la serie de producción de tabaco, los mejores resultados conseguidos con los distintos tipos de sustitución, tanto en pruebas individuales, como en grupos de tres, se presentan en la siguiente tabla:

Tipo de Prueba	Sustitución	ECM	EM	Segundos
Mejor	Parecido Error	0,0073218	0,0630472	1392
	Padres	0,0072688	0,061685	918
Grupo de 3	Parecido Error	0,0074072	0,063563	1407
	Padres	0,007434	0,0626136	916

5.3.4.3 Consideraciones sobre el Tipo de Sustitución

Las opciones de tipo de sustitución, probadas en este punto, tampoco mejoran los resultados conseguidos con la sustitución del peor individuo. Según los resultados de los últimos puntos, la peor sería la "sustitución de los padres" consiguiendo la sustitución de "parecido error" resultados intermedios. Hay que destacar que la sustitución de individuos por parecido error consume entre un 33 y un 50% más de tiempo que las otras dos, debido a que realiza una búsqueda, a través de la población, de los individuos de error parecido y, entre ellos, realiza la selección aleatoria del que se va a eliminar.

Las variaciones utilizadas en este punto de los tipos de sustitución son, normalmente, recursos que mantienen la diversidad genética dentro de la población. La falta de mejora en el ajuste final, con las dos series, puede denotar que no existe un problema grave de homogenización dentro de las poblaciones generadas en el AG, ya sea debido a la codificación utilizada o a la propia naturaleza del problema de entrenamiento de RR.NN.AA. mediante AA.GG. En el caso de que se diera la homogenización, la utilización de estos tipos de sustitución ayudarían a mantener individuos distintos en la población evitando el estancamiento en mínimos locales y mejorando el ajuste final.

5.3.5 Consideraciones sobre los Parámetros del AG

En este punto, se han realizado pruebas con los valores de distintos parámetros y operadores del AG. Los valores conseguidos para el AG que se utiliza en el entrenamiento de las RR.NN.AA. que realizan la predicción sobre las dos series temporales han sido similares. Debido a esto, y, como se refiere en los puntos particulares de este apartado, los valores conseguidos son congruentes con la naturaleza del problema, por lo que se puede decir que, de los valores de parámetros utilizados en este punto, los valores conseguidos son los más adecuados para el entrenamiento de RR.NN.AA. que realicen la predicción de series temporales.

En base a la experiencia de estas pruebas, puede decirse que la característica de no linealidad de los problemas condiciona los parámetros del AG en la dirección de las pruebas realizadas en este apartado.

En todo caso, la diferencia de ajuste conseguida en las distintas pruebas realizadas utilizando la misma arquitectura de red y trabajando sobre las mismas series temporales, indica la necesidad de realizar unas pruebas previas a la aplicación de un AG en la resolución de cualquier tipo de problema que presente una cierta complejidad.

5.4 Automatización del Desarrollo de la Topología

Se van a exponer a continuación los resultados conseguidos con el módulo de arquitecturas. En este módulo, se parte de un conjunto de entrenamiento fijo, para lo que se van a utilizar las dos series temporales empleadas hasta ahora y, como resultado, se va a obtener una arquitectura de red que, en este caso se refiere al número de neuronas ocultas que debe tener la red, así como la matriz de conexiones para realizar el "podaje". En estas pruebas, también se comprobará que la nueva implementación en este módulo de la selección automática de las funciones de activación es la correcta. En este punto, las funciones de activación se representan como en el apartado 5.2.

El número de iteraciones del AG va a ser, para cada prueba, de 200. Aún siendo un número pequeño de iteraciones para un AG, se espera que los resultados conseguidos muestren la potencialidad del sistema implementado. El resto de parámetros del AG serán los siguientes:

Población	Cruces	Mutaciones	Selección	Tipo – Cruce	Sustitución
100	1	1	Aleatoria	Un punto	Peor Individuo

En este módulo, se utiliza como función de evaluación del AG de arquitecturas el módulo de entrenamiento de RR.NN.AA. mediante AA.GG. Para que esta evaluación dure un tiempo acotado, el AG de entrenamiento de RR.NN.AA. simulará con cada arquitectura durante 500 ciclos. En este último módulo se van a utilizar, por tanto, los valores de los parámetros que consiguieron mejores resultados en el punto 5.3.

Población	Cruces	Mutaciones	Selección	Tipo – Cruce	Sustitución
100	5	5	Aleatoria	Un punto	Peor Individuo

Como ya se comentó en el capítulo anterior, el valor de ajuste de cada arquitectura se calcula entrenando una RNA concreta con esa arquitectura y viendo el ajuste final de la red. Obviamente, no es posible entrenar estas redes exhaustivamente como se realizó en puntos anteriores, con simulaciones de 25.000 ciclos. En este caso, lo que interesa es buscar un valor indicativo de lo buena que puede ser una arquitectura. Por este motivo, el número de iteraciones será un valor más pequeño, que permita valorar un gran número de redes en un tiempo reducido. Sin embargo, si el valor es muy pequeño puede que la red no comience el proceso de convergencia y no se consiga un indicador real de la bondad de la arquitectura.

En las pruebas, se ha comenzado con un valor de 100 para el número de iteraciones comprobando que era claramente insuficiente al no generarse, en la mayor parte de los casos, ninguna mejora durante ese número de ciclos. A continuación, se han comprobado valores de iteraciones de 100 en 100 hasta llegar a 500 iteraciones por red. Este valor ya permite que se produzcan varias mejoras en la población, dando tiempo a que se produzca una cierta evolución que ofrezca un valor real de ajuste del conjunto de entrenamiento. Obviamente, es necesario ajustar este parámetro para adaptarlo al problema que se está resolviendo en cada momento, teniendo en cuenta la necesidad de que se produzcan mejoras en la red que se está entrenando.

Por tanto, una vez realizados los 200 ciclos de simulación del AG de automatización del diseño de la arquitectura, se obtiene el valor de ajuste de la mejor arquitectura habiendo entrenado una red con ella durante 500 iteraciones. Para poder comparar los resultados conseguidos en los apartados anteriores, una vez alcanzados los 200 ciclos se va a realizar una simulación de entrenamiento de una RNA con la mejor arquitectura a lo largo de 25.000 iteraciones.

5.4.1 Manchas Solares

A continuación, se muestran los parámetros de la mejor solución conseguida:

Tiempo de Inicialización	Tiempo de Entrenamiento	Número de Neuronas	Número de Conexiones	Funciones de Activación
3:00	2:03:10	4	26	L L E L L H

A la que le corresponde los siguientes errores:

ECM Entrenamiento	EM Entrenamiento
0,0086272	0,072304

5.4.2 Producción de Tabaco

Para la serie temporal de producción de tabaco, los parámetros de la mejor solución conseguida son:

Tiempo de Inicialización	Tiempo de Entrenamiento	Número de Neuronas	Número de Conexiones	Funciones de Activación
2:11	43:55	5	14	L L L L H

A la que le corresponden los siguientes errores:

ECM Entrenamiento	EM Entrenamiento
0,0084106	0,068171

5.4.3 Consideraciones sobre la Automatización del Ajuste de las Arquitecturas

Los resultados conseguidos para las dos series temporales son un poco mayores, pero comparables a los mínimos conseguidos hasta ahora, como se puede comprobar en la tabla que se expone a continuación. Hay que tener en cuenta que el AG de automatización de arquitecturas ha realizado solamente 200 ciclos, lo que se puede considerar poco comparado con los 25.000 ciclos que simula el AG de entrenamiento de las RR.NN.AA. También hay que darse cuenta que este AG está sustituyendo las pruebas que se realizaban en el punto 5.1 sobre el entrenamiento de RR.NN.AA. mediante AA.GG. En este apartado, las simulaciones duraban un total de 11 horas para la serie de manchas solares y más de 5 horas para la serie de producción de tabaco. Además, las pruebas realizadas en ese apartado no cubrían la totalidad del rango de número de neuronas, ni tenían en cuenta el "podaje" de conexiones, ni la selección automática de las funciones de activación.

Series Temporales	Parámetro	Ajuste de parámetros del AG	Ajuste de la Arquitectura	Diferencia
Manchas Solares	Tiempo	40.000 segundos	7.200 segundos	- 82 %
	ECM	0,0077	0,0086	+ 11,7 %
Producción de Tabaco	Tiempo	19.000 segundos	2.700 segundos	- 91,5 %
	ECM	0,0071	0,0084	+18,3 %

También es de destacar la tendencia de las soluciones conseguidas hacia la función de activación lineal, con 4 de 6 en cuanto a la serie de manchas solares y 4 de 5

en la de producción de tabaco. Esto confirma la hipótesis realizada en el punto 5.2.3 sobre el problema de ajuste de las funciones de activación. El problema no radicaba en que la variedad de funciones utilizadas afectara el rendimiento conjunto de la red sino que, esa fase del sistema, no era la más adecuada para la realización de la selección de las funciones de activación, ya que llevaba al sistema a mínimos locales en el ajuste de estas funciones.

En este caso, la función de activación esencial para la resolución del entrenamiento de estas redes es la función lineal, a lo que se llega en los entrenamientos planteados en este apartado, junto con la utilización puntual de otros tipos de función de activación.

5.5 Diseño de los Conjuntos de Entrenamiento

En este punto, se van a realizar las pruebas del módulo de diseño de los conjuntos de entrenamiento a partir de una serie temporal completa. El AG que realiza la selección del conjunto de entrenamiento tendrá los siguientes parámetros:

Población	Cruces	Mutaciones	Selección	Tipo – Cruce	Sustitución
100	1	1	Aleatoria	Un punto	Peor Individuo

Y, como en el apartado anterior, se van a iterar 200 generaciones de este AG en cada prueba. Este valor se considera suficientemente grande para poder conseguir resultados, a la vez que suficientemente pequeño como para realizar la simulación en un tiempo razonable. Este módulo también utiliza como función de evaluación el módulo de entrenamiento de RR.NN.AA. mediante AA.GG. Por ello, en cuanto a los parámetros de las RR.NN.AA., se van a utilizar los valores con los que se consiguieron los mejores resultados en apartados anteriores:

Series Temporales	Neuronas	Función de Activación	Iteraciones
Manchas Solares	6	Lineal	1
Producción de Tabaco	2	Lineal	5

En cuanto a los parámetros del AG de este mismo módulo, también se van a utilizar los valores utilizados en el punto 5.3:

Población	Cruces	Mutaciones	Selección	Tipo – Cruce	Sustitución
100	5	5	Aleatoria	Un punto	Peor Individuo

En este módulo de diseño del conjunto de entrenamiento, una vez consignados los parámetros anteriores que configuran realmente el módulo de entrenamiento de redes, existen dos parámetros que sí son directamente responsables de los conjuntos de entrenamiento que se consiguen.

Por un lado, está el parámetro "número de iteraciones" que se refiere al número de generaciones que se simula en una red. Como ya se comentó en el apartado anterior, el valor no puede ser muy pequeño ya que, de esta forma, no se permite que comience el proceso de convergencia. Se han probado valores de iteraciones de 100 en 100 hasta llegar a 500 iteraciones por red, comprobando que este valor ya permite que se produzcan mejoras en la población, dando tiempo a que se produzca una cierta evolución que ofrezca un valor real de ajuste del conjunto de entrenamiento.

Por otro lado, está el parámetro "grupos" que es el número de muestras de que va a constar cada subserie. Las redes que se utilizan para la simulación predicen el último valor de cada subserie a partir de los restantes valores. Entonces, dependiendo de que el número de muestras sea suficiente o no, la red obtendrá mejores o peores resultados; por lo que también es necesario realizar la adaptación de este parámetro al problema que se quiera resolver en cada momento. Para la predicción de las dos series

temporales que se están utilizando en este estudio, se van a probar, como número de muestras, los siguientes:

Número de Muestras – Parámetro Grupos		
2	4	6

Como se ha realizado hasta ahora, para conseguir una fiabilidad mayor, se han repetido tres veces las pruebas con cada valor para poder conseguir una mayor fiabilidad en los resultados.

Hay que tener en cuenta que, en este módulo, se realiza el entrenamiento de la red con una parte de la serie, definida por las subseries de que conste un individuo. La parte de la serie utilizada no llega, generalmente, como se verá a continuación, al 75% del total. Una vez entrenada la red, para calcular el ajuste del entrenamiento con esas subseries, se realiza el test de la RNA conseguida con la serie completa. Por lo tanto, los valores de ajuste de la red prediciendo la serie temporal serán, en este punto, mucho más importantes debido a que los valores utilizados son, en su mayoría, valores no utilizados en el entrenamiento de la red.

5.5.1 Manchas Solares

Para cada uno de los grupos o número de muestras en cada subserie, se han conseguido los siguientes valores:

Grupos	Tiempo de Inicialización	Tiempo de Entrenamiento	Número de Subseries	% de la Serie Temporal
2	3:01	16:23	19	13,5%
4	4:34	27:00	22	31,4%
6	9:14	34:42	19	40,7%

La configuración que mejor error de test ha conseguido es la siguiente:

Grupos	ECM Entrenamiento	EM Entrenamiento	ECM Test	EM Test
2	0,0077370	0,072233	0,0077859	0,065271

Y la mejor configuración media de errores en grupos de 3 pruebas es:

Grupos	ECM Entrenamiento	EM Entrenamiento	ECM Test	EM Test
2	0,0097441	0,079227	0,0095572	0,071029

5.5.2 Producción de Tabaco

En el caso de esta otra serie, los valores conseguidos para cada grupo o número de muestras en cada subserie, son los siguientes:

Grupos	Tiempo de Inicialización	Tiempo de Entrenamiento	Número de Subseries	% de la Serie Temporal
2	3:10	20:35	23	42,2%
4	7:12	28:10	14	51,3%
6	10:03	41:10	12	66%

La configuración que mejor error de test ha conseguido es la siguiente:

Grupos	ECM Entrenamiento	EM Entrenamiento	ECM Test	EM Test
2	0,00752	0,060745	0,0066468	0,056746

Y la mejor configuración media de errores en grupos de 3 pruebas fue:

Grupos	ECM Entrenamiento	EM Entrenamiento	ECM Test	EM Test
4	0,0105508	0,089025	0,0076151	0,060684

5.5.3 Consideraciones sobre el Diseño de Conjuntos de Entrenamiento

Los resultados obtenidos en este punto mejoran apreciablemente, sobre todo para la serie de tabaco, los conseguidos hasta ahora. Como se puede comprobar en la siguiente tabla, los tiempos empleados en la selección del conjunto de entrenamiento siguen siendo muy inferiores a los empleados en el ajuste manual de parámetros utilizado en el punto 5.1 de este capítulo. Como también se puede ver en la tabla, los niveles de error son inferiores a los conseguidos en otros puntos. Teniendo en cuenta que son errores de la fase de test, no de la fase de entrenamiento, como se comentó en la introducción de este punto, para el caso de las manchas solares, aunque el error es un poco superior, supone una mejora al corresponderse a la fase de test y, en el caso de la producción de tabaco, la disminución del error ya es de por sí significativa sin tener en cuenta la distinción entre entrenamiento y test.

Series Temporales	Parámetro	Ajuste de parámetros del AG	Ajuste del Conjunto de Entrenamiento	Diferencia
Manchas Solares	Tiempo	40.000 segundos	4.500 segundos	- 88,6 %
	ECM	0,0077	0,0078	+ 1,3 %
Producción de Tabaco	Tiempo	19.000 segundos	5.400 segundos	- 71,4 %
	ECM	0,0071	0,0066	- 7 %

También es destacable que, en el caso de la serie de producción de tabaco, el EM conseguido de 0,056 es el menor alcanzado hasta ahora, mejorando el EM del

entrenamiento básico de RR.NN.AA. del punto 5.1.2, partiendo, además, de que se ha conseguido entrenando la RNA con menos del 50% del conjunto de entrenamiento total.

En este sentido, también es reseñable la reducción del número de ejemplos de los conjuntos de entrenamiento que consigue el sistema al utilizar en el entrenamiento de las redes, generalmente, menos de la mitad del conjunto original. Hay que tener en cuenta que no se ha implementado ninguna función de ponderación sobre la función de evaluación que primará la reducción del conjunto de entrenamiento resultante, dejando libertad al sistema para que utilice el número de subseries que ofrezcan un mejor nivel de error en el entrenamiento.

Por último, es interesante observar los resultados conseguidos en cuanto al número de elementos de las subseries para cada serie temporal. Estos resultados indican una tendencia hacia subseries cortas, lo que parece indicar que los valores de la serie dependen de muy pocos valores del pasado inmediato. En este caso, entre 2 y 4. En el caso de la serie de manchas solares es especialmente drástico el empeoramiento de los resultados al utilizar subseries de 4 y 6 valores, empeorando el error un orden de magnitud de valores cercanos a 0,008 a valores alrededor de 0,05.

Aunque los resultados sobre número de subseries no son extrapolables a otras series temporales, es interesante la posibilidad de automatizar este parámetro de forma que busque el número idóneo de valores para las subseries. Esta ampliación del sistema, además de evitar la intervención directa del diseñador en la configuración de este parámetro, haciendo transparente otra parte del desarrollo de la RNA, ofrece una información sobre el comportamiento de los datos de la serie que, probablemente, ayuden a entender mejor la naturaleza del fenómeno que produce la serie.

5.6 Arquitectura de Atenuación Temporal

Se comprueba, en este punto, el funcionamiento del sistema básico de entrenamiento de RR.NN.AA. mediante AA.GG. con la extensión del modelo de "activación atenuada", que intenta acercar el funcionamiento de la neurona artificial al de la neurona natural.

Para poder comparar los resultados de este punto con los ya conseguidos, se van a utilizar los valores de los parámetros de configuración de las RR.NN.AA. alcanzados

en el punto 5.3, en cuanto a número de neuronas de la mejor red de cada serie temporal, realizando cada prueba con 25.000 generaciones del AG. Hay que tener en cuenta que este subsistema es, de momento, independiente de los dos anteriores, por lo que se van a realizar las pruebas con cada función de activación, al no estar implementada en este subsistema la selección automática.

En cuanto a los parámetros propios de esta sección, se van a comprobar valores del número de iteraciones que se deja simular a la red antes de obtener el valor de salida y del valor del tiempo en el eje "x" de la función de activación atenuada.

El número mínimo de iteraciones es uno, suponiendo que se obtiene directamente la salida de la red en el instante siguiente a introducir la última entrada, para dejar que la red realice alguna iteración interna con la información que se le proporciona. Los valores que se van a utilizar son:

Número de Iteraciones		
2	4	6

En cuanto al valor del tiempo, se va a partir del valor por defecto, el valor uno, que iguala el tiempo a las iteraciones de la red, suponiendo que cada iteración de la red se realiza en un instante de tiempo. Este valor se decrementa para suponer que cada instante de simulación se corresponde con valores de tiempo inferiores a uno. Los valores que se utilizan son:

Valor del tiempo – Incrementos del Eje x		
1	0,5	0,1

5.6.1 Manchas Solares

El mejor resultado conseguido en la simulación tiene los siguientes valores:

Función de Activación	Iteraciones	Valor del tiempo	ECM	EM	Segundos
Lineal	4	1	0,010731	0,080343	1814

El mejor resultado medio con tres pruebas, por grupos de funciones e iteraciones es el siguiente:

Función de Activación	Iteraciones	ECM	EM	Segundos
Exponencial	4	0,013875	0,090915	9123

5.6.2 Producción de Tabaco

El mejor resultado conseguido con esta serie es el siguiente:

Función de Activación	Iteraciones	Valor del tiempo	ECM	EM	Segundos
Lineal	4	0,1	0,0069128	0,059497	332

Y, el mejor resultado medio con tres pruebas, por grupos de funciones e iteraciones es el siguiente:

Función de Activación	Iteraciones	ECM	EM	Segundos
Lineal	4	0,0069801	0,059399	356

5.6.3 Consideraciones sobre la Atenuación Temporal

Las pruebas realizadas en este punto, se van a comparar con las realizadas en el punto 5.3, en el que se llevó a cabo el ajuste de los parámetros del AG y se consiguieron

los mejores resultados del módulo de entrenamiento de RR.NN.AA mediante AA.GG. Para favorecer la revisión de los resultados, en la siguiente tabla, se muestra una comparativa para cada una de las series temporales, indicando la diferencia en tanto por ciento del ECM cometido en cada prueba:

Series Temporales	Tipo de Prueba	ECM con ajuste de parámetros del AG	ECM con activación atenuada en el tiempo	Diferencia
Manchas Solares	Mejor	0,0074	0,0107	+ 44%
	Grupo de 3	0,0077	0,0138	+ 79%
Producción de Tabaco	Mejor	0,0069	0,0069	0%
	Grupo de 3	0,0071	0,0069	- 1,5%

Como se puede apreciar en la tabla, los resultados conseguidos para la serie de manchas solares son muy inferiores a los que se habían obtenido con el ajuste de parámetros del AG. Mientras que, los conseguidos para la serie de producción de tabaco sí que los mejoran, sobre todo en el grupo de tres pruebas, mostrando la robustez del método al conseguir en las tres pruebas realizadas unos resultados muy buenos y, además, muy similares.

El empeoramiento de los resultados en la serie de manchas solares, continúa la tendencia mostrada por esta serie en todas las pruebas realizadas hasta ahora sobre la dependencia hacia los datos inmediatamente anteriores. En las primeras pruebas de los apartados 5.1 y 5.2, los mejores resultados se consiguieron con un número de iteraciones internas de uno con cada valor de entrada introducido, indicando que no se necesita conservar un nivel de activación que recuerde valores demasiado lejanos en el tiempo. Esta impresión se confirma en el punto 5.5 en el que los mejores resultados se consiguen al utilizar subseries de 2 datos, empeorando estos resultados de forma drástica al utilizar subseries de 4 y 6 datos.

El subsistema que se prueba en este punto favorece la persistencia de los estados de activación, mejorando la capacidad de almacenamiento de la información temporal

en el funcionamiento de la red. Para conseguir este objetivo, se utilizan valores mayores que uno en el número de iteraciones internas por valor de entrada introducido. Obviamente, y en base a los resultados obtenidos en puntos anteriores, la serie de manchas solares, por su propia naturaleza, no necesita de este tipo de procesado, resultando incluso perjudicial para los resultados obtenidos, como ya sucedió en el caso de utilizar subseries de 4 y 6 datos al diseñar los conjuntos de entrenamiento.

En el caso de la serie de producción de tabaco, los resultados obtenidos son muy interesantes, ya que los valores conseguidos en distintas pruebas son muy homogéneos y son los mejores conseguidos sin utilizar ajuste de arquitecturas o diseño del conjunto de entrenamiento. En este caso, la persistencia de los estados de activación si que mejora, tanto los valores de error, como la robustez del método.

Los resultados obtenidos con respecto a las dos series temporales indican que la utilización de la activación atenuada en las conexiones puede no ser adecuada para todas las series temporales, dependiendo de la naturaleza de sus datos. La decisión de aplicar o no este subsistema a la predicción de series temporales queda ahora en manos del diseñador. Esta decisión se podría automatizar incluyendo un nuevo subsistema con un AG que calcule los valores óptimos de iteraciones y valor del tiempo, que ahora se especifican a mano. Incluir entre los valores posibles del tiempo el cero implica que, en ese caso, no se utilizará la activación atenuada mientras que, valores mayores, que son los probados en este punto, sí que permitirían su aplicación.

5.7 Integración

Por último, en este punto se refieren los resultados de las pruebas del sistema completo. Aquí, se han puesto en común todos los módulos del sistema trabajando en paralelo sobre una red de computadoras.

Para estas pruebas, se han utilizado tres computadoras iguales a las utilizadas en las pruebas de los puntos anteriores. Son computadoras con procesador Pentium II funcionando a una velocidad de reloj de 233MHz y con 64Mb de memoria RAM.

En relación con las RR.NN.AA., el sistema presenta un mínimo de parámetros a configurar ya que, de forma automática, un módulo establece la arquitectura y otro módulo selecciona el conjunto de entrenamiento óptimo. Sobre los parámetros de los

AA.GG., se van a utilizar los mismos valores empleados en los puntos 5.4 y 5.5. Para cada AG del nivel superior, el de ajuste de arquitectura y el de diseño del conjunto de entrenamiento, se van a simular 200 generaciones, como ya se vio en los puntos anteriores. Estos dos AA.GG. van a configurar sus parámetros con los siguientes valores:

Población	Cruces	Mutaciones	Selección	Tipo – Cruce	Sustitución
100	1	1	Aleatoria	Un punto	Peor Individuo

Ambos módulos utilizan como función de evaluación el módulo de entrenamiento de RR.NN.AA. mediante AA.GG. Como ya se ha comentado, para que esta evaluación dure un tiempo acotado, el AG de entrenamiento de RR.NN.AA. simulará con cada arquitectura durante 500 ciclos. En este último módulo, se van a utilizar, por tanto, los valores de los parámetros que consiguieron mejores resultados en el punto 5.3, que son los siguientes:

Población	Cruces	Mutaciones	Selección	Tipo – Cruce	Sustitución
100	5	5	Aleatoria	Un punto	Peor Individuo

En el caso del diseño del conjunto de entrenamiento, el único parámetro que hay que configurar es el número de datos de las subseries que, a partir de la experiencia conseguida en el punto 5.5, se ha configurado con el valor dos, como el más adecuado para las dos series temporales. Por las razones vistas en el punto anterior, la activación atenuada en el tiempo sólo se va a utilizar en el caso de la serie producción de tabaco ya que afecta de modo negativo a la predicción de la serie de manchas solares.

5.7.1 Manchas Solares

Tiempo de Inicialización	Tiempo de Entrenamiento	Número de Subseries	Neuronas Ocultas	Número de Conexiones	Funciones de Activación
3:30	25:30	23	4	14	L L H L H L

Universidade da Coruña
FACULTAD DE INFORMÁTICA
Departamento de Computación

MODELO DE UN SISTEMA PARA LA SELECCIÓN AUTOMÁTICA EN DOMINIOS COMPLEJOS, CON UNA ESTRATEGIA COOPERATIVA, DE CONJUNTOS DE ENTRENAMIENTO Y ARQUITECTURAS IDEALES DE REDES DE NEURONAS ARTIFICIALES UTILIZANDO ALGORITMOS GENÉTICOS

DOCTORANDO: **JULIÁN DORADO DE LA CALLE**
DIRECTOR: **ALEJANDRO C. PAZOS SIERRA**

A Coruña, Marzo 1999

ECM Test	EM Test
0,0074458	0,064756

5.7.2 Producción de Tabaco

Tiempo de Inicialización	Tiempo de Entrenamiento	Número de Subseries	Neuronas Ocultas	Número de Conexiones	Funciones de Activación
3:01	30:35	15	3	14	L E L H L

ECM Test	EM Test
0,0065482	0,056053

5.7.3 Consideraciones de la Integración de Módulos

Como se puede observar en los resultados obtenidos con las dos series, los niveles de error han descendido en ambas destacando, sobre todo, que los tiempos de simulación son similares a los conseguidos en los apartados 5.4 y 5.5.

En el caso de la serie de manchas solares, los resultados son superiores a los del punto 5.3 de ajuste de parámetros del AG, en el que se había logrado un mínimo no superado hasta el momento. En el caso de la serie de producción de tabaco, los resultados superan levemente los conseguidos en el punto 5.5 de diseño del conjunto de entrenamiento. Como ya se comentó, precisamente en ese punto, los resultados de ECM y EM se corresponden a una fase de test ya que, en esta fase del sistema, las RR.NN.AA. se entrenan sólo con parte de los datos disponibles, siendo estos valores de error mucho más relevantes.

Un problema que ha surgido a la hora del desarrollo del sistema final es el de la sincronización del intercambio de datos entre módulos. La forma de abordar este

problema se ha revelado como muy delicada y puede afectar de forma dramática al rendimiento. La actualización de la arquitectura utilizada en el módulo de conjuntos de entrenamiento, o la actualización del conjunto de entrenamiento en el módulo de arquitecturas, si se realiza de forma muy frecuente puede bloquear el trabajo de ambos módulos degradando mucho el rendimiento del sistema integrado. Para controlar el efecto negativo de este problema, en el sistema se ha limitado el número de actualizaciones que se pueden realizar en el tiempo, de forma que cada subsistema acumule las mejoras obtenidas. De esta manera, si en un breve período de tiempo se generan varias mejoras, sólo la última de ellas se envía al otro módulo para su utilización. Obviamente, este problema es mucho más grave en el comienzo de la simulación, cuando el número de mejoras por unidad de tiempo es mayor, que cuando la simulación está en una etapa más avanzada, en la que el número de mejoras que surgen en cada módulo es mucho más limitado.

Otro factor importante de las pruebas realizadas en este punto es la minimización de parámetros a configurar conseguida en esta fase final del sistema. En este punto, gran parte del trabajo de configuración se refiere a los parámetros de los AA.GG., siendo mucho más automática la configuración de los parámetros de las RR.NN.AA. en lo que se refiere a la arquitectura y a la selección del conjunto de entrenamiento.

5.8 Comparación con otras Técnicas de Predicción

En este último punto del capítulo, se van a comparar los resultados conseguidos por el sistema final con distintas técnicas, ya comentadas, que se utilizan para la modelización y predicción de series temporales. En este sentido, la comparación se realizará sobre dos aspectos distintos, por un lado se enfrentará la eficacia del proceso de entrenamiento, en cuanto a consumo de recursos, con distintas aproximaciones tradicionales de entrenamiento de RR.NN.AA.RR. Por otro, se va a comparar la exactitud de la predicciones realizadas mediante un modelo ARIMA y las conseguidas con el sistema cooperativo de entrenamiento de RR.NN.AA.RR. mediante AA.GG.

5.8.1 Algoritmos Tradicionales de Entrenamiento de RR.NN.AA.RR.

El rendimiento conseguido con las RR.NN.AA.RR. desarrolladas por el sistema cooperativo presentado en este trabajo, se va a comparar con el de las RR.NN.AA. entrenadas mediante dos algoritmos clásicos del campo de las redes temporales ya comentados en el Capítulo de Fundamentos: un algoritmo derivado del de retropropagación del error, el Back Propagation Through Time (BPTT) y el algoritmo de entrenamiento de redes realmente recurrentes, Real Time Recurrent Learning (RTRL).

Debido a las altas necesidades de cómputo que demandan estos algoritmos de entrenamiento para redes temporales, se optó por utilizar una computadora del tipo servidor en vez de llevar a cabo el entrenamiento de las RR.NN.AA. en los PC que realizaron el resto de las pruebas expuestas en este capítulo. La estación escogida fue una computadora Silicon Graphics Origin 200 del centro de cálculo de la Facultad de Informática de la Universidade da Coruña. Esta estación cuenta con dos torres con sus placas base conectadas con una conexión de alto rendimiento de tipo CrayLink. En total cuenta con 4 CPUs MIPS R10000 funcionando a una velocidad de reloj de 180MHz y con 384 Mb de memoria RAM. Los programas de simulación de ambos algoritmos de entrenamiento de RR.NN.AA. se implementaron en el entorno Matlab de la empresa MathWorks en su version 5.2. La versión disponible de Matlab no permite la paralelización de los programas de simulación, por lo que no se utilizan los cuatro procesadores en las ejecuciones de programas; sin embargo, las labores propias del sistema operativo sí que pueden llevarse a cabo en otro procesador mientras que uno de ellos puede ser utilizado en exclusiva por el entorno Matlab para las simulaciones.

Para aplicar los dos tipos de algoritmos de aprendizaje se emplearon RR.NN.AA. con 3, 5 y 7 EP, siendo la mejor configuración la de 5 EP. La tasa de aprendizaje utilizada fue de 0,001, siendo con este valor con el que se consiguen mejores resultados. Se realizaron pruebas con las funciones de activación empleadas a lo largo de este trabajo, presentando los mejores resultados la función de activación lineal.

Los resultados conseguidos utilizando esta implementación de los algoritmos de entrenamiento, en cuanto a niveles de error en el aprendizaje, son similares a los conseguidos con el sistema cooperativo expuesto en los trabajos de la tesis, oscilando el ECM de los distintos experimentos entre 0,007 y 0,02 y el EM entre 0,06 y 0,15.

Hay que tener en cuenta, sin embargo, el distinto potencial que presentan una computadora personal con el procesador Intel Pentium II frente a una estación de trabajo SGI Origin 200 con procesadores MIPS R10000. Para tener una referencia objetiva de las diferencias entre estos dos tipos de sistemas, se presentan en la siguiente tabla los resultados de los "benchmarks" o pruebas de rendimiento SPEC CPU95 de estos procesadores realizados sobre estos dos sistemas:

	SPECint_base95	**SPECfp_base95**
PC Pentium II 233MHz	9.38	6.73
SGI Origin 200 MIPS R10000	9.86	14.5

Estas pruebas miden el rendimiento de los distintos tipos de sistemas frente a problemas basados en números enteros (SPECint_base95) y en números en punto flotante (SPECfp_base95). Los problemas que se afrontan en estos trabajos, tanto el desarrollo cooperativo de RR.NN.AA.RR., como el entrenamiento de RR.NN.AA. mediante los algoritmos BPTT y RTRL hacen uso intensivo de computación en punto flotante. Como se puede ver en la tabla el sistema SGI ofrece un 115% más de potencia en números en punto flotante que el sistema basado en el procesador Pentium II. Aún sin tener en cuenta que la máquina SGI utilizada en las pruebas dispone de varias CPUs mientras que los PC son equipos monoprocesador.

Las diferencias más acusadas se presentan en cuanto al tiempo consumido en la realización de los entrenamientos. Los algoritmos de aprendizaje BPTT y RTRL utilizan alrededor de 1.000 ciclos para estabilizar el nivel de error en el proceso de aprendizaje. A partir de este punto se comienza una muy lenta disminución del nivel de error. Cada ciclo consume alrededor de 40 segundos para el caso de la serie de manchas solares y 25 segundos para el caso de la serie de producción de tabaco. Estos tiempos

implican que se necesitan más de 11 horas para conseguir una red entrenada para el caso de la serie de manchas solares y alrededor de 7 para el caso de la serie de producción de tabaco. En el caso de las simulaciones planteadas en este capítulo de resultados, tanto sobre el sistema cooperativo como sobre el funcionamiento de cada módulo independiente, todos oscilan entre media hora y hora y media, con lo que se consigue en cualquier caso una reducción superior al 75%.

Para terminar, destacar que, para conseguir resultados similares en cuanto a nivel de error con el sistema cooperativo, se ha utilizado, de media, una décima parte de tiempo de entrenamiento, utilizando máquinas (PC) de menos de la mitad de rendimiento en punto flotante y que presentan un precio significativamente menor, menos de la décima parte.

5.8.2 Predicción de modelos ARIMA

En este punto se va a realizar una comparación entre las capacidades de predicción de las RR.NN.AA. que se desarrollan en los trabajos de esta tesis con las capacidades de los modelos ARIMA. En el punto 2.3.3.4, se muestran las características de las series temporales utilizadas en este capítulo para realizar las pruebas del sistema. Desde el punto de vista de la teoría de Análisis de Series Temporales, la serie de Manchas Solares se puede modelizar como AR(9); es decir un modelo ARIMA(9,0,0). La serie de Producción de tabaco es más problemática ya que no es estacionaria ni en la media ni en la varianza. Para poder modelizarla es necesario aplicarle una transformación logarítmica. La serie transformada se ajusta entonces a un modelo IMA(1,1); es decir, un modelo ARIMA(0,1,1).

Estos modelos se han desarrollado reservando un conjunto de valores finales de la serie que tampoco han sido utilizados en el entrenamiento de las RR.NN.AA.RR. desarrolladas con el sistema cooperativo. Las RR.NN.AA.RR. que se utilizan para realizar la predicción son las que se han desarrollado en el punto anterior mediante la aproximación cooperativa. Los resultados conseguidos se presentan en las tablas siguientes:

Real	RNA	% Error RNA	ARIMA	% Error ARIMA
66.6	*95.44*	15.17	28.21	20.20
45.9	*56.59*	5.62	16.09	15.68
17.9	41.47	12.40	*12.68*	2.74
13.4	26.55	6.92	*20.99*	3.99
29.3	*24.86*	2.33	45.24	8.39

Tabla 1. Comparación de los valores reales de las manchas solares en los años 1982-1986 con la predicción de la mejor RNA y del modelo ARIMA.

Real	RNA	% Error RNA	ARIMA	% Error ARIMA
1785.89	1678.13	6.02	*1716.16*	2.94
2063.89	*2050.93*	0.62	1740.14	13.77
1994.04	*1969.94*	1.2	1764.30	9.81
1428,90	*1411.87*	1,19	1788.78	15.35
1727.99	1590.33	5.84	*1813.61*	3.67

Tabla 2. Comparación de los valores reales de la producción de tabaco en los años 1980-1984 con la predicción de la mejor RNA y del modelo ARIMA.

En ambas tablas, la primera columna se refiere al valor real de la serie temporal, la columna RNA es la predicción conseguida con la RNA desarrollada y la columna ARIMA es el valor conseguido con el modelo ARIMA. Las columnas de % Error se refieren a los errores cometidos por cada aproximación con respecto al valor real de la serie. Los valores en cursiva son los que más se acercan al valor real.

Como se puede comprobar en las tablas anteriores, los errores conseguidos por la aproximación conexionista son comparables e incluso inferiores en algunos casos a los conseguidos con el modelo ARIMA. Estos resultados son, por tanto, indicadores de la validez del método propuesto en cuanto a predicción de las series temporales de distintos tipos.

Capítulo 6: Conclusiones y Futuros Desarrollos

Chapter 6: Conclusions and future works

6. Conclusions and Future Works

This thesis views ANN development from an integrating approach, which tries to solve traditional problems that an ANN designer must face when trying to apply this technology to the solution of a given problem. The GA technique has been used to carry out the ANN's training, a valid alternative to the learning logarithms based on other techniques such as gradient descent.

From the times when Minsky and Papert [MINS-69] discredited the ANN field by uncovering certain limitations characteristic of the models of those years, ANN technology has greatly developed. On the one hand, its complexity has increased, on the other hand, a great number of new specific and general purpose models have come out.

In view of the Perceptron's limitations exposed by Minsky and Papert, a training logarithm for the multilayer Perceptron was developed. After the recession caused by this controversy, researches such as Hopfield, Grossberg or Kohonen have revitalized this field with the development of new models, closer to neurological neurones and learning theories.

All of these network models present traditional design problems such as the adjustment of the architecture. This is the first problem that the researcher must face when he/she wants to apply an ANN to the solution of a problem. Different solutions to this kind of problems have been suggested. One of the most widespread ones is the incremental network model. Although they present typical problems of local minima or initial situation dependency, they are steps towards the automatization of ANN design.

The development of new specific ANN models for time processing confirm the vitality of the ANN field in the last decade. In these models, the capacity of processing and extracting information has multiplied several magnitude orders, due to the existence of a much greater number of connections among neurones and to the feedback of information inside the network. This greater potential has the disadvantage of convergence and local minima problems, undoubtedly due to the technology's newness. These problems hinder the use of this type of networks, which are the most suitable to tackle time problems or problems of great complexity.

This thesis proposes, above all, the unification of the network architectures and of the neurone models. It is necessary to progress in the development of a generic artificial neurone model which includes a certain number of parameters that make it highly versatile for the treatment of static and dynamic problems. It is also necessary to experiment on artificial neurones all the knowledge we possess about natural ones. Obviously nature is the best guide (having had millions of years of evolution and constant improvement, using the selective pressure of the environment) to build more complete models with a more efficient functioning for information integration and processing.

These generic neurones must be integrated in networks that, obviously, are to have a recurrent architecture where the neurones activation state in a certain moment can be used for the treatment of future information.

All these ideas that, of course, have already been developed somehow in the ANN field from the beginning, are difficult to apply if an evolutive and distributed strategy is not used.

It is not possible to apply new parameters, neither to neurones nor to networks if we do not have an efficient training method that adjusts both the traditional parameters, the weights and the new parameters included in the model. Gradient descent methods are too difficult to adapt so that they are able to train ANNs with different parameter variations. However, the evolutionary search techniques are generic techniques that work well as long as an evaluation function with which to carry out selective pressure on the population can be specified. Notwithstanding, training algorithms based on gradient are not suitable either for training very complex networks such as time networks, given that they show a slow performance in their convergence and at the same time they are prone to fall in local minima during the training process. Evolutionary techniques are also a very suitable alternative for solving these problems, given that, on the one hand, they allow to explore the searching space in a highly parallel way, minimizing the chance to fall in local minima. On the other hand, GAs are parallel algorithms by nature, so that the network distribution of the necessary computation for simulations can be easily carried out.

In this thesis, all these ideas are applied to the development of ANNs. A study has been carried out on how to apply evolutionary techniques to ANN development and to how to integrate the different stages of development of an ANN. In the developed system we have used a recurrent network general pattern based on a generic model of a neurone. From this point on, the system designs the network's architecture adjusting the most classical parameters, such as the number of hidden neurones, or newer ones, such as the type of activation functions of each of the network's neurones.

The system evaluates the architectures proposed by the GA training specific ANNs which try to solve the problem by adjusting the weights of the connections allowed by the connection pruning operation. In a parallel way, the selection of the most relevant data of the original training set is carried out, so that ANNs are only trained with the most significant or important data. This selection process is carrying out, in a concurrent way, the test stage of the developing networks, since in the networks' training only a part of the available database is used, while each network's final adaptation is checked against the data of the whole training set.

The first step in the development of this kind of approach has been the development of a basic system for training ANN with GA. The experience obtained in the development of this part of the system shows that, in order to get a suitable functioning, it is very important to adequate the GA's parameters to the problem we wish to solve. In the studies carried out, we can see that those GA's parameter values which are more suitable are the most coherent with the characteristics of the problem to be solved. The problem, ANN training, is a non-linear one, with a wide searching space, in which there is a dependence among the data we are trying to adjust, whether they are connection weights, pitches or connectivity matrixes. The GA's parameter values obtained in the study are adequate for solving this kind of problems. For that reason, it is necessary to stress the importance of carrying out a study of the problem or, if not, an exhaustive test of the GA's parameters as the previous step to get good results in problem-solving via the use of GA.

We should also note the constraints found in the implementation of GA simulations with regard to simulation times. The linear increase of parameters such as population size, interbreeding or mutation rates, etc., causes an exponential increase in the simulation time. The appearance of these undesired effects, due to GA operational

overload, prevents the testing of parameter values in too wide a range. Obviously, it is not a problem of machine potential, given that the exponential increase in the simulation time saturates any kind of machine with a not excessive increase in the values. The solution to this situation lies in GA parallelization, distributing the GAs, so that each of them deals with some parts of the main problem, or even parallelizing these GAs if a wider distribution is required. This approach is the chosen one for the development of the present research, showing a good adaptation to the solution of the problem of ANN training.

Following this philosophy, one of the problems into which ANN development can be divided is the network's architecture's adjustment. No mathematical ground has been found until now on which to base this problem, that causes the development of non-optimized networks, due to the long time needed to check all the combinations of the network's architecture. This system has implemented a GA which adjusts not only the number of layers and neurones of the network but also selects the activation functions for neurones and prunes the unnecessary connections. This kind of systems, apart from saving global time at simulations, allow the ANN designer to side-step the tedious task of selecting the architecture, which becomes a transparent stage in the ANN development work.

At this point, we must remark the experience obtained from facing the problems of the automatic selection of the activation functions. The selection of the functions was first viewed as part of the basic ANN training system with GA. However, the fact of carrying out simultaneously the weight adjustment and the function selection in the same GA would make the GA fall into local minima, due to the convergence into optimal values of the connection weights of the different activation function types. We can conclude from this problem the fact that it is very important to select the parameters to be adjusted simultaneously in the same GA.

We must avoid their mutual negative influence which would make the GA fall into local minima. In the case of having dependency problems, as it is the case here, it is necessary to build a hierarchy of GAs in which those of most basic level act as evaluation functions of the higher level ones, isolating thus the dependencies and avoiding the local minima.

Another important part of the system is that of adjusting the training set. Many ANN developments do not take these subjects into account. However, the sifting or selection of the examples included in the training set is fundamental. In order to build a good training set, redundancy in the input data must be avoided to accelerate the networks training, apart from focusing on the most significant values or those which give more information about the internal functioning of the series. Networks trained with training sets which have been selected this way tend to be more robust, and at the same time they widen the generalization of the problems.

This research has focused, above all, in the study of temporal series as training sets. In this case, new problems have surfaced due to the time dependency of the examples, that hinders the selection of individual data with which to build a reduced training set. The results obtained with the approach used in this thesis, using half of the values of the whole series, offer error values that are inferior to those obtained using the whole training set. Even in the case that these errors where only comparable, they would mean a progress, given that these modules' error values are generated after testing the whole series, after having trained the networks with a reduced number of examples from the temporal series, so that the generalization capacity of these ANNs is also being checked.

The advantages obtained are, therefore, the natural performance of the series test and the generation of a reduced data base that can be used for the adjustment of other network's characteristics in other modules of the system. Being a reduced system, the networks evaluations will be much faster with this set.

In all the evolutionary subsystems proposed and tested in this research, the evaluation method used has been the same, a module that generates a "phenotype" or network structure to which the chromosome data are applied so as to study its behaviour, taken from a "genotype" i.e., from an individual's chromosome. This module is that of ANN training with GA, and it is previous to the functioning of each of the other modules. The network training stage must be solved first, and once it works it facilitates the growth of the system with regard to parameter adjustment inside the network.

Any kind of parameter which can be integrated in the ANN's functioning equation or in its process elements can be treated with the minimal change system, both in the training GA and in the implementation function of the networks "genotype". Their development causes no problems of network training, that is already solved by the first level module. Due to this, and as an example of application of evolutionary training to the development or enlargement of network models, this research tried to modify the functioning of the networks connections, so that the process of activation of natural neurones is faithfully imitated.

The aim of these modifications is to improve the internal functioning of the ANN, increasing their information-storage capacity. The recurrent ANN model increases its complexity level by incorporating the temporal function of activation weakening. These modifications favour the persistence of the ANN's internal activation states, so that the ANN is able to adapt itself to the solution of this kind of problems in a more flexible way.

Finally, the integration of the different approaches has been obtained so that they co-operate in the search for an optimal solution. The co-operative approach presented here uses two levels of parallelism. On the one hand, there is a parallelism of the ANN training system with GA that searches for the optimal configuration values of specific ANNs. On a higher level, there is a design parallelism in which several GAs search for the best design solutions for networks which are later tested in an inferior level. This second-level parallelism is used in a multiprocessor system, so that each set in a computer network is in charge of the simulation of one GA.

The co-operative characteristic is the other main point in the system, given that progress towards a final solution is not only based on the mere interchange of individuals among the various GAs. In the system, each GA adjusts parameters with a different facet of ANN development. The achievements made by a GA in its own facet are sent to the rest of GAs that instantly make use of them for finding better solutions in the search of their own development facets.

The prediction of temporal series is the application field in which most of the system tests have been made. Nowadays, this field is very interesting for science and industry, which require easy, fast and trustworthy methods for the prediction of different

kinds of phenomena. In the framework of these needs we can find systems like the one developed in this thesis that complement and support the use of more traditional mathematical methods such as ARIMA, which require some mathematical knowledge and a certain degree of experience in their use. In the tests which were carried out, the system, apart from the automatization it brings to ANN development, obtains comparable results to those obtained with ARIMA modelization of temporal series, with a lower error level.

This research exposes the possibilities of evolutionary technique for the development of connectionist systems. This development is divided into parts that are independently adjusted but in co-ordination with the different aspects of ANN development. This distributed and co-operative approach works adequately and it opens the door to the development of new subsystems which complement the results obtained in this thesis. The distribution in a low-cost computer network does not make the enlargement of the system with new modules too expensive or bring about an exponential increase in computation. It also opens the door to the models' development or modification, making it easier to train ANNs with new or modified parameters, making only small changes in the ANN's evaluation part and in the individuals' chromosomes. This avoids the dependency of having to develop or to adapt a gradient-based learning algorithm which includes each modification introduced in ANN models.

The researches carried out until now which have been exposed in this thesis are, as it has been said in the previous paragraph, a starting point in two senses: on the one hand, for the development of new modules that adjust ANN development facets which have not been contemplated in this work, and, on the other hand, for the adjustment and exhaustive test of the developed system in aspects that have not been totally explored yet. In this sense, it is considered necessary to study and increase the automatization of the training sets' development module, which has had a very good performance, given that it is possible to enlarge it so that it explores independently more possibilities regarding to subseries size, offering additional information about the nature of the series that we are trying to predict.

The range of problems to be treated by the system is going to be increased before long, always working in complex dominions. In this sense, the next field to be studied is that of medical diagnosis, due to the affinities of our research group. In this

field, we usually dispose of incomplete, sometimes inconsistent and often imprecise and uncertain input data. These characteristics suppose a complexity level that is difficult to face with traditional ANN techniques.

Finally, it is necessary to go on with the study of GA parameters so as to increase the performance of this search technique. The most logical step to follow in the future development of the system is to apply "multipopulations" in each GA which is integrated in the system as a method to increase diversity, to avoid the population's homogenization and to minimize the chance to fall into local minima during simulation. We must also continue to work on the issue of how the various parameters that are becoming standard in GA theory influence ANN training with GA. Some of these parameters are the "co-evolution" and the "combination" of GAs with other Evolutionary Computation techniques such as Genetic Programming and Evolutionary Strategies.

Capítulo 7: Bibliografía

7. Bibliografía

[ABRA-83] Abraham, B. & Ledolter, J.: *"Statistical Methods for Forecasting"*. Ed. John Wiley. 1983.

[ACKL-87] Ackley, D.H.: *"An Empirical Study of Bit Vector Function Optimization"*. Genetic Algorithms and Simulated Annealing. Ed. Pitman. Eds. L. Davis. Capítulo 13. Pp. 170-204. 1987.

[ALBE-89] Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K. & Watson, J.D.: *"Molecular Biology of the Cell"*. Ed. Garland. 1989.

[ALON-91] Alon, N., Dewdney, A.K. & Ott,T.J.: *"Efficient Simulation of Finite Automata by Neural Nets"*. Journal of the Association of Computing Machinery. Vol 38, nº 2. pp. 495-514. 1991.

[ALPA-90] Alpaydin, A.I.: *"Neural Models of Incremental Supervised and Unsupervised Learning"*. Tesis Doctoral. EPFL. 1990.

[ANDR-85] Andrews, P.E. & Herzberg, A.M.: *"The Data: A Collection of Problems from Statistics"*. Ed. Springer-Verlag. 1985.

[ANGE-92] Angeline, P.J.: *"Antonisse's extension to Schema Notation"*. GA-Digest. Vol. 6. Nº 35. 1992.

[ANTO-87] Antonisse, H.J. & Keller, K.S.: *"Genetic Operators for High-level Knowledge Representations"*. Proceedings of the 2nd International Conference on Genetic Algorithms. Ed. Erbaum. Eds. J.J. Grefentstette. Pp. 69-76. 1987.

[ANTO-89] Antonisse, H.J.: *"A New Interpretation of Schema Notation that Overturns the Binary Encoding Constraint"*. Proceedings of the Third International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds. J.D. Schaffer. Pp. 86-91. 1989.

[ANTO-92] Antonisse, H.J.: *"Re: Antonisse's extension to Schema Notation"*. GA-Digest. Vol. 6. Nº 37. 1992.

[ATMA-76] Atmar, J.W.: *"Speculation on the Evolution of Intelligence and its Possible Realization in Machine Form"*. Doctoral Dissertation. New Mexico State University. 1976.

[ATMA-79] Atmar, J.W.: *"The Inevitability of Evolutionary Invention"*. New Mexico State University. 1979.

[BACK-91] Back, A.D. & Tsoi, A.C.: *"FIR and IIR Synapses, a new Neural Network Architecture for Time Series Modeling"*. Neural Computation. Nº 3. Pp. 375-385. 1991.

[BACK-96] Bäck, T.: *"Evolutionary Algorithms in Theory and Practice"*. Ed. Oxford University Press. 1996.

[BAGL-67] Bagley, J.D.: *"The Behavior of Adaptive Systems which Employ Genetic and Correlation Algorithms"*. Tesis Doctoral. University of Michigan. 1967.

[BAKE-85] Baker, J.E.: *"Reducing Bias and Inefficiency in the Selection Algorihtm"*. Proceedings of the First International Conference on Genetic Algorithms. Ed. Lawrence Erlbaum Associates. Eds. J.J. Grefenstette. Pp. 101-111. 1985.

[BALL-66] Ball, G.H. & Hall, D.J.: *"ISODATA: An Iterative Method of Multivariate Data Analysis and Pattern Recognition"*. Proceedings of the IEEE International Communications Conference. Pp. 116-117. 1966.

[BART-46] Bartlett, M.S.: *"On the Theroretical Specification of Sampling Properties of Autocorrelated Time Series"*. Journal of Royal Statistical Society. B, 8. Pp. 27-41. 1946.

[BART-50] Bartlett, M.S.: *"Periodogram Analysis and Continuous Spectra"*. Biometrika. N° 37. Pp. 1-16. 1950.

[BAUM-89] Baum, E.B. & Haussler, D.: *"What Size Net Gives Valid Generalization"*. Neural Computation. N° 1. pp. 5-19. 1989.

[BEAS-93a] Beasley, D., Bull, D.R. & Martin, R.R.: *"An Overview of Genetic Algorithms: Part 1, Fundamentals"*. University Computing. N° 15. Pp. 58-69. 1993.

[BEAS-93b] Beasley, D., Bull, D.R. & Martin, R.R.: *"A Sequential Niche Technique for Multimodal Function Optimization"*. Evolutionary Computation. N° 2. Pp. 101-125. 1993.

[BELE-91] Belew, R.K. & Booker, L.B. (eds) Proceedings of the 4th International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. 1991.

[BELL-57] Bellman, R.: *"Dynamic Programming"*. Princeton University Press. 1957.

[BETH-81] Bethke, A.D.: *"Genetic Algorithms and Function Optimizers"*. Tesis Doctoral. University of Michigan. 1981.

[BOLT-92] Bolt, G.R.: *"Fault Tolerance in Artificial Neural Networks"*. Tesis Doctoral. York University. Ontario. 1992.

[BOOK-82] Booker, L.: *"Intelligent Behavior as an Adaptation to the Task Environment"*. Tesis Doctoral. University of Michigan. 1982.

[BOOK-85] Booker, L.: *"Improving the Performance of Genetic Algorithms in Classifier Systems"*. Proceedings of the First International Conference on Genetic Algorithms. Ed. Lawrence Erlbaum Associates. Eds. J.J. Grefenstette. Pp. 80-92. 1985.

[BOOK-87] Booker, L.: *"Improving Search in Genetic Algorithms"*. Genetic Algorithms and Simulated Annealing. Ed. Pitman. Eds. L. Davis. Capítulo 5. Pp. 61-73. 1987.

[BOX-57] Box, G.E.P.: *"Evolutionary Operation: A Method for Increasing Industrial Productivity"*. Applications on Statistics. N° 6. Pp. 81-101. 1957.

[BOX-64] Box, G.E.P. & Cox, D.R.: *"An Analysis of Transformations"*. Journal of the Royal Statistical Society. Vol. B. N° 26. Pp. 211-252. 1964.

[BOX-69] Box, G.E.P. & Draper N.P.: *"Evolutionary Operation. A Method for Increasing Industrial Productivity"*. Ed. Willey. 1969.

[BOX-76] Box, G.E.P. & Jenkins, G.M.: *"Time Series Analysis. Forecasting and Control"*. Ed. Holden Day. Segunda Edición. 1976.

[BRAI-77] Braitenberg, V.: *"On the Texture of Brains"*. Ed. Springer-Verlag. 1977.

[BRAM-91] Bramlette M.F.: *"Initialization, Mutation and Selection Methods in Genetic Algorithms for Function Optimization"*. Proceedings of the Fourth International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds. R.K. Belew & L.B. Booker. Pp. 100-107. 1991.

[BRAU-93] Braun, H.: *"Evolving Neural Networks for Application Oriented Problems"*. Proceedings of the 2nd Annual Conference on Evolutionary Programming. Pp. 62-71. 1993.

[BREM-62] Bremermann H.J.: *"Optimization through Evolution and Recombination"*. Self-Organizing Systems. Ed. Spartan. Eds. M.C. Yovits et al. 1962.

[BREM-65] Bremermann H.J., Rogson, M. & Salaff, S.: *"Search by Evolution Biophysics and Cybernetic Systems"*. Proceedings of the 2nd Cybernetic Sciences Symposium. Ed. Spartan. Eds. M Maxfield, A Callahan & L.J. Fogel. Pp. 157-167. 1965.

[BRIN-81] Brindle, A.: *"Genetic Algorithms for Function Optimization"*. Tesis Doctoral. University of Alberta. 1981.

[BROT-95] Brotherson, T. & Simpson, P.: *"Dynamic Feature Set Training of Neural Networks for Classification"*. Proceedings of the 4th Annual Conference on Evolutionary Computation. Pp. 79-90. 1995.

[CAJA-09] Cajal, S.R.: *"Histoligie du Système Nerveux de l'homme et des Vertébrés"*. Ed. A. Maloine. 1909.

[CARL-92] Carlin, M.: *"Radial Basis Function Neural Networks and Nonlinear Data Modelling"*. Proceedings of Neuronimes'92. Pp. 623-631. 1992.

[CARU-88] Caruana, R.A. & Schaffer, J.D.: *"Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms"*. Proceedings of the 5th International Conference on Machine Learning". Ed. Morgan Kaufmann. Pp. 153-161. 1988.

[CHAK-92] Chackraborty, K., Mehrotra, K., Chilukuri, K.M. & Ranka, S.: *"Forecasting the Behavior of Multivariate Time Series Using Neural Networks"*. Neural Networks. Vol. 5. N° 6. Pp. 961-970. 1992.

[CHAL-90] Chalmers, D.J.: *"The Evolution of Learning: an Experiment in Genetic Connectionism"*. Eds. D.S. Touretzky, J.L. Elman & G.E. Hinton. Proceedings of the 1990 Connectionist Models Summer School. Morgan Kaufmann. Pp. 81-90. 1990.

[CHAN-91] Chang, E. & Lippmann, R.: *"Using a Genetic Algorithm to Improve Pattern Classification Performance"*. Advances in Neural Information Processing Systems. Ed. Morgan Kaufmann. Eds. D. Touretsky. Pp. 797-803. 1991.

[CIOS-92] Cios, K.J. & Liu, N.: *"A Machine Learning Method for Generation of a Network Architecture: A Continous ID3 Algorithm"*. IEEE Transactions on Neural Networks. Vol. 3. N° 2. Pp. 280-291. 1992.

[COLL-91] Collins, R.J. & Jefferson, D.R.: *"Selection in Massively Parallel Genetic Algorithms"*. Proceedings of the Fourth International Joint Conference on Genetic Algorithms. Ed. Morgan Kaufmann Publishers. Eds. R.K. Belew & L.B. Booker. Pp. 249-256. 1991.

[CONN-94] Connor, J.T., Martin, R.D. & Atlas, L.E.: *"Recurrent Neural Networks and Robust Time Series Prediction"*. IEEE Transactions on Neural Networks. Vol. 5. N° 2. Pp. 240-254. 1994.

[CRIC-89] Crick, F.H.C.: *"The Recent Excitement about Neural Networks"*. Nature. N° 337. Pp. 129-132. 1989.

[CROM-91] Compton, W. & Stephens, N.M.: *"Using Genetic Algorithms to Search for Binary Sequences with Large Merit Factor"*. Proceedings of the Third IMA Conference on Cryptography and Coding. 1991.

[CYBE-89] Cybenko, G.: *"Approximation by Superposition of Sigmoidal Function"*. Mathematics of Control, Signals and Systems. N° 2. Pp. 303-314. 1989.

[DARD-91] Darden, L. & Cain, J.A.: *"Selection Type Theories"*. Philosophy Science. N° 56. Pp. 106-129. 1987.

[DARW-59] Darwin, C.R.: *"On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life"*. Ed. Murray. 1859.

[DAVD-91a] Davidor, Y.: *"A Genetic Algorithm Applied to Robot Trajectory Generation"*. Handbook of Genetic Algorithms. Ed. Van Nostrand Reinhold. Eds. L. Davis. Capítulo 12. Pp. 144-165. 1991.

[DAVD-91b] Davidor, Y.: *"A Naturally Occurring Niche and Species Phenomenon: The Model and First Results"*. Proceedings of the Fourth International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds. R.K. Belew & L.B. Booker. Pp. 18-23. 1991.

[DAVI-85a] Davis, L.: *"Applying Adaptive Algorithms to Epistatic Domains"*. Proceedings of the International Conference on Artificial Intelligence. Pp. 162-164. 1985.

[DAVI-85b] Davis, L.: *"Job Shop Scheduling with Genetic Algorithms"*. Proceedings of the First International Conference on Genetic Algorithms. Ed. Lawrence Erlbaum Associates. Eds. J.J. Grefenstette. Pp. 136-140. 1985.

[DAVI-89] Davis, L.: *"Adapting Operator Probabilities in Genetic Algorithms"*. Proceedings of the Third International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Pp. 61-69. 1989.

[DAVI-91] Davis, L.: *"Hybridization and Numerical Representation"*. The Handbook of Genetic Algorithms. Ed. Van Nostrand Reinhold. Ed. L. Davis. Pp. 61-71 1991.

[DAWK-86] Dawkins, R.: *"The Blind Watchmaker"*. Ed. Claredon. Oxford, UK. 1986.

[DAY-93] Day, S.P. & Davenport, M.R.: *"Continous-time Temporal Back-Propagation with Adaptive Time Delays"*. IEEE Transactions on Neural Networks. Nº 4. Pp. 348-354. 1993.

[DEB-91] Deb, K. & Goldberg, D.E.: *"Analyzing Deception in Trap Functions"*. Technical Report IlliGal 91009. 1991.

[DEJO-75] DeJong, K.A.: *"Analysis of Behavior of a Class of Genetic Adaptive Systems"*. Tesis Doctoral. University of Michigan. 1975.

[DEJO-85] DeJong, K.A.: *"Genetic Algorithms: A 10 Years Perspective"*. Proceedings of the First International Conference on Genetic Algorithms. Ed. Lawrence Erlbaum Associates. Eds. J.J. Grefenstette. Pp. 169-177. 1985.

[DEJO-90] DeJong, K.A. & Spears, W.M.: *"An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms"*. Parallel Problem Solving from Nature. Ed. Springer-Verlag. Eds. H.P. Schewefel & R. Männer. Pp. 38-47. 1990.

[DELI-96] DeLiang Wang, Xiaomei Liu & Stanley C. Ahalt: *"On Temporal Generalization of Simple Recurrent Networks"*. Neural Networks. Vol. 9. Nº 7. Pp. 1099-1118. 1996.

[DEME-93] DeMers, D. & Cottrell, G.: *"Non-Linear Dimensionality Reduction"*. Advances in Neural Information Systems. Morgan-Kaufmann. Eds. S.J. Hanson, J.D. Cowan & C.L. Giles. 1993.

[DOBZ-70] Dobzhansky, T.: *"Genetics of the Evolutionary Processes"*. Columbia University Press. 1970.

[DODD-91] Dodd, N., Macfarlane, D. & Marland, C.: *"Optimization of Artificial Neural Networks Structure Using Genetic Techniques Implemented on Multiple Transputers"*. Proceedings of Transputing '91. 1991.

[DORA-96] Dorado, J., Santos, A. y Pazos, A.: *"Methodology for the Construction of more Efficient Artificial Neural Networks by Means of Studying and Selecting the Training Set"*. International Conference on Neural Networks ICNN96. pp. 1285-1290. 1996.

[DUDA-73] Duda, R. & Hart, P.: *"Pattern Classification and Scene Analysis"*. Ed. J. Wiley & Sons. 1973.

[EAST-93] East, I.R. & Macfarlane, D.: *"Implementation in Occam of Parallel Genetic Algorithms on Transputer Networks"*. Parallel Genetic Algorithms. Ed. IOS Press. Eds. J. Stender. Pp. 43-63. 1993.

[ELMA-90] Elman, J.L.: *"Finding Structure in Time"*. Cognitive Science nº 14. Pp. 179-211. 1990.

[EPAN-69] Epanechnikov, V.A.: *"Non-parametric Estimation of a Multivariate Probability Density"*. Theory Probability Application. Vol. 14. Pp. 153-158. 1969.

[ENSL-92] Ensley, D. & Nelson, D.E.: *"Extrapolation of Mackey-Glass Data using Cascade Correlation"*. Simulation. Vol. 58. Nº 5. Pp. 333-339. 1992.

[ERIC-91] Ericson, C. & Ordonez-Reinoso, I.: *"Dialogue on Uniform Crossover"*. GA-Digest. Nº 5. 1991.

[ERSO-90] Ersoy, O.K. & Hong, D.: *"Parallel, Self-Organizing, Hierarchical Neural Networks"*. IEEE Transactions on Neural Networks. Vol 1. Nº 2. Pp. 167-178. 1990.

[ESHE-89] Eshelman, L.J., Caruna, R. & Schaffer, J.D.: *"Biases in the Crossover Landscape"*. Proceedings of the Third International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds. J.D. Schaffer. Pp. 10-19. 1989.

[ESHE-91a] Eshelman, L.J. & Schaffer, J.D.: *"GAs and Very Fast Simulated Re-annealing"*. GA-Digest. Nº 5. 1991.

[ESHE-91b] Eshelman, L.J.: *"Bit-Climbers and Naïve Evolution"*. GA-Digest. Nº 5. 1991.

[ESHE-93] Eshelman, L.J. & Schaffer, J.D.: *"Real-coded Genetic Algorithms and Interval Schemata"*. Foundations of Genetic Algorithms. Ed. Morgan Kaufmann. Eds. L.D. Whitley. Nº 2. Pp. 187-202. 1993.

[FAHL-90] Fahlman, S.E. & Lebiere, C.: *"The Cascade Correlation Learning Architecture"*. Report Carnegie Mellon University. Nº CMU-CS-90-100. 1990.

[FERN-84] Fernandez, R.E.: "Análisis de Identificación de Modelos ARIMA mediante Simulación con Métodos de Montecarlo". Tesis Doctoral. Universidad de Santiago de Compostela. 1984.

[FOGA-89] Fogarty, T.C.: *"Varying the Probability of Mutation in the Genetic Algorithm"*. Proceedings of the Third International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds. J.D. Shaffer. Pp. 104-109. 1989.

[FOGE-62] Fogel, L.J.: *"Autonomous Automata Industial"*. Res. 4. 1962.

[FOGE-66] Fogel, L.J., Owens A.J. & Walsh M.J.: *"Artificial Intelligence through Simulated Evolution"*. Ed. Wiley. 1966.

[FOGE-92] Fogel, D.B. & Atmar, J.W. (eds) Proceedings of the 1st Annual Conference on Evolutionary Programming. Ed. Evolutionary Programming Society. 1992.

[FOGE-95] Fogel, D.B.: *"Evolutionary Computation: Toward a New Philosophy of Machine Intelligence"*. IEEE. Piscataway, NJ, USA.1995.

[FONT-91] Fontaneri, J.F. & Meir, R.: *"Evoving a Learning Algorithm for the Binary Perceptron"*. Networks. Vol. 2. Pp. 353-359. 1991.

[FRAN-72] Franz, D.R.: *"Non-liniarities in Genetic Adaptive Search"*. Tesis Doctoral. University of Michigan. 1972.

[FRAS-57] Fraser, A.S.: *"Simulation of Genetic Systems by Automatic Digital Computers"*. Australian Journal Biological Sciences. Nº 10. Pp. 484-499. 1957.

[FREA-90] Frean, M.: *"The Upstart Algorithm: A Method for Constructing and Training Feedforward Neural Networks"*. Neural Computation. N° 2. Pp. 198-209. 1990.

[FRIE-58] Friedberg, R.M.: *"A Learning Machine: Part I"*. IBM. J. 2. Pp. 2-13. 1958.

[FRIE-59] Friedberg, R.M., Dunham, B. & North, J.H.: *"A Learning Machine: Part II"*. IBM. J. 3. Pp. 282-287. 1959.

[FUTU-86] Futuyma, D.J.: *"Evolutionary Biology"*. Ed. Sinauer. 1986.

[FUKU-75] Fukushima, K.: *"Cognitron: A Self-Organizing Multilayered Neural Network"*. Biological Cybernetics n°20. pp. 121-136. 1975.

[GALL-86] Gallant, S.I.: *"Optimal Linear Discriminants"*. Proceedings of the 8th International Conference on Pattern Recognition. Vol. 2. Pp. 849-854. 1986.

[GOLD-83] Goldberg, D.E.: *"Computer-Aided Gas Pipeline Operation using Genetic Algorithms and Rule Learning"*. Tesis Doctoral. Universidad de Michigan. 1983.

[GOLD-85] Goldberg, D.E. & Lingle, R.L.: *"Alleles, Loci and the Traveling Salesman Problem"*. Proceedings of the 1st International Conference on Genetic Algorithms. Ed. Erbaum. Eds. J.J. Grefenstette. Pp. 154-159. 1985.

[GOLD-87a] Goldberg, D.E. & Segrest, P.: *"Finite Markov Chain Analysis of Genetic Algorithms"*. Proceedings of the Second International Conference on Genetic Algorithms. Ed. Lawrence Erlbaum Associates. Eds. J.J. Grefenstette. Pp. 1-8. 1987.

[GOLD-87b] Goldberg, D.E.: *"Simple Genetic Algorithms and the Minimal Deceptive Problem"*. Genetic Algorithms and Simulated Annealing. Ed. Pitman. Eds. L. Davis. Capítulo 6. Pp. 74-88. 1987.

[GOLD-87c] Goldberg, D.E. & Richardson J.: *"Genetic Algorithms with Sharing for Multimodal Function Optimization"*. Proceedings of the Second International Conference on Genetic Algorithms. Ed. Lawrence Erlbaum Associates. Eds. J.J. Grefenstette. Pp. 41-49. 1987.

[GOLD-89a] Goldberg, D.E.: *"Genetic Algorithms in Search, Optimization and Machine Learning"*. Ed. Addison-Wesley. 1989.

[GOLD-89b] Goldberg, D.E.: *"Zen and the Art of Genetic Algorithms"*. Proceedings of the Third International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds. J.D. Schaffer. Pp. 80-85. 1989.

[GOLD-90a] Goldberg, D.E. & Bridges C.L.: *"An Analysis of a Reordering Operator on a GA-hard Problem"*. Biological Cybernetics. N° 62. Pp. 397-405. 1990.

[GOLD-90b] Goldberg, D.E.: *"The Theory of Virtual Alphabets"*. Parallel Problem Solving from Nature. Ed. Springer-Verlag. Eds. J.P. Schewefel & R. Männer. Pp. 13-22. 1990.

[GOLD-91a] Goldberg, D.E., Deb, K. & Korb, B.: *"Don't Worry, Be Messy"*. Proceedings of the 4th Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds R.K. Belew & L.B. Booker. Pp. 24-30. 1991.

[GOLD-91b] Goldberg, D.E. & Deb, K.: *"A Comparative Analysis of Selection Schemes Used in Genetic Algorithms"*. Foundations of Genetic Algorithms. Ed. Morgan Kaufmann. Eds. G.J.E. Rawlins. Pp 69-93. 1991.

[GOLD-92] Goldberg, D.E., Deb, K. & Horn J.: *"Massive Multimodality, Deception and Genetic Algorithms"*. Parallel Problem Solving from Nature. Ed. North-Holland. Eds. R. Männer & B. Manderick. N° 2. Pp. 37-46. 1992.

[GORD-68] Gordon, T.J. & Hayward, H.: *"Initial Experiment with the Cross-impact Matrix Method of Forecasting"*. Futures. N° 1. 1968.

[GORG-89] Gorges-Schleuter, M.: *"ASPARGOS: An Asynchronous Parallel Genetic Optimization Strategy"*. Proceedings of the Third International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds. J.D. Schaffer. Pp. 422-427. 1989.

[GREE-90] Green, N.P.O., Stout, G.W. & Taylor, D.J.: *"Biological Science 1 & 2"*. Cambridge University Press. 1990.

[GREF-84] Grefenstette, J.J.: *"GENESIS: A System for Using Genetic Search Procedures"*. Proceedings of the 1984 Conference on Intelligent Systems and Machines. Pp. 161-165. 1984.

[GREF-85a] Grefenstette, J.J. (ed) Proceedings of the 1st International Conference on Genetic Algorithms and their Applications. Ed. Erlbaum. 1985.

[GREF-85b] Grefenstette, J.J., Gopal, R., Rosmaita,B.J. & Van Gucht, D.: *"Genetic Algorithms for the Traveling Salesman Problem"*. Proceedings of the 1st International Conference on Genetic Algorithms. Ed. Erbaum. Eds. J.J. Grefenstette. Pp. 160-168. 1985.

[GREF-86] Grefenstette, J.J.: *"Optimization of Control Parameters for Genetic Algorithms"*. IEEE Transactions SMC. N° 16. Pp. 122-128. 1986.

[GREF-87] Grefenstette, J.J.: *"Incorporating Problem Specific Knowledge into Genetic Algorithms"*. Genetic Algorithms and Simulated Annealing. Ed. Pitman. Eds. L. Davis. Capítulo 4. Pp. 42-60. 1987.

[GREF-91] Grefenstette, J.J.: *"Strategy Acquisition with Genetic Algorithms"*. Handbook of Genetic Algorithms. Ed. Van Nostrand Reinhold. Eds. L. Davis. Capítulo 14. Pp. 186-201. 1991.

[GREF-93] Grefenstette, J.J.: *"Deception Considered Harmful"*. Foundations of Genetic Algorithms. Ed. Morgan Kaufmann. Eds. L. Darrel Whitley. N° 2. Pp. 75-91. 1993.

[GRUA-92] Gruau, F.C.: *"Cellular Encoding of Genetic Neural Networks"*. Technical Report. LIP-IMAG Ecole Normale Superieure de Lyon. 1992.

[HARP-89] Harp, S.A., Samad, T. & Guha, A.: *"Towards the Genetic Synthesis of Neural Networks"*. Third International Conference on Genetic Algorithms. Eds. J.D. Schaffer. Morgan Kauffmann. Pp. 360-369. 1989.

[HART-88] Hartl, D.L.: *"A Primer of Population Genetics"*. Ed. Sinauer Associates Inc. 1988.

[HEBB-49] Hebb, D.O.: *"The Organization of Behavior: A Neuropsychological Theory"*. Ed. Wiley. 1949.

[HEE-97] Hee Yeal, & y Sung Yan B.: *"An Improved Time Series Prediction by Applying the Layer-by-Layer Learning Method to FIR Neural Networks"*. Neural Networks. Vol. 10 N°. 9. Pp. 1717-1729. 1997.

[HELM-59] Helmer, O & Rescher, N.: *"On the Epistemology of the Inexact Sciences"*. Management Science. N° 6. 1959.

[HOFF-89] Hoffmann, A.: *"Arguments on Evolution: A Paleontologist's Perspective"*. Oxford University Press. New York. 1989.

[HOLL-62] Holland J.H.: *"Outline for a Logical Theory of Adaptive Systems"* Journal of ACM. N° 9. Pp. 297-314. 1962.

[HOLL-67] Holland, J.H.: *"Nonlinear Environments Permitting Efficient Adaptation Computer and Information Sciences II"*. Ed. Academic. New York. 1967

[HOLL-69] Holland J.H.: *"Adaptive Plans Optimal for Payoff-only Environments"*. Proceedings of the 2nd Hawaii International Conference on System Sciences. Pp. 917-920. 1969.

[HOLL-71] Holland, J.H.: *"Processing and Processors for Schemata Associative Information Processing"*. Ed. Elsevier. Eds.E.L. Jacks. Pp. 127-146. 1971.

[HOLL-73] Holland, J.H.: *"Genetic Algorithms and the optimal Allocation of Trials"*. Journal of Computing. N° 2. Pp. 88-105. 1973.

[HOLL-75] Holland J.H.: *"Adaptation in Natural and Artificial Systems"*. University of Michigan Presss. Ann Arbor, MI, USA. 1975.

[HOLL-87] Holland J.H.: *"Genetic Algorithms and Classifier Systems: Foundations and Future Directions"*. Proceedings of the Second International Conference on Genetic Algorithms". Ed. Lawrence Erlbaum Associates. Eds. J.J. Grefenstette. Pp. 82-89. 1987.

[HOLS-71] Hollstien, R.B.: *"Artificial Genetic Adaptation in Computer Control Systems"*. Tesis Doctoral. University of Michigan. 1971.

[HUXL-63] Huxley, J.: *"The Evolutionary Process Evolution as a Process"*. Ed. Collier. Eds. J. Huxley, A.C. Hardy & E.B. Ford. Pp. 9-33. 1963.

[JANI-91] Janikow, C.Z. & Michaelewicz, Z.: *"An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms"*. Proceedings of the 4th International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds. R.K. Belew & L.B. Booker. Pp. 31-36. 1991.

[JOYA-93] Joya, G., Frias, J.J., Marín, M.M. & Sandoval, F.: *"New Learning Strategies from the Microscopy Level of an Artificial Neural Networks"*. Electronics Letters. Vol. 29. N° 20. Pp.1775-1777. 1993.

[KARN-90] Karnin, E.D.: *"A Simple Procedure for Pruning Back-Propagation Trained Neural Networks"*. IEEE Transactions on Neural Networks. Vol. 1. N° 2. Pp. 239-242. 1990.

[KITA-90] Kitano, H.: *"Designing Neural Networks Using Genetic Algorithms with Graph Generation System"*. Complex Systems. Vol. 4. Pp. 461-476. 1990.

[KNER-91] Knerr, S., Personnaz, L. & Dreyfus, G.: *"A New Approach to the Design of Neural Network Classifiers and its Application to the Automatic Recognition of Handwritten Digits"*. International Joint Conference on Neural Networks. Pp. 91-96. 1991.

[KOHO-82] Kohonen, T.: *"Self-Organized Formation of Topologically Correct Feature Map"*. Biological Cybernetics. Vol.43. pp. 59-69. 1982.

[KOHO-88] Kohonen, T.: *"Self-Organization and Associtive Memory"*. Ed. Springer Verlag. 1988.

[KOZA-92] Koza, J.R.: *"Genetic Programming"*. MIT Press. 1992.

[LAMA-14] Lamarck, J.B.: *"Zoological Philosophy"*. 1914. Reimpreso por Ed. Haffner. 1963.

[LANG-88] Lang, K.J. & Hinton, G.E.: *"The Development of the Time-Delay Neural Network Architecture for Speech Recognition"*. Technical Report CMU-CS-88-152. Carnegie-Mellon University. 1988.

[LANG-87] Langton, C.G.: *"Artificial Life"*. Ed. Addison-Wesley. Eds. C.G. Langton. Pp. 1-47. 1987.

[LAPE-87] Lapedes, A. & Farber, R.: *"Nonlinear Signal Processing Using Neural Network: Prediction and System Modelling"*. Technical Report LA-UR-87-2662. Los Alamos National Lab. 1987.

[LAPE-88] Lapedes, A. & Farber, R.: *"How Neural Nets Works"*. Neural Information Processing Systems. Eds. D.Z. Anderson. American Institute of Physics. Pp. 442-456. 1988.

[LEE-91] Lee, S. & Rhee, M.K.: *"A Gaussian Potential Function Network with Hierarchically Self-Organizing Learning"*. Neural Networks. Nº 4. Pp. 207-224. 1991.

[LEVE-91] Levenick, J.: *"Inserting Introns Improves Genetic Algorithm Success Rate: Taking a Cue from Biology"*. Proceedings of the Fourth International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds. R.K. Belew & L.B. Booker. Pp. 123-127. 1991.

[LEWI-90] Lewin, B.: *"Genes IV"*. Oxford University Press. Oxford. 1990.

[LEWO-74] Lewontin, R.C.: *"The Genetic Basis of Evolutionary Change"*. Columbia University Press. 1974.

[LIN-92] Lin, D.T., DayHoff, J.E. & Ligomenides, P.A.: *"Adaptive Time-Delay Neural Network for Temporal Correlation and Prediction"*. SPIE Intelligent Robots and Computer Vision XI: Biological, Neural Net and 3-D Methods. Vol. 1826. Pp. 170-181. 1992.

[LIN-93] Lin, D.T., DayHoff, J.E. & Ligomenides, P.A.: *"Learning Spatiotemporal Topology Using an Adaptive Time-Delay Neural Network"*. World Congress on Neural Networks. Vol.1. Pp.291-294. 1993.

[LOUI-91] Louis, S.J. & G.J.E. Rawlins.: *"Designer Genetic Algorithms: Genetic Algorithms in Structure Design"*. Proceedings of the Fourth International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds. R.K. Belew & L.B. Booker. Pp. 53-60. 1991.

[MCDO-94] McDonnell, J.R. & Waagen, D.: *"Evolving Recurrent Perceptrons for Time Series Modeling"*. IEEE Transactions on Neural Networks. Vol. 5. Nº 1. 1994.

[MAKR-78] Makridakis, S. & Wheelwright, S.C.: *"Forecasting Methods and Applications"*. Ed. John Wiley. 1978.

[MALT-26] Malthus, T.R.: *"An Essay on the Principle of Population, as Affects the Future Improvement of Society"*. 6th edn. Ed. Murray. 1826.

[MANN-92] Männer, R. & Manderick, B. (eds): *"Parallel Problem Solving from Nature 2"*. Ed. Elsevier-North-Holland. 1992.

[MARC-90] Marchand, M., Golea, M. & Ruján. P.: *"A Convergence Theorem for Sequential Learning in Two-Layer Perceptrons"*. Europhysics Letters. Nº 11. Pp. 487-492. 1990.

[MARU-92] Maruyama, T., Konagaya, A. & Konishi, K.: *"An Asynchronous Fine-Grained Parallel Genetic Algorithm"*. Parallel Problem Solving from Nature 2. Ed. Elsevier Science Publishers B.V. Eds. Männer, R. & Manderick, B. Pp. 563-572.

[MATH-94] Mathias, K.E. & Whitley, L.D.: *"Changing Representations During Search: a Comparative Study of Delta Encoding"*. Evolutionary Computation. Nº 2. 1994.

[MAYN-89] Maynard Smith J.: *"Evolutionary Genetics"*. Oxford University Press. 1989.

[MAYR-63] Mayr, E.: *"Animal Species and Evolution"*. Ed. Belknap. Cambridge, MA, USA. 1963.

[MAYR-82] Mayr, E.: *"The Growth of Biological Thouht: Diversity, Evolution and Inheritance"*. Ed. Belknap. Cambridge, MA, USA. 1982.

[MAYR-88] Mayr, E.: *"Toward a New Philosophy of Biology: Observations of an Evolutionist"*. Ed. Belknap. 1998.

[MCBR-65] McBride, L.E. & Narendra, K.S.: *"Optimization of Time-Varying Systems"*. IEEE Transactions on Automatic Control. AC-10. Pp. 289-294.1965.

[MCCL-86] McClelland, J. & Rumelhart, D.: *"Explorations in Parallel Distributed Processing"*. Vol. 1 y 2. MIT Press. Cambridge, MA. 1986.

[MCCU-43] McCulloch, W.S. & Pitts, W.: *"A Logical Calculus of Ideas Immanent in Nervous Activity"*. Bulletin of Mathematical Biophysics. N° 5. Pp. 115-133. 1943.

[MCDO-96] McDonnell, J.R. & Waagen, D.: *"Evolving Recurrent Perceptrons for Time Series Modeling"*. IEEE Transactions on Neural Networks. Vol. 5 N° 1. Pp. 24-38. 1996.

[MEAD-91] Mead, W.C., Jones, R.D., Lee, Y.C., Barnes, C.W., Flake, G.W. Lee, L.A. & Rourke, M.K.: *"Using CNLS-Net to Predict the Mackey-Glass Chaotic Time Series"*. Proceedings of the International Joint Conference on Neural Networks. Vol. 2. Pp. 485-490. 1991.

[MEZA-89] Mezard, M. & Nadal, J.P.: *"Learning in Feedforward Layered Networks: The Tiling Algorithm"*. Journal of Physics. N° A22. Pp. 2191-2203. 1989.

[MICH-91] Michalewicz, Z. & C.Z. Janikow: *"Handing Constraints in Genetic Algorithms"*. Proceedings of the Fourth International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds. R.K. Belew & L.B. Booker. Pp. 151-157. 1991.

[MICH-92] Michalewicz, Z.: *"Genetic Algorithms + Data Structures = Evolution Programs"*. Ed. Springer –Verlag. 1992.

[MICH-94] Michalewicz, Z. et al (eds) Proceedings of the 1st IEEE Conference on Evolutionary Computation. Orlando, Florida, USA. 1994.

[MILL-89] Miller, G.F., Todd, P.M. & Hedge, S.U.: *"Designing Neural Networks Using Genetic Algorithms"*. Eds. J.D. Schaffer. Third International Conference on Genetic Algorithms. Ed. Morgan Kauffmann. Pp. 379-384. 1989.

[MINS-67] Minsky, M.: *"Computation: Finite and Infinite Machines"*. Ed. Prentice-Hall. Englewood Cliffs. 1967.

[MINS-69] Minsky, M. & Papert, S.: *"Perceptrons"*. MIT Press. Cambridge, MA.1969.

[MINS-88] Minsky, M. & Papert, S.: *"Perceptrons. Expanded Edition"*. MIT Press. Cambridge, MA.1988.

[MJOL-89] Mjolsness, E., Sharp, D.H. & Alpert, B.K.: *"Scaling, Machine Learning and Genetic Neural Nets"*. Advances in Applied Mathematics. Vol. 10. Pp. 137-163. 1989.

[MONT-89] Montana, D.J. & Davis, L.: *"Training Feedforward Neural Networks Using Genetic Algorithms"*. Proceedings of Eleventh International Joint Conference on Artificial Intelligence". Pp. 762-767. 1989.

[MORE-93a] Moreno, J.M., Castillo, F. y Cabestany, J.: *"Enhanced Unit Training for Piecewise Linear Separation Incremental Algorithms"*. European Symposium on Artificial Neural Networks. Pp. 33-38. 1993.

[MORE-93b] Moreno, J.M., Castillo, F. y Cabestany, J.: *"Optimized Learning for Improving the Evolution of Piecewise Linear Separation Incremental Algorithms"*. New Trends in Neural Computation. Ed. Springer Verlag. 1993.

[MORE-94] Moreno, J.M., Castillo, F. y Cabestany, J.: *"Improving Piecewise Linear Separation Incremental Algorithms Using Complexity Reduction Methods"*. European Symposium on Artificial Neural Networks. 1994.

[MUSA-92] Musavi, M.T., Ahmed, W., Chan, K.H., Faris, K.B. & Hummels, D.M. *"On the Training of Radial Basis Function Classifiers"*, Neural Networks, N° 5. Pp.595-603. 1992.

[NEUM-56] von Neumann, J.: *"Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components"*. Automata Studies (C.E. Shannon & J. McCarthy, eds). Princeton University Press. pp. 43-98. 1956.

[NEUM-58] von Neumann, J.: *"The Computer and the Brain"*. Yale University Press. 1958.

[PARA-90] Parallel Computing, Especial Artificial Neural Networks. Vol. 14. N° 3. 1990.

[PARK-85] Parker, D.B.: *"Learning-Logic: Casting the Cortex of the Human Brain in Silicon"*. Technical Report TR-47. Center for Computational Research in Economics and Management Science. MIT. 1985.

[PARZ-62] Parzen, E.: *"On Estimation of a Probability Density Function Interpolation"*. Annual Mathematical Statistics. N° 33. Pp. 1065-1076. 1962.

[PASH-91] Park, D.C., El-Sharkawi, M.A., Marks II, R.J., Atlas, L.E. & Damborg, M.J.: *"Electric Load Forecasting Using an Artificial Neural Network"*. IEEE Transactions on Power Systems. Vol. 6. Pp. 442-449. N° 2. 1991.

[PAZO-96a] Pazos, A., Dorado, J. y Santos A.: "Mixing AI Techniques (ANN & GA) to Recognize Hip Joint Patterns in X-Ray Image". 4° Congresso Português de Engenharia Biomédica. Aveiro Portugal. Pp. X.2.1-X.2.4. 1996.

[PAZO-96b] Pazos, A., Dorado, J. y Santos A.: "Detection of Patterns in Radiographs using ANN Designed and Trained with GA". Proceedings of the Genetic Programming 96 Conference. Pp. 432. 1996.

[PAZO-98] Pazos, A., Dorado, J. y Santos A.: "AG para el Entrenamiento de RNA Recurrentes con Activaciones Temporales". Revista Iberoamericana de Inteligencia Artificial. N° 5. Pp. 26-31. 1998.

[PETT-87] Pettey, C.B. Leuze, M.R. & Grefenstette, J.J.: *"A Parallel Genetic Algorithm"*. Proceedings of the Second International Conference on Genetic Algorithms. Ed. Lawrence Erlbaum Associates. Eds. J.J. Grefenstette. Pp. 155-162. 1987.

[PLAT-91] Platt, J.: *"A Resource Allocating Network for Function Interpolation"*. Neural Computation. N° 3. Pp. 213-225. 1991.

[POLA-93] Polani, D. & Uthmann T.: *"Training Kohonen Feature Maps in Different Topologies: An Analysis Using Genetic Algorithms"*. Proceedings of the 5th International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Pp. 326-334. 1993.

[RADC-93] Radcliffe, N.J.: *"Genetic Set Recombination and its Application to Neural Network Topology Optimization"*. Neural Computing and Applications. Vol. 1. N° 1. Pp. 67-90. 1993.

[RAY-91] Ray, T.: *"An Approach to the Synthesis of Artificial Life II"*. Ed. Addison-Wesley. Eds. C.G. Langton, C. Taylor, J.D. Farmer & S. Rasmussen. Pp. 371-408. 1991.

[RECH-65] Rechenberg, I.: *"Cybernetic Solution Path of an Experimental Problem Royal Aircraft"*. Establishment Library Translation UK.1965.

[RECH-94] Rechenberg, I.: *"Evolutionsstrategies '94"*. Ed. Frommann-Holzboorg. 1994.

[REIL-82] Reilly, D.L., Cooper, L.N. & Elbaum, C.: *"A Neural Model for Category Learning"*. Biological Cybernetics. Nº 45. Pp. 35-41. 1982.

[ROBB-93] Robbins, G.E., Hughes, J.C., Plumbley, M.D., Fallside, F. & Prager, R.: *"Generation and Adaptation of Neural Networks by Evolutionary Techniques (GANNET)"*. Neural Computing and Applications. Vol. 1. Nº 1. Pp. 22-30. 1993.

[ROBI-91] Robinson, A.J. & Fallside, F.: *"A Recurrent Error Propagation Speech Recognition System"*. Computer Speech and Language. Nº 5. Pp. 259-274. 1991.

[ROSE-58] Rosenblatt, F.: *"The Perceptron: A Probabilistic Model for Information Storage and Oganization in the Brain"*. Psylogical Review. Nº 65. 1958.

[ROSE-62] Rosenblatt, F.: *"Principles of Neurodynamics"*. Ed. Spartan. 1962.

[ROSN-67] Rosenberg, R.: *"Simulation of Genetic Populations with Biochemical Properties"*. Tesis Doctoral. University of Michigan. 1967.

[RUME-86] Rumelhart, D.E., Hinton, G.E. & Williams, R.J.: *"Learning Internal Representations by Error Propagation"*. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press. Eds. D.E. Rumelhart & J.L. McClelland. Vol. 1. Capítulo 8. 1986.

[RUTE-89] Rutenbar, R.A.: *"Simulated Annealing Algorthims: An Overview"*. IEEE Circuits and Devices Magazine. Pp. 19-26. Enero. 1989.

[SAMP-81] Sampson, J.R.: *"A Synopsis of the Fifth Annual Ann Arbor Adaptive Systems Workshop"*. Department of Computing and Communication Science, Logic of Computers Group. Technical Report. University of Michigan. 1981.

[SANC-94] Sánchez, E., Barro, S. y Regueiro, C.V.: *"Artificial Neural Networks Implementations on Vectorial Supercomputers"*. IEEEE International Conference on Neural Networks. Pp. 3938-3943. Orlando. Florida. 1994.

[SANG-91] Sanger, T.D.: *"A Tree-Structured Adaptive Network for Function Approximation in High-Dimensional Spaces"*. IEEE Transactions on Neural Networks. Vol. 2. Nº 2. Pp. 285-293. 1991.

[SANT-98] Santos del Riego, Antonino: *"Desarrollo de una Metodología e Implementación de un Sistema basado en el Conocimiento de Filosofía Híbrida. Una Aplicación para la Evaluación del Impacto Ambiental"*. Tesis Doctoral. Universidade da Coruña. 1998.

[SATT-59a] Satterthwaite, F.E.: *"Random Balance Experimentation"*. Technometrics. Nº 1. Pp. 111-137. 1959.

[SATT-59b] Satterthwaite, F.E.: *"REVOP or Random Evolutionary Operation"*. Merrimack College Technical Report 10-10-59. 1959.

[SATO-91] Sato, A., Yamada, K., Tsukumo, J. & Temma, T.: *"Neural Network Models for Incremental Learning"*. Artificial Neural Networks. Elsevier Science Publishers. Eds. T. Kohonen, K. Makisata & O. Simula. 1991.

[SCHA-87] Schaffer, J.D. & Morishma, A.: *"An Adaptive Crossover Distribution Mechanism for Genetic Algorithms"*. Proceedings of the Second International Conference on Genetic Algorithms". Ed. Lawrence Erlbaum Associates. Eds. J.J. Grefenstette. Pp. 36-40. 1987.

[SCHA-89] Schaffer, J.D., Caruna, R.A., Eshelman L.J. & Das, R.: *"A Study of Control Parameters Affecting Online Performance of Genetics Algorithms for Function Optimization"*. Proceedings of the Third International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds. J.D. Schaffer. Pp. 51-60. 1989.

[SCHA-91] Schaffer, J.D. & Eshelman L.J.: *"On Crossover as an Evolutionarily Viable Strategy"*. Proceedings of the Fourth International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds. R.K. Belew & L.B. Booker. Pp. 61-68. 1991.

[SCHR-92] Schraudolph, N.N. & Belew, R.K.: *"Dynamic Parameter Encoding for Genetic Algorithms"*. Machine Learning. Vol. 9. N° 1. Pp. 9-21. 1992.

[SCHU-90] Schultz A.C. & Grefenstette J.J.: *"Improving Tactical Plans with Genetic Algorithms"*. Proceedings of the IEEE Conference of Tools for AI. Ed. IEEE Society Press. Pp. 328-344. 1990.

[SCHW-65] Schwefel H.P.: *"Kybernetische Evolution als Strategie der Experimentellen Forschung in der Strömungstechnik"*. Diploma Thesis. Technical University of Berlin. 1965.

[SCHW-91] Schwefel H.P. & Männer, R.: *"Parallel Problem Solving from Nature"*. Proceedings of the 1st Workshop PPSN I. Ed. Springer. Berlin, Alemania. 1991.

[SCHW-95] Schwefel H.P.: *"Evolution and Optimum Seeking"*. Ed. Wiley. New York. 1995.

[SEBA-92] Sebald, A.V. & Fogel, D.B.: *"Design of Fault Tolerant Neural Networks for Pattern Classification"*. Proceedings of the 1st Annual Conference on Evolutionary Programming. Pp. 90-96. 1992.

[SHAE-87] Shaefer C.G.: *"The ARGOT strategy: Adaptive Representation Genetic Optimizer Technique"*. Genetic Algorithms and their Applications: Proceedings of the 2nd International Conference on Genetic Algorithms. Ed. Erlbaum. Eds. J.J. Grefenstette. Pp. 50-58. 1987.

[SHAM-89] Shama, S.A.: *"Spatial and Temporal Processing in Central Auditory Networks"*. MIT Press. Methods in Neural Modeling. Koch, C. & Segev, I. Eds. Pp.247-289. 1989.

[SHAR-90] Sharda, R & Patil, R.B.: *"Neural Networks as Forecasting Experts: An Empirical Test"*. Proceedings of the IJCNN Meeting. Pp. 491-494. 1990.

[SHEP-90] Shepher, G.M.: *"The Significance of Real Neuron Architectures for Neural Network Simulations"*. Computational Neuroscience. MIT Press. Editor E.L. Schwartz. Pp. 82-96. 1990.

[SHIS-67] Shiskin, J., Young, A.H. & Musgrave, J.C.: *"The Variant of Census II Method Seasonal Adjustment Program"*. Bureau of the Census. Technical Paper. N° 15. 1967.

[SIEG-91] Siegelman, H. & Sontag, E.D.: *"Neural Networks are Universal Computing Devices"*. Technical Report SYCON-91-08. Rutgers Center for Systems and Control. 1991.

[SIMP-49] Simpson, G.G.: *"The Meaning of Evolution: a Study of the History of Life and its Significance for Man"*. Yale University Press. 1949.

[SINN-58] Sinnot, E.W., Dunn L.C. & Dobzhansky, T.: *"Principles of Genetics"*. Ed McGraw-Hill. New York. 1958.

[SIRA-90] Sirat, J.A. & Nadal, J.P.: *"Neural Trees: A New Tool for Classification"*. Technical Report. Laboratories d'Electronique Philips. 1990.

[SMIT-80] Smith, S.F.: *"A Learning System Based on Genetic Adaptive Algorithms"*. Tesis Doctoral. University of Pittsburgh. 1980.

[SPEA-91] Spears, W.M. & DeJong, K.: *"An Analysis of Multi-point Crossover"*. Foundations of Genetic Algorithms. Ed. Morgan Kaufmann. Eds. G.J.E. Rawlins. Pp. 301-315. 1991.

[SPEA-93] Spears, W.M.: *"Crossover or Mutation?"*. Foundations of Genetic Algorithms. Ed. Morgan Kaufmann. Eds. L. Darrel Withley. N° 2. Pp. 221-237. 1993.

[SPEC-90] Specht, D.: *"Probabilistic Neural Networks"*. Neural Networks. Vol. 3. Nº 1. 1990.

[SPEN-62] Spendley, W., Hext, G.R. yHimsworth, F.R.: *"Sequential Application of Simplex Designs in Optimization and Evolutionary Operation"*. Technometrics. Nº 4. Pp. 441-461. 1962.

[SPIE-91] Spiessens, P. & Manderick, B.: *"A Massively Parallel Genetic Algorithm: Implementation and First Analysis"*. Proceedings of the Fourth International Joint Conference on Genetic Algorithms. Ed. Morgan Kaufmann Publishers. Eds. R.K. Belew & L.B. Booker. Pp. 279-287. 1991.

[STAD-87] Stadnyk I.: *"Schema Recombination in a Pattern Recognition Problem"*. Proceedings of the Second International Conference on Genetic Algorithms. Ed. Lawrence Erlbaum Associates. Eds. J.J. Grefenstette. Pp. 27-35. 1987.

[STAN-75] Stanley, S.M.: *"A Theory of Evolution above the Species Level"*. Proceedings of the Natural Academy of Sciences. Nº 72. Pp. 646-650. 1975.

[STOR-89] Stork, D.: *"Is Backpropagation Biologically Plausible"*. International Joint Conference on Neural Networks". Vol. 2. Pp. 241-246. 1989.

[SUH-87] Suh J.Y. & Van Gucht D.: *"Incorporating Heuristic Information into Genetic Search"*. Proceedings of the Second International Conference on Genetic Algorithms. Ed. Lawrence Erlbaum Associates. Eds. J.J. Grefenstette. Pp. 100-107. 1987.

[SUN-88] Sun, G.Z., Chun, H.H. & Lee, Y.C.: *"Parallel Sequential Induction Network: A New Paradigm of Neural Network Architecture"*. IEEE International Conference on Neural Networks. Pp. 489-496. 1988.

[SUND-90] Sunderam, V.S.: *"PVM: A Framework for Parallel Distributed Computing"*. Concurrency Practice and Experience. Vol. 2. Nº 4. Pp. 315-339. 1990.

[SUTT-88] Sutton, R.S.: *"Learning to Predict by the Methods of Temporal Differences"*. Machine Learning. Vol. 3 nº 9. 1988.

[SYSW-89] Syswerda, G.: *"Uniform Crossover in Genetic Algorithms"*. Proceedings of the Third International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds. J.D. Schaffer. Pp. 2-9. 1989.

[SYSW-91] Syswerda, G.: *"Schedule Optimization Using Genetic Algorithms"*. Handbook of Genetic Algorithms. Ed. Van Nostrand Reinhold. Eds. L. Davis. Capítulo 21. Pp. 332-349. 1991.

[TANA-87] Tanese, R.: *"Parallel Genetic Algorithms for a Hypercube"*. Proceedings of the Second International Conference on Genetic Algorithms. Ed. Lawrence Erlbaum Associates. Eds. J.J. Grefenstette. Pp. 177-184. 1987.

[TANG-91] Tang, Z., Almeida, C. & Fishwick, P.A.: *"Time Series Forecasting Using Neural Networks vs. Box-Jenkins Methodology"*. Simulations. Vol. 57. Nº 5. Pp. 303-310. 1991.

[TINT-40] Tintner, G.: *"The Variate Difference Method"*. Ed. Principia Press. Bloomington. Indiana. 1940.

[TINT-63] Tintner, G. & Rao, J.N.K.: *"On the Variate Difference Method"*. Australian Journal of Statistics. Nº 5. Pp. 106-. 1963.

[TORR-91] Torras, C.: *"Neural Learning Algorithms and their Applications in Robotics"*. Self-Organization, Emerging Properties and Learning, Plenum Press. pp. 161-176. New York. 1991.

[VASS-96] Vassilios Petridis & Athanasios Kehagias: *"A Recurrent Network Implementation of Time Series Classification"*. Neural Computation. Nº. 8. Pp. 357-372. 1996.

[WAIB-89] Waibel, A., Hanazawa, T., Hinton, G.E., Shikano, K. & Lang, K.J.: *"Phoneme Recognition Usin Time-Delay Neural Networks"*. IEEE Transactions on Acoustics, Speech and Signal Processing n°37. pp. 390-398. 1989.

[WALD-61] Waldhmeirer, M.: *"The Sunspot Activity in the Years 1610-1960"*. Ed. Schulthess. 1961.

[WAN-90a] Wan, E.A.: *"Temporal Backpropagation for FIR Neural Networks"*. IEEE International Joint Congress on Neural Networks. Vol. 1. pp. 575-580. 1990.

[WAN-90b] Wan, E.A.: *"Temporal backpropagation: An Efficient Algorithm for Finite Impulse Response Neural Networks"*. Proceedings of the 1990 Connectionist models Summer School. Ed. Morgan Kaufmann. pp. 131-140. 1990.

[WAN-93] Wan, E.A.: *"Time Series Prediction by Using a Connectionist Network with Internal Delay Lines"*. Time Series Prediction: Forecasting the Future and Understanding the Past. Ed. Addison-Wesley. Weigend, A.S. & Gershenfeld N.A., Eds. Pp. 195-217. 1993.

[WANG-96] Wang, D., Liu X.. & Ahalt S.C.: *"On Temporal Generalization of Simple Recurrent Networks"*. Neural Networks. Vol. 9 N°. 7. Pp. 1099-1118. 1996.

[WEIG-91] Weigend, A.S., Rumelhart, D.E. & Huberman, B.A.: *"Generalization by Weight-Elimination with Application to Forecasting"*. Advances in Neural Information Processing Systems. Ed. Morgan-Kaufman. N° 3. Pp. 875-882. 1991.

[WERB-74] Werbos, P.J.: *"Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences"*. Tesis Doctoral. Harvard University. 1974.

[WERM-98] Wermter S.: *"Hybrid Neural and Symbolic Language Processing"*. Proceedings of the Interdisciplinary Conference, Guenne. 1998.

[WETZ-83] Wetzel, A.: *"Evaluation of the Effectiveness of Genetic Algorithms in Combinatorial Optimization"*. University of Pittsburgh. 1983.

[WHCH-94] Whitehead, B. & Choate, T.: *"Evolving Space-Filling Curves to Distribute Radial Basis Functions over an Input Space"*. IEEE Transactions on Neural Networks. N° 5. Pp. 15-23. 1994.

[WHIL-54] Whittle, P.: *"A Statistical Investigation of Sunspot Observations with Special Reference to h. Alfven's Sunspot Model"*. The Astrophysical Journal. N° 120. Pp. 251-260. 1954.

[WHIT-89] Whitley, D.: *"The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best"*. Proceedings of the Third International Conference on Genetic Algorithms. Ed. Morgan Kaufmann. Eds. J.D. Shaffer. Pp. 116-121. 1989.

[WHIT-90a] Whitley, D. & Starkweather, T.: *"Optimizing Small Neural Networks Using a Distributed Genetic Algorithm"*. Proceeding of the International Joint Conference on Neural Networks. Lawrence Erlbaum Ed. Pp. 206-209. 1990.

[WHIT-90b] Whitley, D., Starkweather, T. & Bogart, C.: *"Genetic Algorithms and Neural Networks. Connections and Connectivity"*. Parallel Computing. Vol. 14. Pp. 347-361. 1990.

[WIDR-60] Widrow, B. & Hoff, M.E.: *"Adaptive Switching Circuits"*. IRE WESCON Convention Record. pp. 96-104. 1960.

[WILL-76] Willshaw, D.J. & von der Malsburg, C.: *"How Patterned Neural Conections Can Be Set up by Self-Organization"*. Proceedings of the Royal Society of London. Series B. N° 194. Pp. 431-445. 1976.

[WILL-89] Williams, R.J. & Zipser, D.: *"A Learning Algorithm for Continually Running Fully Recurrent Neural Networks"*. Neural Computation nº 1. Pp. 270-280. 1989.

[WILL-90] Williams, R.J. & Peng, J.: *"An Efficient Gradient-Based Algorithm for on-line Training of Recurrent Networks Trajectories"*. Neural Computation. Nº 2. Pp.490-501. 1990.

[WOLD-38] Wold, H.: *"A Study in the Analysis of Stationary Time Series"*. Ed. Almquist and Wicksell. Upsala. Segunda Edición. 1954.

[WOLP-96] Wolpert, D.H. & Macready, W.G.: *"No Free Lunch Theorems for Search"*. Technical Report SFI-TR-95-02-010. Santa Fe Institute. 1996.

[WOOL-68] Wooldridge, D.E.: *"The Mechanical Man: The Physical Basis of Intelligent Life"*. Ed. McGraw-Hill. 1968.

[WRIG-31] Wright, S.: *"Evolution in Mendelian Populations"*. Genetics. Nº 16. Pp. 97-159. 1931.

[WRIG-32] Wright, S.: *"The roles of Mutation, Inbreeding, Crossbreeding and Selection in Evolution"*. Proceedings of 6th International Congress on Genetics. Vol. 1. 1932.

[WRIG-60] Wright, S.: *"The Evolution of Life, panel discussion Evolution After Darwin"*. Issues in Evolution. University of Chicago Press. Eds. S. Tax & C. Callender. Vol. 3. 1960.

[WRIT-91] Wright, A.: *"Genetic Algorithms for Real Parameter Optimization"*. Foundations of Genetic Algortithms. Ed. Morgan Kaufmann. Eds. G.J.E. Rawlins. Pp. 205-218. 1991.

[YANG-55] Yanglom, A.M.: *"The Correlation Theory of Processes whose nth Difference Constitute a Stationary Process"*. Matem. Sb. Nº 37. Pp. 141-196. 1955.

[YEE-92] Yee, P.: *"Clasification Experiments involving Backpropagation and Radial Basis Function Networks"*. Communications Research Laboratory. Report nº 249. McMaster university.1992.

[YULE-21] Yule, G.U.: *"On the Time-Correlation Problems with Special Reference to the Variate Difference Correlation Method"*. Journal of Royal Statistical Society. Nº 84. Pp. 497-526. 1921.

[YULE-26] Yule, G.U.: *"Why Do We Sometimes Get Nonsense Correlations Between Time Series?: A Study in Sampling and the Nature of Time Series"*. Journal of Royal Statistical Society. Nº 80. Pp. 1-64. 1926.

[YULE-27] Yule, G.U.: *"On a Method for Investigating Periodicities in Disturbed Series with Special Reference to Wolfer's Sunspot Numbers"*. Philosophical Transactions of the Royal Society. London. A, 226. 1927.

[ZAKN-91] Zaknich A., deSilva, C.J.S. & Attikiouzel Y.: *"A Modified Probabilistic Neural Network (PNN) for Nonlinear Time Series Analysis"*. Proceedings of the International Joint Conference on Neural Networks. Pp. 1530-1535. 1991.

[ZIPS-90] Zipser, D., Rumelhart, D.E.: *"The Neurobiological Significance of the New Learning Models"*. Computational Neuroscience. MIT Press. Ed E.L. Schwartz. Pp. 192-200. 1990.

Anexo 1 sobre Fundamentos

1. Anexo Sobre Fundamentos

1.1 Redes de Neuronas Artificiales

1.1.1 Neurofisiología Elemental y Modelización de la Neurona Natural

La tecnología de las RR.NN.AA. surge de la observación de la estructura, organización y funcionamiento del cerebro y, más concretamente, de ver cómo, a partir de un conjunto elevado de elementos biestables muy simples, las neuronas, se pueden conseguir comportamientos complejos e, incluso, comportamientos "inteligentes". En este punto, se va a hacer un repaso de la anatomía, fisiología, funcionamiento y proceso de aprendizaje de la neurona natural y de cómo se ha modelado, en los inicios del campo de las RR.NN.AA., de una forma simple su comportamiento y la interacción entre ellas.

1.1.1.1 Fisiología de una Neurona Individual

La Figura 1.1 representa los componentes principales de una célula nerviosa típica perteneciente al sistema nervioso central. Consta de tres partes bien definidas: el cuerpo de la célula o soma, que es la zona integradora de la información que llega a la neurona, el axón que se origina en al zona cónica del soma y termina en forma de diversas ramificaciones siendo la zona transmisora o emisora de la información y el árbol dendrítico que también tiene su origen en el soma, está muy ramificado y es la zona receptora de información [CAJA-09].

Lo más interesante de la funcionalidad de las neuronas es su capacidad para recibir, almacenar, procesar y transmitir una enorme cantidad de mensajes simultáneamente, sin que se confundan entre sí, en un breve tiempo y muy limitado espacio. Son de los pocos tipos de células que están especialmente diseñadas para ser excitables e incluso poder transmitir señales a lo largo de ellas. Estas características vienen dadas por: su capacidad para generar un flujo de corriente unidireccional e intracelular; la capacidad de modificar la velocidad de conducción intracelular e

intercelular; la existencia de un alto grado de aislamiento de la membrana que impida el lenguaje cruzado erróneo entre las células nerviosas; la existencia de un significativo número de mensajeros intercelulares (de una célula a otra vecina) diferentes y especializados, denominados neurotransmisores para que sólo el receptor adecuado traduzca la información apropiada, y el desarrollo de dispositivos químicos especiales destinados a garantizar no sólo una transmisión rápida y precisa, sino también una igualmente rápida y precisa desaparición de la señal.

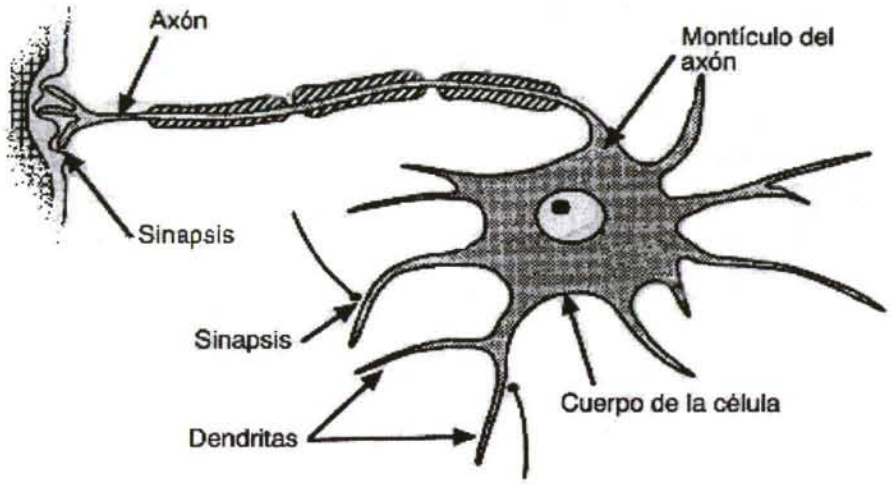


Figura 1.1.- Anatomía de la Neurona Natural

Este comportamiento viene dado, sobre todo, por la composición físico-química tanto de la neurona como de su entorno. Los cambios de la composición están regulados por la membrana de la neurona que separa el plasma intracelular del fluido extracelular que, como se verá a continuación, mantienen una composición diferente. La membrana es permeable para ciertas especies iónicas, y actúa de tal forma que se mantenga una diferencia de potencial entre el fluido intracelular y el extracelular.

En la composición del líquido extracelular destacan las concentraciones elevadas de Sodio (Na^+) y de Cloruro (Cl^-); mientras que, en el intracelular, destacan la concentración de Potasio (K^+) y Fosfatos. Estas diferencias de concentración se logran con mecanismos de transporte a través de unos canales de la membrana celular que es una bicapa lipídica atravesada por moléculas de proteínas. La mayor parte del transporte es muy selectivo respecto a las moléculas o iones que pueden atravesar la membrana.

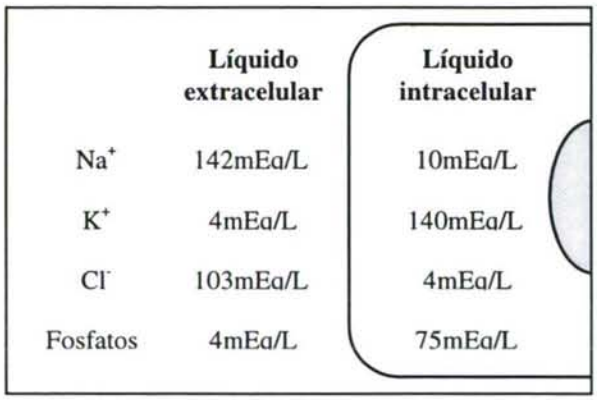


Figura 1.2.- Distribución de iones dentro y fuera de la membrana celular

Los mecanismos básicos de transporte son dos: la difusión y el transporte activo de sustancias.

La **difusión** es el paso al azar de sustancias a través de los canales de la membrana. Puede ser *simple*, de forma que las sustancias hidrosolubles suficientemente pequeñas y con suficiente energía cinética pasen por los intersticios de la bicapa lípida o *facilitada*, en la que la sustancia se liga a ciertas proteínas de la membrana alterando su configuración y permitiendo su paso.

La velocidad a la que se produce la difusión a través de la membrana de la célula se denomina "magnitud neta de la difusión" y se ve afectada por: la permeabilidad de la membrana, la diferencia de concentración de la sustancia a difundir entre ambos lados de la membrana, la diferencia de presión a través de la membrana y, en el caso de iones, la diferencia de potencial eléctrico (se difundirán mejor hacia donde haya mayor carga contraria a la suya, aún en contra de diferencia de concentración). La sustancia que más y mejor difunde la membrana celular es el agua, pues pasa aproximadamente 100 veces el volumen celular en cada segundo, en ambas direcciones por ósmosis (dependiendo de un mecanismo de concentración).

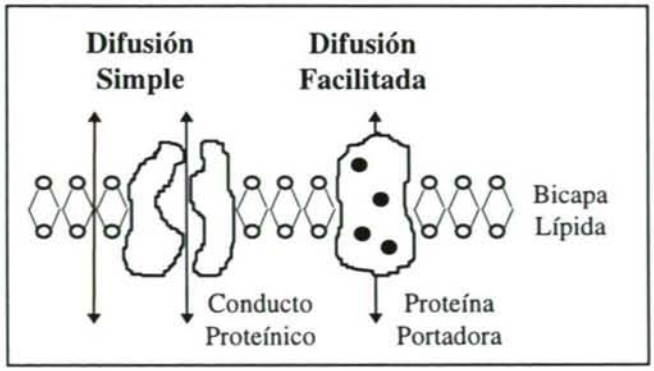


Figura 1.3.- Esquema de difusión simple y facilitada a través de la membrana celular

El **transporte activo** depende de las proteínas transportadoras que facilitan la energía necesaria, (a partir del ATP - adenosin trifosfato). Se usa para concentrar o eliminar sustancias del medio intracelular, aún a pesar de ir contra gradientes electroquímicos, razón por la cual consume energía. Algunas de las sustancias que se transportan por este medio son: sodio, potasio, calcio, cloruro, hierro, urato, azúcares y aminoácidos.

La "bomba de sodio-potasio" es el ejemplo más representativo del transporte activo. Consigue unos valores de sodio externo de 142 meq/l (miliequivalentes/litro) e interno de 14, potasio externo de 4 e interno de 40 meq/l. Esta "bomba" es un complejo de dos proteínas globulares, una mayor que la otra y que tienen, entre otras, la función básica de controlar el volumen celular. Cuando la célula aumenta de volumen saca tres sodios a cambio de meter dos potasios, creando así una presión osmótica hacia fuera de la célula y, por tanto, un potencial eléctrico negativo en el interior.

Todos los iones se pueden difundir a través de la membrana, con la excepción de los iones orgánicos, que son demasiado grandes. Dado que los iones orgánicos no pueden salir de la célula por difusión, su carga negativa neta dificulta la entrada en la célula de iones cloro por este mecanismo; por tanto, habrá una concentración más alta de iones cloro fuera de la célula. La bomba de sodio-potasio determina una concentración más alta de potasio dentro de la célula y una concentración más alta de sodio fuera de ella.

La membrana celular es selectivamente más permeable para los iones de potasio que para los iones de sodio. El gradiente químico del potasio tiende a hacer que los iones de potasio salgan de la célula por difusión, pero la fuerte atracción de los iones orgánicos negativos tiende a mantener dentro el potasio. El resultado de estas fuerzas opuestas es que se alcanza un equilibrio en el cual hay más iones de sodio y cloro fuera de la célula, y más iones orgánicos y de potasio dentro de ella. El equilibrio resultante produce una diferencia de potencial a través de la membrana de la célula de unos 70 a 100 mv (milivoltios), siendo más negativo el fluido intracelular. Este potencial se denomina potencial de reposo de la célula.

Las influencias de las entradas excitatorias que llegan a la célula desde otras neuronas se suman en el montículo axonal y producen una despolarización sucesiva de

la membrana. Se invierte entonces el potencial hasta +35 mv en sólo 0'1 milisegundos. La despolarización resultante en el montículo del axón altera la permeabilidad de la membrana celular a efectos de los iones de sodio. Como resultado hay un flujo entrante de iones sodio positivos, que penetran en la célula, contribuyendo aún más a la despolarización. Este efecto autogenerado da lugar al potencial de acción (ver Figura 1.4). Después de un potencial de acción, el potencial de la membrana se vuelve más negativo que el potencial de reposo durante unos milisegundos hasta que se restablece el equilibrio iónico. Para iniciar el potencial de acción se requiere un incremento súbito de aproximadamente 15-30 mv, considerándose -65 mv el umbral de estimulación. Este umbral no es fijo. Aumenta, e incluso no llega a producirse el potencial de acción, si el incremento de potencial de membrana ocurriese lentamente, fenómeno que se conoce como "acomodación". En una fibra excitable no puede producirse un segundo potencial de acción mientras la membrana se encuentre despolarizada. Al período durante el cual no puede desencadenarse otro potencial de acción, incluso con un estímulo muy poderoso, se llama "período refractario absoluto" y dura aproximadamente 0'5 milisegundos. A continuación, le sigue un "período refractario relativo" que dura entre 1 y 1'5 milisegundos, donde se precisa un estímulo mayor que lo normal para producir el potencial de acción, al no haber recuperado todavía el equilibrio iónico del potencial de reposo. Ese punto no puede volver a ser excitado en, aproximadamente, 1 milisegundo, que es el tiempo que tarda en volver a su potencial de reposo. Este período refractario limita la frecuencia de transmisión de los impulsos nerviosos a unos 1.000 por segundo.

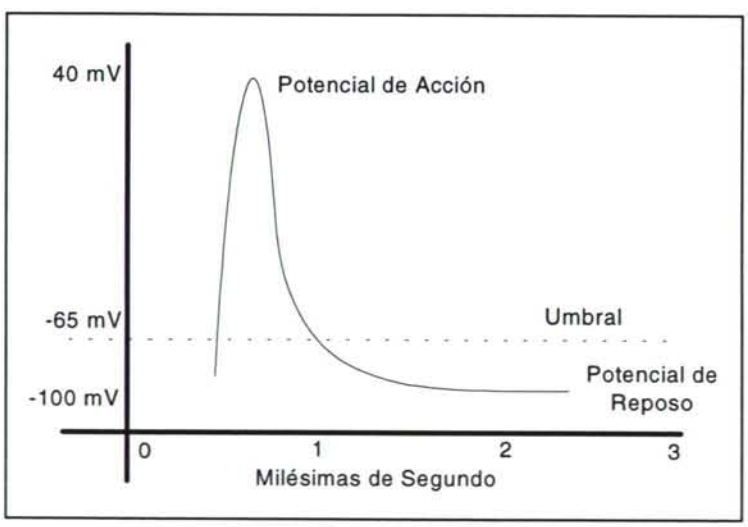


Figura 1.4.- Función de Activación de la Neurona

Ya sólo queda transmitir el impulso nervioso a lo largo del axón para hacerlo llegar a las siguientes neuronas.

1.1.1.2 La Unión Sináptica

Una vez expuesto cómo se produce la activación de una neurona y el transporte del impulso eléctrico a lo largo del axón, se comenta en este apartado el paso del impulso de una neurona a la siguiente. Este paso se produce en el espacio existente entre las neuronas, que se denomina unión sináptica o sinapsis. La comunicación tiene lugar como resultado de la liberación de unas sustancias llamadas neurotransmisores por parte de la célula presináptica y la absorción de estas sustancias por la célula postsináptica. La Figura 1.5 muestra un esquema de esta actividad. Cuando el potencial de acción llega a la membrana presináptica, los cambios de permeabilidad de la membrana dan lugar a un flujo entrante de iones de calcio. Estos iones dan lugar a que las vesículas que contienen los neurotransmisores se fundan con la membrana sináptica, liberando así sus neurotransmisores en la separación sináptica.

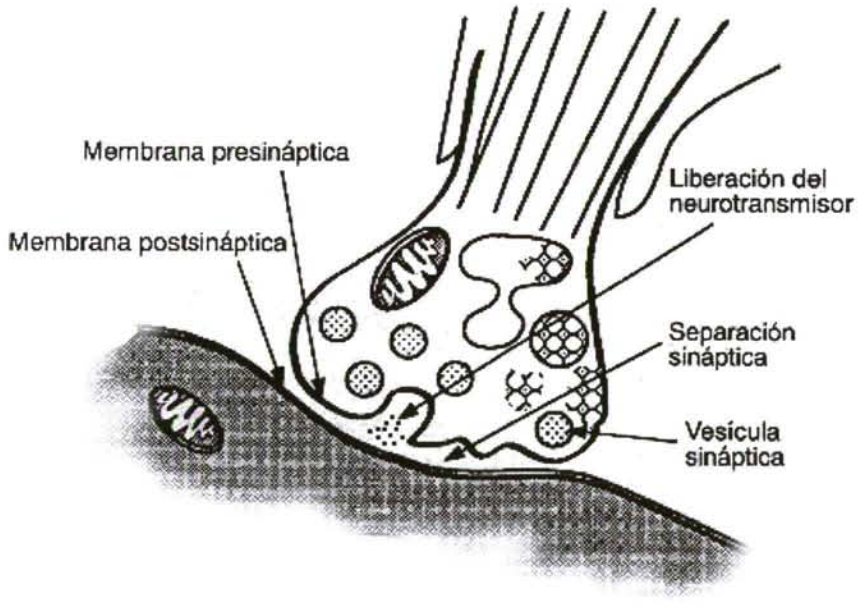


Figura 1.5.- Unión sináptica

Los neurotransmisores se difunden a través de la unión y se unen a la membrana postsináptica en ciertos lugares llamados receptores. La acción química que se produce en los receptores da lugar a cambios de permeabilidad de la membrana postsináptica para ciertas especies iónicas. Un flujo entrante de especies positivas hacia la célula tenderá a despolarizar el potencial de reposo; este efecto es excitatorio. Si entran iones negativos, se producirá un efecto hiperpolarizante; este efecto es inhibitorio. Estos dos efectos son locales, y actúan tan sólo a lo largo de una pequeña distancia hacia el interior de la célula; integrándose con otros estímulos en su camino hacia el montículo

del axón. Si la suma en el montículo axonal es mayor que un cierto valor umbral se genera un potencial de acción propio de esa neurona.

El montículo axonal producirá un estímulo o un "tren de estímulos" todos iguales como respuesta a la amplitud de la estimulación que se genera en el montículo axonal como consecuencia de la integración de todos los estímulos simultáneos que recibe esa neurona en sus receptores. La codificación de la información pasa de ser en amplitud a ser en frecuencia.

1.1.2 Funciones de Activación

1. Lineal: no se realiza ningún tipo de potencial sobre el $neta_i$ salvo una ponderación con un valor constante, η, por lo que, la salida no está limitada y puede tomar cualquier valor real. Véase la parte superior izquierda de la Figura 1.6.

2. Paso: es una función con dos niveles de salida distintos. Un nivel bajo antes de un determinado valor y un valor alto después. Un caso concreto de la función paso es la función escalón donde los valores de salida son 0 y 1 y el cambio de valor se produce en la coordenada 0. Esta función se puede encontrar en la parte superior derecha de la Figura 1.6. Otro caso concreto es la función signo, igual que la anterior pero en la que los valores de salida van entre –1 y 1.

3. Sigmoide: donde T es un parámetro que controla la forma de la función. Si T aumenta la función sigmoide tiende a la función escalón. El rango de salida de esta función es [0,1]. Este tipo de función es muy utilizada debido a que el rango de salida es continuo, acotado y además es derivable lo cual es imprescindible para poder utilizar después un método de descenso de gradiente en el entrenamiento. La representación gráfica de esta función se puede contemplar en la parte inferior izquierda de la Figura 1.6. Una variación también muy utilizada es la tangente hiperbólica que tiene la misma forma pero su rango de salida es [-1,1]. En este caso si el parámetro T aumenta la función tiende a la función signo.

4. Rampa: esta función se suele utilizar como aproximación o simplificación de la función sigmoidal por ser más fácil de calcular y porque conserva las características de derivabilidad. La representación gráfica de esta función se puede contemplar en la parte inferior derecha de la Figura 1.6. Si los valores límite de la salida se

configuran con –1 y 1 la función rampa pasa a ser una aproximación de la función hiperbólica tangente.

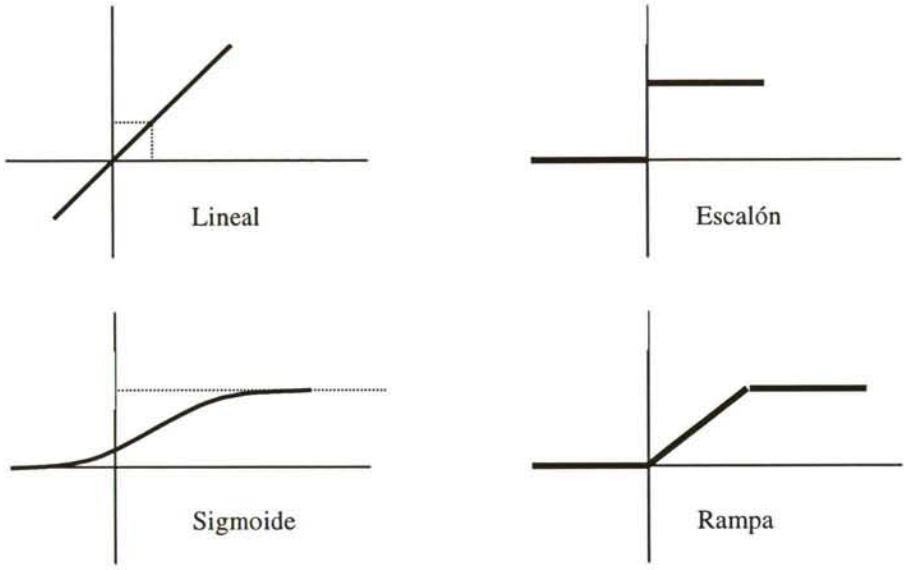


Figura 1.6.- Representación gráfica de las funciones de activación

1.1.3 Tipos de Entrenamiento

1.1.3.1 Entrenamiento no Supervisado

Existen diferentes grupos de algoritmos de entrenamiento no supervisado aunque todos ellos siguen unos principios comunes. Los siguientes son los más comunes:

Algoritmos basados en la regla de Hebb: La base de este tipo de algoritmos ya se ha comentado en un punto anterior de este mismo capítulo [HEBB-49]. La idea fundamental es que si la activación de un EP provoca a su vez la activación de otro EP posterior, es normal suponer que existe una relación entre ambos EP y que, por tanto, su conexión se debe mantener o reforzar. De una forma numérica, el incremento de w_{ij} es función de los valores de activación de las neuronas i y j, respectivamente x_i y x_j. La ecuación básica es:

$$w_{ij}(t+1) = w_{ij} + \alpha\, x_i(t)\, x_j(t)$$

Donde α es la constante que determina la velocidad de aprendizaje. Dependiendo del rango de salida de los EP de la red varía la forma de combinar las activaciones de las neuronas.

Algoritmos basados en Hebb de Diferencias: Se basa en los mismos principios que el anterior pero en vez de trabajar con valores instantáneos, utiliza las variaciones o tendencias de las salidas de las neuronas. La ecuación básica es:

$$w_{ij}(t+1) = w_{ij} + \alpha\, \Delta x_i(t)\, \Delta x_j(t)$$

Donde $\Delta x_i(t) = x_i(t) - x_i(t-1)$ es la representación del incremento del valor de las activaciones del EP i.

Aprendizaje Competitivo: Las ideas sobre este tipo de aprendizaje parten de los trabajos sobre autoorganización en RNA de [FUKU-75][WILL-76]. Se basa en reforzar aquel vector de pesos que mejor se asocie con la entrada actual. En este tipo de aprendizaje, el EP que presente un mayor valor de salida para una cierta configuración de entradas actualiza los pesos de sus conexiones para que se obtenga un valor de salida. La modificación de los pesos se realiza de acuerdo a la siguiente ecuación:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(t)\, x_i(t) \left(x_j(t) - w_{ij}(t)\right)$$

Aprendizaje de Kohonen: Es muy similar al aprendizaje competitivo y parte de la misma base pero, en este caso, además de modificar los pesos del EP con mayor valor de salida, también se modifican los pesos de sus vecinos [KOHO-82] [KOHO-88]. Debido a esto, es necesario definir como un parámetro el ámbito de la vecindad de los EP. Para ello existen varias opciones, se puede definir una vecindad de 2 o más dimensiones e incluso se puede definir la vecindad como un parámetro dinámico.

La ecuación que modifica los pesos es una pequeña variación de la del punto anterior. En vez de utilizar una constante α para la velocidad de aprendizaje, se dispone de un conjunto α_{ij} que pertenecen al intervalo [0,1] y decrecen con el tiempo según 1/t. Otra condición que tienen que cumplir las α_{ij} es que tienen que decrecer según se alejen del EP ganador i. La ecuación para la modificación de los pesos queda como:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha_{ij}(t)\, x_i(t) \left(x_j(t) - w_{ij}(t)\right)$$

1.1.3.2 Entrenamiento Supervisado

La característica distintiva de este tipo de entrenamiento es que el algoritmo utiliza la salida correcta que debería producir la RNA y la producida realmente para

corregir el valor de las conexiones. Este tipo de algoritmos son los más utilizados y se basan en la definición del error cometido por la red. La función del algoritmo es minimizar la magnitud del error utilizando métodos clásicos de minimización como el descenso de gradiente en donde el ajuste de cada peso es proporcional a su contribución al error total, E, del sistema. Así el incremento de un peso viene dado por la función:

$$\Delta w_{ij} \cong \frac{-\partial E}{\partial w_{ij}}$$

Estos son los principales métodos:

Regla Delta: también llamada LMS (Least Mean Square) o de Widrow-Hoff [WIDR-60], se basa, como se ha comentado, en minimizar el error que, en este caso, viene definido como la distancia euclídea entre el vector de la salida deseada de la red A y el vector de la salida obtenida X:

$$E = \|A - X\|^2 = \frac{1}{2}\sum_i (a_i - x_i)^2$$

Es un método de minimización basado en el descenso de gradiente, que modifica los pesos basándose en la derivada del Error con respecto al valor de los pesos:

$$\frac{\partial E}{\partial w_{ij}} = (a_i - x_i)x_j$$

Donde x_j es una entrada y x_i es una salida de la red y a_i es la salida correcta que debería obtener el EP_i. La fórmula para el cambio de los pesos es:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha \ \big(x_i(t) - a_i(t)\big)x_j(t)$$

Como se puede ver en la fórmula anterior, todos los EP's tienen una salida conocida. Esto quiere decir que sólo existe una capa en la RNA. La consecuencia de esto es que este tipo de redes sólo pueden aprender funciones linealmente separables [MINS-69]. Para poder resolver problemas que no sean linealmente separables es necesario poder implementar más capas ocultas en la RNA. Se hace por tanto necesario desarrollar un nuevo algoritmo de aprendizaje. La solución al entrenamiento de este tipo de redes surgió de una variación de este algoritmo.

Regla Delta Generalizada: también conocido como algoritmo de Retropropagación del Error (Backpropagation) [WERB-74][PARK-85][MCCL-86]. Es una generalización del método anterior para resolver el problema del ajuste de los pesos en las capas ocultas, incluso utilizando funciones de activación no lineales. Para resolver el problema de definir el error que comenten los EP de las capas ocultas se utilizó la regla de la cadena para calcular el gradiente del error. El error de la red se define igual que en el caso anterior:

$$E = \|A - X\|^2$$

El valor de los pesos en el siguiente paso de entrenamiento se calcula según la fórmula:

$$w_{ij}^{mn}(t+1) = w_{ij}^{mn}(t) + \alpha \lambda_i^m(t) x_j^n(t)$$

Donde n es de donde parte la conexión, m es la capa de destino de la conexión y λ_i^m es la contribución al error final del EP_i^m. La contribución de los EP al error global se calcula de forma distinta si:

- El EP está en la capa de salida: en este caso se conocen los valores de activación deseados para esos EP's, entonces:

$$\lambda_i^S(t) = \left(a_i(t) - x_i^S(t)\right) f'\left(net_i^S(t)\right)$$

Donde S se refiere a la última capa de la RNA, $net_i^S(t)$ es la suma de las activaciones ponderadas que llegan al Ep_i y $f'(\ )$ es la derivada de la función de activación, que normalmente es una función sigmoidal.

- El EP está en una capa intermedia: λ se calcula retropropagando los valores del error de la capa de salida hacia la capa en la que se quieren calcular según la siguiente ecuación:

$$\lambda_i^m(t) = f'\left(net_i^m(t)\right) \sum_n \sum_{j=0}^{|n|} w_{ji}^{nm}(t) \lambda_j^n(t) \quad \forall m \in \{1,\ldots,S-1\}, n \in \{m+1\ldots,S\}$$

Donde $|n|$ representa el número de EP de la capa n (posterior a m). El error que comete el EP_j de una capa oculta es una combinación lineal de los errores cometidos en

la capa de salida y en las capas anteriores. La aportación al error de ese EP depende de lo que haya influido en la generación de cada salida.

1.1.3.3 Entrenamiento con Refuerzo

En este tipo de algoritmos la red obtiene una ayuda exterior que guía en el proceso de entrenamiento. Esta ayuda viene dada por un valor de refuerzo aplicado como una entrada de la red con la que se le indica el acierto o el fallo de sus salidas. El valor de refuerzo puede venir dado por un proceso externo a la red, puede venir dado por variables del entorno o lo puede calcular la red a partir del comportamiento de algunas de sus entradas. En todo caso el valor de refuerzo no da, en ningún caso, indicación sobre cuál debería ser la salida de la red sino que valora de una forma conjunta la actuación de la RNA. Así, en los algoritmos basados en refuerzo se puede entender que, un valor positivo del refuerzo refleja una recompensa, mientras que un valor negativo es el equivalente de un castigo.

Lo que se le indica a la red es si sus resultados son más o menos buenos con respecto al problema que hay que resolver pero sin indicar qué habría que cambiar para hacerlo mejor. Además de este tipo de incertidumbre, existe un problema adicional en el entrenamiento que es el de la asignación temporal del refuerzo. Es decir, si se distribuye el refuerzo ante cada salida de la red o si se acumulan los valores de refuerzo y se aplican después de un cierto número de ciclos para evaluar las tendencias. En este tipo de entrenamiento los pesos de conexión de los EP´s se ajustan en función de los valores de refuerzo obtenidos por la RNA. La dificultad estriba en obtener de un valor único de refuerzo los incrementos de los pesos para cada EP individual.

Aprendizaje por búsqueda asociativa: es el algoritmo más simple de esta categoría [TORR-91]. Su ecuación es la del aprendizaje de Hebb pero ponderado con el valor del refuerzo. La expresión más simple de este tipo de aprendizaje es:

$$w_{ij}(t+1) = w_{ij} + \alpha\ x_i(t)\ x_j(t)\ r(t)$$

Donde r(t) es el valor del refuerzo en el instante t. Esta regla de aprendizaje tiene el problema de que sólo puede implementar funciones linealmente separables, lo cual limita su aplicación a problemas relativamente simples.

Existen algoritmos más complejos, como el de Diferencias Temporales [SUTT-88] en los que su mayor logro es la implementación de refuerzos asíncronos que se aplican a la red a intervalos irregulares de tiempo para reorientar el entrenamiento.

1.1.4 Modelos Básicos de RR.NN.AA

1.1.4.1 Perceptrón

Para plasmar los estudios teóricos de Rosenblatt se desarrolló el "fotoperceptrón" que es un dispositivo que responde a señales ópticas. En la Figura 1.7 se muestra un diagrama genérico. En este dispositivo, la luz incide en los puntos sensibles (S) de la estructura de la retina. Cada punto S responde en forma todo-nada a la luz entrante. Los impulsos generados por los puntos S se transmiten a las unidades de asociación (A) de la capa de asociación. Cada unidad A está conectada a un conjunto aleatorio de puntos S, denominados conjunto fuente de la unidad A, y las conexiones pueden ser tanto excitatorias como inhibitorias. Las conexiones tienen los valores posibles +1, -1 y 0. Cuando aparece una trama de estímulos en la retina, una unidad A se activa si la suma de sus entradas sobrepasa algún valor umbral. Si está activada, la unidad A produce una salida, que se envía a la siguiente capa de unidades.

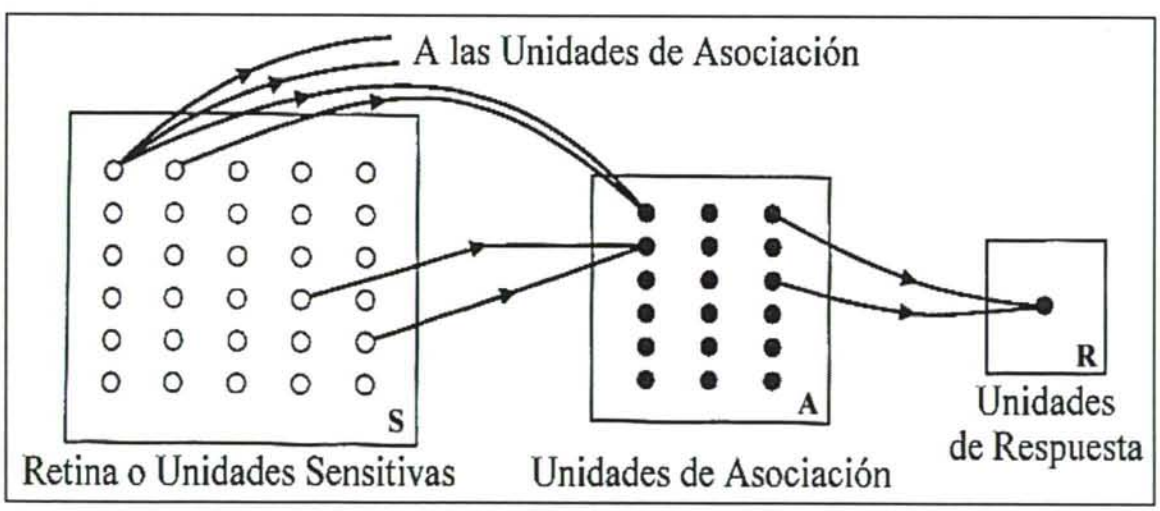


Figura 1.7.- Estructura del Fotoperceptrón.

De forma similar, las unidades A están conectadas a unidades de respuesta (R) dentro de la capa de respuesta. La trama de conectividad vuelve a ser aleatoria entre capas, pero se añaden conexiones inhibitorias de realimentación procedentes de la capa de respuesta, que llegan a la capa de asociación. También hay conexiones inhibitorias

entre las unidades R. Todo el esquema de conexiones está descrito en forma de diagrama en la Figura 1.8 para un perceptrón sencillo con dos unidades R.

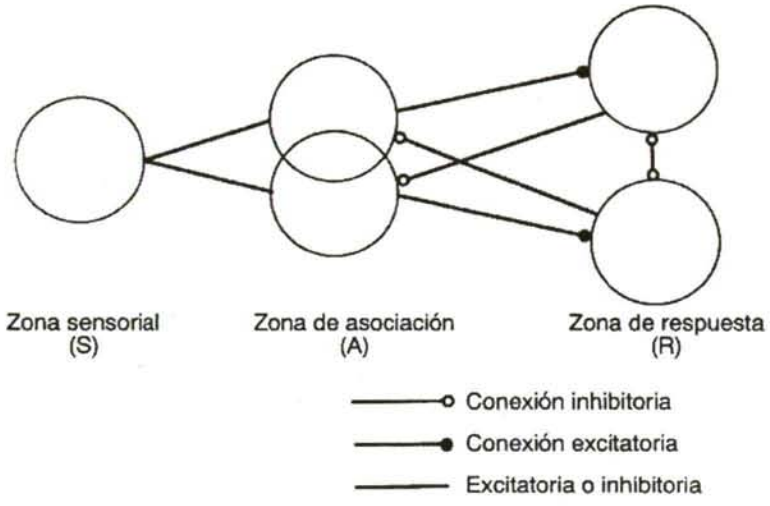


Figura 1.8.- Esquema de las conexiones del Fotoperceptrón.

Este dibujo muestra que cada una de las unidades R inhibe a las unidades A en el complemento de su propio conjunto fuente. Además, cada una de las unidades R inhibe a la otra. Estos factores ayudan a establecer que una sola unidad R sea la ganadora para cada trama de estímulos que aparezca en la retina. Las unidades R responden de una forma muy similar a la de las unidades A. Si la suma de sus entradas sobrepasa un umbral, producen un valor de salida de +1; en caso contrario, su salida vale -1. Un mecanismo alternativo de realimentación conectaría conexiones excitatorias de realimentación procedentes de cada unidad R a los respectivos conjuntos fuentes de esa unidad R dentro de la capa de asociación.

Se puede utilizar un sistema como el que se acaba de describir para clasificar tramas que aparezcan en la retina por categorías, de acuerdo con el número de unidades de respuesta que haya en el sistema. Las tramas que sean suficientemente parecidas deberían excitar a la misma unidad R. Por tanto, se trata de un problema de separabilidad.

El perceptrón era un dispositivo que no estaba preparado para realizar su labor hasta después de superar un proceso de aprendizaje. En su configuración inicial sin entrenar, el perceptrón no era capaz de distinguir entre grupos de tramas; sin embargo, mediante el proceso de aprendizaje, era capaz de adquirir esta capacidad. En esencia, el entrenamiento implicaba un proceso de refuerzo mediante el cual la salida de las

unidades A se incrementaba o se decrementaba dependiendo de si las unidades A contribuían o no a las respuestas correctas del perceptrón para una trama dada. El proceso de aprendizaje era el siguiente: se aplicaba una trama a la retina y el estímulo se propagaba a través de las capas hasta que se activase una unidad de respuesta. Si se había activado la unidad de respuesta correcta, se incrementaba la salida de las unidades A que hubieran contribuido. Si se activaba una unidad R incorrecta, se hacía disminuir la salida de las unidades A que hubiesen contribuido.

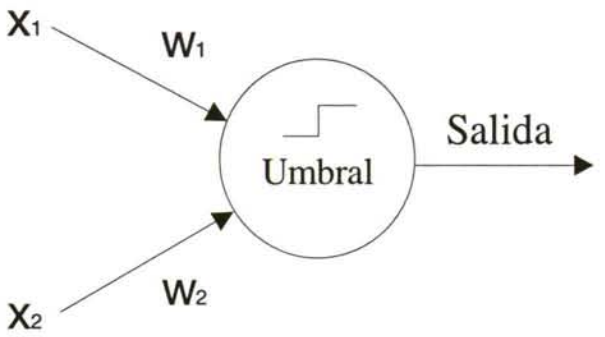
Utilizando un sistema de estas características, Rosenblatt pudo demostrar que el perceptrón era capaz de clasificar tramas correctamente, en lo que él denominaba un entorno diferenciado, en el cual cada clase estaba formada por tramas que eran similares unas a otras en algún sentido. El perceptrón también era capaz de responder de manera congruente frente a tramas aleatorias, pero su precisión iba disminuyendo a medida que aumentaba el número de tramas que intentaba aprender.

Los trabajos de Rosenblatt dieron lugar a que se demostrase un importante resultado conocido con el nombre de "Teorema de convergencia del perceptrón". El teorema se demuestra para un perceptrón con una unidad R que está aprendiendo a diferenciar tramas de dos clases diferentes. Afirma, en esencia, que si la clasificación puede ser aprendida por el perceptrón, entonces el procedimiento que se ha descrito garantiza que será aprendida en un número finito de ciclos de entrenamiento.

1.1.4.2 Restricciones en el funcionamiento del Perceptrón

Minsky y Papert se apartaban de la aproximación probabilística que propugnaba Rosenblatt, y volvían a las ideas de cálculo de predicados en su análisis del perceptrón.

El conjunto $\Theta = \{\varphi_1, \varphi_2, ..., \varphi_n\}$ es un conjunto de predicados de forma que $\varphi_i = 1$ cuando el i-ésimo punto de la retina está activado, y $\varphi_i = 0$ en caso contrario. Cada uno de los predicados de entrada viene ponderado mediante un número del conjunto $\{a_{\varphi 1}, a_{\varphi 2}, ..., a_{\varphi n}\}$. La salida, Ψ, es 1 si $\sum_n a_{\varphi n}\varphi_n > \Theta$, en donde Θ es el valor umbral. Uno de los ejemplos más sencillos de problemas que no se pueden resolver mediante un perceptrón es el problema XOR. Una red como la de la Figura 1.9 no es capaz de resolver este problema.



x_1	x_2	Salida
0	0	0
0	1	1
1	0	1
1	1	0

Figura 1.9.- Red para tratar la función XOR.

En ella la función de salida de la unidad de salida es una función umbral.

$$f(neta) = \begin{cases} 1 \text{ si neta} \geq \Theta \\ 0 \text{ si neta} < \Theta \end{cases}$$

donde Θ es el valor umbral. Este tipo de nodo se denomina unidad de umbral lineal. La activación del nodo de salida es

$$neta = w_1 x_1 + w_2 x_2$$

y el valor de salida es

$$o = f(neta) = \begin{cases} 1 \text{ si } w_1 x_1 + w_2 x_2 \geq \Theta \\ 0 \text{ si } w_1 x_1 + w_2 x_2 < \Theta \end{cases}$$

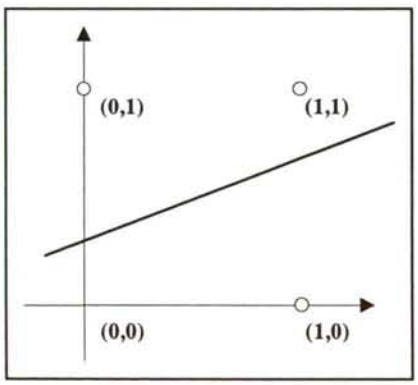


Figura 1.10.- Comparación de la separación de la red y los cuatro puntos del problema XOR.

Para resolver el problema hay que seleccionar unos valores de pesos tales que, todos los pares de valores de entrada den lugar a los valores de salida correctos. Realmente no existen los valores para hacer esto. Esto se debe a que la ecuación:

$$\Theta = w_1 x_1 + w_2 x_2$$

es la ecuación de una línea en el plano x_1, x_2 y es también la función de activación de la red. Este plano se ilustra en la Figura 1.10, junto con los cuatro puntos que son las posibles entradas de la red. El problema consiste en subdividir este espacio en regiones y asignar después rótulos a estas regiones de tal modo que correspondan a la respuesta correcta para los puntos de esa región. La línea parte el plano en dos regiones distintas como máximo. Entonces se pueden clasificar los puntos de una región como pertenecientes a la clase que posee una salida de 1, y los de la otra región como pertenecientes a la clase que posee una salida nula; sin embargo, no hay ninguna forma de disponer la situación de la línea de tal forma que los dos puntos correctos para cada clase se encuentren en la misma región. Como se puede ver la sencilla unidad de umbral lineal no es capaz de llevar a cabo correctamente la función XOR.

Antes de demostrar la forma de resolver esta dificultad, se va a presentar el concepto de hiperplanos. Esta idea aparece ocasionalmente en la literatura, y puede ser útil para evaluar el rendimiento de ciertas redes neuronales.

En el espacio tridimensional que nos es familiar, un plano es un objeto de dos dimensiones. Un único plano puede descomponer el espacio tridimensional en dos regiones distintas; dos planos pueden dar lugar a tres o cuatro regiones distintas, dependiendo de sus orientaciones relativas, y así sucesivamente. Por extensión, en un espacio n-dimensional, los hiperplanos son objetos de (n-1) dimensiones (los espacios n-dimensionales suelen recibir el nombre de hiperespacio). Una colocación adecuada de los hiperplanos permite descomponer los espacios n-dimensionales en varias regiones diferentes.

Hay muchos problemas reales que implican la separación de regiones de puntos de un hiperespacio en categorías individuales o clases, que deben distinguirse de otras clases. Una forma de hacer estas distinciones consiste en seleccionar hiperplanos que descompongan el espacio en regiones adecuadas. Esta tarea podría parecer difícil de realizar en un espacio de búsqueda con muchas dimensiones (esto es, más de las dos dimensiones habituales), y lo es. Afortunadamente, como se verá más adelante, ciertas redes neuronales pueden aprender la descomposición adecuada, así que no es preciso determinarla por anticipado. En un espacio general n-dimensional, la ecuación de un hiperplano se puede escribir en la forma

$$\sum_{i=1}^{n} a_i x_i = C$$

donde los a_i son constantes con, al menos, un $a_i \neq 0$, y los x_i son las coordenadas del espacio.

Este concepto de hiperplano se puede aplicar al problema XOR. El grafo de la Figura 1.11 sugiere que se podría descomponer correctamente el espacio si se tuvieran tres regiones. Una región abarcaría una de las clases de salida, y las otras dos abarcarían a la segunda clase. No hay razón por la cual regiones disjuntas no puedan pertenecer a una misma clase. La Figura 1.11 muestra una red de unidades de umbral lineal que lleva a cabo la descomposición correcta, junto con el diagrama correspondiente de hiperplanos.

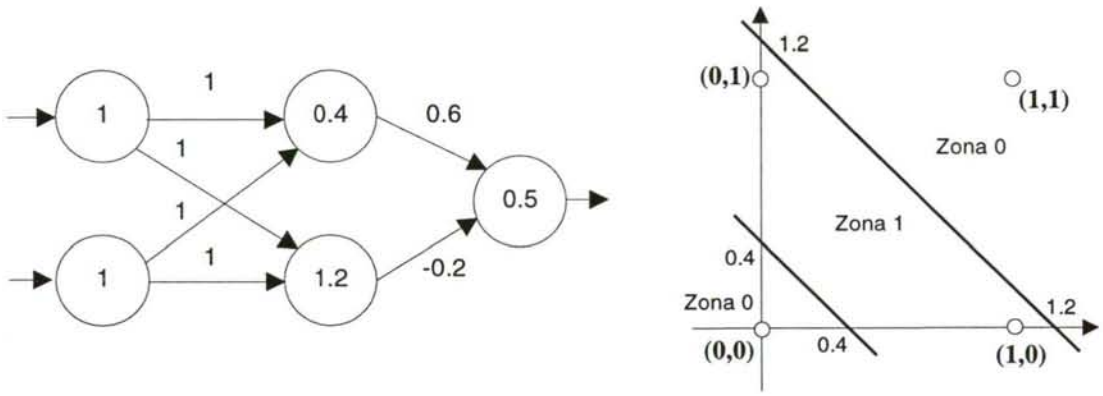


Figura 1.11.- Modelo de red que resuelve el problema de la XOR y gráfica de la separación lineal.

La adición de dos unidades de capa oculta, o capa intermedia, dan a la red la flexibilidad necesaria para resolver el problema. De hecho, la existencia de capas ocultas es lo que da la capacidad de construir redes que puedan resolver problemas complejos.

1.1.5 RR.NN.AA. para Procesado Temporal

1.1.5.1 RR.NN.AA. Basadas en Retardos

1.1.5.1.1 Time Delay Neural Network – TDNN

La TDNN es una red multicapa de alimentación hacia delante en la que las capas ocultas y la de salida se replican por cada instante de tiempo. Para realizar el

entrenamiento se desarrolla la red a lo largo del tiempo hasta conseguir una RNA con retropropagación tradicional. Esta topología TDNN se basa en un preceptrón multicapa en el que cada sinapsis está implementada por un filtro de impulso de duración finita (finite-duration impulse response filter - FIR) [WAN-93]. Esta red se conoce como "Perceptrón Multicapa FIR" y para su entrenamiento se construye una versión estática equivalente desdoblando el perceptrón FIR en el tiempo y usando después un algoritmo de aprendizaje de retropropagación temporal.

Aunque este sistema realiza su función de predicción en problemas temporales, sufre un conjunto de problemas, siendo el más importante la pérdida de la simetría entre la propagación hacia delante de los estados de activación y la propagación hacia atrás de los términos necesarios para calcular el gradiente del error debido al desdoblamiento de la red en el tiempo.

En este modelo no existe ni recurrencia en la arquitectura de la red, ni una fórmula recursiva que permita la propagación de los términos de error. Fue la primera opción de que se dispuso ya que supone el camino más fácil hacia el procesamiento temporal. Debido a los problemas ya mencionados pronto se descartó en favor del algoritmo de aprendizaje Temporal Back-Propagation.

1.1.5.1.2 Algoritmo Temporal Back Propagation

Surge como solución a los problemas asociados con la aproximación del cálculo del gradiente [WAN-90a][WAN-90b] vista en el punto anterior para los perceptrones FIR. En este algoritmo, se considera la siguiente expresión como la derivada de la función de error ε_{TOTAL} con respecto al vector de pesos $w_{ji}(t)$:

$$\frac{\partial \varepsilon_{TOTAL}}{\partial w_{ij}(t)} = \sum_{t} \frac{\partial \varepsilon_{TOTAL}}{\partial v_j(t)} \frac{\partial v_j(t)}{\partial w_{ij}(t)} \qquad \text{Ecuación 1-1}$$

Al disponer de una función del gradiente del error total, ya se puede utilizar la idea tradicional del gradiente para modificar el vector de pesos $w_{ji}(t)$ de la forma:

$$w_{ji}(n+1) = w_{ji}(n) - \eta \frac{\partial \varepsilon_{TOTAL}}{\partial v_j(t)} \frac{\partial v_j(t)}{\partial w_{ij}(t)} \qquad \text{Ecuación 1-2}$$

Las fórmulas de cálculo de la variación de los pesos de este algoritmo representan una generalización vectorial con respecto al algoritmo de retropropagación estándar.

Este algoritmo de aprendizaje aporta ventajas de tipo práctico con respecto al anterior. Por un lado, se mantiene la simetría entre la propagación de los estados hacia delante y la propagación hacia atrás de los gradientes del error. Por otro lado, cada peso de las conexiones se usa una única vez en el cálculo de los gradientes. Ya que no se desdobla la red en el tiempo, no hace falta repetir las conexiones para realizar los cálculos en el tiempo.

Una limitación que se puede encontrar en el algoritmo de entrenamiento que se acaba de presentar es que los retardos de las conexiones son fijos. Pronto surgieron generalizaciones que permitían la existencia de retardos variables en las sinapsis que se podían adaptar mediante entrenamiento [DAY-93][LIN-92][LIN-93]. Este nuevo tipo de neurona consiste en el encadenamiento de dos tipos de unidades funcionales:

Un retardo temporal en la sinapsis, denotado como $\tau_{ji}(t)$ y el vector de pesos en el tiempo visto hasta ahora y que se denotaba $w_{ji}(t)$. Con este tipo de arquitectura, estos dos parámetros varían sus valores de forma continua en "t".

La idea a destacar de esta implementación es que aporta un nuevo tipo de red más flexible y eficiente ya que es capaz de almacenar más información acerca del flujo temporal de datos.

1.1.5.2 RR.NN.AA.RR.

1.1.5.2.1 Backpropagation Through Time - BTT

Como se puede ver en la Figura 1.12, para una RNAR de dos neuronas con interconexión total, se genera una red alimentada hacia delante desde t=0 hasta t=n duplicando estas dos neuronas para cada instante de tiempo t. Al ser todas las neuronas de la red desdoblada, réplicas de las neuronas originales, no sólo en la capa de salida existe salida deseada, como es habitual en las redes alimentadas hacia delante, sino que las neuronas de cada capa tienen también sus propias salidas deseadas en un instante t.

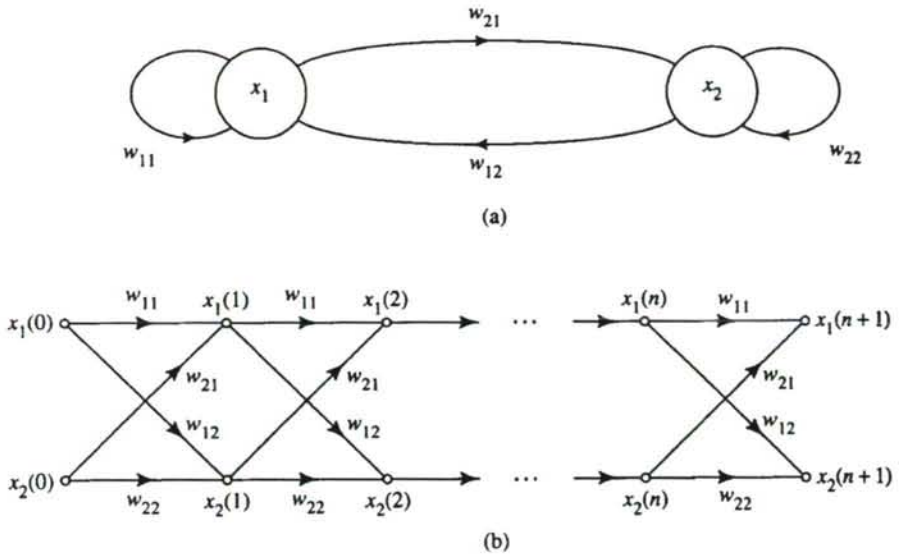


Figura 1.12.- a) RNA Recurrente con dos EP totalmente conectados
b) Desdoblamiento de la RNA para aplicar el aprendizaje BTT

Los cálculos de este algoritmo de entrenamiento aplican el algoritmo de retropropagación estándar a esta red multicapa alimentada hacia delante.

Aunque esta primera aproximación a las redes recurrentes se basa en el desdoblamiento hacia una red estándar de retropropagación, ya utiliza los bloques necesarios para realizar un procesado temporal más natural al incluir conexiones de retropropagación. Sin embargo, la propia naturaleza del proceso de entrenamiento, que desdobla la red en el tiempo, lo hace desaconsejable para la operación de la red en tiempo real. Los mismos autores describen una variante de este algoritmo que intenta modificar el aprendizaje para que se pueda realizar el empleo continuo de la red recurrente [WILL-90].

1.1.5.2.2 Real-Time Recurrent Learning – RTRL

En el caso de estas redes al poder realizar conexiones entre todas las neuronas de la red, desaparece el concepto de capas. Ahora sólo existen un conjunto de neuronas interconectadas de las cuales unas cuantas aceptarán entradas del exterior y otras ofrecerán las salidas al exterior, pudiendo incluso coincidir alguna de ellas. Se va a trabajar con una red de N neuronas con M entradas. Esto quiere decir que x(n) será un vector de Mx1 valores que se aplica a la entrada de la red en el instante n. Mientras que, y(n+1) será un vector de Nx1 valores de salida generados en el instante siguiente, n+1.

El vector de entrada x(n) y la salida retrasada un instante de tiempo y(n) se concatenan para formar el vector u(n) que será de magnitud (M+N)x1.

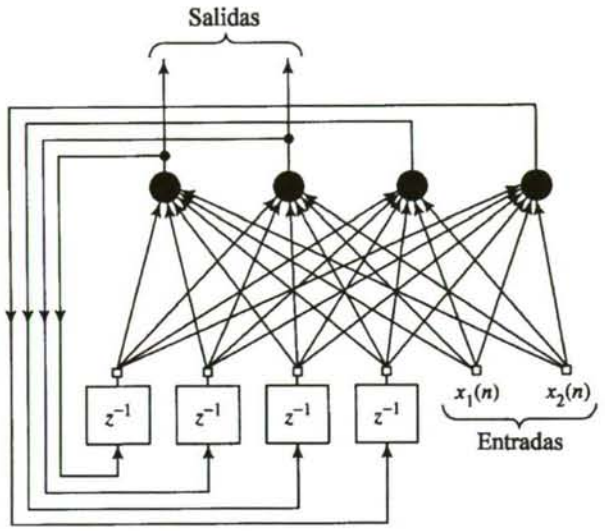


Figura 1.13.- Arquitectura de una Red RTRL

Se va a definir el conjunto A como el conjunto de neuronas para las cuales $u_i(n)$ es una entrada externa y el conjunto B como el conjunto de neuronas para los cuales $u_i(n)$ es la realimentación de la salida de una neurona. Este se puede plasmar como:

$$u_i(n)=\begin{cases} x_i(n) & \text{si } i\in A \\ y_i(n) & \text{si } i\in B \end{cases}$$

Se pueden distinguir dos capas distintas en la red, una capa con las entradas-salidas concatenadas y la capa de proceso. La red está totalmente interconectada entre estas dos capas, lo que quiere decir que existen NxM conexiones hacia delante, mientras que hay N^2 conexiones recurrentes. La matriz de pesos W estará compuesta entonces por Nx(N+M) pesos.

Una vez vista la estructura de la red, el nivel de activación para una neurona j, tal que $j\in B$, será:

$$v_j(n)=\sum_{i\in A\cup B} w_{ji}(n)u_i(n) \qquad \text{Ecuación 1-3}$$

En el siguiente instante la salida de la red se calcula como:

$$y_j(n+1)=\varphi(v_j(n)) \qquad \text{Ecuación 1-4}$$

Hay que tener en cuenta que para el proceso de entrenamiento el vector de entradas externo x(n) no tiene influencia en la salida de la red hasta el instante n+1. Los pasos para aplicar el algoritmo de entrenamiento serían los siguientes:

1. Para cada instante "n", empezando en n=0, se usan las ecuaciones del funcionamiento de las neuronas (Ecuación 1-3 y Ecuación 1-4) para computar las salidas de cada una de ellas. Para los valores iniciales de los pesos se escoge un conjunto de números uniformemente distribuido generado aleatoriamente.

2. Una vez calculadas las salidas se calculan las variables $\pi_{kl}^{j}(n)$ para todos los j, k y l necesarios.

3. Se usan los $\pi_{kl}^{j}(n)$ obtenidos en el paso anterior y el error de salida $e_j(n)$ para calcular la diferencia de los pesos:

$$\Delta w_{kl}(n) = \eta \sum_{j \in C} e_j(n) \pi_{kl}^{j}(n)$$

4. Y, por último, se actualizan los pesos incrementandos con las diferencias que se acaban de calcular:

$$w_{kl}(n+1) = w_{kl}(n) + \Delta w_{kl}(n)$$

5. Para repetir de nuevo todos los pasos.

Como se puede comprobar por la evolución de este capítulo, el algoritmo de aprendizaje Real-Time Recurrent Learning es el que mejor se adapta al tratamiento de problemas temporales, debido a la utilización de las conexiones recurrentes entre neuronas y debido a que es un tipo de aprendizaje que se realiza en tiempo real. Esta ventaja plantea el inconveniente del almacenamiento de la información que, en forma de $\pi_{kl}^{j}(n)$, es necesario almacenar para cada instante de tiempo. En el caso de una red con N neuronas y M entradas externas existirán $N(N^2+NM)$ variables $\pi_{kl}^{j}(n)$ para cada instante "n", siendo "j" y "k" índices asociados a las neuronas de salida y "l" un índice sobre todas las neuronas de la red. Obviamente, para redes con muchas neuronas, el número de $\pi_{kl}^{j}(n)$ crecerá rápidamente, siempre teniendo en cuenta que no sólo hay que calcular sus valores sino que, después, es necesario almacenar estas variables para poder calcular los incrementos de los pesos en el instante siguiente.

1.1.5.2.3 RR.NN.AA. Parcialmente Recurrentes

Este tipo de RR.NN.AA. fue propuesto originalmente por Robinson y Fallside [ROBI-91], aunque esta estructura puede ser vista como una versión reducida de la red descrita por Elman [ELMA-90] eliminando las neuronas ocultas.

La estructura de una red de este tipo es la que se muestra en la Figura 1.14. La red consiste en una capa de entrada que recoge los datos de entrada y los de realimentación y una capa de salida que contiene las neuronas de proceso. Esta última capa está dividida en dos partes, un conjunto K de neuronas de salida y un conjunto L de neuronas de contexto. La capa de contexto genera un vector r(n) que retardado un instante de tiempo se concatena a los valores de las entradas del vector x(n) para conseguir el vector de entrada u(n). Este vector de entrada produce el vector de salida y(n+1) y un nuevo vector de realimentación r(n+1) mediante una función de activación φ, de forma que:

$$y(n+1)=\varphi(v_o(n))$$
$$r(n+1)=\varphi(v_c(n))$$

Por último, el vector de salida y(n+1) se compara con el vector de respuestas deseadas d(n+1) de acuerdo con una función de coste que, en la descripción original de [ROBI-91], era una función basada en la entropía relativa.

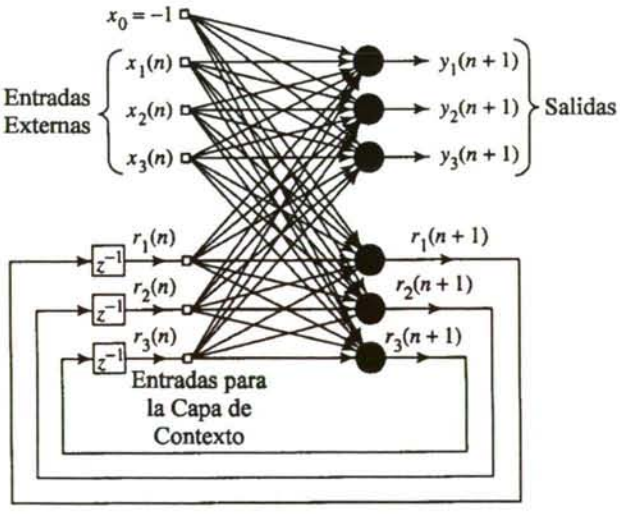


Figura 1.14.- Arquitectura de un RNA parcialmente recurrente.

1.1.6 Modelos de RR.NN.AA. Incrementales

1.1.6.1 Algoritmos Incrementales de Separación Lineal

Como ya se ha visto, el objetivo de un sistema clasificador PLS es determinar combinaciones de funciones lineales que construyan discriminantes para resolver un problema dado. Cada una de las funciones lineales estará asociada a las neuronas generadas por el algoritmo incremental. Si se tiene en cuenta que estas unidades son de tipo perceptrón, cada una de ellas supondrá un discriminante lineal de dimensión n-1.

Debido a esto, en los algoritmos PLS se suele comenzar a entrenar la red con una única neurona. Si el problema de clasificación que se está tratando es linealmente separable, esta única neurona encontrará la solución al problema y el algoritmo PLS no generará nuevas unidades. En caso contrario, la solución que ofrece esta neurona se supone que es una primera aproximación y el algoritmo continuará generando nuevas neuronas hasta que la salida obtenida mediante la combinación de los separadores lineales de las neuronas sea satisfactoria.

Para ver el funcionamiento de este tipo de algoritmos, se va a comentar un ejemplo de problema de clasificación que se soluciona mediante el algoritmo "Neural Trees" [SIRA-90] (Ver Figura 1.15). Se parte de una estructura de red muy simple con una única neurona a la que se denomina unidad 0 y se entrena mediante la "regla delta" [ROSE-62] ya comentada o mediante el "algoritmo Pocket" [GALL-86]. Este algoritmo de entrenamiento consiste en la ejecución continuada de la "regla delta" sobre el conjunto de vectores de entrada, pero manteniendo almacenado como solución provisional el vector de pesos que es capaz de clasificar correctamente el mayor número de vectores de entrada.

La neurona generará como salida los valores 0 ó 1 para definir las dos categorías de salida. Como el ejemplo que se presenta no es linealmente separable, es preciso que el algoritmo incremental genere nuevas unidades para poder encontrar la solución del problema. En el caso del algoritmo "Neural Trees", a partir de cada unidad se generan dos nuevas unidades, la unidad 1 y la unidad 2. Cada una de estas unidades se entrena con una parte del conjunto de entrenamiento que divide la clasificación de la unidad 0. Las unidades 1 y 2 estarán conectadas a la unidad 0 y una se entrenará con los patrones

que provocan salida 0 en la unidad 0 y la otra con los que provocan salida 1 en la unidad 0. Como todavía, con las unidades 1 y 2 no existe una solución lineal, el algoritmo generará 2 nuevas unidades para cada una de las unidades 1 y 2. Este proceso termina cuando todos los subproblemas generados por medio de la inclusión de nuevas unidades tienen una solución lineal.

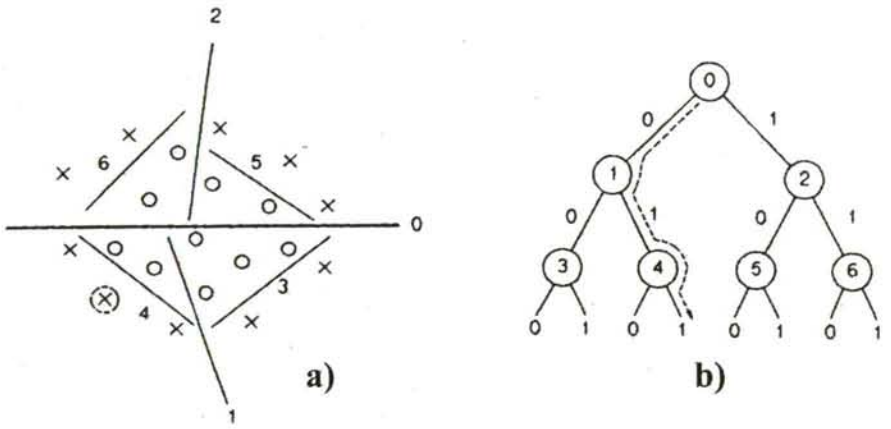


Figura 1.15.- a) Clasificación del espacio de soluciones conseguida a partir de una Red Entrenada mediante el algoritmo "Neural Trees" b) Estructura final de la red

Todas las unidades van a dividir el espacio de entradas en dos partes identificadas como salida 0 y salida 1. La estructura final de la red tiene la apariencia de un árbol de decisión en el cual una vez aplicada una entrada para clasificar, se irá propagando su clasificación por las ramas del árbol, lo que significa que irá cayendo en clasificaciones parciales de zona 0 ó 1 hasta llegar a una de las neuronas de la capa de salida. Hay que tener en cuenta que en este tipo de red los enlaces entre las neuronas o unidades no representan pesos sino meras conexiones que sirven para dividir el espacio de búsqueda.

La aplicación de este tipo de algoritmos puede presentar algún problema, en el caso de que la unidad del primer nivel falle al clasificar y provoque que todos los patrones coincidan en la misma clase. En el caso de darse este problema, el árbol de clasificación contendría una única rama que clasificaría en cada etapa todos los patrones en la misma clase. Para evitar este problema, que se da utilizando los dos métodos de entrenamiento, regla delta y pocket, se han propuesto una serie de soluciones [MORE-93a][MORE-93b][MORE-94] que se orientan a comprobar que, en cada paso del algoritmo, el número de vectores clasificados debe ser mayor que el que producía el vector de pesos anterior. Hay que tener en cuenta que el campo de los algoritmos

incrementales está actualmente en constante evolución y las mejoras y avances en sus algoritmos son constantes.

Los algoritmos "Incrementales de Separación Lineal" se pueden clasificar siguiendo dos criterios: la forma en que evoluciona la red hacia la solución y el tipo de arquitectura final generada. Desde el punto de vista de la arquitectura existen dos tipos básicos:

- Los que, como el algoritmo "Neural Trees", hacen evolucionar la estructura de red mediante sucesivas divisiones lineales del espacio de búsqueda, consiguiendo en cada división una mayor precisión en la resolución del problema. A este grupo pertenecen el algoritmo de "Tiling" [MEZA-89], el algoritmo de "Marchand" [MARC-90], el algoritmo "PSIN" [SUN-88], el algoritmo de "Knerr" [KNER-91] y el algoritmo "CID3" [CIOS-92], entre otros.

- Y aquellos algoritmos en los que las nuevas unidades añadidas a la red tienen como misión la corrección de los errores de clasificación que cometen las unidades que actualmente contiene la red. Pertenecen a esta categoría, entre otros, el algoritmo "Upstart" [FREA-90], el algoritmo de "Sato" [SATO-91], el algoritmo "PSOHNN" [ERSO-90] o el algoritmo "Cascade Correlation" [FAHL-90].

Con respecto a la topología de red generada, también se pueden distinguir dos grupos:

- Los que generan una estructura de red del tipo árbol de decisión, como los algortimos "Upstart" y "Neural Trees".

- Los que generan una estructura de red del tipo "perceptrón multicapa", como por ejemplo los algoritmos "Cascade Correlation", "Tiling", "CID3", etc.

1.1.6.2 Redes de Región de Influencia

En este tipo de clasificadores no se calculan discriminantes que separen regiones del espacio de soluciones sino que se almacenan patrones alrededor de los cuales se definen las regiones que se corresponden con las clases. La formulación para un clasificador como este, también denominado de vecino más próximo, es la siguiente:

Si $d(X,X^p) < d(X,X^m)$, $\forall m \neq p$ asociar X a la clase representada por θ^p

Donde X^i representa un patrón i almacenado por el clasificador, θ^p es la etiqueta que indica la pertenencia de X^p a una clase y $d(X,X^n)$ es la distancia entre el patrón X y el X^n. En este tipo de red, la clase que tiene asociado el patrón más cercano al que se le presenta en la entrada, es a la que se le adjudica este patrón. Por tanto, los cálculos que ha de realizar la red son las distancias entre el patrón de entrada y cada una de las muestras almacenadas. Matemáticamente, este tipo de red consigue sus objetivos, en labores de clasificación, debido a que funciona como una regla de decisión aleatoria que clasifica los patrones de entrada, seleccionando la clase con una mayor probabilidad "a posteriori" de contenerlos [DUDA-73].

1.1.6.2.1 Algoritmo Reduced Coulomb Energy (RCE)

En el algoritmo RCE [REIL-82] se va a asociar a cada patrón un radio que va a ser el parámetro que define el tamaño de la región asociada a ese patrón. El aprendizaje se efectúa presentando patrones de forma secuencial a la red y calculando si los nuevos patrones caen dentro de la región de influencia de algún patrón ya almacenado. Si es así, y el nuevo patrón pertenece a esa clase, no se realizan cambios sobre el clasificador. Si no cae en la región de ningún patrón almacenado, entonces el nuevo patrón se almacena en el clasificador como una nueva clase y se le asigna un radio de influencia. Por último, si existe un conflicto con una clase ya creada; es decir, el nuevo patrón cae dentro de su radio pero no pertenece a esa clase, se modifica el radio de esa clase de forma que el nuevo patrón quede fuera de esa clase. Entonces al nuevo patrón se le asigna un radio inicial más pequeño.

Otro algoritmo ROI, variante de éste último, se denomina "Grow and Learn" (GAL) [ALPA-90]. También es un clasificador de vecino más próximo pero modifica la forma de entrenamiento. En este algoritmo, sólo los patrones presentados a la entrada que estén mal clasificados se almacenan como nuevos patrones del clasificador. Si un patrón está bien clasificado se desecha y se pasa al siguiente patrón. De esta forma, se construye el clasificador sin necesidad de almacenar todos los patrones del conjunto de entrenamiento.

Estos dos algoritmos son capaces de producir redes muy aceptables sólo con presentar los patrones de entrada una única vez, aunque es conveniente realizar la presentación varias veces para que los resultados no dependan del orden de presentación. También puede ser necesario utilizar algún criterio para eliminar patrones poco representativos [ALPA-90] con lo que se ganaría capacidad de generalización en la red.

1.1.6.2.2 Redes de Función de Base Radial (RBF)

Los fundamentos de este tipo de redes se basan en los modelos clásicos de la teoría de decisión y clasificación [DUDA-73]. Trabajos recientes [LEE-91][PLAT-91][MUSA-92] modificando los algoritmos de agrupación (clustering) de patrones, utilizados tradicionalmente para el entrenamiento de las redes RBF, consiguen redes con una estructura más adecuada para la solución de cada problema.

Este tipo de redes están compuestas de dos capas. La primera capa se compone de unidades cuyos valores de salida dependen de la proximidad del vector de entrada al vector que representa esa unidad (por eso el nombre de radiales). El valor de salida de estas unidades se obtiene del cálculo de la distancia por medio de una función como puede ser la gausiana normalizada:

$$o_i = e^{-\left(\frac{(x-w_i)^T(x-w_i)}{2\sigma_i^2}\right)} \qquad i=1..N$$

Donde, o_i es la salida de la unidad i del primer nivel, x es el vector de entrada, w_i es el vector que define a la unidad i, σ_i es el parámetro de normalización para la unidad i y N es el número de unidades en el primer nivel. Existen otros tipos de funciones [CARL-92] que pueden se utilizadas en RBF para las unidades del primer nivel. Las unidades de salida realizan únicamente una combinación lineal de las salidas obtenidas por las unidades del primer nivel.

La fase de entrenamiento de este tipo de redes tiene dos partes bien diferenciadas debido a los diferentes tipos de neuronas de las dos capas de la red. El entrenamiento de las neuronas del primer nivel se lleva a cabo utilizando algoritmos de agrupación. Mediante un proceso de agrupamiento por similitud de distancia se calculan los vectores

que definen a las unidades del primer nivel. Los algoritmos más habituales para realizar este proceso son el de las k-medias [DUDA-73] y el ISODATA [BALL-66].

El principal problema que se presenta en estos algoritmos (K-medias e ISODATA) es que el número de agrupaciones que representan las unidades de la capa de entrada es un parámetro que debe fijarse "a priori", y no es calculado de forma automática por la red. Además, el valor que se escoja para este parámetro determina el correcto funcionamiento de la red como sistema clasificador. Por esto, se han desarrollado estos algoritmos incrementales que son capaces de calcular, por sí mismos, el número de clases necesarias para cada problema.

En el caso del algoritmo de Lee [LEE-91], el proceso de entrenamiento no sólo se encarga de añadir nuevas unidades sino que también modifica las funciones radiales. Al principio comienza con un número reducido de unidades que abarcan una amplia área de influencia (al tener un valor elevado del parámetro de normalización σ) para, posteriormente, ir añadiendo unidades con menor área de influencia, con el fin de obtener una aproximación progresivamente más precisa.

Por último, el entrenamiento de las conexiones entre las unidades de salida y las de la primera capa se suele realizar mediante algoritmos tradicionales de descenso de gradiente, como el algoritmo LMS [WIDR-60].

1.1.6.3 Redes Probabilísticas

Son RR.NN.AA. que estiman directamente la probabilidad "a posteriori" o que se basan en aproximar regiones o discriminantes a partir de esa probabilidad. La desventaja más grande de este tipo de redes es que, al operar directamente sobre muestras, pueden llegar a un sobre-entrenamiento si se llega a aprender todas muestras, incluso las que no son representativas. Todas estas técnicas se basan en la división del espacio de búsqueda en varias regiones y tratan de estimar la densidad de probabilidad en cada una de ellas.

Lo que se necesita obtener son estimaciones de probabilidad en regiones pequeñas. Esto implica que, a medida que se toman muestras, se aminoran los volúmenes iniciales de las regiones, de forma que todo el espacio de entrada quede

cubierto por regiones pequeñas, con lo cual, dentro de cada una, se obtiene una buena estimación de probabilidad.

Para estimar la probabilidad de cada zona o volumen se suele, bien utilizar una función "gaussiana" o la función "epanechnikov" [EPAN-69]. Si el tamaño de todas las regiones es el mismo, se está empleando el algoritmo de "Parzen" [PARZ-62][SPEC-90]. Si el tamaño de la región puede cambiar, se dice que son adaptativas. Estos métodos son incrementales ya que almacenan los patrones en la red y no hay ningún problema en añadir nuevos patrones ya que con la nueva información modificarán el clasificador ya construido.

1.2 AA.GG.

1.2.1 Genética

La mayoría de las plantas y los animales crean su descendencia mediante reproducción sexual, en la cual el núcleo de la célula espermatozoide se une a la célula óvulo. Ambos núcleos contienen, cada uno, partes complementarias de material genético organizadas en unas estructuras denominadas cromosomas. Cuando el espermatozoide se une al óvulo, se produce una célula denominada zigoto que contiene dos copias de cada cromosoma dentro del citoplasma del óvulo. La mitad complementaria de cromosomas se denomina conjunto haploide (abreviado como n) y el conjunto de pares complementarios de cromosomas se denomina conjunto diploide (2n). Los gametos (células sexuales) son haploides mientras que la mayoría del resto de células son diploides. Esto supone que en el proceso de formación de los gametos (gametogénesis) el número de cromosomas en la célula sea dividido por dos. La gametogénesis se produce mediante una clase especial de división celular llamada "meiosis". Los mecanismos de este tipo complejo de división aseguran que los gametos contengan sólo una copia de cada cromosoma. El genotipo es el material genético que cada organismo hereda de sus padres. En un organismo diploide, la mitad del genotipo se hereda de un progenitor y la otra mitad del otro. Las células diploides contienen dos copias de cada cromosoma.

Aunque desde hace mucho tiempo se lleva criando animales domésticos y cultivando plantas, no es hasta mediados del siglo diecinueve que, con los trabajos de

Gregor Mendel (referenciados en [SINN-58]), empiezan los trabajos de la genética moderna. Mendel investigó la herencia de ciertos rasgos en alubias. Por ejemplo, tomó plantas que tenían semillas estriadas y plantas que tenían semillas lisas y las cruzó con plantas del mismo fenotipo, las estriadas entre sí y las lisas entre sí. Siguió este procedimiento un cierto número de generaciones hasta que las plantas con semillas lisas siempre daban semillas lisas y las plantas con semillas estriadas siempre daban semillas estriadas. A estas plantas las llamó plantas de cruce puro. Entonces, cruzó entre sí plantas de las dos características. En la siguiente generación (abreviada como F1) las semillas eran todas lisas. Entonces Mendel cruzó los híbridos F1 entre ellos y se dio cuenta que en la siguiente generación, la F2, tenía tanto semillas lisas como estriadas en la proporción: 3 lisas por 1 estriada.

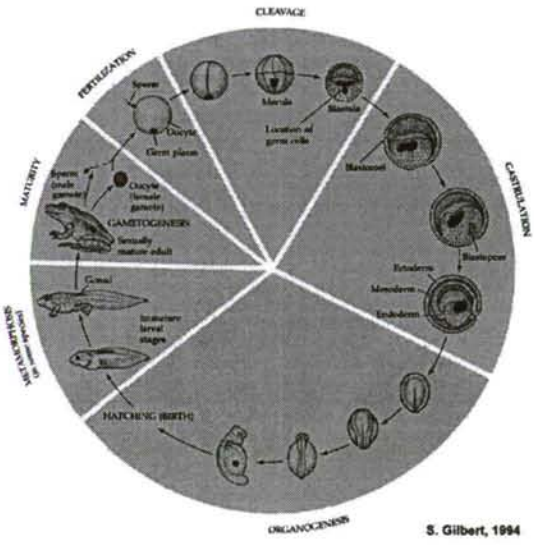


Figura.1.16.- Ciclo de Vida

Mendel repitió este experimento con distintas características de la alubias incluyendo: color de los cotiledones (amarillo o verde), color de las flores (rojo o blanco), color de las semillas (marrón o blanco) y longitud del tallo (largo o corto). En cada caso, encontró que los híbridos de la generación F1 eran todos iguales mientras que las dos características iniciales resurgían en la generación F2. Mendel calificó a la característica que tenía toda la descendencia en F1 como dominante y a la característica que reaparece en la generación F2 como recesiva.

La moderna interpretación de la herencia se basa en el entendimiento de los genes y de cómo éstos se expresan en el fenotipo. La naturaleza de los genes es bastante compleja [ALBE-89][LEWI-90][FUTU-86], existiendo distintas formas de cada gen

"alelos" dependiendo del lugar que ocupen en el cromosoma "locus". La idea de unidades que representan características distinguibles (como los ejemplos de los trabajos de Mendel) se comenzó a reconocer en el campo de la genética a principios de este siglo mediante los trabajos de Bateson, de Vries y Correns (recopilados en [DARD-91]).

Existen, además del cruce, otros mecanismos de cambio, que pueden alterar los genes o cambiar el número de genes presentes en un genoma. Una mutación se puede ver como un cambio en la secuencia del DNA de un individuo. Las mutaciones pueden ser de dos tipos: la mutación puntual, en la cual se modifica una única base y la mutación de desplazamiento de marco, en la que una o más bases (pero nunca múltiplos de tres) se insertan o se borran en el DNA – Ácido Desoxiribonucléico. Estos cambios varían el marco en el que las tripletas de bases son transcritas en el RNA – Ácido Ribonucléico y utilizadas luego para formar las proteínas. Además, algunos genes son capaces de cambiar su posición a cualquier lugar dentro del genoma. Poniendo toda esta información en común, se puede decir que los cambios dentro de los cromosomas pueden ser, por borrado (pérdida de una sección), duplicación (se repite una sección), inversión (la sección se codifica en el orden inverso) y transposición (la sección cambia de posición dentro del genoma). Otro nivel donde se pueden dar cambios es en el del genoma. Hasta ahora se ha visto que por el número de copias de los cromosomas que tiene una célula, estas podían ser "haploides" (una copia) o "diploides" (dos copias), pero también existe la posibilidad de la "trisomía" (tres copias) o de células "tetraploides" (cuatro copias) comunes en algunos cultivos como el trigo y el algodón.

1.2.2 Historia de la CE

Desde mediados de los años 50, existen evidencias del uso de ordenadores para avanzar en la comprensión del proceso de la evolución natural. Uno de los primeros trabajos que utilizan un proceso evolutivo para la resolución de problemas en computadores es Friedberg [FRIE-58][FRIE-59]. Estos artículos representan los primeros trabajos en aprendizaje automático y describen la utilización de un algoritmo evolutivo para programación automática. El objetivo era encontrar un programa que calculara una función de entrada-salida. También el artículo de Fraser [FRAS-57] sirvió de influencia en los primeros trabajos de esta área.

En esta misma época Bremermann presentó los primeros intentos de aplicar evolución simulada a problemas de optimización numérica [BREM-62]. Más adelante, también comenzó el desarrollo de la teoría de algoritmos evolutivos (AE), demostrando que el ratio óptimo de mutación para problemas linealmente separables debería tener el valor "1/m" donde "m" es el número de bits que codifica cada individuo [BREM-65].

También durante este período Box desarrolló las ideas de la operación evolutiva (evolutionary operation EVOP) que utilizaba una técnica evolutiva para el diseño y análisis de experimentos industriales [BOX-57][BOX-69]. Las ideas de Box no fueron nunca implementadas en computador aunque Spendley [SPEN-62] las usó como base para su método denominado "simplex design".

Como suele suceder con el comienzo de cualquier nuevo campo de estudio, estos inicios se veían con un grado elevado de escepticismo. Sin embargo, a mediados de los 60 ya estaban establecidas las bases de lo que hoy son las tres ramas principales de la CE. Así, por un lado están los trabajos de Fogel [FOGE-66] en San Diego, California, en el campo de la Programación Genética. Por otro lado están los Algoritmos Genéticos desarrollados por John Holland [HOLL-69] en Ann Arbor, en la Universidad de Michigan. Por último, en el lado europeo, se desarrollan las Estrategias Evolutivas [RECH-65] por un grupo de tres estudiantes de la Universidad de Berlín, Bienert, Rechenberg y Schwefel.

Durante los siguientes 25 años, cada una de estas ramas se desarrolló de una forma bastante independiente trazando rutas paralelas. En el año 1990 se hizo un esfuerzo por acercar a los distintos investigadores de CE. Para ello, se organizó un *Workshop* internacional con el nombre de "Parallel Problem Solving from Nature" [SCHW-91] en Dortmund, Alemania. Desde este primer contacto, la interacción y cooperación entre los investigadores de las distintas ramas de la CE continuó creciendo. Mediante esta creciente interacción entre las áreas se consiguió llegar, en el año 1991, al nombre actual de este campo, "Computación Evolutiva" y el inicio de la publicación, en el año 1993, de una revista con el mismo nombre por MIT Press. El interés por este campo se puede ver ya consolidado cuando en el año 1994, en la celebración del IEEE World Congress on Computational Intelligence (WCCI) en Orlando, Florida [MICH-94], una de las tres conferencias simultáneas estaba dedicada a la CE, junto con las de RNA y de Lógica Difusa.

1.2.3 Historia de los AA.GG.

Las primeras ideas en las que se basan los AA.GG. se pueden encontrar en los artículos de Holland de principios de los años 60 [HOLL-62]. En ellos, se establece una amplia y ambiciosa agenda para la comprensión y el establecimiento de los principios básicos de los sistemas adaptativos. Estos sistemas son capaces de automodificarse en respuesta a su interacción con el medio en el que están funcionando. Estas teorías sobre los sistemas adaptativos estarían orientadas a facilitar, por un lado, la comprensión de las formas complejas de adaptación que aparecen en los sistemas naturales y, por otro, la habilidad para diseñar sistemas adaptativos robustos.

Según Holland, la característica fundamental de los sistemas adaptativos naturales es el éxito en el uso de la competición y la innovación que les proporciona la habilidad de responder dinámicamente a sucesos nuevos y a cambios en el medio. Para llegar a estas ideas, Holland se basó en modelos simples de evolución biológica, a través de la supervivencia del más adaptado y la producción continua de nueva descendencia.

A mediados de los años 60, las ideas de Holland empezaron a plasmarse en modelos implementados en ordenador a través de varios estudiantes de doctorado que trabajaban con él. Cada uno de estos sistemas ya tenía un "aire evolutivo" ya que los objetos que evolucionaban en estos sistemas se basaban en el uso de genomas y los mecanismos de evolución y herencia eran abstracciones de operadores genéticos como mutación, cruce e inversión.

La tesis de Bagley [BAGL-67] trata del ajuste de los pesos que se usan en las funciones de evaluación de los programas de juegos y es uno de los primeros trabajos experimentales en el uso de representaciones diploides, la inversión y los mecanismos de selección. En otro sentido, la tesis de Rosenberg [ROSN-67] trabaja en la simulación de la evolución de un sistema bioquímico simple en el cual había organismos unicelulares capaces de producir encimas. Estos organismos tenían una representación diploide y evolucionaban para producir concentraciones químicas adecuadas.

La tesis de Cavicchio [CAVI-70] estaba enfocada en tratar estas ideas como una forma de búsqueda adaptativa y probarlas experimentalmente en problemas de búsqueda complicados, como reconocimiento de patrones. En este trabajo, se pueden encontrar los

primeros estudios con selección del tipo elitista e ideas para ratios adaptativos de mutación y cruce. La tesis de Hollstien [HOLS-71] estudia detalladamente, por primera vez, las diferentes posibilidades de selección y cruce. Usando dos espacios de soluciones diferentes, Hollstien experimentó con una amplia variedad de operadores de cruce. También es interesante en estos trabajos la utilización de codificación binaria en el genoma y las primeras observaciones de las virtudes de la codificación Gray.

En paralelo con estos estudios experimentales, Holland continuó trabajando en una teoría general de los sistemas adaptativos [HOLL-67]. Durante este período desarrolló el análisis del esquema en los sistemas adaptativos [HOLL-69] y utilizó estas ideas para completar el análisis teórico de sus planes reproductivos (AG simples) [HOLL-71][HOLL-73]. Después, Holland compiló todas estas ideas en su libro más importante de esta época, "Adaptation in Natural and Artificial Systems" [HOLL-75].

Es interesante destacar que muchas de las propiedades de los AA.GG. identificadas por Holland de forma teórica no pudieron ser observadas experimentalmente debido a la falta de recursos computacionales. Esto obligaba a trabajar con un número limitado de generaciones y con un número muy pequeño de individuos, habitualmente menos de 20. En estos casos, las desviaciones en el comportamiento esperado de los AA.GG. se debía al conocido fenómeno del "genetic drift", que produce la pérdida de diversidad genética debido a los fenómenos estadísticos que se producen en la selección y reproducción en poblaciones pequeñas.

A principios de los 70 ya había un gran interés en entender mejor el comportamiento de las implementaciones de AA.GG. En concreto, estaba bastante claro que la elección del tamaño de la población, la representación de los individuos, la elección de los operadores y sus ratios, tenían efectos relevantes en el funcionamiento de los AA.GG. La tesis de Franz [FRAN-72] refleja estas nuevas preocupaciones estudiando en detalle la influencia de cruce y mutación en poblaciones de 100 individuos. En estos trabajos se empiezan a utilizar los operadores de cruce multipunto.

En la tesis de De Jong [DEJO-75] se amplían estos estudios analizando de forma teórica y práctica los efectos cruzados de la modificación del tamaño de la población, cruce y mutación en el comportamiento de una familia de AA.GG. utilizados para optimizar un conjunto fijo de funciones de test. A partir de estos estudios se empezó a

vislumbrar el enorme potencial de los AA.GG. para resolver problemas de optimización complejos.

A mediados de los años 70, se empieza a generalizar la investigación sobre AA.GG. al iniciarse los trabajos en esta área en otras universidades americanas. Esta generalización fue muy lenta al principio debido a cierto escepticismo creado por la publicidad que habían tenido técnicas de IA como los sistemas auto-organizativos y los perceptrones, que finalmente no cubrieron las espectavivas creadas.

A pesar de estas malas perspectivas, algunos grupos de universidades como Michigan, Pittsburgh y Alberta, organizaron un *Workshop* sobre sistemas adaptativos en el verano de 1976 en Ann Arbor, Michigan. Asistieron alrededor de 20 personas, emplazando una nueva celebración del *Workshop* para el siguiente año. Este evento se repitió sucesivamente durante varios años hasta que en 1979, los organizadores, Holland, De Jong y Sampson consiguieron fondos de la NSF (National Science Foundation) para la celebración del *"An Interdisciplinary Workshop in Adaptive Systems"* que se celebró en la Universidad de Michigan en el verano de 1981 [SAMP-81].

En esta época, ya había varios grupos consolidados trabajando en AA.GG. En la Universidad de Michigan, Bethke, Goldberg y Booker continuaban el desarrollo de los AA.GG. y empezaban a estudiar los sistemas clasificadores de Holland como parte de sus tesis doctorales [BETH-81][BOOK-82][GOLD-83]. En la Universidad de Pittsburgh, Smith y Wetzel estaban trabajando con De Jong en varias mejoras de los AA.GG. incluyendo la regla Pittsburgh de ponderación para aprendizaje de reglas [SMIT-80][WETZ-83]. En la Universidad de Alberta, Brindle trabajaba en las aplicaciones de optimización de los AG [BRIN-81].

Debido al crecimiento continuo del interés en los AA.GG. se organiza el primer *"International Congress on Genetic Algorithms (ICGA)"* en Pittsburgh, Pennsylvania, en 1985. Hubo 75 participantes presentando artículos tanto sobre teoría como sobre aplicaciones prácticas de los AA.GG. [GREF-85a]. El éxito de esta edición del congreso consiguió que se instaurara su celebración cada dos años. A partir de la edición de 1989 de ICGA, las actividades en el campo de los AA.GG. habían crecido suficientemente como para necesitar de una institución que ayudara a la organización y planificación de

eventos y actividades en esta área, por ello se fundó la *"International Society for Genetic Algorithms (ISGA)"*.

En 1990, Schwefel organizó la primera conferencia *"Parallel Problem Solving from Nature"* PPSN-91 [SCHW-91] en Dormund y fue la primera interacción entre investigadores de AA.GG. y EE. Esta relación continuó con la celebración del ICGA'91 en San Diego donde se acordó que los congresos ICGA y PPSN se celebraran en años alternos y se constató la necesidad de iniciar una publicación periódica en esta área. Para garantizar el éxito de esta publicación, se quiso incluir otras técnicas de CE dentro de los temas a cubrir por la revista, como por ejemplo la PG (que ya empezaba a organizar sus propios congresos en 1992). Así, en la primavera 1993, se publicó el primer número de la revista *"Evolutionary Computation"* como aglutinadora de los artículos en esta área.

A partir de 1990, la tendencia ha sido la de un tremendo crecimiento y diversificación de la comunidad investigadora en AA.GG., como se refleja en el éxito de organización de las distintas conferencias y la proliferación de libros y revistas periódicas. La interacción con las otras técnicas de CE se ha traducido en un muy productivo intercambio de ideas y en nuevas aplicaciones híbridas en el entorno de la CE. Desde este punto, las aplicaciones de AA.GG. continúan desarrollándose abarcando un amplio abanico de campos, desde la ingeniería hasta la investigación operativa y la programación automática.

1.2.4 Otras Técnicas de Búsqueda

1.2.4.1 Búsqueda Aleatoria

La forma más directa de maximizar funciones complejas es utilizar una aproximación aleatoria o una búsqueda consecutiva. Los puntos del espacio de soluciones se seleccionan al azar o de alguna manera sistemática y después se evalúan. Esta es la estrategia menos inteligente y que consume más recursos. Obviamente se utiliza en pocas ocasiones.

1.2.4.2 Métodos de Gradiente

Se han desarrollado un gran número de métodos para optimizar funciones continuas usando la información del gradiente de la función para guiar la dirección de la búsqueda. Sin embargo, si la derivada de la función no puede ser calculada, porque, por ejemplo, existen discontinuidades, estos métodos fallan.

Tales métodos son conocidos como de "hill-climbing" o escalado de colinas y funcionan muy bien cuando existe un único máximo. Pero en funciones con varios máximos locales, tienen el problema de que el máximo que encuentren puede no ser el máximo global de la función. Una vez que alcanzan lo alto del máximo local, no se puede conseguir ningún progreso más. Un ejemplo de todo esto se muestra en la figura (figura hill-climbing). El método de escalado comienza desde un punto X escogido aleatoriamente. Se escala en la dirección del pico A y este es localizado. Sin embargo los picos B y C no son encontrados.

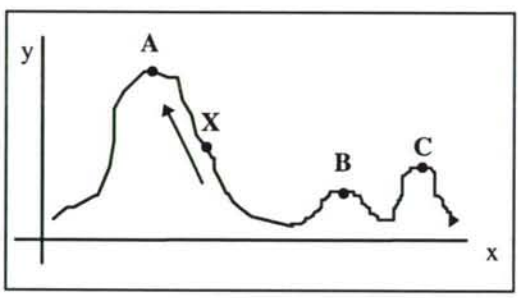


Figura 1.17.- Funcionamiento del Hill-Climbing ante máximos locales

1.2.4.3 Búsqueda Iterativa

La búsqueda aleatoria y la de gradiente se pueden combinar para conseguir la búsqueda de escalado de colinas iterativa. Una vez que se ha localizado un pico, el hill-climbing comienza de nuevo, pero con otro punto de inicio escogido aleatoriamente. Esta técnica tiene la ventaja de la simplicidad y puede funcionar bien si la función no tiene demasiados máximos locales.

Sin embargo, ya que cada intento aleatorio se realiza de una forma aislada, no se consigue una imagen global de la forma del dominio estudiado. Mientras continúa la búsqueda aleatoria, se continúan realizando pruebas en todo el espacio de búsqueda. Esto significa que se evaluará el mismo número de puntos en regiones de baja adaptación como en regiones donde la adaptación es alta.

Un AG, en comparación, empieza con una población inicial aleatoria, y al pasar las generaciones, realiza un mayor número de pruebas en aquellas áreas del espacio de soluciones en las cuales el ajuste es mayor. Esto es una desventaja si el máximo reside en una región pequeña rodeado por todas partes de regiones de ajuste bajo. Sin embargo, esta clase de función es difícil de optimizar usando cualquier tipo de método. En este tipo de problemas tan específicos, la simplicidad de la búsqueda iterativa, será la que funcione de una forma más eficiente [ACKL-87].

1.2.4.4 Enfriamiento Simulado

Esta técnica (en inglés *Simulated Annealing*) fue inventada por Kirkpatrick en 1982 (se explica en profundidad en [RUTE-89]). Esencialmente consiste en una versión modificada del hill-climbing. Empezando desde un punto aleatorio del espacio de soluciones, se realiza un desplazamiento aleatorio. Si el movimiento se realiza a un punto de mayor ajuste, este se acepta. Si se realiza a un punto peor, se acepta sólo con probabilidad p(t) donde t es el tiempo. La función p(t) empieza cerca de 1 para, gradualmente, reducirse hasta el cero, realizando una analogía del enfriamiento. Inicialmente, cualquier movimiento es válido, pero cuando la temperatura se reduce, la probabilidad de aceptar un movimiento a peor se reduce. Los movimientos negativos son esenciales algunas veces si hay que escapar de un máximo local pero obviamente, demasiados movimientos negativos, acaban alejando la búsqueda del máximo.

Como en la búsqueda aleatoria, el Enfriamiento Simulado, sólo trata con un candidato de solución en cada momento y, por tanto, no se hace una idea completa del espacio de soluciones. No se utiliza la información sobre antiguos movimientos para guiar la realización de los nuevos. Esta nueva técnica es todavía objeto de una intensa investigación (re-annealing, parallel annealing), y ha sido utilizada con éxito en muchas aplicaciones, por ejemplo, diseño de circuitos VLSI [RUTE-89].

1.2.5 Operadores de Selección

1.2.5.1 Métodos de Ponderación Explícita de Ajuste

Como ya se ha mencionado, la piscina de cruce tiene el mismo tamaño que la población original para ello, la media del número de copias de cada individuo de la

población en la piscina de cruce debería ser uno. El número de copias de cada individuo viene dado por su ajuste dividido por el ajuste medio de la población. Este esquema de ponderación reparte las posibilidades de cruce de acuerdo al ajuste de cada individuo, y se corresponde con la versión original de las teorías de Holland.

Este método sufre el problema de la convergencia prematura, así que se han desarrollado un conjunto de variantes que intentan paliarlo. Existe una descripción exhaustiva de estas técnicas en [BAKE-85] y las más importantes son:

- Escalado de ajuste (fitness scaling): es un método muy común. En él, el número máximo de posibles copias de un individuo se fija a un determinado valor, típicamente 2. El número máximo se consigue de restar este valor del ajuste y dividir el resultado por la media de ajuste de la población. Restando un valor fijo se incrementa el ratio entre máximo ajuste y ajuste medio. El escalado de ajuste tiende a comprimir el rango de valores de ajuste al comienzo de la simulación, disminuyendo la velocidad de convergencia e incrementando la cantidad de exploración.

Sin embargo, la presencia de sólo un individuo de muy alta adaptación (por ejemplo con un ajuste 10 veces más grande que cualquier otro) puede llevar a una sobre-compresión del rango de valores. Si la escala de ajuste se comprime de tal forma que el ratio máximo/media es 2:1, entonces el resto de la población tendrá un ajuste muy cercano a uno. Así, aunque prevengamos la convergencia prematura, esto se hace a expensas del fattening out de la función de evaluación. Como ya se mencionó, si la función de ajuste es demasiado plana, el genetic drift puede llegar a ser un problema, así que la sobre-compresión puede no sólo disminuir el rendimiento sino también alejarnos del máximo.

- Ventana de ajuste (fitness windowing): básicamente es igual que el escalado de ajuste, excepto que la cantidad que se resta es diferente. El mínimo ajuste de cada generación se graba y la cantidad restada es el mínimo ajuste observado durante las n generaciones anteriores, donde n es típicamente 10. Con este esquema la selección varía durante la simulación. La presencia de un individuo de muy poca adaptación causa una infra-expansión, mientras que un individuo altamente adaptado puede causar convergencia prematura, ya que no influyen en el grado de escala aplicado. Este método se implementa en el sistema de AG GENESIS de Gefenstette [GREF-84].

El problema con estas dos técnicas, escalado de ajuste y ventana de ajuste, es que el grado de compresión lo dicta un solo individuo, el más extremo, el más adaptado o el peor. El rendimiento sufrirá si el individuo extremo es excepcionalmente extremo.

- Ranking de ajuste (fitness ranking): es otro método muy común, al que no le afecta la presencia de individuos extremos. Los individuos se ordenan de acuerdo a su ajuste, y entonces los valores de ajuste reproductivos se asignan de acuerdo a su posición en este ranking. El número de copias o intentos reproductivos de un individuo será siempre el mismo para una misma posición en el ranking, independientemente de su valor de ajuste o el de sus vecinos. Esta asignación puede ser lineal [BAKE-85] o exponencial [DAVI-89]. Esto consigue un resultado similar al escalado de ajuste, en el que el ratio de ajuste máximo/media se normaliza a un valor concreto. Sin embargo, se asegura también que los ajustes ponderados de los individuos intermedios están distribuidos uniformemente. Debido a esto, el efecto de uno o dos individuos extremos no será negativo, independientemente de lo mayor o menor que sea su valor de ajuste con respecto al resto de la población. El efecto de esto es que la sobre-compresión deja de ser un problema.

Experimentos realizados por [BAKE-85][WHIT-89] han demostrado que esta técnica es superior al escalado de ajuste. Otros métodos (métodos híbridos incluyendo tamaño dinámico de población) se describen en [Bak85], pero no presentan mejoras de rendimiento.

1.2.5.2 Métodos de Ponderación Implícita de Ajuste

Estos métodos llenan las piscina de cruce sin el paso intermedio de ponderar los valores de ajuste.

- Selección por torneo (tournament selection) [BRIN-81][GOLD-91b]: existen numerosas variantes. La más simple de ellas es el torneo de selección binario. En él se escogen parejas de individuos al azar de la población. El que tenga un mayor ajuste se copia a la piscina de cruce (y entonces ambos son reemplazados de la población original). Esto se repite hasta que la piscina está llena. Los torneos pueden ser de más candidatos, de forma que sólo se copia el mejor de n individuos escogidos al azar. El uso de torneos de múltiples candidatos tiene el efecto de incrementar la presión de la

selección, ya que así los individuos con nivel de ajuste bajo tienen menos probabilidades de ganar un torneo.

Una generalización del método anterior es la selección por torneo binario probabilístico. En él, el individuo mejor gana el torneo con una probabilidad p, donde $0.5 < p < 1$. Usando valores más bajos en p, decrece la presión en la selección, ya que así es más probable que ganen torneos los individuos de menor adaptación. Ajustando el número de individuos que intervienen en el torneo o la probabilidad del ganador, la presión en la selección puede variar entre muy grande y muy pequeña.

Goldberg y Deb [GOLD-91b] comparando 3 esquemas diferentes; selección proporcional, ranking de ajuste y selección por torneo concluyen que, ajustando los parámetros adecuados, todos estos esquemas (menos la selección proporcional) pueden ofrecer un rendimiento similar, por lo que no existe un método definitivo.

1.2.6 Operadores de Cruce

El AG estándar utiliza el operador de cruce con un único punto corte generando de cada individuo padre dos partes que se combinan para generar dos individuos hijo. A partir de este operador original se han desarrollado diversos operadores de cruce, frecuentemente utilizando varios puntos de cruce. DeJong [DEJO-75] investigó la efectividad de varios de estos operadores de punto de cruce múltiple y concluyó que el cruce con 2 puntos de corte incrementa el rendimiento del AG, pero que, a partir de dos puntos de cruce el rendimiento vuelve a decrecer. La razón de este empeoramiento es que al utilizar muchos puntos de cruce, se incrementa la probabilidad de romper los bloques constructivos de los individuos eliminando las razones de su mejor ajuste. Además de poder aplicar varios puntos de corte al operador de cruce se puede también pensar en una generalización que llegue a decidir gen por gen su destino en la descendencia, es lo que se llama cruce uniforme.

1.2.6.1 Cruce con 2 Puntos de Corte

En este tipo de cruce, y en cualquier tipo de cruce con múltiples puntos de corte, no se representa la codificación de los individuos como cadenas lineales sino como anillos resultantes de unir la cadena de genes por sus extremos. Para intercambiar un

segmento de uno de estos anillos con otro individuo, es necesario realizar dos cortes. El cruce con un punto de corte puede verse como un cruce con dos puntos de corte en el que uno siempre está fijo al principio de los individuos. Teniendo esto en cuenta, se puede decir que el operador con dos puntos de cruce, realiza la misma tarea que el de uno pero es más general. Además, un cromosoma en anillo contiene más bloques constructivos, ya que pueden empezar justo antes del final del individuo y continuar por el principio. Por estas razones, en la actualidad, es comúnmente aceptado que el operador de cruce de 2 puntos de corte puede aportar mejoras al rendimiento de los AG.

1.2.6.2 Cruce Uniforme

Este tipo de cruce es radicalmente diferente a los operadores con n puntos de cruce. Cada gen de la descendencia se crea copiando los genes de uno u otro de los padres, de acuerdo con una máscara de cruce generada aleatoriamente. Esta máscara está compuesta por tantas posiciones como tienen los individuos y en cada posición habrá valores 1 ó 0. Si en una posición de la máscara hay un uno al primer individuo de la descendencia se le copiará el gen de esa posición del primer padre y al segundo individuo de la descendencia el del segundo padre. Si en la máscara hay un cero, al segundo descendiente se le copia el gen del primer padre y al primer descendiente el del segundo padre. En cada operación de cruce se genera aleatoriamente una nueva máscara de cruce.

Como consecuencia de este tipo de cruce, la descendencia tiene una mezcla de los genes de ambos padres y se puede equiparar a un cruce con n puntos de cruce donde n será de media la mitad de la longitud de los individuos.

1.2.6.3 Otros Tipos de Cruce

Además de las técnicas de cruce ya comentadas, se han desarrollado multitud de variantes. Por un lado, se ha intentado emular la idea de que en la Naturaleza algunos genes del cromosoma son candidatos más probables para punto de cruce que otros. En este sentido se han desarrollado varios métodos [SCHA-87][HOLL-87][DAVD-91a][LEVE-91][LOUI-91]. En ellos se intenta que el propio AG descubra qué lugares deberían ser favorecidos como puntos de cruce. Esta información se guarda en una cadena de puntuación que se incorpora al cromosoma y se pasa a la descendencia con el

resto de información genética. Como es habitual en CE, las cadenas de puntuación que consigan mejores individuos tendrán más posibilidades de ser propagadas a la descendencia.

Por otro lado, se han desarrollado modificaciones de los operadores de cruce para problemas que se basan en el orden de los genes del individuo, como el problema del viajante. En este tipo de problemas no se cruzan los valores de los genes sino el orden en que aparecen. La descendencia hereda la información de la ordenación que tiene cada uno de sus padres. Para este tipo de problemas desarrolló Goldberg [GOLD-85][GOLD-89a] un operador denominado *partially matched crossover* (PMX). Syswerda [SYSW-91] y Davis [DAVI-91] describen también operadores para este tipo de problemas.

1.2.7 Teoría de Funcionamiento de los AA.GG.

1.2.7.1 Plantillas y Teorema del Esquema

El teorema del esquema de Holland [HOLL-75] fue la primera explicación rigurosa de cómo funcionaban los AA.GG. Un esquema es un patrón de valores de genes que puede ser representado (en un código binario) por una cadena de caracteres con el alfabeto {0, 1, #}. Se dice que un cromosoma concreto contiene un esquema en particular si entra dentro de la plantilla, con el símbolo "#" haciendo de comodín. Así, por ejemplo, el cromosoma "1010" contiene, entre otras, las plantillas "10##", "#0#0" "##1#" y "101#". El orden de un esquema es el número de símbolos no-# que contiene: 2, 2, 1 y 3 respectivamente en el ejemplo. La longitud definida de un esquema es la mayor distancia entre símbolos no-#: 2, 3, 1 y 3 respectivamente en el ejemplo.

El teorema del esquema explica el poder de los AA.GG. en términos del procesado de plantillas. A los individuos de la población se les da la posibilidad de reproducirse y producir descendencia. El número de tales oportunidades que recibe cada individuo es proporcional a la valoración de su ajuste; así, los mejores individuos contribuyen con más de sus genes a la nueva generación. Se asume el hecho de que un individuo con un nivel de ajuste alto contiene una buena plantilla. Pasando más de estas buenas plantillas a la nueva generación, se incrementará la probabilidad de encontrar mejores soluciones que las de la generación actual.

Holland enunció que el camino óptimo para explorar el espacio de soluciones es generar nuevos individuos a partir de los ya existentes, proporcionalmente a su nivel de ajuste. En este sentido, las buenas plantillas sufren un incremento exponencial en sucesivas generaciones. Esto es lo que se denomina "Teorema del Esquema". También se demostró que, ya que cada individuo contiene un gran número de plantillas distintas, el número de plantillas que son realmente procesadas en cada generación es del orden de n^3, donde "n" es el tamaño de la población. Esta propiedad se conoce como paralelismo implícito y es una de las explicaciones del buen rendimiento de los AA.GG.

1.2.7.2 La Hipótesis de los Bloques Constructivos

De acuerdo con Goldberg [GOLD-89a], la gran ventaja de los AG reside en ser capaz de encontrar buenos bloques constructivos. Estos bloques son plantillas de longitud definida y corta que están compuestos de bits que trabajan juntos, y tienden a mejorar el rendimiento cuando existen en un individuo. Un buen esquema de codificación es aquel que favorece la creación de bloques constructivos asegurando que:

1. los genes relacionados permanecen juntos en el cromosoma

2. hay poca interacción entre genes.

La interacción entre genes, denominada epístasis significa que la contribución de un gen al ajuste depende del valor de otros genes del cromosoma. Por ejemplo, para la localización mediante eco, los murciélagos son capaces de generar pulsos ultrasónicos, y tienen un buen sistema de oído para detectar los ecos. Así, los genes para un buen oído sólo pueden incrementar el valor de adaptación de un murciélago si también tiene los genes adecuados para la generación de pulsos. En realidad siempre existe alguna interacción entre los genes en las funciones de ajuste multimodal. Esto es muy significativo porque las funciones multimodales son las únicas interesantes desde el punto de vista de la investigación de AA.GG., ya que las funciones unimodales pueden ser resueltas más fácilmente utilizando métodos más simples.

Si se siguen las dos reglas enunciadas, entonces el AG será tan efectivo como predice el teorema del esquema. Desafortunadamente, las condiciones 1 y 2 no son siempre fáciles de cumplir. Los genes relacionados pueden estarlo de tal forma que no permita que todos ellos estén próximos dentro de una cadena unidimensional (por

ejemplo, si se relacionan de forma jerárquica). En la mayoría de los casos, la naturaleza exacta de la relación que se da entre los genes no puede ser conocida por el diseñador, así, incluso existiendo sólo relaciones simples, puede ser imposible adecuar el código para reflejarlo.

La condición 2 es una precondición para la 1. Si la contribución al ajuste total de cada gen fuera independiente de todos los demás genes, entonces sería posible resolver el problema utilizando el "hill-climbing" para cada gen consecutivamente. Por supuesto, esto no es posible normalmente. Si se puede asegurar que cada gen sólo tiene relación con un pequeño número de genes y que pueden ponerse todos juntos en el cromosoma, entonces se cumplen las condiciones 1 y 2. Pero si existe mucha interacción entre los genes, no se puede cumplir ninguna de las condiciones.

Claramente, se debería intentar diseñar esquemas de codificación de acuerdo con las recomendaciones de Goldberg, ya que asegura que el AG funcionará de la mejor manera posible. Se pueden plantear ahora las siguientes cuestiones:

1. ¿Es posible, en general, encontrar esquemas de codificación que se ajusten a las recomendaciones de la hipótesis de los bloques constructivos?, y si la respuesta es sí, ¿cómo se puede diseñar este esquema?.

2. Si no es posible encontrar este esquema de codificación ideal, ¿pueden los AA.GG. modificarse para mejorar su rendimiento en estas circunstancias?, y si es así, ¿cómo?.

Estas cuestiones son temas muy actuales e importantes en la investigación de AA.GG. que deben plantearse antes de la aplicación de las técnicas de AA.GG. en cualquier problema.

1.3 Series Temporales

1.3.1 Proceso Estocástico

Un proceso estocástico se define como una familia de variables aleatorias que se corresponden a momentos sucesivos en el tiempo. Se designa por Y(t,u) donde t es el tiempo y u es la variable aleatoria. Si se fija t en un valor t_0, $Y(t_0,u)$ será una variable

aleatoria. Si, en cambio, se fija u en u_0, para cada momento del tiempo, el proceso tomaría un único valor $Y(t, u_0)$.

La determinación de las características de un proceso estocástico puede hacerse mediante dos formas alternativas, bien a partir de funciones de distribución conjunta o bien a partir de los momentos. Esta determinación realizada mediante las funciones de distribución es, en general, un procedimiento complicado, por lo que se acostumbra a utilizar, preferentemente, el método de los momentos. En una distribución de probabilidad, los momentos de primer y segundo orden, son los más utilizados para su caracterización.

En un proceso estocástico, denotado por Y_t, la media o momento de primer orden se define como:

$$\mu = E(Y_t) \qquad \text{Ecuación 1-5}$$

El subíndice t señala que la media será, en general, distinta para cada período de tiempo. Como momentos de segundo orden respecto a la media es preciso considerar, además de la varianza, las covarianzas entre variables referidas a distintos momentos de tiempo o autocovarianzas, que se definen como:

$$\gamma_{t,s} = \text{cov}(Y_t, Y_s) = E(Y_t - \mu_t)E(Y_s - \mu_s) \qquad \text{Ecuación 1-6}$$

Cuando s=t se obtiene la definición de la varianza

$$\gamma_{t,t} = \text{var}(Y_t) = E(Y_t - \mu_t)^2 \qquad \text{Ecuación 1-7}$$

Como forma alternativa de caracterización de un proceso estocástico se utilizan los coeficientes de autocorrelación que se definen:

$$\rho_{t,s} = \frac{\text{cov}(Y_t, Y_s)}{\sqrt{\text{var}(Y_t)\text{var}(Y_s)}} \qquad \text{Ecuación 1-8}$$

Las autocorrelaciones conjuntamente con las varianzas proporcionan idéntica información que las covarianzas. Sin embargo, es preferible utilizar las autocorrelaciones ya que éstas proporcionan unas medidas relativas, mientras que las covarianzas vienen afectadas por la escala que se utiliza. La caracterización d un proceso estocástico mediante los momentos de primero y segundo orden es, en

principio, más incompleta que cuando se hace mediante funciones de distribución. Ahora bien, si el proceso es normal, éste queda perfectamente caracterizado a través de los dos primeros momentos.

Una serie temporal sería un proceso estocástico en el que se dispone de una observación para cada período de tiempo. Tiene siempre un carácter aleatorio, ya que si se no se podría obtener directamente de la aplicación de una fórmula. Una serie temporal se puede interpretar como una muestra de tamaño uno tomada en períodos sucesivos de tiempo dentro de un proceso estocástico. A diferencia del muestreo aleatorio simple donde cada extracción es independiente de las demás, en una serie temporal el dato extraído para un período concreto no será, en general, independiente de los datos extraídos para períodos anteriores.

1.3.1.1 Procesos Estacionarios

Se dice que un proceso estocástico es estacionario en sentido estricto cuando al realizar un mismo desplazamiento en el tiempo de todas las variables de cualquier distribución finita, resulta que esta distribución no varía. Considerando la función de distribución conjunta:

$$F\left(Y_{t_1}, Y_{t_2}, \ldots, Y_{t_k}\right) \qquad \text{Ecuación 1-9}$$

Si se adopta el supuesto de que todos los elementos de la anterior distribución se desplazan m períodos, la nueva definición de distribución conjunta sería:

$$F\left(Y_{t_1+m}, Y_{t_2+m}, \ldots, Y_{t_k+m}\right) \qquad \text{Ecuación 1-10}$$

Si el proceso es estacionario en sentido estricto se deberá verificar que

$$F\left(Y_{t_1}, Y_{t_2}, \ldots, Y_{t_k}\right) = F\left(Y_{t_1+m}, Y_{t_2+m}, \ldots, Y_{t_k+m}\right) \qquad \text{Ecuación 1-11}$$

e, igualmente, se deberá obtener un resultado análogo para cualquier otra distribución conjunta que tenga carácter finito.

También en el caso del estudio de la estacionariedad es más complejo el análisis a partir de las funciones de distribución que si se efectúa a partir de los momentos. Sin embargo, al realizarlo mediante los momentos, el concepto de estacionariedad será más

limitado. Se dice que un proceso es estacionario de primer orden, o en la media, si se verifica que:

$$E(Y_t) = \mu \qquad \forall t \qquad \text{Ecuación 1-12}$$

Por tanto, en un proceso estacionario en media, la esperanza matemática, o media teórica, permanece constante a lo largo del tiempo.

Se dice que un proceso es estacionario de segundo orden, o en sentido amplio, cuando se verifican las dos condiciones siguientes:

1. La varianza es finita y permanece constante a lo largo del tiempo, es decir:

$$E(Y_t - \mu)^2 = \sigma^2 < \infty \qquad \forall t \qquad \text{Ecuación 1-13}$$

2. La autocovarianza entre dos períodos distintos de tiempo únicamente viene afectada por el lapso de tiempo transcurrido entre esos dos períodos. Así:

$$E(Y_{t+k} - \mu)(Y_t - \mu) = \gamma_k \qquad \forall t \qquad \text{Ecuación 1-14}$$

que sería una autocovarianza de orden k, por ser éste el lapso que separa a Y_t de Y_{t+k}. Su valor, γ_k, es independiente de cuál sea el período t que se considere.

1.3.1.2 Procesos Ergódicos

Además de la estacionariedad, es necesario que un proceso estocástico goce de la propiedad de ergodicidad, con objeto de que el proceso de inferencia pueda realizarse de una forma adecuada. Este concepto se explica mejor de forma intuitiva.

Cuando valores de la serie temporal alejados en el tiempo están muy correlacionados, es decir, cuando ρ_k se mantiene en unas cotas elevadas para un k grande, sucederá que al aumentar el tamaño de la muestra se añade poca información nueva. La consecuencia de este hecho en el plano estadístico será que los estimadores obtenidos no son consistentes, ya que el aumento del tamaño de la muestra no tendrá una especial utilidad, puesto que se tendrá que calcular un mayor número de autocovarianzas para caracterizar adecuadamente el proceso.

Cuando se verifica la propiedad de ergodicidad se pueden obtener estimadores consistentes y una condición necesaria, aunque no suficiente, es que:

$$\lim_{k \to \infty} \rho_k = 0 \qquad \text{Ecuación 1-15}$$

Cuando un proceso es estacionario y también ergódico, todo el problema de inferencia se simplifica de forma considerable. Ahora bien, existen una gran cantidad de series temporales que no están generadas por procesos estacionarios. A la vista de esto, una conclusión que se podría extraer es que puede tener poco interés la utilización de procesos estocásticos. Sin embargo esto no es así ya que, mediante sencillas transformaciones, en la mayor parte de los casos, las series no estacionarias se pueden convertir en series aproximadamente estacionarias, siendo entonces aplicable el proceso de inferencia correspondiente a procesos de este tipo.

1.3.1.3 Procesos Lineales

Dentro de los procesos estacionarios y ergódicos se presta más atención a una clase especial de procesos denominados procesos lineales. Estos procesos se caracterizan porque se pueden representar como una combinación lineal de variables aleatorias. En el año 1938 el profesor Wold [WOLD-38] enunció y demostró el siguiente teorema:

Cualquier proceso estacionario Y_t puede representarse unívocamente como la suma de dos procesos mutuamente incorrelacionados.

$$Y_t = D_t + X_t \qquad \text{Ecuación 1-16}$$

donde D_t es linealmente determinista y X_t es un proceso de medias móviles puramente no determinista. Debido a esto, los procesos lineales que centran la mayor atención son: los procesos puramente aleatorios, los procesos autorregresivos, los procesos de medias móviles y los obtenidos como combinación de estos dos últimos.

El proceso puramente aleatorio es el más simple de todos. Podría expresarse de la forma:

$$Y_t = \varepsilon_t \qquad \text{Ecuación 1-17}$$

donde ε_t satisface las siguientes propiedades:

$$E[\varepsilon_t]=0 \qquad \forall t$$

$$E[\varepsilon_t]^2=\sigma^2 \qquad \forall t \qquad \text{Ecuación 1-18}$$

$$E[\varepsilon_t,\varepsilon_{t'}]=0 \qquad t\neq t'$$

Así pues, ε_t se caracteriza porque su media y su varianza son constantes a lo largo del tiempo y porque no existe relación entre valores referidos a momentos distintos del tiempo. En el tratamiento de series temporales, se suele designar a un proceso puramente aleatorio con la denominación de "ruido blanco". Naturalmente, al no existir relación entre valores referidos a momentos distintos del tiempo, el ruido blanco está alejado de la concepción intuitiva que se ha dado de proceso estocástico, es decir, para manejar variables aleatorias del tipo ε_t no haría falta la teoría de los procesos estocásticos. Sin embargo, el ruido blanco es una pieza clave en la construcción de procesos estocásticos más complicados como pueden ser los procesos autorregresivos y los procesos de medias móviles.

El proceso autorregresivo de orden p, o utilizando la notación usual, un proceso AR(p) se expresa de la siguiente forma:

$$Y_t=\phi_1 Y_{t-1}+\phi_2 Y_{t-2}+...\phi_p Y_{t-p}+\varepsilon_t \qquad \text{Ecuación 1-19}$$

Como puede verse en un proceso AR(p), aparece un ruido blanco referido al momento actual y la variable desfasada para distintos períodos, siendo p el retardo máximo que aparece en el proceso. La denominación de autorregresivo procede de que Y_t se obtiene mediante regresión sobre valores desfasados de la propia variable. Los procesos autorregresivos fueron introducidos por primera vez por Yule [YULE-27].

Un proceso de medias móviles (*moving average* en terminología inglesa) de orden q, o un proceso MA(q) viene dado por:

$$Y_t=\varepsilon_t-\theta_1\varepsilon_{t-1}-\theta_2\varepsilon_{t-2}-...-\theta_q\varepsilon_{t-q} \qquad \text{Ecuación 1-20}$$

La expresión de medias móviles hace referencia a que la variable Y_t se obtiene como un promedio de variables de ruido blanco, siendo las θ_i los coeficientes de ponderación. Como las variables que forman parte de este promedio varían a lo largo

del tiempo, reciben el apelativo de móviles. La introducción de estos procesos también se debe a Yule [YULE-21][YULE-26].

Finalmente, mediante una combinación de un proceso autorregresivo y un proceso de medias móviles se obtiene un proceso ARMA(p,q), donde p indica el retardo máximo de la parte autorregresiva y q señala el correspondiente a la parte de las medias móviles. La divulgación y popularización de este tipo de procesos se debe fundamentalmente a la obra de Box y Jenkins [BOX-76], aunque fueron estudiados anteriormente por Wold [WOLD-38] y Bartlett [BART-46]. La expresión de un proceso ARMA(p,q) es la siguiente:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... \phi_p Y_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - ... - \theta_q \varepsilon_{t-q} \quad \text{Ecuación 1-21}$$

1.3.1.4 Procesos no Estacionarios

En los procesos descritos en el punto anterior se ha impuesto las condición de estacionariedad. Sin embargo, buena parte de las series temporales se debe considerar que están generadas por procesos no estacionarios. Así pues, si se desea obtener un tratamiento de las series basado en la teoría de los procesos estocásticos es necesario ampliar el campo de estudio para, al menos, incluir algunos tipos de procesos no estacionarios. En principio cabe imaginar distintas formas por las cuales se puede introducir la no estacionariedad en un proceso. Sin embargo, interesa considerar solamente ciertos tipos de procesos no estacionarios que sean adecuados para describir el comportamiento de los fenómenos en estudio y que sean fácilmente transformables en procesos estacionarios con objeto de poder utilizar las ventajas que ofrecen estos últimos. Desde esta perspectiva, dentro de los modelos no estacionarios se considerará en primer lugar y a modo de introducción el modelo:

$$Y_t = Y_{t-1} + \varepsilon_t \quad \text{Ecuación 1-22}$$

El modelo anterior es un AR(1) con $\phi_1=1$. A este modelo se le denomina también "paseo aleatorio". Cuando el proceso se inicia en un pasado remoto, mediante sustituciones sucesivas se puede expresar así:

$$Y_t = \sum_{j=0}^{\infty} \varepsilon_{t-j} \quad \text{Ecuación 1-23}$$

La varianza del proceso es infinita y el proceso es, en consecuencia no estacionario. Sin embargo, el proceso puede transformarse fácilmente en estacionario. Así, el proceso:

$$w_t = Y_t - Y_{t-1} = \Delta Y_t \qquad \text{Ecuación 1-24}$$

De acuerdo con la Ecuación 1-22, se tiene que:

$$w_t = \Delta Y_t = \varepsilon_t \qquad \text{Ecuación 1-25}$$

Es decir, el proceso transformado, obtenido al tomar primeras diferencias en el proceso original, es ruido blanco. Como se ve tomando diferencias de primer orden se pasa de Y_t al proceso w_t. Hay que ver ahora cómo realizar el proceso inverso, es decir, cómo obtener Y_t a partir del proceso w_t. Por aproximaciones sucesivas se tiene que:

$$Y_t = w_t + Y_{t-1} = w_t + w_{t-1} + Y_{t-2} = \ldots = w_t + w_{t-1} + w_{t-2} + w_{t-3} + \ldots \qquad \text{Ecuación 1-26}$$

Por tanto, el proceso Y_t se obtiene sumando, o lo que es lo mismo, integrando el proceso w_t. Por esta razón, se dice que "el paseo aleatorio" pertenece a la clase de modelos integrados. Esta clase está constituida por todos aquellos modelos que se pueden transformar en estacionarios mediante la toma de diferencias de un determinado orden, o dicho de otra forma, modelos integrados son aquellos que se pueden obtener mediante suma o integración de un proceso estacionario. A estos modelos se les denomina también modelos no estacionarios homogéneos y han sido estudiados por Tintner [TINT-40], Tintnery Rao [TINT-63], Yanglom [YANG-55] y por Box y Jenkins [BOX-76], siendo estos últimos autores los que más han contribuido a su divulgación.

A un proceso integrado Y_t se le denomina proceso ARIMA(p,d,q) si tomando diferencias de orden d se obtiene un proceso estacionario w_t del tipo ARMA(p,q). La I central del término ARIMA indica, pues, integrado. Los procesos ARIMA constituyen una clase particular de procesos no estacionarios. Sin embargo, en muchos casos son suficientes para representar el comportamiento de las series temporales. En otros casos se puede tener en cuenta alguna transformación previa, como la toma de logaritmos sobre los valores de la serie, cuando se observa que en un período dilatado de la serie, la varianza se ve afectada por la tendencia. Los estadísticos Box y Cox [BOX-64] definieron en este sentido una transformada de carácter más general.

Anexo 2 Manual de Usuario

2. Anexo Manual de Usuario de la Aplicación

2.1 Introducción

En este punto de los Anexos se expone el manual de utilización del sistema desarrollado para comprobar las hipótesis expuestas en este trabajo. Este programa es el que se ha utilizado para realizar las pruebas expuestas en el Capítulo de Resultados.

En primer lugar se comentan los requerimientos para la instalación del programa y los pasos a seguir en esa instalación. A continuación, se indican las labores a realizar para el entrenamiento básico de RR.NN.AA. en todas sus fases: creación del conjunto de entrenamiento, configuración de los parámetros de la RNA y del AG, seguimiento del proceso de entrenamiento, almacenamiento de las redes ya entrenadas y realización de pruebas. El siguiente punto comenta cómo utilizar el módulo de ajuste de la arquitectura, sobre todo en el seguimiento del proceso de entrenamiento. Después, se explica la forma de utilizar el módulo de diseño de los conjuntos de entrenamiento y, en el último punto, se comenta cómo configurar los distintos módulos para que trabajen juntos de forma cooperativa.

2.2 Instalación

2.2.1 Requerimientos

Este sistema funciona sobre plataforma PC y, para la parte de integración de módulos, es necesario que estas computadoras estén conectadas a una red de ordenadores y que tengan instalado el protocolo TCP/IP.

En las pruebas realizadas durante el desarrollo del sistema se han utilizado varias computadoras con procesador Pentium II a una velocidad de reloj de 233MHz, con 64Mb de memoria RAM de tipo DIMM de 100MHz. En las últimas pruebas realizadas con el sistema, que no se han reflejado en el Capítulo de Resultados para mantener la consistencia en las comparaciones con resultados previos, se han podido utilizar los nuevos procesadores Pentium II a velocidades de reloj de hasta 400MHz. Obviamente, el rendimiento del sistema mejora notablemente, tanto por la mayor capacidad de

cálculo del procesador como por la mayor tasa de transferencia en los buses de las placas base.

Los requerimientos mínimos son, en cuanto a procesador, el que tenga una potencia comparable a un Pentium II con una velocidad de reloj de 200MHz y en cuanto a memoria RAM un mínimo de 32Mb siendo aconsejable 64Mb. Si se quiere utilizar el sistema cooperativo de desarrollo de RR.NN.AA. se necesita un mínimo de tres computadoras.

2.2.2 Pasos

El sistema se distribuye en un CD-ROM de instalación que consta de una serie de ficheros para la descompresión e instalación, y otros con el sistema propiamente dicho.

Figura 2.1.- Pantalla de la instalación solicitando el nombre del directorio destino

Para comenzar la instalación se ejecuta, desde el administrador de Windows, el programa "Setup.exe", en el directorio raíz de la unidad del CD-ROM, con lo que se iniciará el programa de instalación. La primera consulta que se realiza al usuario desde

la instalación es el directorio en el que se va a instalar la aplicación (Figura 2.1). El programa escoge por defecto el directorio C:\ERNAAG, pero el usuario puede modificar este valor. Una vez escogido el directorio se pulsa el botón "Next".

Figura 2.2.- Pantalla de la instalación solicitando el grupo de programas

La siguiente cuestión que se plantea es la creación de un grupo de programas propios en el menú de inicio del sistema operativo (Figura 2.2). El programa de instalación ofrece de nuevo un valor por defecto que el usuario puede modificar. Una vez escogido el nombre se pulsa de nuevo el botón "Next".

A partir de este momento se procede a la descompresión de archivos del programa, su copia al directorio especificado, la creación del grupo de programas y los iconos que lanzarán las distintas aplicaciones del sistema (Figura 2.3). La instalación también configura el módulo de acceso a BD, de la empresa Borland, denominado BDE, necesario para que la aplicación acceda a la BD con los conjuntos de datos y las RR.NN.AA. Por último la instalación registra la aplicación en el sistema operativo para que conste como software instalado.

Figura 2.3.- Pantalla con la evolución de la instalación de ficheros y configuración del sistema

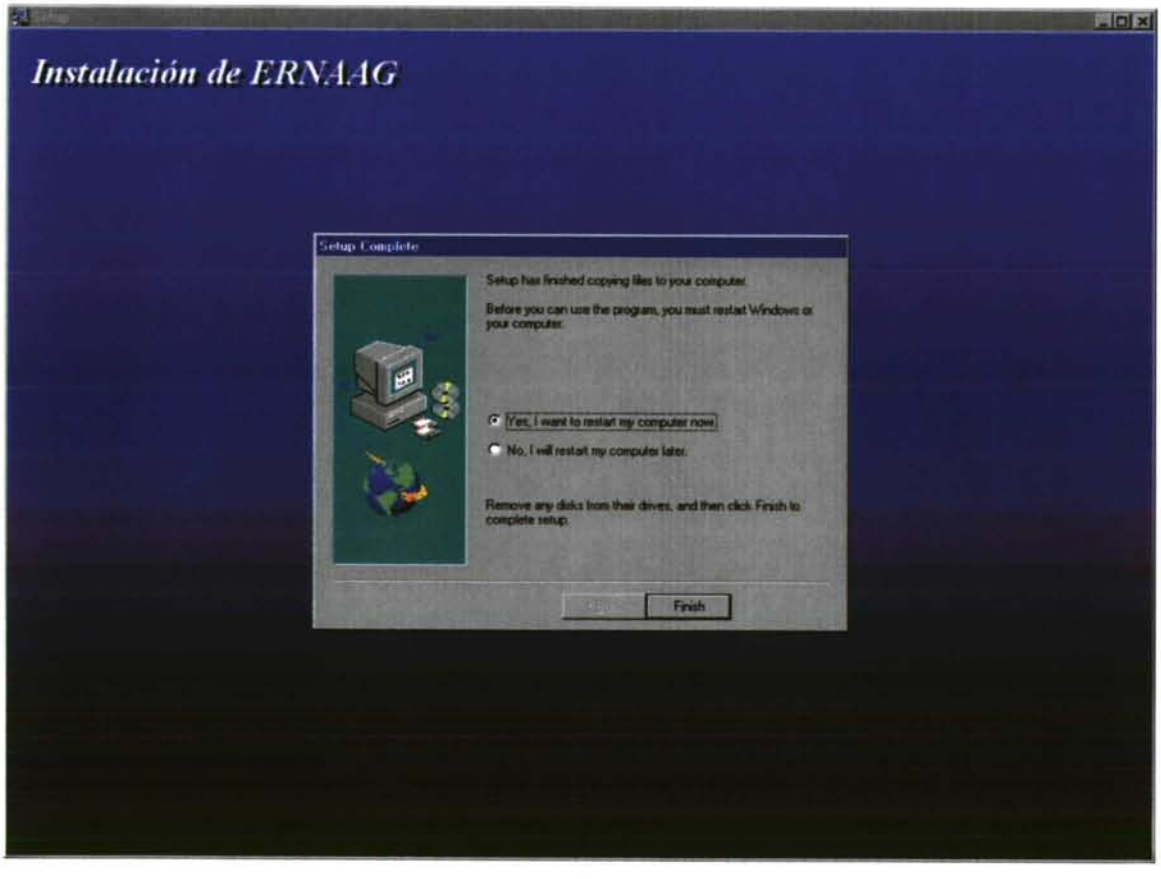


Figura 2.4.- Pantalla final de la instalación para reiniciar el sistema y actualizar el sistema operativo

Después de acabar la fase de copia de archivos y configuración del sistema, el programa de instalación le pregunta al usuario si desea reiniciar la computadora para que tengan efecto todos los cambios realizados (Figura 2.4). Es conveniente reiniciar el equipo antes de proceder a utilizar el programa por lo que se puede optar por decir que sí para que realice el reinicio automático el programa de instalación.

Una vez terminado el proceso de instalación, se crea un grupo de programas con accesos directos a cada uno de los programas de que consta la aplicación. Para lanzar los programas se abre el menú "Inicio" del sistema operativo, dentro de éste el de "Programas" y ahí, se puede encontrar el grupo de programas con el nombre que se le haya dado durante la instalación (Figura 2.5).

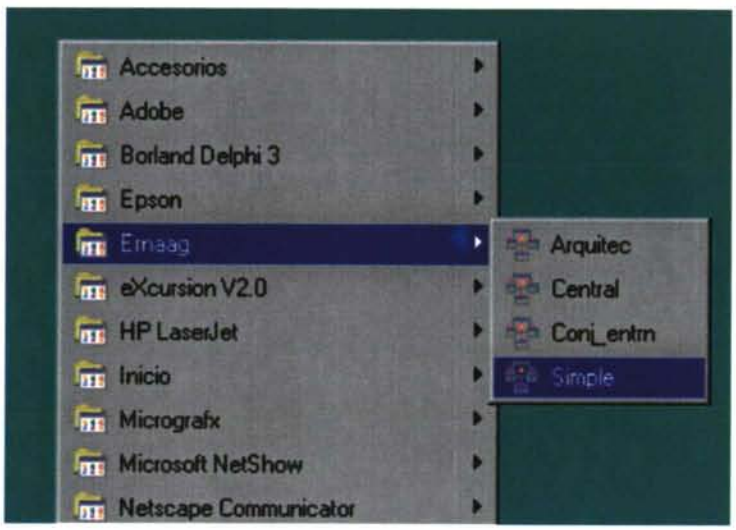


Figura 2.5.- Menú del inicio de programas desde donde se pueden lanzar las aplicaciones del sistema

2.3 Entrenamiento Básico de RR.NN.AA.

En este primer punto, se expone una guía para el desarrollo de RR.NN.AA. entrenadas mediante AA.GG. En este punto se utiliza el módulo básico de entrenamiento de RR.NN.AA. mediante AA.GG. que se corresponde con el programa denominado Simple.exe del grupo de programas de la aplicación. Los pasos a seguir en su utilización son, la creación y definición del conjunto de entrenamiento que se va a utilizar en el entrenamiento, la configuración de los parámetros de la RNA, la especificación de los parámetros del AG, el seguimiento del proceso de entrenamiento, el almacenamiento de la RNA entrenada, la realización de las pruebas y, por último, la utilización de la RNA con nuevas entradas; es decir, la puesta en producción.

2.3.1 Gestión de los conjuntos de entrenamiento

En esta fase primera fase, se selecciona uno de los conjuntos de entrenamiento almacenados en la BD o, si no se dispone todavía del conjunto de entrenamiento en la BD, se puede crear uno nuevo e ir añadiendo los valores de entrada-salida de los distintos ejemplos. En la Figura 2.6 se muestra la pestaña "Ficheros de Entrenamiento" del programa de entrenamiento de RR.NN.AA. mediante AA.GG. en la que se pueden llevar a cabo las tareas comentadas.

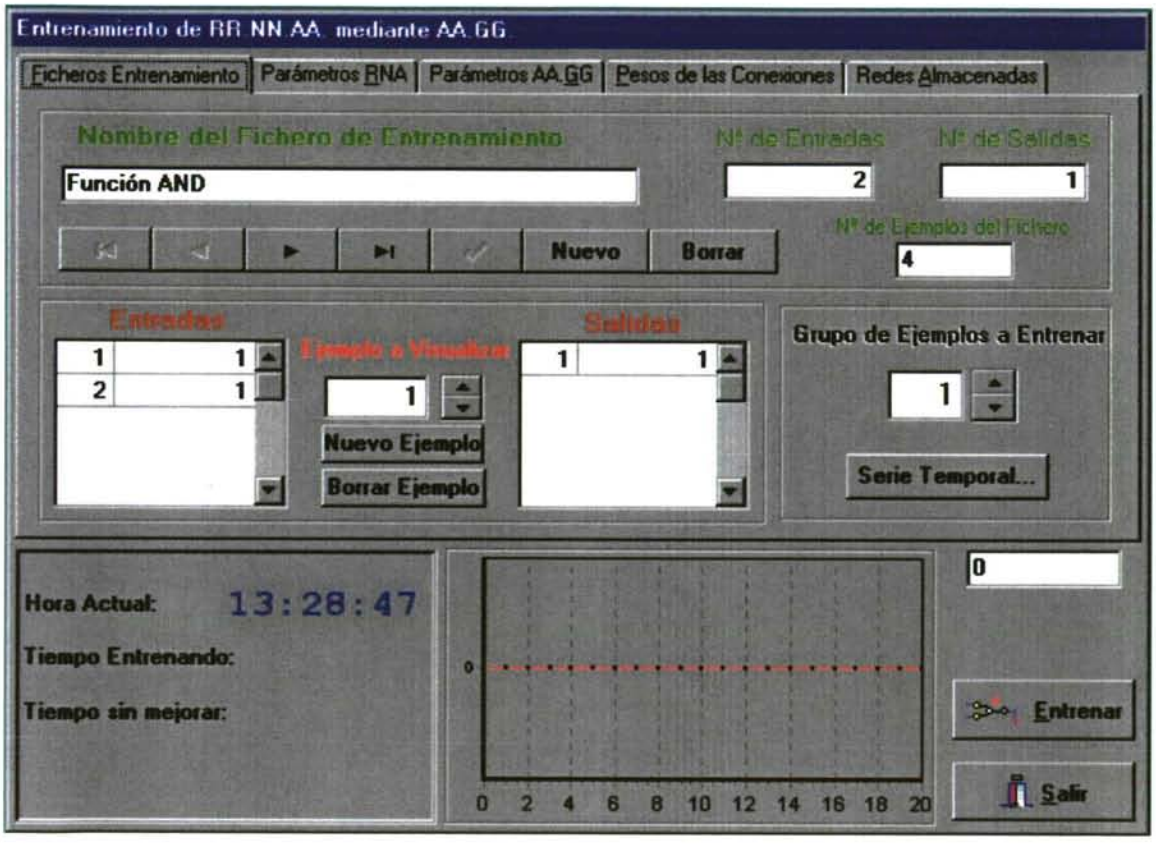


Figura 2.6.- Pestaña de selección del conjunto de entrenamiento

Si se quiere escoger un conjunto de entrenamiento que ya reside en la BD se utiliza el control de navegación por la BD para desplazarse de uno a otro (Figura 2.7).

Figura 2.7.- Navegador de la BD

Al ir pasando de un conjunto a otro, a la derecha del nombre del conjunto de entrenamiento, se va mostrando la información del número de entradas, de salidas y de ejemplos de ese conjunto de entrenamiento. Por debajo de esta parte de la ventana está el navegador de ejemplos (Figura 2.8).

Figura 2.8.- Navegador de ejemplos

En este grupo de controles se pueden visualizar los valores de las entradas y salidas de un ejemplo concreto y cambiar de ejemplo en el control central. También existen dos botones para añadir o borrar ejemplos. Los valores de los ejemplos añadidos se cubren en los campos de texto: "Entradas" y "Salidas".

Si se desea crear un nuevo conjunto de entrenamiento se pulsa el botón "Nuevo" del navegador de la BD, que establece un nombre prefijado para el conjunto de entrenamiento. Este nombre se puede cambiar sustituyéndolo en el campo de texto que lo visualiza. Una vez creado el conjunto de entrenamiento hay que ajustar el número de entradas y de salidas en los campos de texto correspondientes. Cada vez que se realice una de estas modificaciones hay que confirmarlas pulsando el botón con el símbolo ✓ del navegador de la BD. Una vez fijados estos valores se añaden ejemplos al conjunto de entrenamiento como se acaba de comentar. Para borrar un conjunto de entrenamiento se pulsa el botón "Borrar" del navegador de la BD.

En el caso de estar añadiendo un conjunto de entrenamiento que representa una serie temporal, las salidas de una entrada se tienen que corresponder con los valores de la serie del instante siguiente a los de la entrada. En este caso se puede definir una ventana temporal para su utilización con redes sin retropropagación poniendo el tamaño de la ventana en el campo de texto titulado: "Grupos de Ejemplos a Entrenar" y pulsando el botón "Serie temporal". En la BD que se distribuye con el programa ya están incluidas las series temporales de manchas solares y producción de tabaco.

2.3.2 Caracterización de las RR.NN.AA.

En este paso se especifican los parámetros de la RNA que se va a entrenar. Para realizar esta configuración se pulsa en la pestaña denominada Parámetros RNA (Figura 2.9). La primera opción a configurar es si se va a utilizar una red con o sin retropropagación. Para indicarlo, existe un control que indica si se permiten o no

conexiones recurrentes dentro de la red. Al pulsarlo se modifica la apariencia de la ventana para poder especificar número de capas y EP en el caso de redes alimentadas hacia delante o sólo número de EP en RR.NN.AA.RR. En la Figura 2.9 el control está sin marcar por lo que se está configurando una red alimentada hacia delante. En este tipo de redes, se indica el número de capas ocultas en el control de la esquina superior izquierda. Al cambiar el valor en este control se modifica el número de líneas del control de tipo lista que está bajo él para tener tantas líneas como capas ocultas tiene la red. En cada línea, se especifica el número de EP que tendrá la capa oculta correspondiente. En el ejemplo de la Figura 2.9 se indica que la red va a tener dos capas ocultas con tres EP la primera y uno la segunda. A la derecha de estos controles se indica la suma de EP que va a contener la red en sus capas ocultas. En el caso del ejemplo, cuatro.

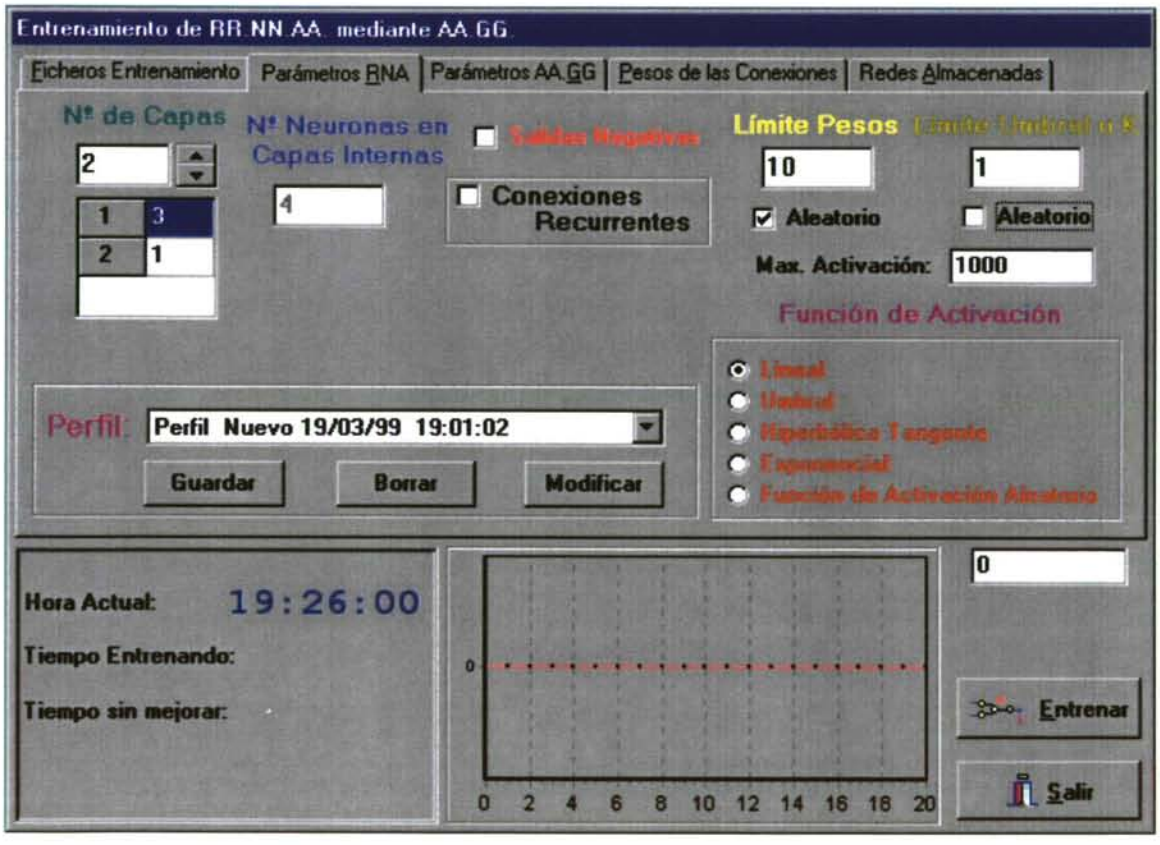


Figura 2.9.- Pestaña de configuración de parámetros de la RNA

En el caso de que se opte por una red con recurrencias en las conexiones, el aspecto de esta ventana cambia al marcar la opción de recurrencias (Figura 2.10). En este caso no se especifican el número de capas sino solamente el número de EP internos a la red. En el caso de las redes recurrentes hay parámetros específicos a configurar,

como el número de iteraciones que se deja iterar una red antes de comprobar los valores de las salidas y el modo de entrenamiento: discreto o continuo.

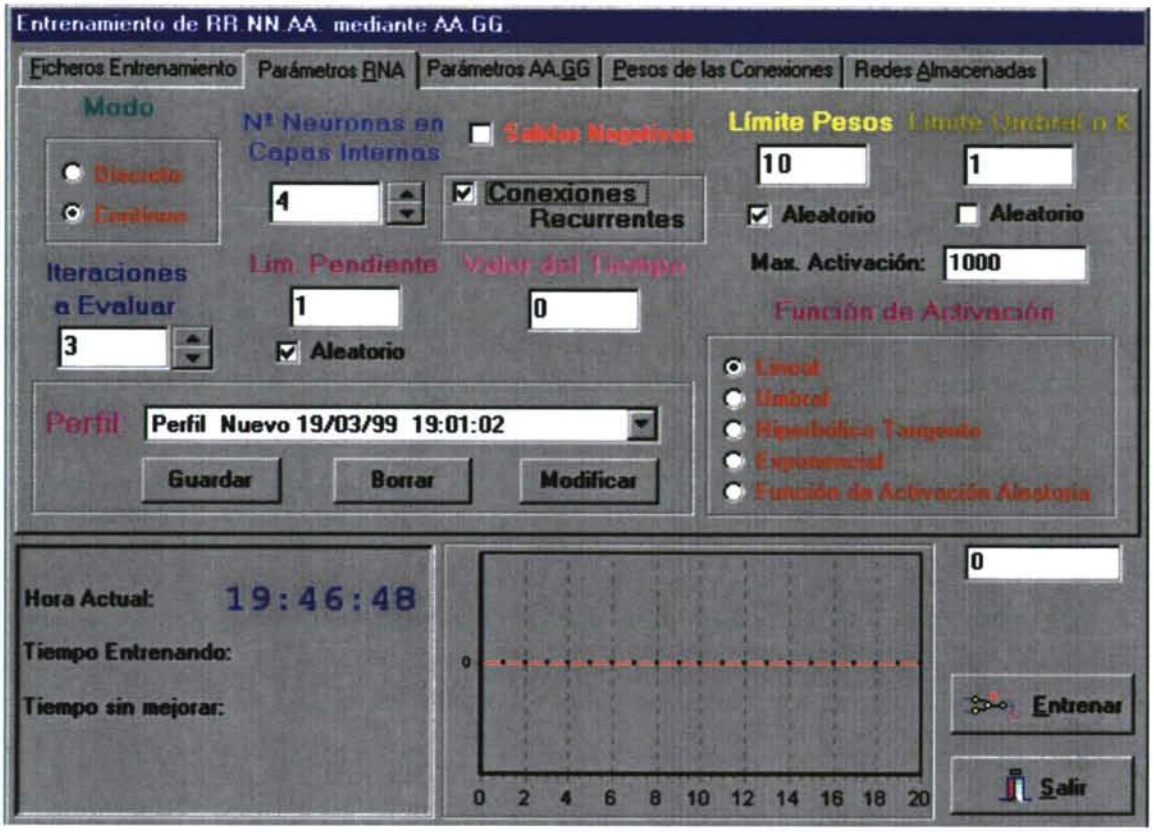


Figura 2.10.- Configuración de parámetros para RR.NN.AA.RR.

Siguiendo por la pantalla hacia la derecha, se especifican parámetros como si se permiten salidas negativas a los EP y el límite de los valores para los pesos de conexión. Este valor es el valor absoluto del límite y, por tanto, en el ejemplo los valores de los pesos están comprendidos en el intervalo [-10,10]. Si no se marca el control denominado "aleatorio" debajo de la especificación de límite todos los pesos se inicializan al valor especificado. En caso de marcarse se inicializan al azar dentro del rango antes comentado.

El resto de parámetros tienen que ver con las funciones de activación que se escogen para los EP. En primer lugar se puede escoger uno de los cuatro tipos: lineal, umbral, hiperbólica tangente o exponencial, para todos los EP de la red o indicar que cada EP va a tener una función de activación distinta escogiendo la opción de aleatoria. Si se escoge la función lineal es conveniente limitar los valores de activación, en el control "Max. Activación", para que no se produzcan "overflow" en los cálculos de activación de los EP. También se puede especificar el límite de la pendiente de la

función lineal y el límite para el valor de umbral si se utiliza este otro tipo de función de activación. El control denominado "Aleatorio", debajo del control de límite, funciona igual que el comentado para el caso del límite de los pesos.

Por último, para facilitar la utilización continua de la aplicación, se ha implementado un gestor de perfiles (Figura 2.11). Esta herramienta permite almacenar los valores de una configuración determinada de parámetros de una RNA en esta pestaña y darle un nombre. Una vez guardada es posible recuperar los valores de todos los parámetros de esta ventana escogiendo este perfil del control desplegable de la zona de perfiles.

Figura 2.11.- Gestor de perfiles

2.3.2.1 Configuración de la Activación Atenuada

Dentro de la pestaña de Parámetros RNA, cuando se ha escogido la opción de "Conexiones Recurrentes" (Figura 2.10), también se activan los controles de los parámetros de la activación atenuada (Figura 2.12). Como ya se comentó en la descripción del sistema, los parámetros a configurar son el límite de la pendiente de la recta que simula el potencial de acción y el valor del tiempo que se considera para evaluar la activación a la salida de una neurona. Como en el punto anterior si no se marca el control "Aleatorio", inicialmente todas las pendientes valen uno. Si se marca, las pendientes se inicializan aleatoriamente.

Figura 2.12.- Parámetros de la activación atenuada

2.3.3 Caracterización de los AA.GG.

El siguiente paso es configurar los parámetros del AG de entrenamiento. En la Figura 2.13 se observa el aspecto de la pestaña "Parámetros AA.GG." donde se configura el funcionamiento del AG. Desde la parte izquierda a la derecha, se

configuran el número de individuos de la población y el número de cruces y mutaciones por generación. Después aparecen los parámetros de la operación de cruce: el tipo y la selección de individuos. Por último, se configura el tipo de inserción de individuos nuevos en la población (ordenada, sustitución de padres y sustitución por igual error).

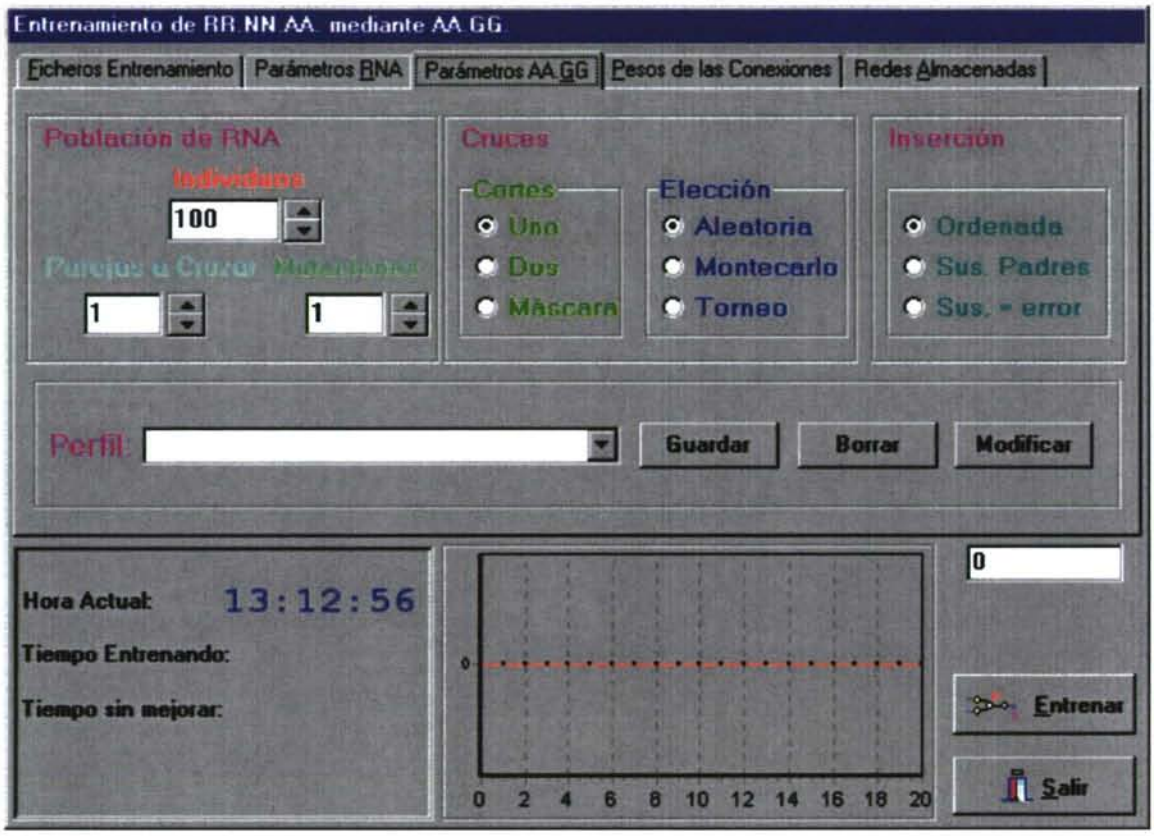


Figura 2.13.- Configuración de parámetros del AG

Al igual que en el caso de los parámetros de las RR.NN.AA., con los parámetros de los AA.GG. también se pueden almacenar configuraciones concretas en perfiles de forma que su reutilización sea muy fácil. La forma de funcionar es idéntica, una vez configurados los parámetros se almacenan pulsando el botón "Guardar" pudiendo darle un nombre o mantener el que el sistema le pone por defecto.

2.3.4 Evolución del Proceso de Entrenamiento

Una vez escogido el conjunto de entrenamiento y configurados los parámetros de la RNA y el AG se puede iniciar el proceso de entrenamiento. Para ello se pulsa el botón "Entrenar" localizado en la parte inferior derecha de la pantalla del programa. Después de un período de inicialización de la población en el que se valoran todos los individuos, comienza la simulación. La mejor forma de seguir la simulación es desde la

pestaña "Pesos de las Conexiones" en la que se visualiza, cada vez que se mejora el mejor individuo, los valores actualizados del error cuadrático medio, del error medio, de los pesos de todas las conexiones entre neuronas y de las pendientes o umbrales de las funciones de activación (Figura 2.14).

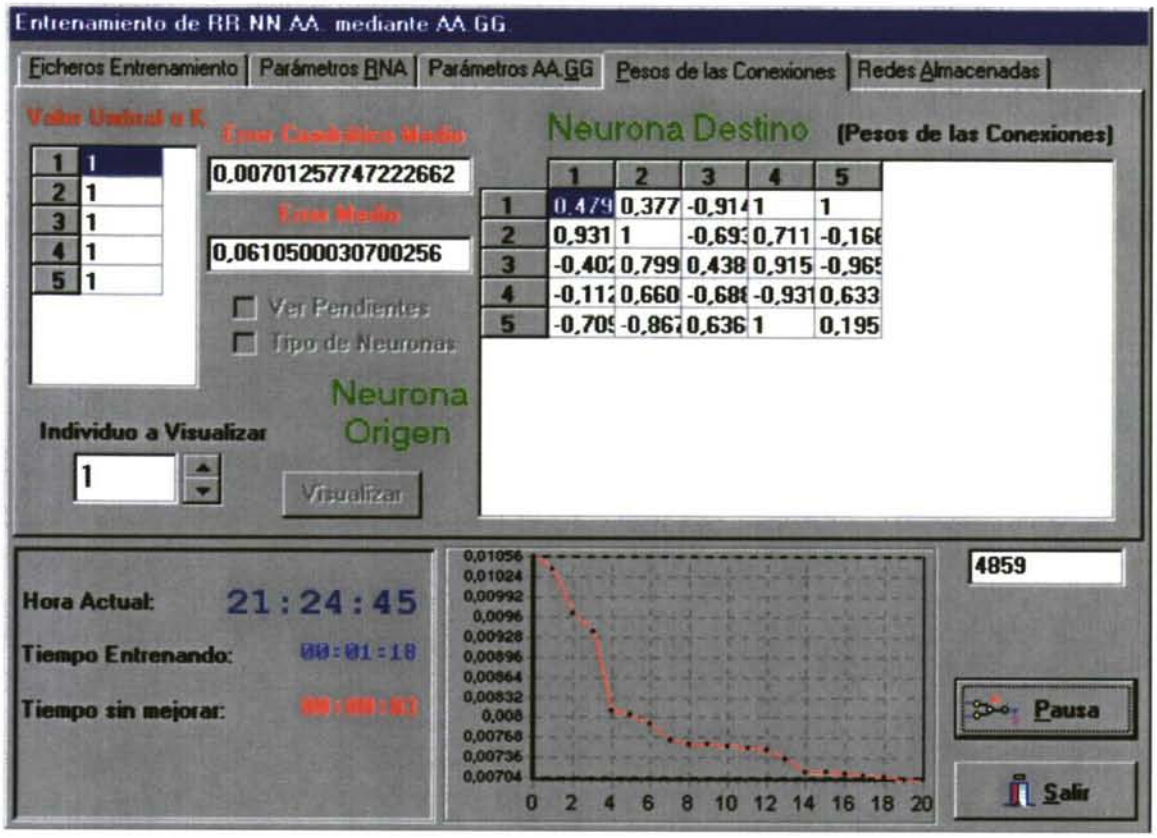


Figura 2.14.- Pestaña de pesos de las conexiones y evolución del proceso de entrenamiento

Además, en la parte inferior se visualiza un gráfico del error cuadrático medio conseguido en las últimas 20 mejoras del primer individuo de la población, lo que proporciona una información muy importante para valorar el proceso de entrenamiento de la red. En la parte izquierda se visualiza el tiempo de entrenamiento y el tiempo transcurrido desde la última mejora del primer individuo. A la derecha del gráfico de error se indica el número de generaciones de la simulación del AG.

En cualquier momento del entrenamiento se puede realizar una pausa en la simulación, de forma que se puedan estudiar con calma los resultados conseguidos hasta ese punto. Una vez parada la simulación pulsando el botón "Pausa", este botón cambia su etiqueta por "Volver" y aparece un nuevo botón con la etiqueta "Terminar" que pone fin a la simulación y permite modificar parámetros tanto de la RNA como del AG para comenzar una nueva simulación (Figura 2.15). Con la simulación detenida se pueden

ver los valores de los parámetros para el resto de los individuos de la población y comprobar así el grado de homogenización de la población. Para ello se cambia el número de orden en el control Individuo a Visualizar y se pulsa el botón "Visualizar". Así se puede comprobar los valores de Umbral, K, Error Cuadrático Medio, Error Medio y Pesos de las Conexiones para ese individuo. Los valores de Tiempo Entrenando y Tiempo sin mejorar se detienen hasta que se pulse el botón de "Volver".

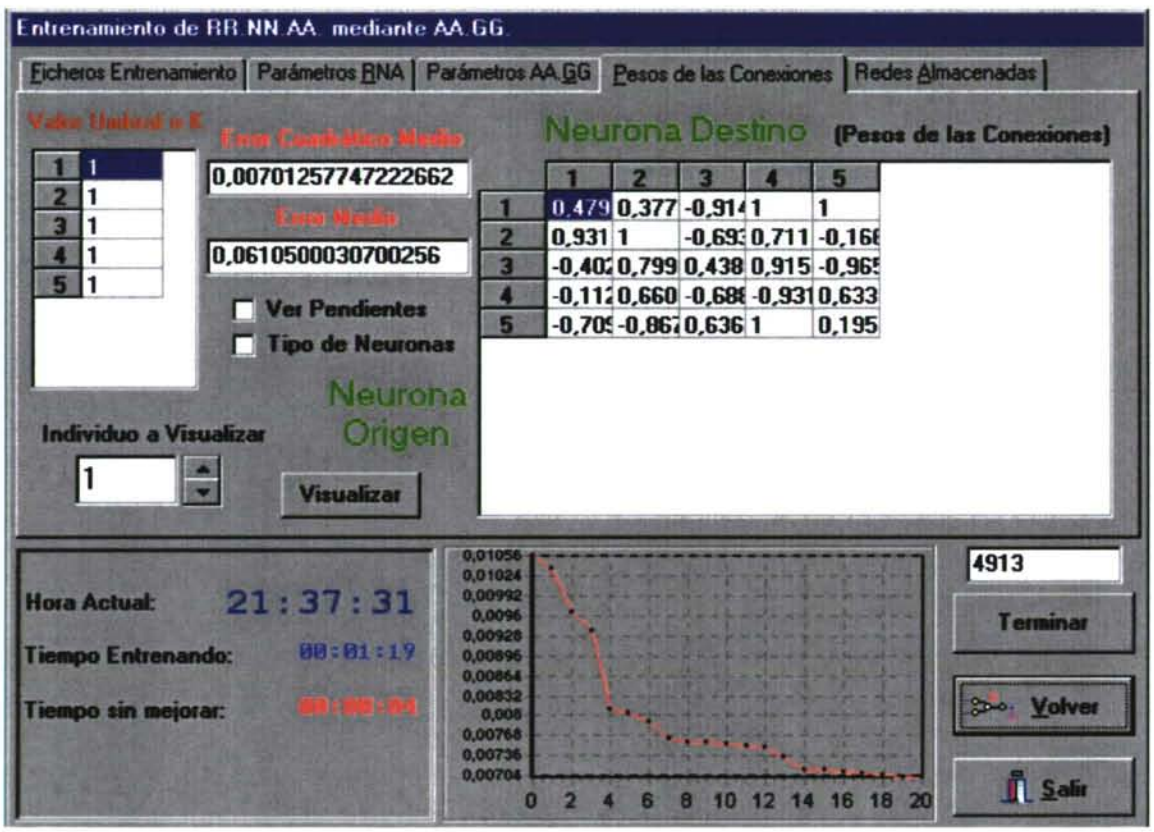


Figura 2.15.- Pantalla de entrenamiento en situación de pausa

En el caso de que se pulsen los controles de "Ver pendientes" y "Tipo de Neuronas" (Figura 2.16) cambia la información de los controles principales que, de mostrar el Valor de Umbral o K, pasa a mostrar el tipo de funciones de activación y, de mostrar el valor de los pesos de las conexiones, pasa a mostrar el valor de las pendientes de las conexiones.

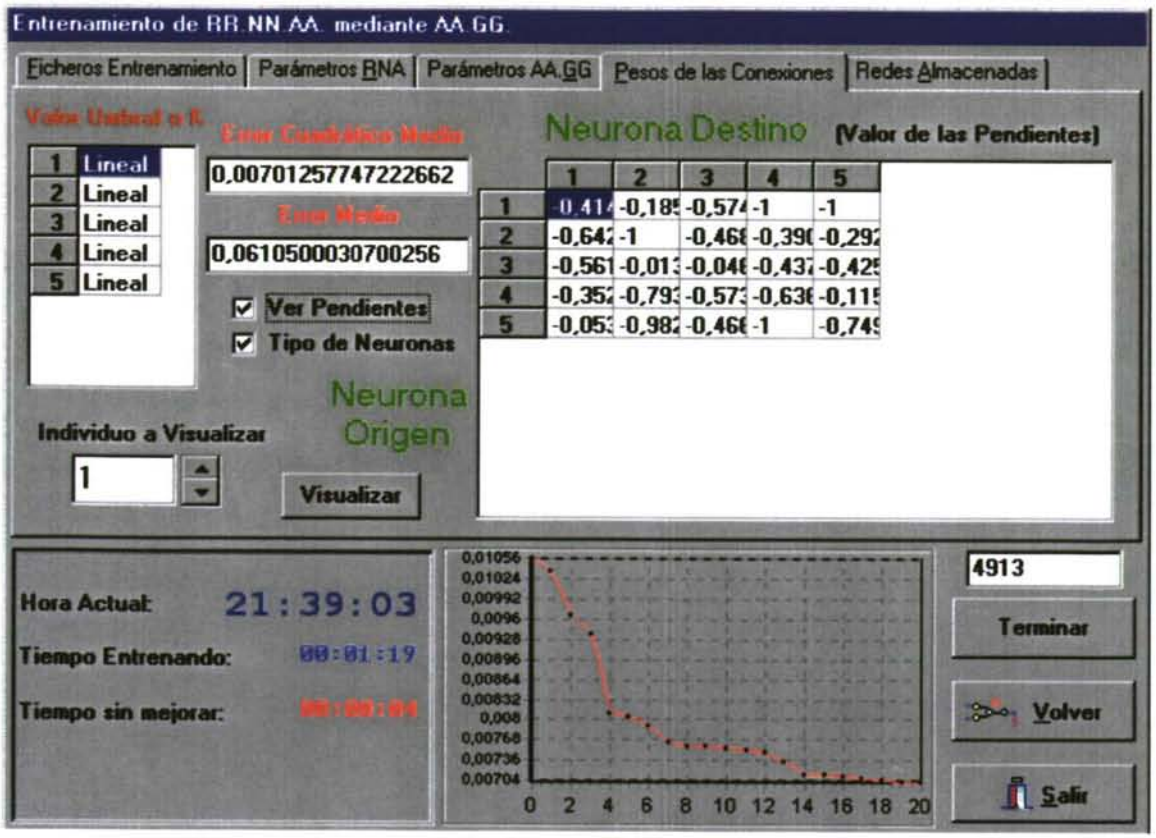


Figura 2.16.- Pantalla de entrenamiento en pausa visualizando pendientes y funciones de activación

2.3.5 Almacenamiento de las RR.NN.AA. Entrenadas

Una vez que se alcanza el nivel de error deseado en la fase de entrenamiento, se puede proceder a guardar la configuración de RNA conseguida para su prueba y utilización posterior. Para ello, se pulsa en la pestaña denominada "Redes Almacenadas" donde se puede proceder al almacenamiento en la BD de la red resultado del entrenamiento realizado (Figura 2.17).

En la BD se almacena toda la información referente a la RNA. Por un lado se guarda la información de la arquitectura como el número de capas, el número de EP y las funciones de activación utilizadas. Por otro, los resultados del entrenamiento: los valores de los pesos y las pendientes referidos a las conexiones y la pendiente o K referido a las funciones de activación. Hay que tener en cuenta que lo que se está almacenando es la información de la mejor RNA de la población en un momento dado. No se está almacenando información alguna sobre el AG; es decir, la población sobre la que se está trabajando y los parámetros del AG. Esto quiere decir que, en caso de querer repetir un experimento dado, lo que se puede hacer es guardar los perfiles de

configuración de la RNA y del AG para generar una nueva población y realizar una nueva simulación con los mismos parámetros.

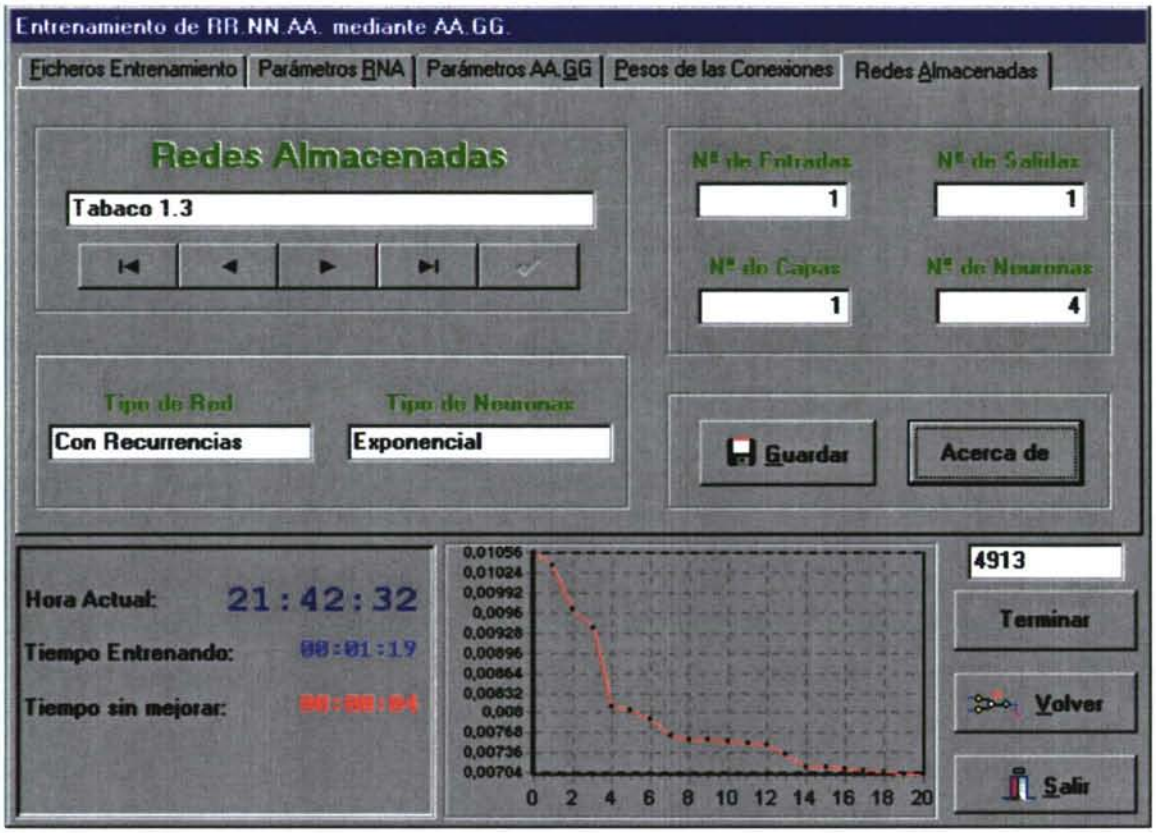


Figura 2.17.- Pantalla para el almacenamiento de RR.NN.AA. ya entrenadas

El proceso de grabación es muy sencillo. Una vez se consigue el error deseado en el proceso de entrenamiento se pulsa el botón de "Pausa". A continuación, se pasa a la pestaña de "Redes Almacenadas" y se pulsa el botón "Guardar". Esto guarda la información sobre la RNA comentada en el párrafo anterior y le asigna un nombre prefijado a la red. Como en el caso de la grabación de conjuntos de entrenamiento, se puede cambiar el nombre de la red pulsando después el símbolo ✓ del control navegador de redes. La información que se visualiza de la red es a título informativo y, como es lógico, no se puede cambiar manualmente en esta pantalla.

2.3.6 Fase de Test de RR.NN.AA.

Después de haber salvado las características de una RNA se puede proceder a su evaluación, considerando ésta como la fase de prueba o test junto con la fase de puesta en producción. Para entrar en las pantallas del programa en las que se realizan la fase de prueba hay que situarse en la pestaña de "Redes Almacenadas" y pulsar el botón

"Evaluar Redes" (Figura 2.18). El botón es el mismo que se pulsa para almacenar las redes. En el caso de que se esté realizando un entrenamiento la etiqueta aparece como "Guardar" mientras que, si no se está realizando ningún entrenamiento la etiqueta aparece como "Evaluar Redes". Al pulsar el botón "Evaluar Redes" la ventana se transforma (Figura 2.19) en la pantalla de Evaluación o puesta en producción. Para realizar el test se pulsa el botón "Test de la RNA" de la parte inferior de la pantalla.

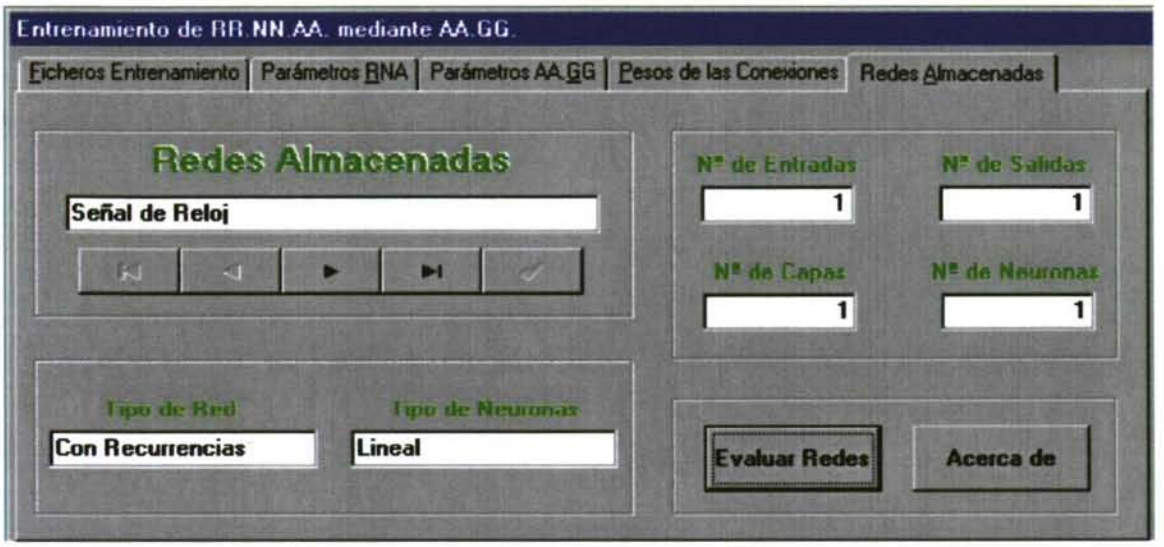


Figura 2.18.- Entrada a las pantallas de prueba y puesta en pruducción

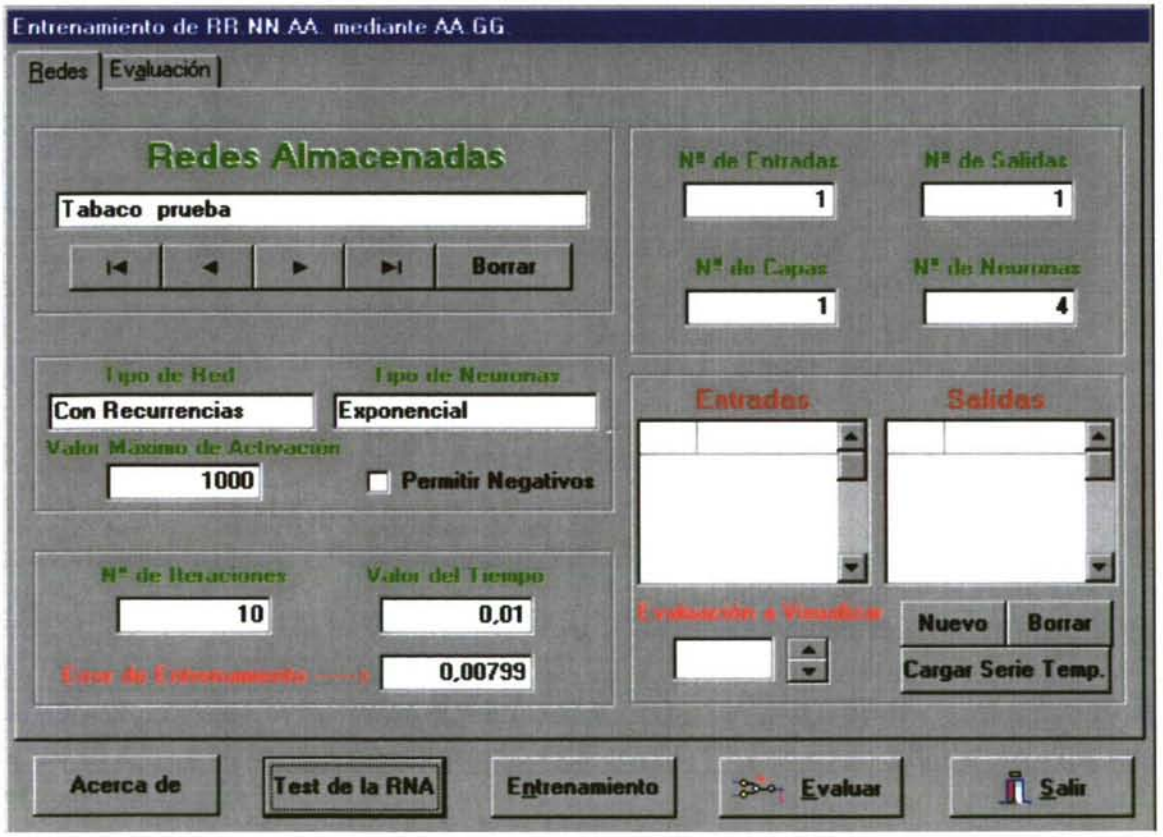


Figura 2.19.- Pantalla de puesta en producción de RR.NN.AA.

Nuevamente la pantalla cambia para mostrar la ventana donde se realizará la prueba de las RR.NN.AA. desarrolladas (Figura 2.20). En esta ventana, en la pestaña "Redes", al escoger la red a la que se va a aplicar la fase de test, se visualiza toda la información disponible sobre ella. En la parte derecha se muestra la información de capas y EP. En la parte izquierda la información de existencia de recurrencias, tipo de funciones de activación utilizadas, utilización de atenuación temporal y nivel de error conseguido en la fase de entrenamiento. En la parte inferior derecha se muestran los valores de entrada-salida de los ejemplos con los que se va a realizar el test. Estos ejemplos son parte del fichero de entrenamiento utilizado que, previamente a la realización del test, ha de ampliarse con nuevos ejemplos en la ventana de gestión de conjuntos de entrenamiento (Figura 2.8).

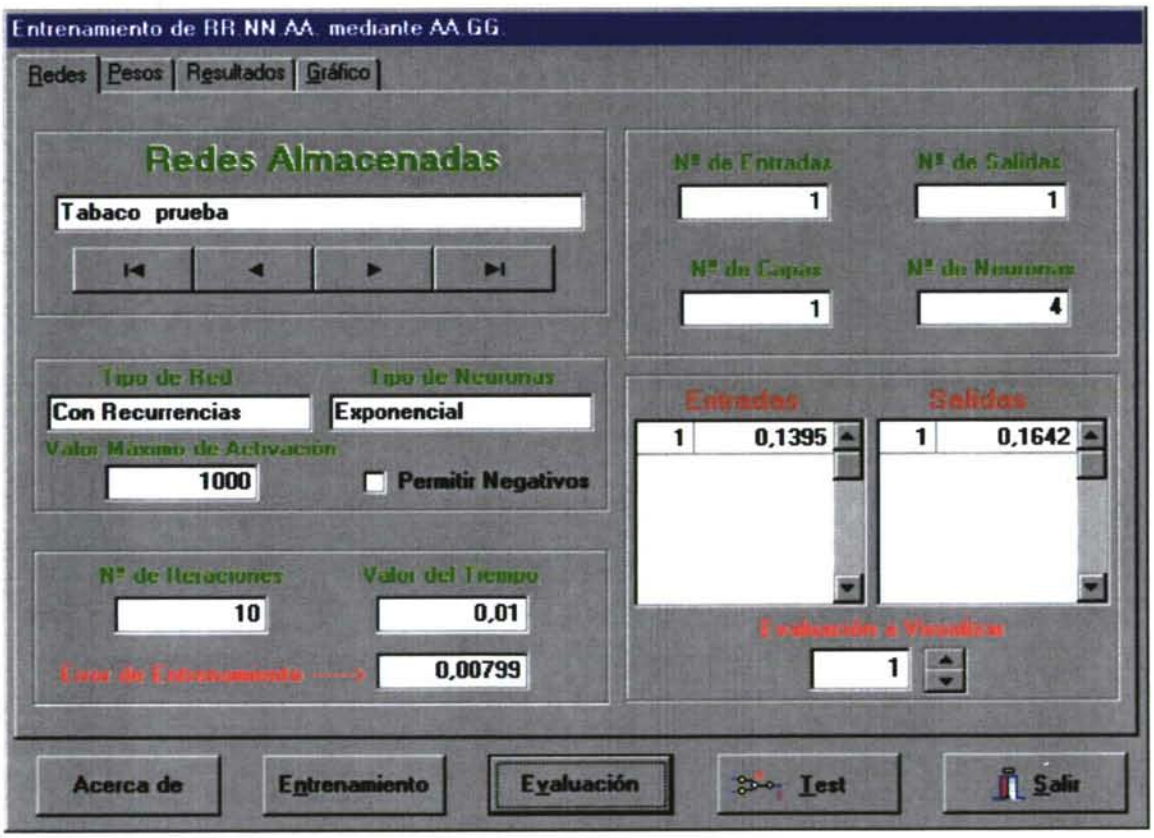


Figura 2.20.- Pantalla de test de las RR.NN.AA ya entrenadas

También desde esta ventana se pueden estudiar los pesos de la red y los valores relativos a las funciones de activación que utiliza. Si se pulsa en la pestaña "Pesos" y, dentro de ella, en el botón "Visualizar", se muestran los parámetros mencionados de la red que se haya seleccionado en la pestaña Redes (Figura 2.21).

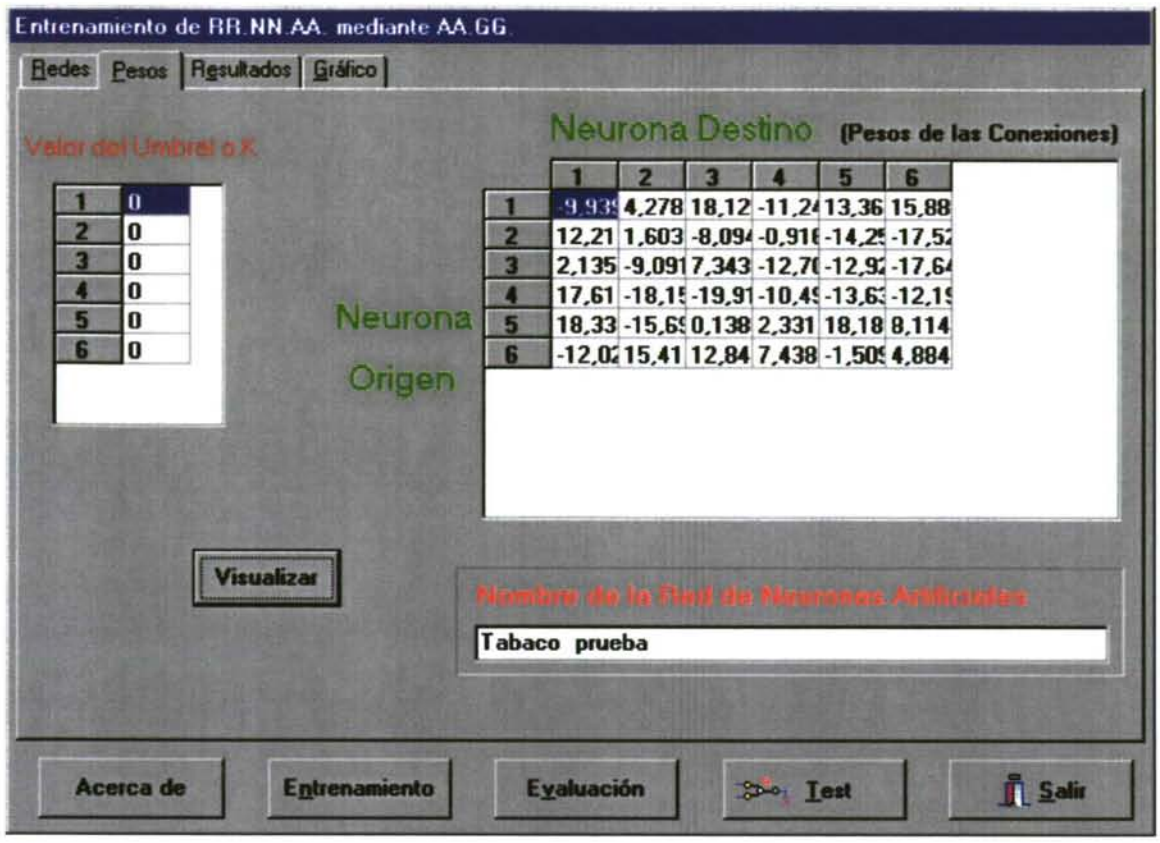


Figura 2.21.- Parámetros de la red seleccionada para el proceso de prueba

Una vez seleccionada la red y añadidos los ejemplos del test se pulsa el botón "Test" de la parte inferior de la ventana. Con esta acción se construye la estructura de una RNA con los valores de diseño almacenados, se cargan los valores de los pesos, las pendientes y las funciones de activación; y se presentan en las entradas de la red los valores del conjunto de ejemplos del test. Una vez aplicados los valores del conjunto de prueba se puede pasar a comprobar los resultados obtenidos.

Para comprobar los resultados se presentan dos opciones, por un lado, en la pestaña Resultados se muestran los resultados numéricos comparados entre las salidas obtenidas de la red y las salidas deseadas. Por otro, en la pestaña Gráfico se muestran de forma de gráfica estos mismos resultados para poder realizar una comparación mucho más fácil. Para volver a las pantallas de entrenamiento se ha de pulsar el botón "Entrenamiento" de la parte inferior de la Ventana. Pulsando el botón "Salir" se termina la ejecución del programa.

Figura 2.22.- Pantalla de resultados comparados en formato numérico

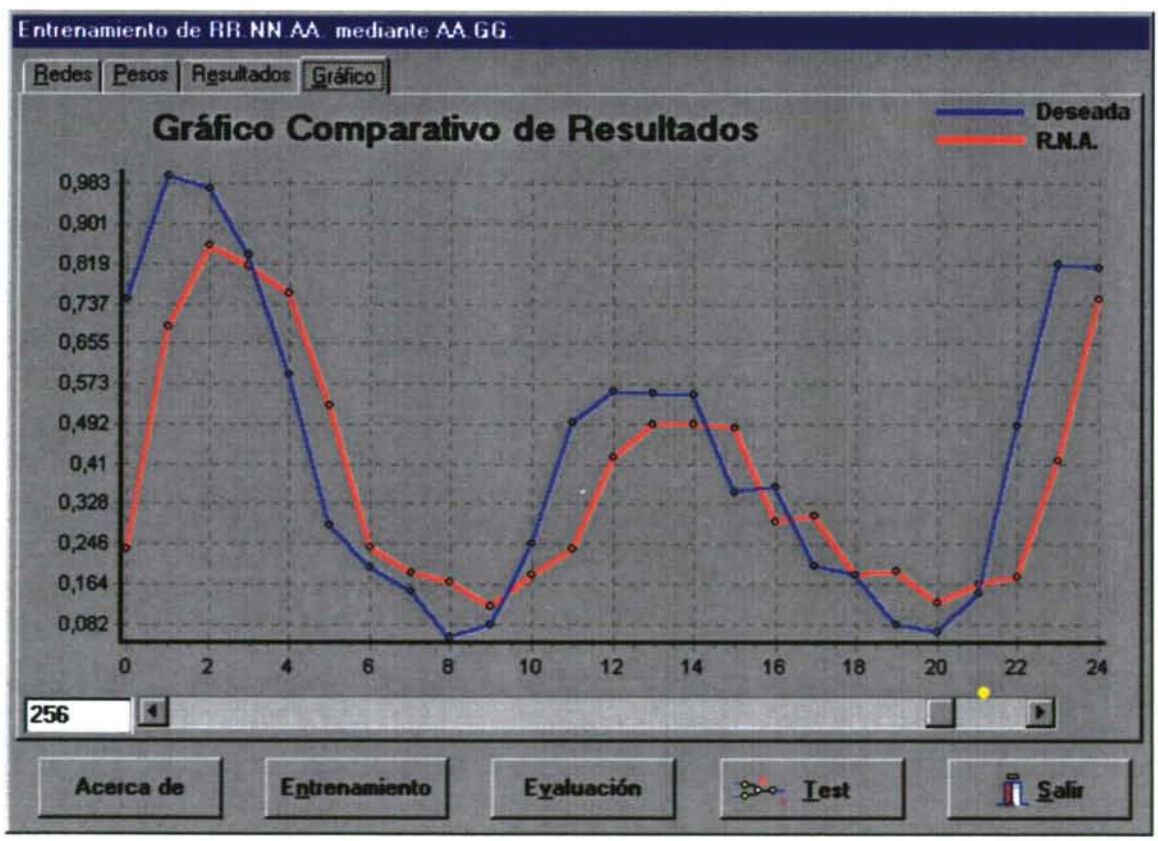


Figura 2.23.- Pantalla de resultados comparados en formato gráfico

2.3.7 Puesta en Producción de RR.NN.AA.

Como se ha visto en el punto anterior, para llegar a la ventana de Evaluación o puesta en producción de redes se pulsa en la pestaña "Redes Almacenadas" (Figura 2.18) y se pulsa el botón "Evaluar Redes". Una vez cambia la ventana (Figura 2.19), también como en el caso anterior, lo primero es escoger la RNA que se quiere emplear utilizando el navegador de la BD. Una vez fijada la RNA, se muestran los parámetros de su arquitectura. Igual que en el caso anterior, se pueden estudiar los valores de los pesos de las conexiones y los parámetros referidos a las funciones de activación de los EP, en la pestaña denominada "Pesos" (como en el caso del test, Figura 2.21).

Una vez escogida la red y repasados sus parámetros, se procede a presentar los valores para los que se desea obtener una salida de la red. Dentro de la ventana de "Evaluar Redes" (Figura 2.19) en la parte inferior derecha se sitúa el navegador de los datos a evaluar (Figura 2.24). En este navegador se pueden insertar y borrar datos para las entradas utilizando los botones "Nuevo" y "Borrar" o cargar la serie temporal, previamente modificada en el gestor de conjuntos de entrenamiento, que se utilizó en el entrenamiento.

Figura 2.24.- Detalle de la pantalla de evaluación para la gestión de datos de entrada

Una vez introducido el conjunto de datos que se quiere evaluar se pulsa el botón "Evaluar" de la parte inferior de la ventana para conseguir, en la parte derecha del navegador de datos (Figura 2.24), las salidas de los valores de entrada que se han introducido.

2.4 Diseño de Arquitecturas

A partir de este punto, se empieza a trabajar con los módulos integrados en el sistema cooperativo de desarrollo de RR.NN.AA. mediante AA.GG. que consta de tres partes: esta primera de diseño de arquitecturas, la de ajuste del conjunto de entrenamiento y el módulo de comunicaciones. Los dos primeros módulos pueden funcionar de manera cooperativa si se utilizan desde el módulo de comunicaciones pero también pueden funcionar de forma independiente, que es el modo de funcionamiento que se describe en este punto. El módulo de este punto se corresponde en el grupo de programas con Arquitec.exe. La puesta en funcionamiento de estos módulos tiene muchos puntos en común con el entrenamiento básico que se acaba de describir por lo que algunos pasos de utilización no se volverán a describir.

Para realizar el entrenamiento de una RNA con este módulo, los pasos comunes con el entrenamiento básico son: la creación y definición del conjunto de entrenamiento que se va a utilizar en el entrenamiento (con el mismo funcionamiento que en el punto 2.3.1), la configuración de los parámetros de la RNA con la única salvedad que, en este caso, el número de EP que se indica en la pantalla de parámetros de la RNA sería el número máximo de EP que va a tener la red, pudiendo tener las redes desarrolladas un número menor (el resto de parámetros se configuran de la misma forma que en el punto 2.3.2).

Lo siguientes pasos: la especificación de los parámetros del AG y el seguimiento del proceso de entrenamiento, difieren de los comentados en los puntos anteriores y se comentarán a continuación. Los últimos pasos: el almacenamiento de la RNA entrenada, la realización de las pruebas y, la utilización de la RNA con nuevas entradas, no cambian, ni en filosofía, ni en funcionamiento a los comentados en los puntos 2.3.5, 2.3.6 y 2.3.7 por lo que, estos puntos sirven como referencia de manejo.

2.4.1 Caracterización de los AA.GG. para el Diseño de RR.NN.AA.

En este punto se procede a la configuración de los AA.GG. que se utilizan en el ajuste de arquitecturas de las RR.NN.AA. Existen dos AA.GG., uno para el diseño de las arquitecturas y otro para el entrenamiento de las RR.NN.AA. Este último es el que

se ha utilizado en el entrenamiento básico y su funcionamiento y configuración ya han sido descritas. En la Figura 2.25 se puede comprobar el contenido de la pestaña AA.GG. donde se configuran los parámetros de ambos AA.GG. En la parte izquierda se configuran los parámetros del AG de diseño de arquitecturas: tamaño de la población y número de cruces y mutaciones por generación. El parámetro iteraciones se refiere al número de iteraciones del AG de entrenamiento que generarán el valor de ajuste de una arquitectura determinada.

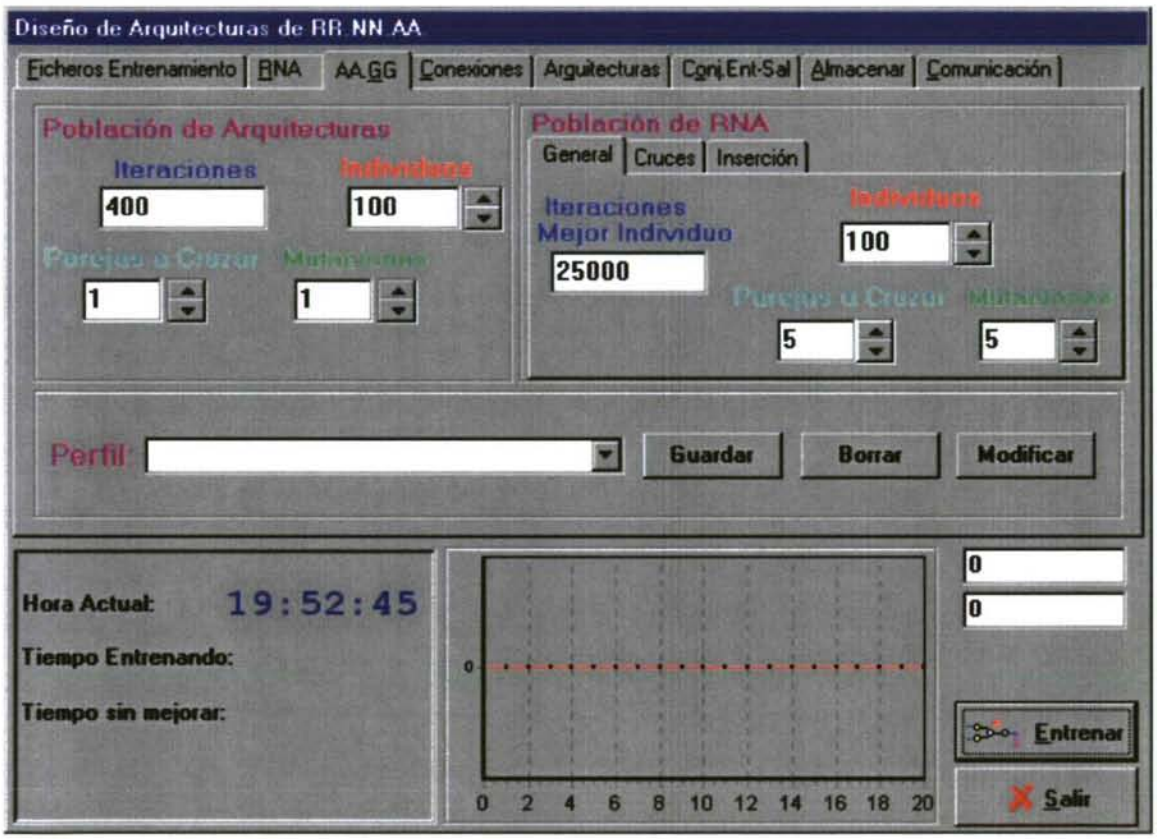


Figura 2.25.- Parámetros de los AA.GG.

En la parte derecha se configuran los parámetros para el AG de entrenamiento de las RR.NN.AA. y todos los parámetros son similares a los comentados en el punto 2.3.3. En este caso los mismos parámetros se presentan en un control con pestañas que da acceso a: tamaño de la población, ratios de cruce y mutación, tipos de cruce y selección y sustitución de individuos. Hay un parámetro nuevo que aparece como "Iteraciones Mejor Individuo" que permite evaluar un individuo de la población con un número mayor de iteraciones que las especificadas en el campo "Iteraciones" de los parámetros del AG de diseño de arquitecturas. La utilidad y funcionamiento de estos dos parámetros se explican en detalle en la parte final del siguiente punto.

2.4.2 Evolución del Proceso de Ajuste de las Arquitecturas

En esta parte del sistema, igual que en el caso del entrenamiento básico, se dispone de unas pantallas para realizar el estudio del avance en el proceso de entrenamiento de las redes. En este caso, no se está entrenando una única arquitectura de RNA con distintos valores de pesos de conexiones y de funciones de activación, sino que se está evaluando una población de arquitecturas con un nivel de ajuste individual para cada una de ellas.

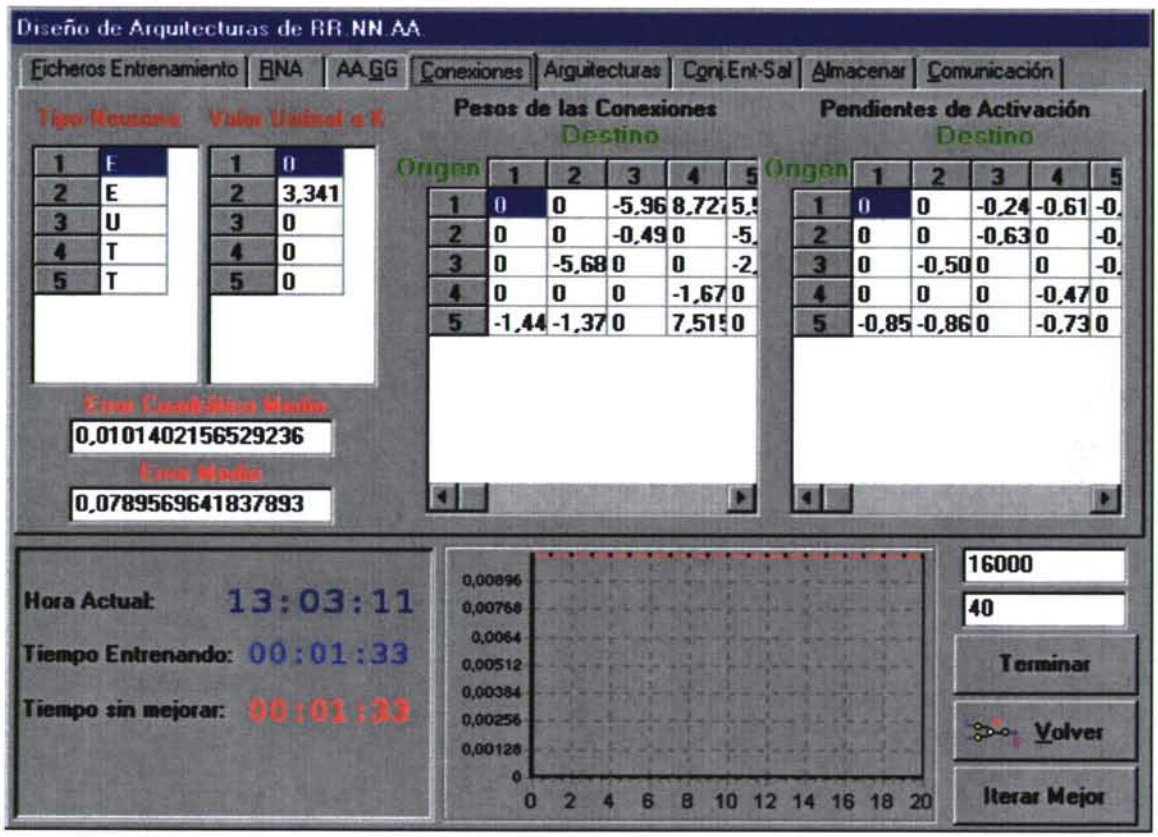


Figura 2.26.- Pantalla de los valores del mejor individuo de la mejor arquitectura actual

Debido a la existencia de estos dos niveles, en el programa existen dos pantallas distintas donde comprobar la evolución de la simulación en cada uno de ellos. La primera pantalla (Figura 2.26) es parecida a la comentada en la Figura 2.15 y en la Figura 2.16. En ella se pueden comprobar los valores de los pesos de conexión, de las pendientes de activación y de las funciones de activación para el mejor individuo de la población en la que se consiguió la mejor arquitectura hasta ese momento. También se muestra el Error Cuadrático Medio y el Error Medio cometido en el entrenamiento de esta red. Hay que destacar que, en esta parte del programa, ya se aplica el podaje de conexiones por lo que, tanto en la matriz de pesos, como en la de pendientes de

activación se puede observar un grupo de conexiones puestas a cero. Realmente, este es el resultado de la aplicación de la matriz de conexiones a la matriz de pesos y de pendientes ya que, como se ha explicado en el Capítulo 4, la matriz de pesos y pendientes mantienen valores en todas sus posiciones y no los ceros que se muestran en pantalla. Para acabar con esta pantalla comentar que, además del número total de iteraciones del AG, habitual en las pantallas del entrenamiento básico, aquí aparece un nuevo contador que indica las iteraciones del AG de diseño; es decir, se aumenta en una iteración el AG de diseño por "n" iteraciones del AG básico. Este número "n" se ha definido en la pestaña "RNA", en el campo "Iteraciones" de los controles de "Población de Arquitecturas".

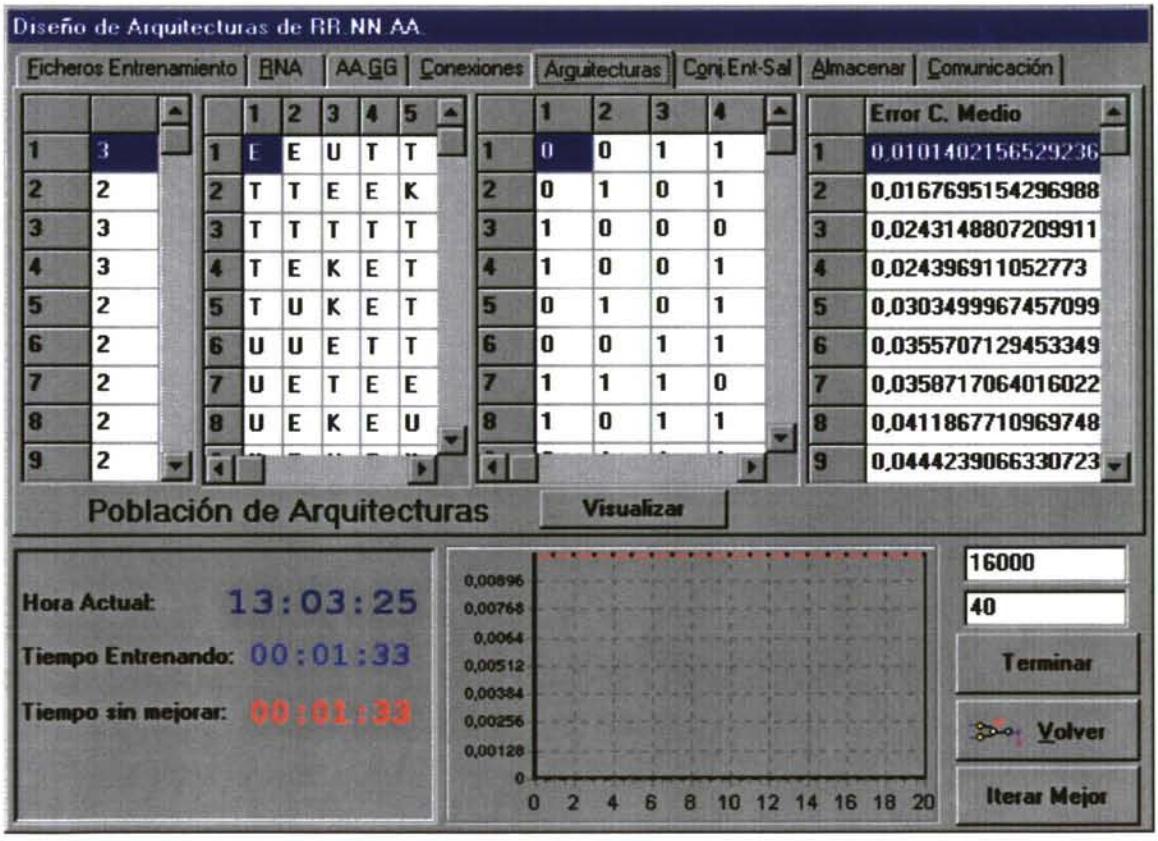


Figura 2.27.- Pantalla con la evolución de las arquitecturas de RNA

La segunda pantalla (Figura 2.27) permite comprobar la evolución de la población de arquitecturas de las RR.NN.AA. Para poder estudiar los valores reflejados en esta pantalla es necesario detener el proceso de entrenamiento y pulsar el botón "Visualizar", ya que el refresco de los datos no se realiza automáticamente para no ralentizar la simulación. Una vez parada la simulación y pulsado el botón, cada fila de cada control en la pantalla se corresponde a una arquitectura. La primera fila contiene un resumen de los datos de la pantalla comentada anteriormente. De izquierda a

derecha, se muestran el número de neuronas ocultas, las funciones de activación de los EP, la matriz de conexiones y el Error Cuadrático Medio. En el caso de las funciones de activación y de la matriz de conexiones, como ya se ha comentado en capítulos anteriores, se utilizan matrices de un tamaño superior al necesario de forma que cada individuo mantiene almacenada más información de la que utiliza. Debido a esto, en los controles de estos dos parámetros se pueden ver más datos de los que se utilizan en la evaluación de las redes. Esta información no afecta directamente al valor de adaptación de esa arquitectura pero sí puede afectar a la adaptación de la descendencia.

Por último, como cada arquitectura se evalúa un número limitado de veces para obtener un indicador de su nivel de ajuste y no se realiza un entrenamiento completo, se ha incluido la posibilidad de poder realizar un número mayor de iteraciones sobre la mejor arquitectura de la población para poder conseguir una RNA completamente entrenada a partir de esa definición de arquitectura. Así, en cualquier momento de la simulación se pulsa el botón de "Pausa" y, una vez detenida, aparece un botón "Iterar Mejor". Al pulsar este botón se realiza el entrenamiento de una RNA con la arquitectura del mejor individuo el número de veces reflejado en la pantalla de configuración del AG (Figura 2.25) en el control "Iteraciones Mejor Individuo". No es necesario terminar aquí la simulación y se puede reanudar pulsando el botón "Volver". Después de continuar la simulación se puede nuevamente parar en cualquier momento para volver a iterar un número mayor de veces el mejor individuo de la simulación.

2.5 Diseño del Conjunto de Entrenamiento

En este punto se expone el modo de utilización del otro módulo de diseño de la aplicación, el de conjuntos de entrenamiento. Como en el caso del módulo de diseño de arquitecturas, este módulo puede funcionar de forma cooperativa con el resto de módulos del sistema o de forma independiente, que es el tipo de funcionamiento que se comenta en este punto. Este módulo se denomina Conj_entr.exe en el grupo de programas de la aplicación.

Los pasos a seguir para la selección de un conjunto de entrenamiento óptimo son los siguientes: primero es necesario escoger el conjunto de entrenamiento que se quiere utilizar. Esto se realiza en la primera pantalla de la aplicación (Figura 2.28) pulsando los botones del navegador de la BD. Los conjuntos seleccionados se visualizan en el gráfico

de la parte superior de la ventana. Una vez seleccionado el conjunto de entrenamiento se pulsa el botón "AG" para pasar a realizar la configuración de los parámetros del AG del nivel de diseño. Después se configuran los parámetros de las RR.NN.AA. que se utilizan para valorar los conjuntos de entrenamiento y, por último, se realiza la simulación del AG para conseguir el conjunto de entrenamiento óptimo para el entrenamiento de una RNA que resuelva el problema planteado. Estos últimos pasos se comentan en los puntos siguientes.

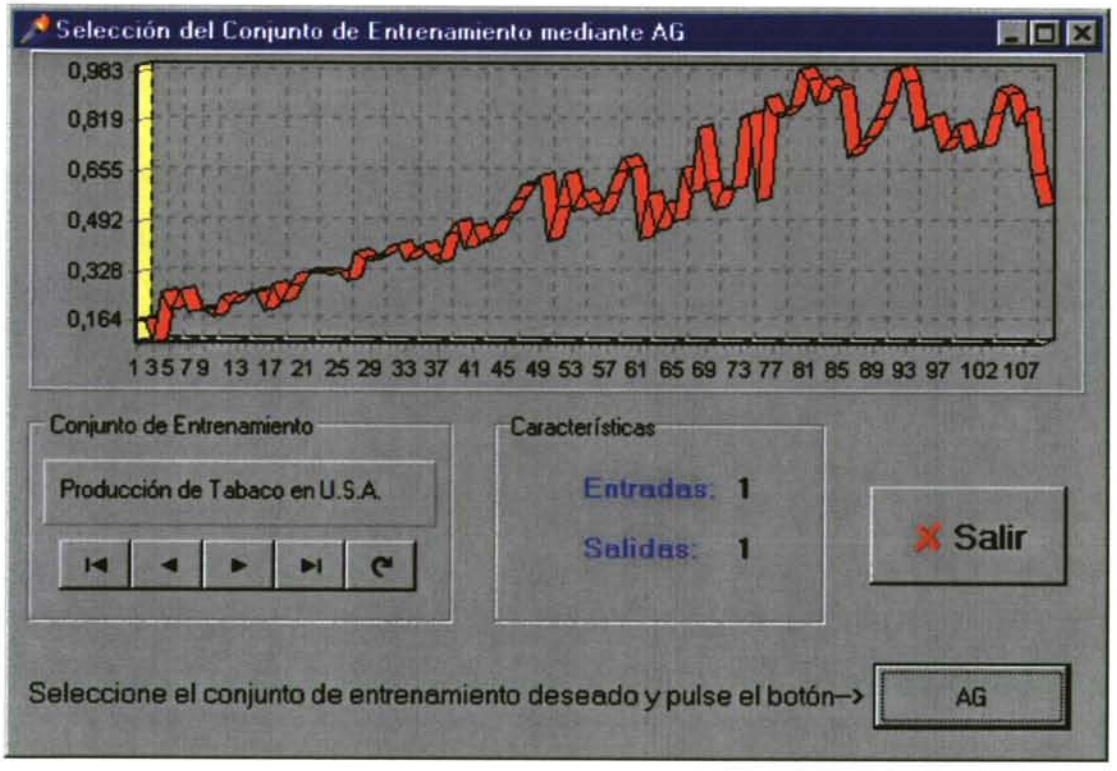


Figura 2.28.- Pantalla de selección del conjunto de entrenamiento

2.5.1 Configuración del AG de Selección del Conjunto de Entrenamiento

Este punto es el siguiente en la consecución del conjunto de entrenamiento óptimo. Al igual que en el caso del diseño de arquitecturas, el AG de selección de conjuntos de entrenamiento tiene como parámetros el tamaño de la población, el ratio de cruces y mutaciones y el número de iteraciones que se va a iterar el AG de entrenamiento para evaluar los distintos conjuntos de entrenamiento (Figura 2.29).

El campo denominado "Mejorar" se refiere al número de iteraciones que se puede simular una RNA utilizando el conjunto de entrenamiento de mejor adaptación al

pulsar el botón "Iterar Mejor" que aparece en esta pantalla al comenzar la simulación. Su comportamiento es igual al comentado en el punto de desarrollo de arquitecturas. El campo de "Grupos" se refiere al tamaño de las subseries y el de "Ciclos" es el número de iteraciones del AG del nivel de selección.

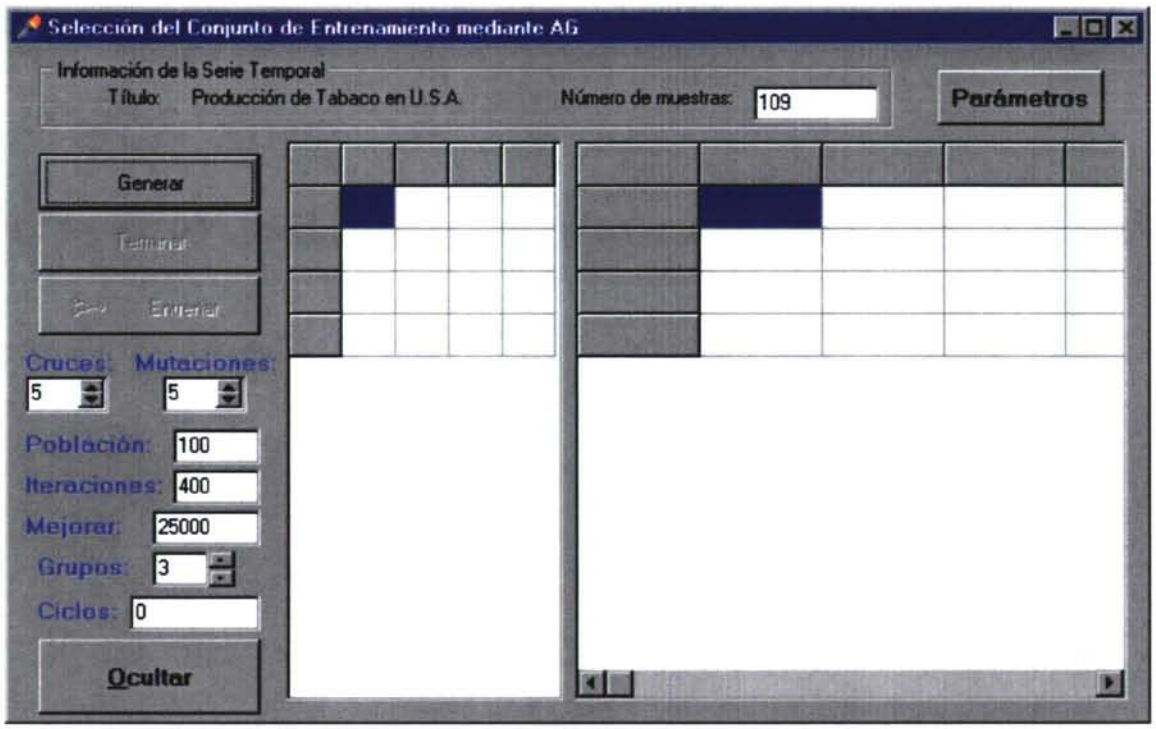


Figura 2.29.- Pantalla de configuración de parámetros del AG de selección de conjuntos de entrenamiento

Una vez configurados los parámetros del AG del nivel de selección sólo queda, antes de la simulación, configurar los parámetros de las RR.NN.AA. que se van a entrenar para conseguir el nivel de ajuste de los distintos conjuntos de entrenamiento. Para ello se pulsa el botón "Parámetros" de la pantalla mostrada en la Figura 2.29. Aparece así una ventana de configuración de RR.NN.AA. muy similar a la que se describe en el punto 2.3 de este capítulo. En esta ventana se configuran, por un lado, los parámetros de las RR.NN.AA. en una pestaña como la que se muestra en la Figura 2.10 y, por otro, los parámetros del AG de entrenamiento en otra pestaña como la mostrada en la Figura 2.13. Una vez configurados estos parámetros se pulsa el botón "Generar" para inicializar la población del AG y, después, el botón "Entrenar" para comenzar la simulación.

2.5.2 Evolución del Proceso de Selección

Una vez comenzada la simulación, la evolución del nivel de error se puede seguir en la misma pantalla comentada en el punto anterior (Figura 2.30). Se muestran dos tablas, la de la izquierda indica el número y posición de las subseries dentro de la serie temporal y la tabla derecha muestra los errores para cada individuo. En concreto, se muestran cuatro errores, los errores cuadrático medio y medio, tanto de las RR.NN.AA. entrenadas con los conjuntos de entrenamiento que representan los individuos, que es a lo que se refiere en la pantalla como errores de entrenamiento "E Cuadr tr" y "E Medio tr", como los errores de esas mismas RR.NN.AA. ya entrenadas al realizar la predicción sobre la serie temporal completa, que es lo que se refiere como errores de test "E Cuadr ts" y "E Medio ts".

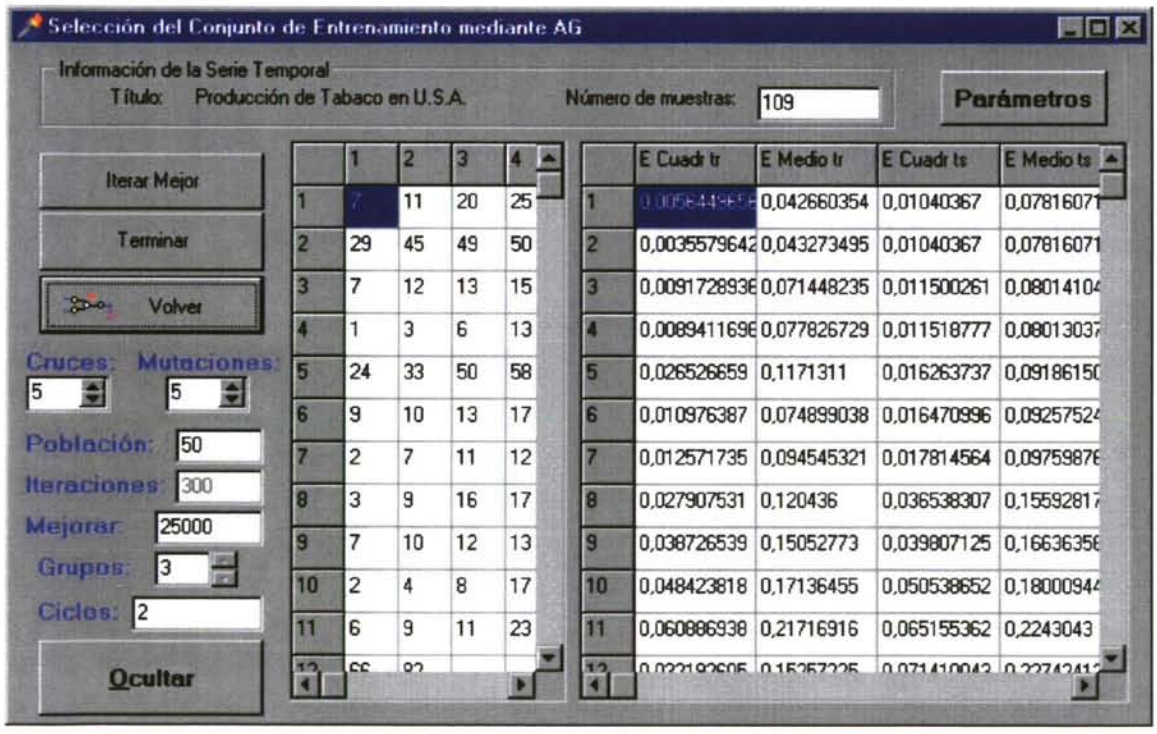


Figura 2.30.- Evolución del AG de selección de conjuntos de entrenamiento

Como en el caso del desarrollo de arquitecturas es posible realizar un entrenamiento con un número de ciclos mayor, especificando el número de ciclos en el campo "Mejorar", para tener una idea más precisa de cuál es el nivel de error de una RNA completamente entrenada. Para realizar esto, es preciso detener el proceso de entrenamiento pulsando el botón "Pausa" (que es el botón "Volver" de la Figura 2.30 mientras se está simulando el AG). Una vez pulsado el botón de "Pausa" este cambiará su etiqueta por "Volver" y se activará el botón "Iterar Mejor" en al parte superior. Al

pulsarlo se entrenará una RNA con el mejor individuo, el número de iteraciones especificadas y, al terminar, se mostrarán los errores conseguidos en la primera fila de la tabla de errores. También existe un botón para terminar la simulación que sólo puede ser pulsado mientras la simulación está en pausa.

2.6 Sistema Cooperativo Distribuido

Como último punto se va a comentar el funcionamiento del sistema cooperativo de desarrollo de RR.NN.AA. mediante AA.GG. realizando el ajuste de la arquitectura y la selección del conjunto de entrenamiento. Para utilizar el sistema completo se necesita un mínimo de tres computadoras conectadas en red mediante el protocolo TCP/IP. En una de las computadoras hay que ejecutar el programa denominado Central.exe que es el que se encarga de coordinar las comunicaciones entre los módulos de simulación tanto de arquitecturas como de conjuntos de entrenamiento. Después hay que lanzar cada uno de los dos módulos de simulación en, al menos, una computadora. No es necesario tener el mismo número de computadoras corriendo cada uno de los módulos.

Una vez arrancados los distintos programas en las distintas computadoras se ponen en comunicación los distintos módulos, se escoge el conjunto de entrenamiento, se configuran los parámetros de RR.NN.AA. y AA.GG. y se comienza la simulación de los módulos en modo cooperativo. Estas tres tareas son las que se comentan en los siguientes puntos. Por último, las tareas de almacenamiento de la mejor RNA conseguida, pruebas con valores de test y la ejecución de la red frente a nuevos datos se realizan de la misma forma que en los puntos 2.3.5, 2.3.6 y 2.3.7 referidos al entrenamiento básico de RR.NN.AA.

2.6.1 Conexión de Equipos

En el módulo Central no se realiza simulación alguna, sino que se coordina el intercambio de información entre los módulos de simulación: los de desarrollo de arquitecturas y los de selección del conjunto de entrenamiento.

La apariencia de este módulo es similar al de arquitecturas salvo que, en el módulo Central, existe una nueva pestaña pestaña denominada "Comunicación" (Figura 2.31) que permite seleccionar las computadoras que van a participar en el sistema.

Previamente estas computadoras tienen que tener lanzado uno de los dos tipos de módulos de simulación: de arquitecturas o de conjuntos de entrenamiento. Una vez en esta ventana se añaden a la lista "Equipos a Conectar" las computadoras haciendo doble click con el ratón en la lista de "Equipos de la Red de Computadoras". Para incluir un nuevo equipo en esta lista se pulsa el botón "Nuevo" y para borrarlo el botón "Eliminar".

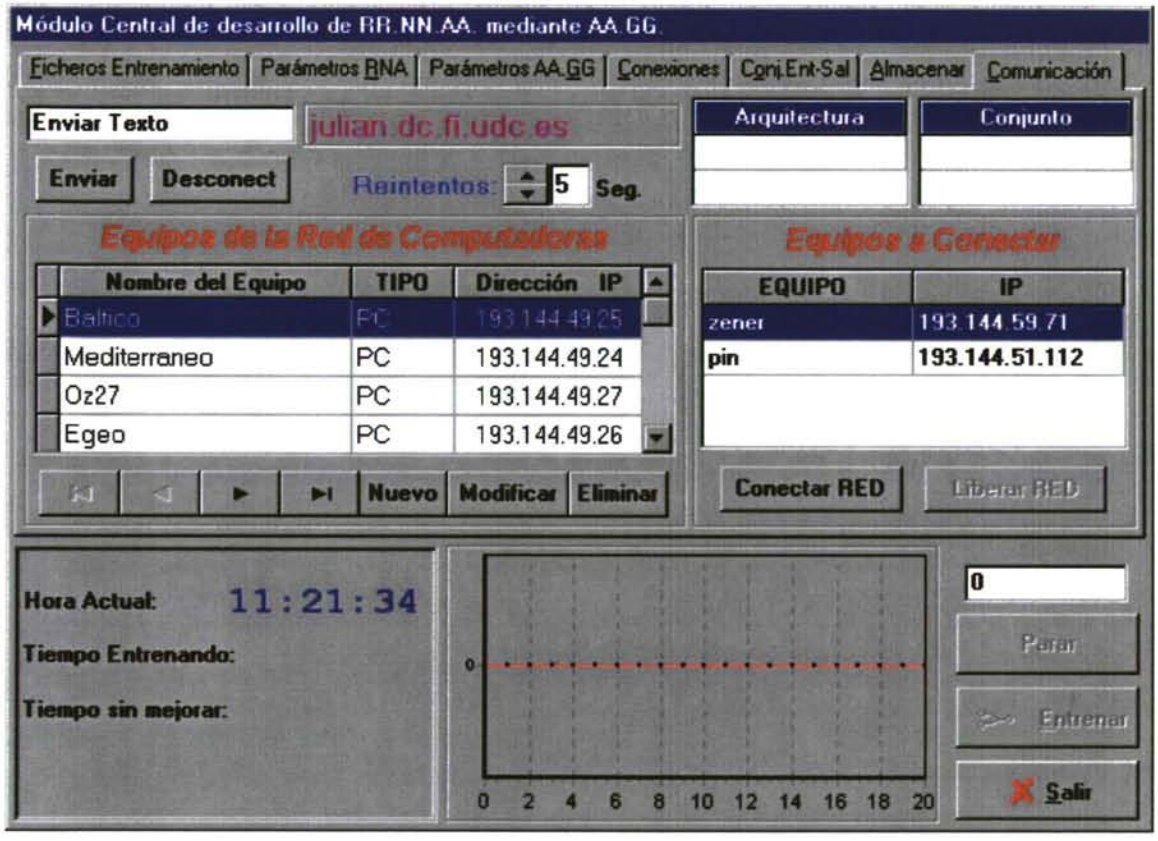


Figura 2.31.- Especificación de computadoras del módulo central de comunicaciones

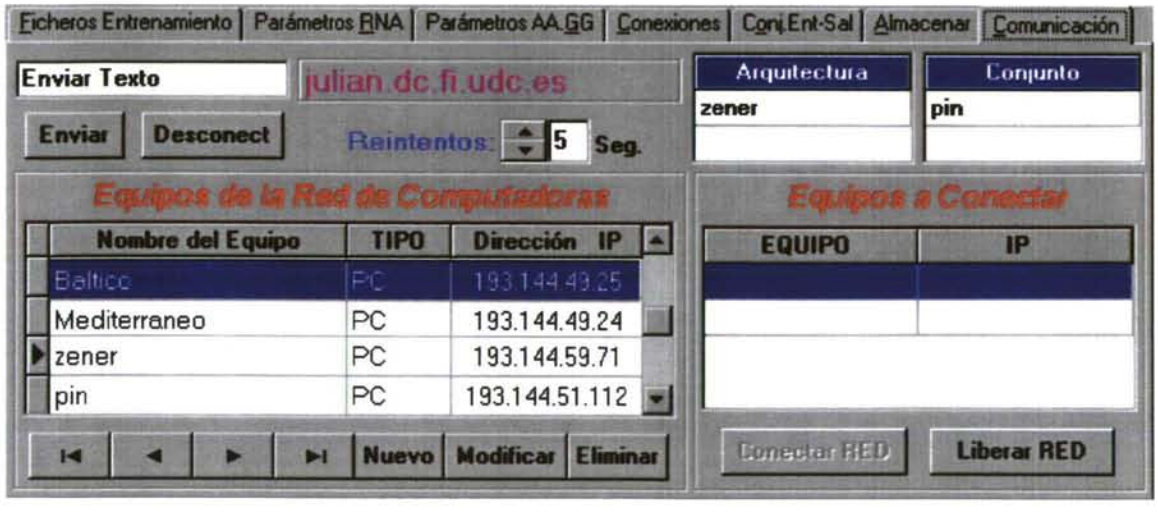


Figura 2.32.- Detalle de la pestaña Comunicación después de comenzar las simulaciones

Cuando se inicia la simulación, en la parte superior de la ventana se indica qué equipos están realizando la simulación del módulo de arquitecturas y cuáles están realizando la simulación del módulo de conjuntos de entrenamiento (Figura 2.32). Que realicen una u otra simulación depende del programa que tenga lanzado el equipo en el momento en que el módulo Central contacta con él.

2.6.2 Configuración de Módulos de Diseño

Una vez seleccionadas las computadoras que van a realizar el proceso de desarrollo de las RR.NN.AA. hay que configurar los parámetros, tanto de las RR.NN.AA. como de los AA.GG. que se van a utilizar en el sistema. Estas tareas de configuración se realizan ahora en el módulo Central, tanto para los módulos de arquitecturas como para el de conjuntos de entrenamiento y, una vez que comienza la simulación, el módulo Central se encarga de hacer llegar estos datos a los módulos de todos los equipos. El primer paso es escoger el conjunto de entrenamiento que se va a utilizar, esto se realiza en el módulo Central una única vez por simulación y todos los módulos utilizan en un primer momento este conjunto de entrenamiento.

La pestaña de configuración de las RR.NN.AA. es idéntica a la mostrada en la Figura 2.10 y es la que recoge los parámetros de las RR.NN.AA. que se utilizan en la valoración del nivel de ajuste, tanto de arquitecturas como de conjuntos de entrenamiento. La pestaña de configuración de AA.GG. es idéntica a la mostrada en la Figura 2.25 y aquí se configuran los distintos parámetros de los dos tipos de AA.GG. utilizados en el sistema: el del nivel básico para el entrenamiento de RR.NN.AA. y los del nivel de diseño para el desarrollo de arquitecturas y el ajuste del conjunto de entrenamiento. Como ya se mencionó en la explicación de los módulos individuales, también existe la posibilidad de utilizar perfiles para facilitar la reutilización de valores para los parámetros de las RR.NN.AA. y de los AA.GG.

2.6.3 Evolución del Proceso de Entrenamiento

Una vez están configurados todos los parámetros y seleccionado el conjunto de entrenamiento sólo resta iniciar la simulación pulsando el botón "Conectar Red" de la pestaña Comunicación. En ese momento el módulo Central transfiere a cada uno de los

equipos conectados la información de configuración sobre RR.NN.AA. y AA.GG. para que inicialicen sus poblaciones y comiencen la simulación.

Una vez que la simulación está en marcha no es posible modificar opciones en los programas de los módulos de simulación, que ceden el control al módulo Central. Sólo es posible utilizar y cerrar estos programas una vez se le indique al módulo Central que la simulación ha terminado. Las mejoras que consigue cada módulo de simulación son enviadas al módulo Central para que las remita al resto de los módulos y se utilicen las mejoras conseguidas. Aprovechando que toda esta información pasa por el módulo Central, es aquí donde se visualizan los mejores individuos y los niveles de error que consiguen, tanto los módulos de arquitecturas como los de conjuntos de entrenamiento.

Por un lado, la pestaña "Conexiones" del módulo Central es igual a la comentada a partir de la Figura 2.26 dentro del punto 2.4.2 y, en ella, se puede estudiar cuál es la arquitectura que, en cada momento de la simulación, presenta una mejor adaptación. De cada arquitectura se visualizan los valores de las funciones de activación y de los pesos (ver punto 2.4.2 para una mayor explicación).

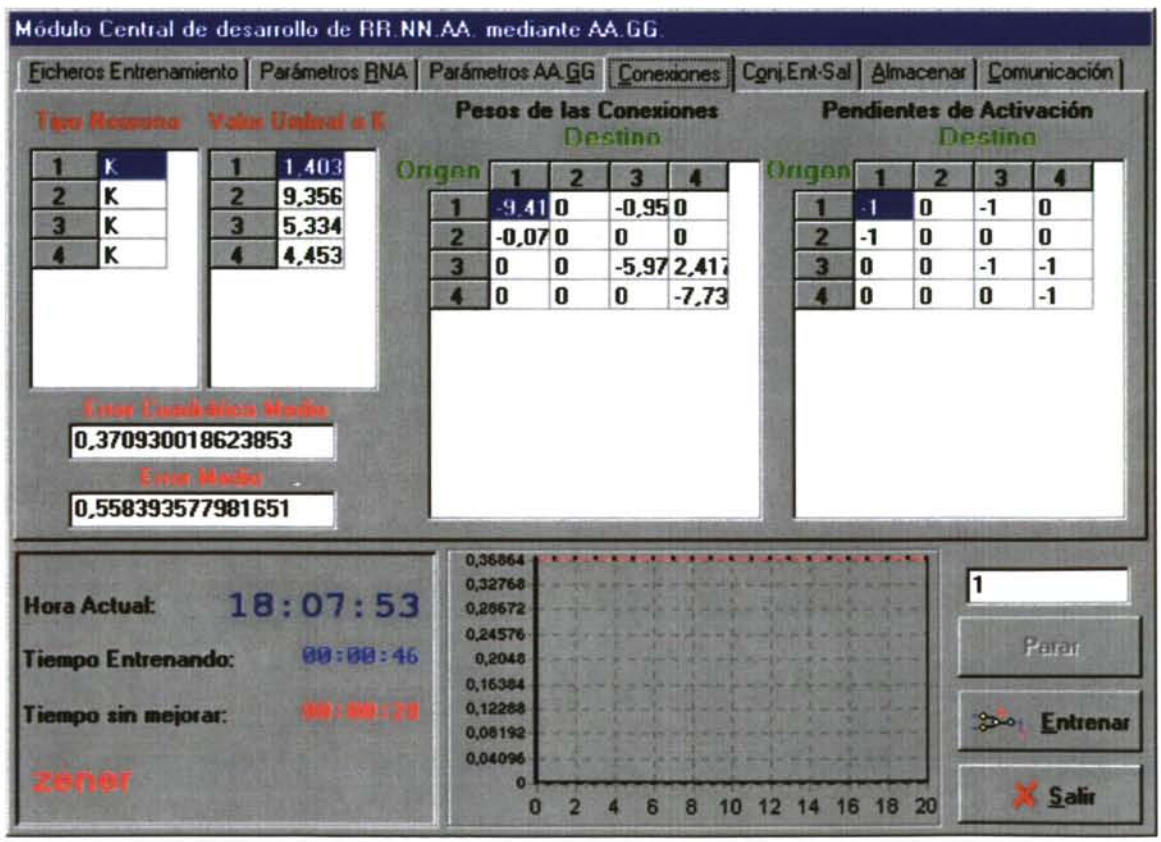


Figura 2.33.- Pestaña para el seguimiento de resultados de los módulos de arquitecturas

En la pestaña "Conj. Entr-Sal" del módulo Central se muestra el mejor individuo de los módulos de conjuntos de entrenamiento. En este caso se visualiza el número de subseries de que se compone el individuo, qué posiciones representa y el nivel de error alcanzado.

Figura 2.34.- Pestaña para el seguimiento de resultados de los módulos de conjuntos de entrenamiento

Cada vez que se consigue una mejora en uno de los módulos que se ejecutan en las computadoras de la red, esta información se visualiza en estas dos pantallas del módulo Central pudiendo detener el entrenamiento en todos los módulos pulsando el botón "Parar" de la parte inferior de la ventana. Para continuar con el entrenamiento hay que pulsar el botón "Entrenar" y, si se pulsa el botón "Salir" se cierra la aplicación Central y se libera el control sobre los módulos de la red.