Artificial Neural Networks Manipulation Server: Research on the Integration of Databases and Artificial Neural Networks

A. Santos, B. Arcay, J. Dorado, A.B. Rodríguez and A. Pazos
Department of Communications and Information Technology, School of Computer Science, A Coruña University, A Coruña, Spain

This paper proposes a new whole and distributed integration approach between Artificial Neural Networks (ANNs) and Databases (DBs) taking into account the different stages of the former's lifecycle (training, test and running). The integration architecture which has been developed consists of an ANN Manipulation Server (AMS) based on a client-server approach, which improves the ANNs' manipulation and experimentation capabilities considerably, and also those of their training and test sets, together with their modular reuse among possibly remote applications. Moreover, the chances of integrating ANNs and DBs are analysed, proposing a new level of integration which improves the integration features considerably. This level has not been contemplated yet at full reach in any of the commercial or experimental tools analysed up to the present date. Finally, the application of the integration architecture which has been developed to the specific domain of Environmental Impact Assessments (EIAs) is studied. Thus, the versatility and efficacy of that architecture for developing ANNs is tested. The enormous complexity of the functioning of the patterns which rule the environment's behaviour, and the great number of variables involved, make it the ideal domain for experimenting on the application of ANNs together with DBs.

Keywords: Artificial neural networks; Databases, Artificial neural networks manipulator server; Client-server architectures; Environmental impact assessment; Hybrid systems

1. Introduction

If, as a starting point, we take the statement 'Algorithms are not important, the important thing are the data given to the ANN. The problem must be tackled from the point of view of data and their representation', finding a connection between ANNs and DBs seems to be easy [1].

Although many of the developments carried out with ANNs use DBs, in most cases this fact is not stressed. Moreover, we are not aware of the existence of references which have studied this subject deeply and extensively, either directly or as a result of the research done.

After a thorough research of a set of ANNs representing many application domains and many connectionist tools of development and experimentation (both commercial and freely distributed), it has been noted that most of them present a monolithic and weakly structured character. Besides, they use imbibed codes or simple text files for representing the entities involved in the different stages of the ANNs' lifecycle.

The purpose of this paper is not to evaluate or count up the tools available for developing ANNs, but to study the features and flaws in the integration of present systems, proposing ideal integration configurations and conditions which reflect the real needs of this kind of application. References for the

systems analysed are not included, since in some instances they are commercially-distributed tools.

The research carried out has analysed the integration levels with general-purpose commercial connectionist 4-tool DBs (listed with letters A, B, C and D), six non-commercial ones (E to J) and 13 ANNs applied to different specific domains (K to W). Table 1 shows the degree of integration with DBs of these three groups of tools in the following regard:

- The representation of the ANNs' architectures and the processes involved (*architecture* column).
- The representation of the training sets (*training* column).
- The representation of the test sets (*testing* column).
- The representation of the flows towards and from the ANNs in running time (*running* column).

Another column is included, labelled *no integration*, which stands for those tools which do not include any kind of integration with DBs in each and every one of the stages of the ANNs' lifecycle. Thus, it may be noted how connectionist tools C and D offer a certain degree of integration with DBs in the training and test stages, although they do not include that support with regard to their architectures and running process. Of all the 23 systems studied, none presents an all-level integration degree with DBs. A great number of tools (A, B, E, H, K, N) offer no integration with DBs at all.

The research carried out analyses, according to the 'divide and conquer' principle, the capabilities and advantages of integration with DBs of each stage of the ANNs' lifecycle (*architectures, training, test* and *running*) in order to unify all the results into one integrated approach.

Finally, the resulting integrating approach is applied and analysed in the EIAs domain. The incorporation of connectionist systems to conventional EIA applications will facilitate the treatment of new cases, incomplete or imprecise data, etc. The variability of the information used (different environments, projects, actions, affected Environmental Factors (EFs), etc.) advises the use of connectionist approaches. The subjective character of the variables used, together with the lack of general experimental and theoretical knowledge on environmental interactions, supports this idea. Although three ANNs which separately tackle different problems within the domain have been developed, the simplest of them is reviewed in this paper with the purpose of explaining the functioning of the proposed approach clearly. This ANN tackles the problem of the definition of impact identifiers, which allow the determination of the impacts produced by the actions of a project on the EFs which are a feature of our environment.

2. Integration of Connectionist Architectures

DBs offer great versatility when experimenting with various architectures, learning algorithms, process element models, etc. They also support weights, thresholds, outputs, error rates, etc., of trained or training ANNs.

The integration between ANNs and DBs at this level will give rise to the following advantages:

- It facilities the analysis or parametric estimation of most aspects which may influence the ANNs' learning, performance, generalisation capacity, etc.
- It offers the system great versatility for the study and optimisation of ANNs.
- DBs offer ANNs a high degree of integrity and consistency. These aspects are, according to Codd

Table 1. Study on the integration levels between ANNs and DBs.

Level Tools	Architecture	Training	Testing	Running	No integration
Tools General Purpose Commercial [A..D] (4)		C, D	C, D		A, B
Tools General Purpose Non commercial [E..J] (6)		I, J	I, J		E, F, G, H
ANNs in specific application domains [K..W] (13)	O, P	V, W	V, W	O, P, Q, R, S, T, U	K, L, M, N

and Ceri [2–4], intrinsic to DBs, improving the research and experimentation potential of the ANN domain considerably.
- It enables the integration of security levels and the whole potential of query languages, aspects of which are analysed by Codd and Cardenas [5,6] as elements of DBs and DB Management Systems within the connectionist domain.
- It facilitates the manipulation of extended ANNs or connectionist hierarchies with many architectures and models.
- It will enable the improvement of experimentation, design and integration possibilities of general-purpose connectionist tools and ANN-based applications.

The proposed integration approach is based on the ANNs Manipulation Server (AMS), which is independent from those client-systems which demand connectionist resources. The AMS stores the ANNs into a DB to be used during the training, testing and running stages. This DB will support the various ANNs generated by the system, thus including a whole hierarchy of connectionist systems. Besides, the AMS allows different applications, both local and remote, to use the ANNs generated. Therefore, these ANNs are developed from a client-server architecture.

The AMS architecture has been designed according to its application possibilities in a wide range of domains. The needs of connectionist schemes in bigger domains have also been considered. This kind of development is common in knowledge engineering. These domains can usually be divided into subproblems, which may be tackled by applying several connectionist approaches separately. The AMS must facilitate the integration of all of them into a single co-operative environment, which enables an approach to the domain as a whole. It must also facilitate the integration with other approaches or techniques. This is the case of the EIA's application domain, where the variability of the information and the great number of variables involved makes the application of multiple ANNs the easiest way to recognise certain situations. On the other hand, the AMS must be just as efficient in small, well defined domains. Figure 1 shows the system's higher-order architecture.

This architecture differentiates three elements which are distributed in a data network, and which are part of one or more DBs, either local or remote from each element:

- The first element or AMS is in charge of storing and managing all the indexed ANNs in the connectionist DB designed and developed for that purpose. This DB is considered to be local to the AMS, for reasons of the system's performance and complexity.
- The second element, accessible by the AMS, consists of the DB servers of the various application domains involved.
- The third element consists of the client-applications, which are able to call the first element's AMS or connectionist level at any time. From a functional point of view, it is formed by the connectionist systems at the running time.

The systems running control is focused on the client-applications (element #3), which may, at certain steps of its running, call the AMS or transfer control to it (element #1). This assigns and executes the

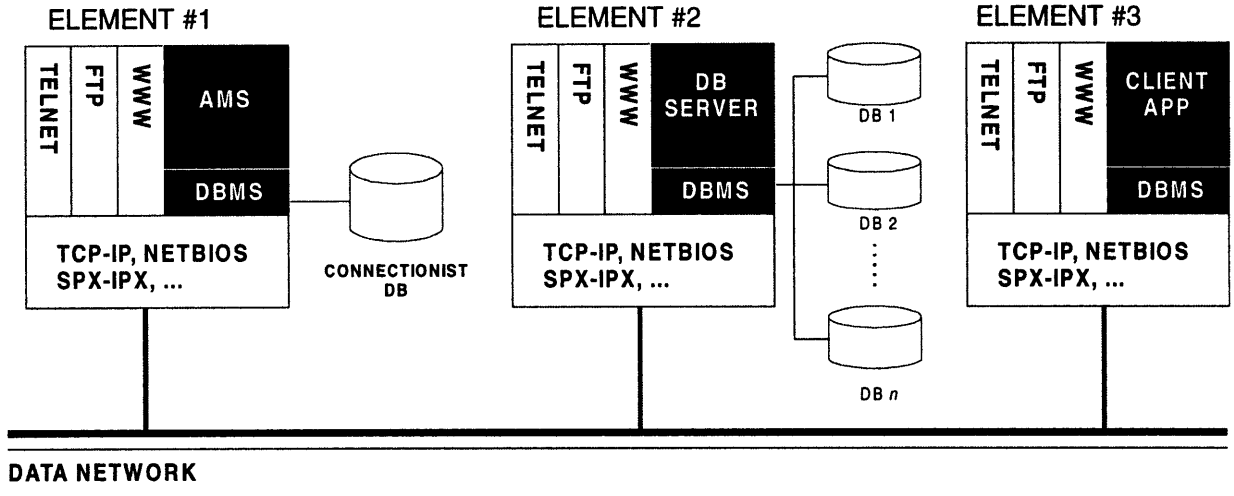

Fig. 1. AMS higher-order architecture and elements involved.

relevant ANNs according to the point from which the call comes. Thus, the connectionist level of client-applications becomes independent, facilitating the integration of ANNs with other approaches which may be part of client-applications. That is, a connectionist learning level is made available to any kind of system which may require it due to its features, and which has a sufficient amount of data or patterns. In this respect, the AMS offers a connectionist level which may be easily incorporated into the user's final applications. Moreover, application programmers need not worry about all the innate tasks of connectionist systems, since the AMS makes them available in a visual and simple way.

3. The Connectionist DB Model

Figure 1 shows how the AMS incorporates a DB which supports the entities corresponding to the three elements of the architecture. Therefore, the DB research or modelling must cover every aspect related to the DB servers of the specific application domains, architectures and ANN models, activation functions, learning rules and methods, training and test sets, client-applications, etc.

The following entities may be identified in the analysis of the ANN domain:

- Associated architecture and parameters (ARCHITECTURE entity).
- Layers (LEVELLAYER).
- Weights (WEIGHTS).
- Activation functions (ACTIVATIONFUNCTION).
- Learning rules (LEARNINGRULE).
- Client applications (CLIENTAPP).
- Client-applications forms (FORMSAPP).
- Client-applications and DB servers addresses (NETWORKADDRESS).
- Network-directing schemes or protocols (NETWORKPROTOCOL).
- Training and test sets queries (QUERYTRAININGTEST).
- DBs used for ANN test and training (DATABASES).
- Application domains DB tables (TABLES).

Figure 2 shows the connectionist DB model which supports the information handled by the AMS. The entities attributes have been left out due to lack of space.

The entities ARCHITECTURE, LEVELLAYER, WEIGHTS, ACTIVATIONFUNCTION and LEARNING RULE model the structures and parameters of the ANNs which correspond to element #1 of the architecture (Fig. 1). The AMS modular scheme allows the integration of different models of feed-forward and feedback ANNs using these entities, together with their architectures, learning rules, activation functions, etc. The corresponding table of the ARCHITECTURE entity will reflect the models and features of all the system's specific

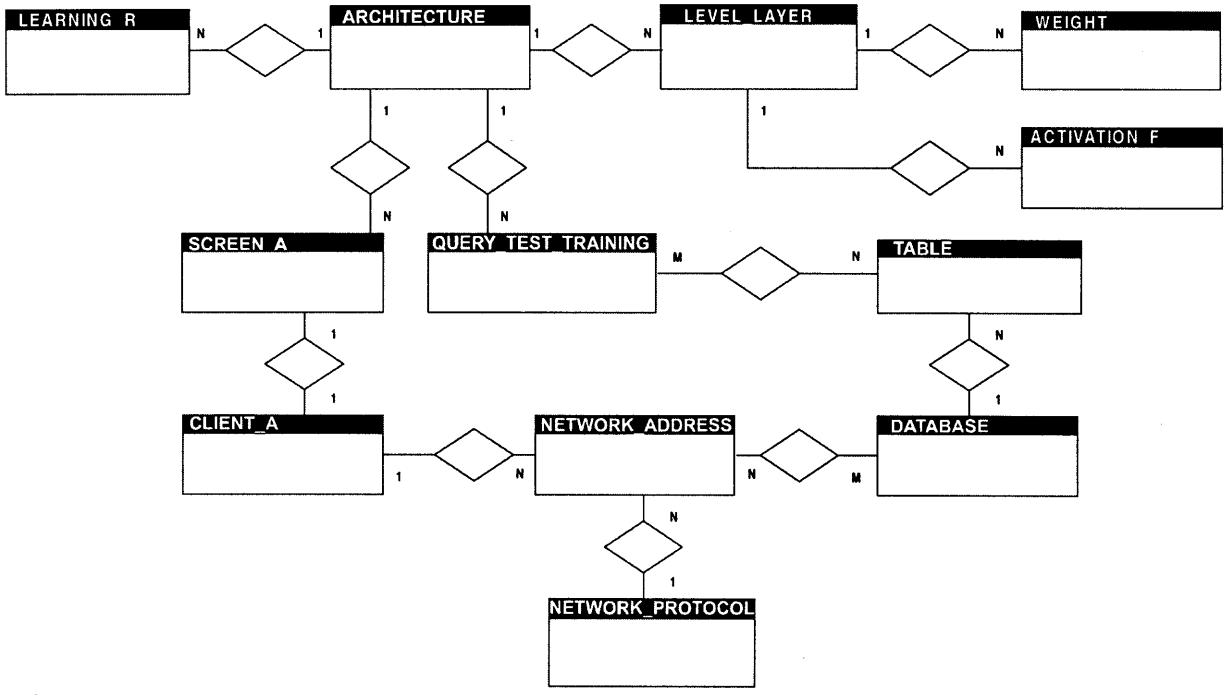

Fig. 2. Model of the AMS connectionist DB.

ANNs. This table will incorporate the various ANNs developed in several application domains. In addition, the AMS includes a set of adjustment approaches for the selection of parameters such as the learning rate or the momentum which facilitate ANN training.

This approach initially considers an update of the synaptic weights by epochs, and not in each pattern, and then undertakes a search for the right learning rate and momentum. For adjustment of the learning rate, we have chosen a scheme which consists of a progressive decrease of the learning rate throughout the learning process [7]. On the other hand, the AMS also incorporates a scheme, proposed by Silva and Almeida, based on the independent assignment and adjustment of different learning rates. Thus, a different rate will be learning rate will be assigned to each of the ANN's synaptic weights [8]. The use of multiple learning rates allows the achievement of an optimal rate for each synaptic weight. This scheme also introduces various turn-back strategies for those cases in which an error increase takes place, with the purpose of avoiding local minima, or for instances in which we are not following the right strategy.

This kind of approach, together with others which may be incorporated into future AMS versions, such as the use of evolutionary strategies [9–11] or genetic algorithms [12–17] for architecture design and optimisation, will further increase the chances of automating ANN integration at all levels and for every type of user.

In turn, the tables corresponding to LEVEL-LAYER, WEIGHTS, ACTIVATIONFUNCTION and LEARNINGRULE entities reflect the layers, synaptic weights, activation functions and learning algorithms of the different ANNs of the system. The FORMSAPP, CLIENTAPP, NETWORKADDRESS and NETWORKPROTOCOL entities support the relationships between the ANNs of the AMS and the specific forms of client-applications at a conceptual level. Therefore, they model the structures and parameters corresponding to element #3 of the general architecture in Fig. 1. The corresponding tables will characterise the features of the client-applications forms, including their network addresses and protocols supported in the case of remote applications which may require the AMS's ANNs to run. When it is running, a client-application will call the AMS, passing on the client-application's identifier and the identifier of the form which has generated that call as parameters. Thus, any call to the AMS is perfectly defined by two identifiers: the client-application which calls; and the form of that application which generates the call. As a result, the AMS sends back the identifiers of the ANNs which may be relevant to the client-application at that time, considering the point or call form. The following stage would be that of running those ANNs identified as relevant, sending the results back to the client-application.

The QUERYTRAININGTEST, TABLES and DATABASES entities are used for the process of generating training and test sets. That is, they model element#2 of the Fig. 1 architecture. The corresponding tables will refer to the tables or queries to be used for the ANNs' training and test processes. These tables or queries may belong both to local and remote AMS's DBs. This process, which constitutes another level of integration between ANNs and DBs, will be studied separately in Section 4.

Finally, it would be interesting to stress the advantages of using DBs in any system which requires information treatment, and which can, therefore, be applied to a greater or lesser extent to the different stages of the ANN lifecycle (the possibility of including rules and restrictions into the data model, the diminution of redundancy, a greater availability of the information, information protection, efficiency, definition of security levels, etc.).

4. Training and Test

This section will explore the AMS possibilities of automating the generation of multiple training and test sets through the use of logical operators, and relationships which will also facilitate any preprocessing of patterns that may be required. This section will focus mainly on element#2 of the AMS architecture (Fig. 1).

The AMS, through its DB:

- facilitates the extraction of sufficiently big and significant fact sets;
- improves the test stage considerably, facilitating the manipulation of multiple ANNs and the automatic generation of their corresponding test sets;
- the training and test sets will always reflect the real situation of the application domain DB:
- it incorporates a high degree of integrity, consistency and security to the training and test sets;
- it places the distributed training and test sets throughout a computer network, making them available to the AMS as if they were situated at the local host.

For instance, in a feed-forward architecture, the AMS uses the relational sentences in order to compose the ANN's information or input layer patterns. The possibility of using complex relational sentences

on both local and remote DBs enormously increases the capacity of generating and preprocessing training sets. The AMS will be able to assign various table domains belonging to any DB server of the researched application domain to the corresponding process elements of the studied ANN input and output layers. In addition, domains resulting from views or queries related to their assignment to the process elements can be used, thus improving the capabilities and possibilities in the generation of training and test sets.

It must be noted that we are always referring to two different DBs, the one maintained by the AMS to support the ANNs, and the other maintained by the corresponding DB server as support to the training and test process of the application domain involved. The process of generating the training and test sets considers the assignment or mapping among different DB fields of the researched application domain and the process elements of the corresponding DB. This task carries out a JOIN operation among the DB tables involved in the learning process. Later on, a projection is carried out on the relevant fields which constitute the training or definition sets of the patterns to be learned.

These queries may produce three well-differentiated types of fields: those used as ANN inputs, those corresponding to the desired outputs (in the case of a scheme based on a supervised learning paradigm), and finally, the semantic descriptions of the previous fields. This third type of field allows both the AMS and the client-applications to manipulate concepts or descriptions instead of simple codes. On the other hand, the second type of field would disappear when the AMS considers a non-supervised learning scheme at a given time. Figure 3 shows the generation scheme for the training and test sets. It can be seen how a query on the application domain DB generates a set of fields which shape the training set. The domains have been labelled and related to a generic ANN architecture. Thus, fields labelled 1 constitute the ANN input patterns, and those labelled 3 constitute the desired outputs, assuming a supervised training scheme. The fields labelled 2 and 4 will be the semantic descriptions of fields 1 and 3, respectively. As one can easily imagine, both the client-applications and the DBs use fields 2 and 4 for their interactions with the users and ANN managers. That is, while fields 1 and 3, used for the training process, are merely numeric fields without meaning, fields 2 and 4 provide that meaning. For instance, one of the fields labelled 1 could be an identification code for a certain product. This code as such would provide no information to the user. In this sense, a field labelled 2 *can contain* descriptions of those product codes as an element of user-interaction. As has already been seen, the AMS's DB must contain every kind of information about the server involved and the corresponding application domain DB: its queries, tables, and the type of resulting field (input, output or semantic).

Thus, the AMS automates the generation process of the training and test sets, which will always be updated. The AMS will also be able to incorporate various preprocessing procedures through the use of logical and relational operators of the DB's query language.

5. Integration in Running Time

At this point, the core for handling and manipulating ANNs is available. Nevertheless, the problem is the

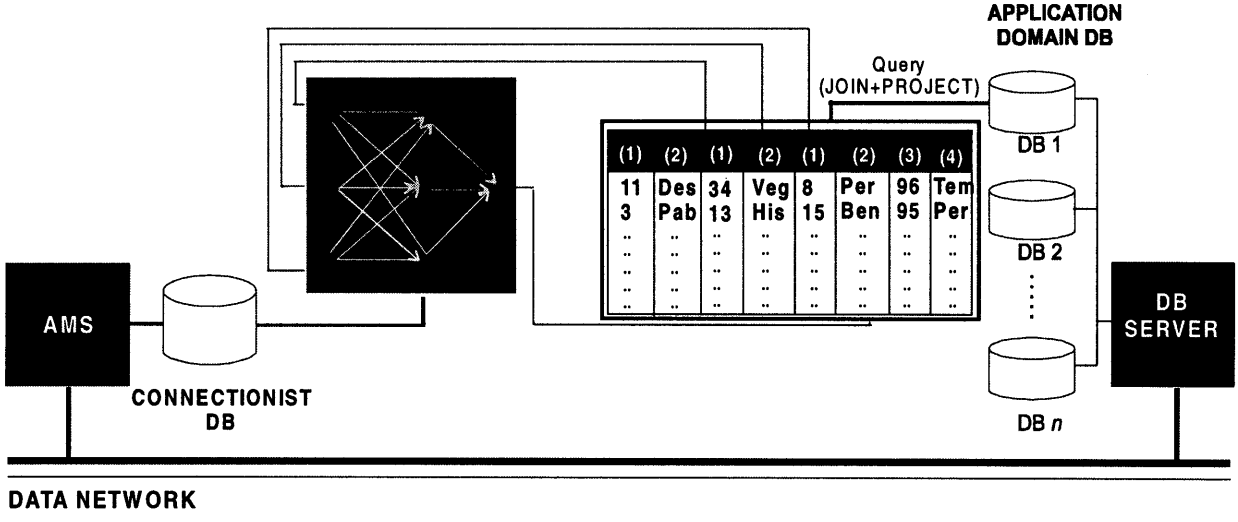

Fig. 3. Generation of the training and test sets.

reuse of the various trained ANNs in any application, either developed or to be developed (element#3 of the architecture).

A scheme based on the assignment of ANNs to client-applications screens or forms is included, in order to facilitate the integration of ANNs with other approaches. This relationship is expressed by means of the CLIENTAPP, FORMSAPP and ARCHITECTURE entities, as may be seen in Fig. 2, which determine those ANNs that can be applied according to the information available in the forms of the client-applications in running time. Similarly, the data from those forms may be processed by using other techniques, turning the forms into the integrating elements of the various approaches. Generally speaking, a set of ANNs may be assigned to the forms of the various client-applications, and vice versa. Figure 4 shows the system's functioning and main information flows.

From the connectionist DB, through the 1.1 information flow, the AMS obtains the characterisation of the various ANNs (architecture, weights, activation functions, learning rules, etc.), and the definition of their training set queries. The AMS includes a visual environment for the creation of these ANNs and the definition of the parameters which characterise their architectures, and of the training and test sets. Later on, the query is carried out against the application domain's DB server for the automated generation of the training set (flow 1.2). Once the ANN architecture and training set have been obtained, the learning stage will start. The test process follows the same steps as the learning process, the only difference being that, instead of using a query which generates a training set, that used generates a test set. Flows 2.1 and 2.2 of Fig. 4 stand (respectively) for the query on the connectionist DB of the ANN which is being tested, and the corresponding test set query. The AMS is structured in two modules, training and test modules, for carrying out these tasks. Although the connectionist DB is considered to be local to the AMS on the grounds of performance, the application domains' DBs could be either local or remote.

In the ANNs' running stage, a client-application demands the identifiers of the relevant ANNs through the 3.1 information flow. This application can be either local or remote with regard to the AMS and the application domain DB Server. For this purpose, the client-application's identifier and that of the corresponding form which generates the call are passed as parameters. Thus, the AMS identifies, through the connectionist DB, those ANNs which may be of interest to the present running stage of the client-application demanding connectionist resources (information flow 3.2). Later on, the AMS notifies those ANNs which are relevant to its process to the client-application, while the client-application is in charge of defining and notifying the AMS of the relevant ANN(s).

Finally, the client-application will pass on to the AMS those parameters required as inputs to the ANN researched, to go on with the running stage. The AMS will send back to the client-application, through the 3.4 information flow, the results obtained at the ANN running stage. Generally, all these AMS management processes include the typical tasks with DBs (inserts, deletes and updates).

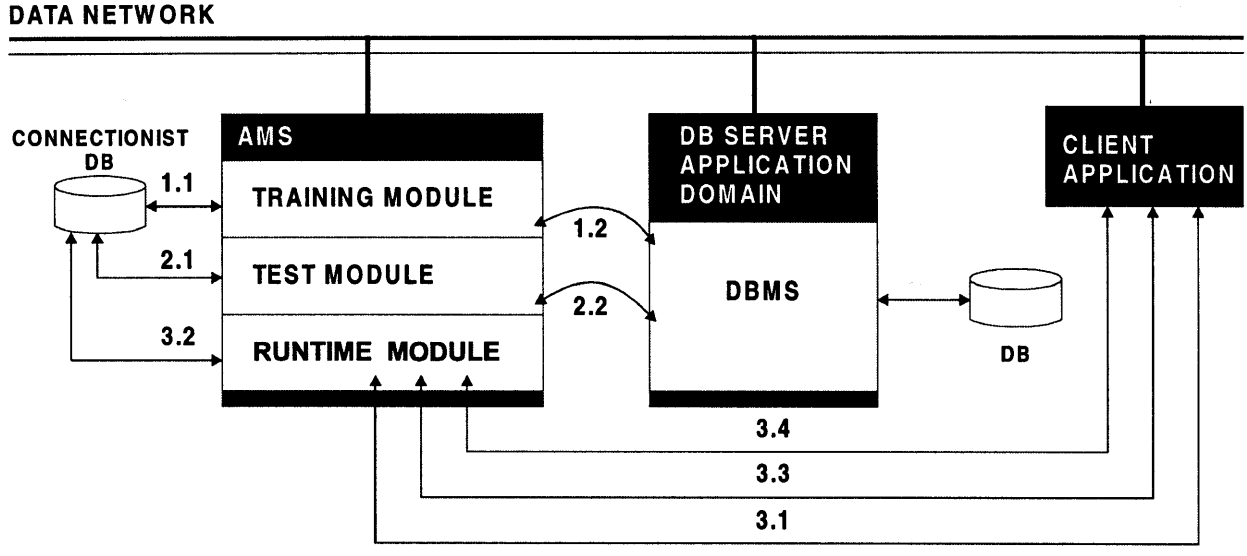

Fig. 4. System's functioning environment.

As the AMS's DB increases, a core of data and experiences with ANN design and modelling will be defined. This will allow us, in future, to apply the AMS scheme on its own DB, defining the connectionist level as the application domain to be studied. In this sense, the application domain's DB and the AMS's connectionist DB will become the same DB.

6. Results

Nowadays, there is increasing concern about the control of all those activities which may damage the environment. For this reason, the implementation of a correct Environmental Impact Assessment (EIA) will contribute to the rational integration of infrastructural and entrepreneurial projects, trying to make development and technological progress compatible with the necessary quality of environmental life. The progress made in recent years in the fields of computation and artificial intelligence makes highly sophisticated and powerful systems available which have become an essential tool when carrying out EIAs, due to the great complexity of this application domain.

A system has been implemented in the EIA field which integrates an ANN set into its reasoning scheme, in the framework of the research on DB and ANN integration. This system or client-application uses the connectionist level offered by the AMS. At an early stage, a DB is designed and developed into that application domain.

This DB includes a wide information set about:

- actions which produce impact;
- Environmental Factors (EFs);
- environmental crossovers between actions and EFs;
- indicators and functions to evaluate those impacts;
- involved environmental parameters;
- corrective actions;
- environmental legislation;
- The relationships among them and the possible restrictions [18,19].

The results and conclusions obtained by carrying out new EIAs are stored into this DB with the purpose of using that information as experience for future EIAs. The DB presented in Fig. 5 is the result of studying all the information involved in this application domain. The entities attributes have been left out due to lack of space.

It seems obvious that the integration of a connectionist level into any final application through the use of the AMS will totally or partially involve the corresponding application domain DB.

From a functional point of view, this client-application is in charge of characterising and evaluating the hypotheses linked to the impacts suggested from the environmental parameters suggested by all those ANNs considered relevant by the AMS. The AMS suggests the relevant ANN set according to the client-application's call point, and the information which may be available to those applications at that time. Finally, the results obtained by the ANNs will become part of the client-application.

From a structural point of view, the EIA system has been organised into five levels of abstraction:

- project level;
- action level – those which may cause some environmental impact;
- Impact level – which establishes cause–effect links between the project actions and the possible EFs which may be affected;
- EF level–which characterises the environment;
- The level of corrective measures which may decrease all those impacts that, for reasons of legal constraints or environmental advice, may inflict considerable damage on the environment.

Figure 6 shows the interaction among the various levels of abstraction of the system.

The system input is composed of the set of projects targeted by the EIA. These projects suggest a series of possible actions which cause impacts that, in turn, may suggest other actions. From the final set of actions, cause–effect links are established between these actions and the EFs affected. These links evaluate the impacts on the EFs from the characteristics of the impact and an impact indicator. The impact characteristics include its sign (beneficial, harmful or unknown), intensity, extension, momentum, duration, possibility of reconstructing the initial conditions or reversibility, and possibility of including corrective actions. Finally, various corrective actions are suggested and studied in order to reduce the environmental effects which, according to their features, may produce a strong impact, and/or whose parameters are above the threshold established by law. These measures are oriented both towards the improvement of the positive effects and the minimisation of the negative ones. This system will enable a classification of impacts, identifying those which cause a deeper impact on the environment, those which are inevitable and those which require constant monitoring [20–22].

The AMS functioning pattern with regard to this EIA system will be identical to that of any other

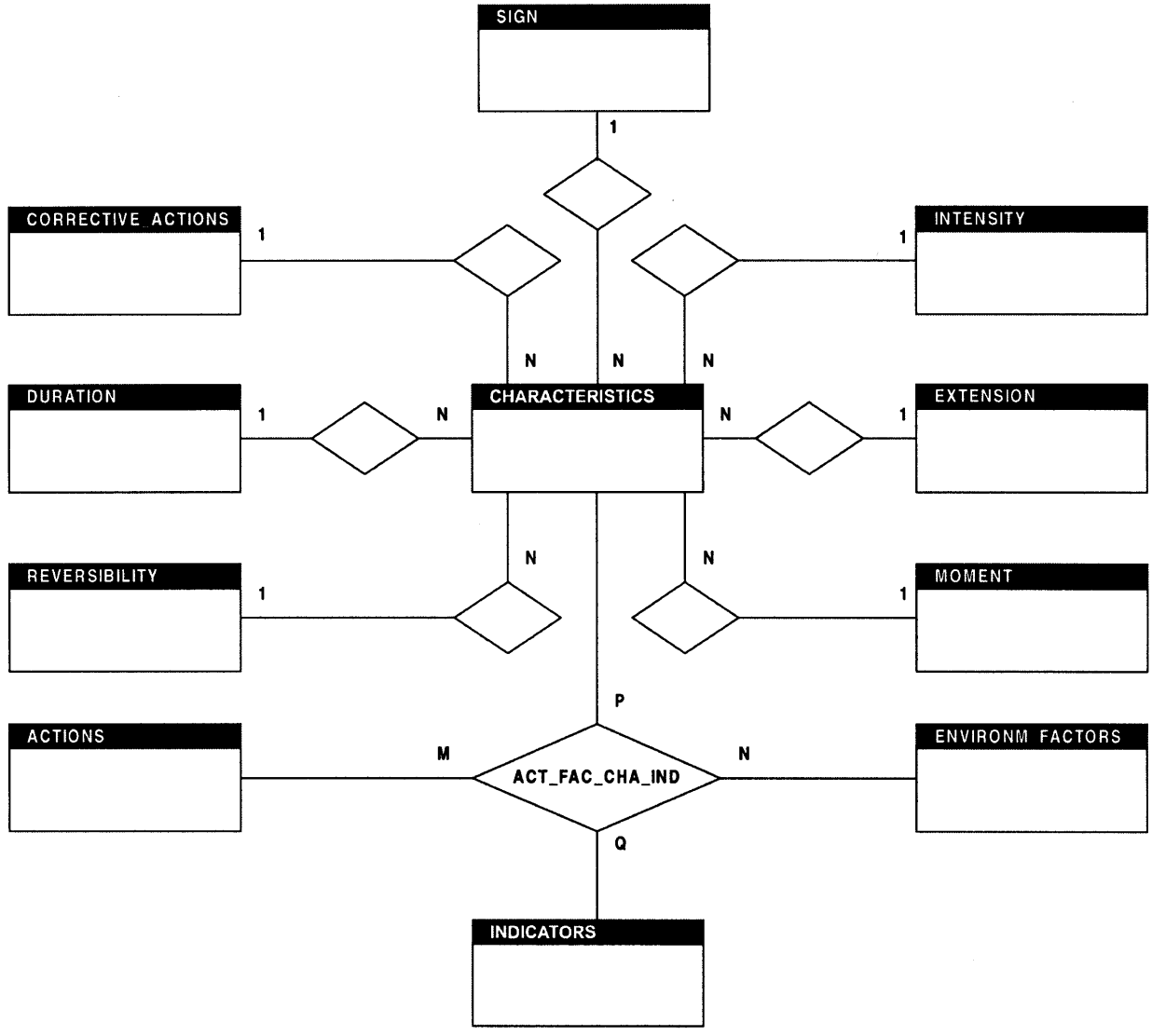

Fig. 5. Model for the EIA domain DB.

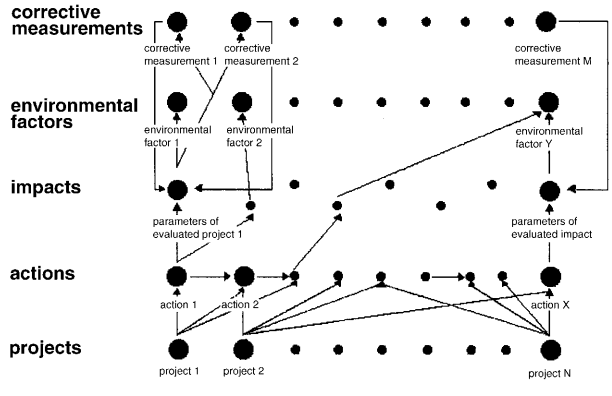

Fig. 6. Abstraction levels of the EIA domain.

client-application or application domain. As one can easily imagine, the 'quality' of the results and the ANNs' generalisation capacity will depend, among other parameters, upon the number, type and representative quality of the training patterns available in the application domain DB. The convergence criterion obtained will also characterise the quality of the resulting ANNs.

We shall now analyse the integration between the AMS and the client-application of the EIA domain, considering each of the three basic stages of the ANNs' lifecycle.

The integration of ANNs and DBs is of great significance during the ANNs' training process, both in supervised and non-supervised learning. At that stage of the ANNs' lifecycle, the importance of this

integration becomes apparent. We must note that reference will be made to two DBs which may be located in different hosts. The first DB models those architectures and parameters characteristic of ANNs (shown in Fig. 2) supported by the AMS. The second DB models those belonging to the EIA domain (shown in Fig. 5), and supported by the client-application. The AMS uses relational sentences to compose the patterns which will constitute the training sets. Generally, it will be able to assign various fields of the EIA's DB tables to the process elements of an ANN input layer. In addition those fields resulting from queries or views [23] will have the chance of being used in this assignment.

For instance, to generate a training set, according to the DB model in Fig. 5, the following parameters are considered as inputs: impact-causing action; affected EF; sign or feature of the effect produced; degree of incidence on the EF; influence area of the impact on the environment; time elapsed between action and effect; time that the effect will last; possibility of reconstructing the initial conditions once the effect has been produced; and possibility of adopting measures in order to compensate for the effects.

The desired output, assuming a supervised learning approach, will be the indicator which allows quantification of the effect. Considering all the previous parameters, a relational sentence must be defined to perform a JOIN operation among the following tables: ACTFACCHAIND, ACTIONS, ENVIRONMFACTORS, CHARACTERISTICS, SIGN, INTENSITY, EXTENSION, MOMENT, CORRECTIVEACTIONS, DURATION, REVERSIBILITY and INDICATORS of the DB corresponding to the model in Fig. 5. Later, a projection is made on the relevant fields of the training set. The results of the query include the parameters used as ANN inputs, those desired as output, and the descriptions of the parameters involved in the training set. Thus, the AMS and the client-applications will handle concepts or descriptions instead of simple codes. The information which integrates these patterns may be of different sorts: quantitative, qualitative, graphic, etc. Generally, this information must be transformed into a numeric representation capable of being preprocessed and used by the AMS.

Thus, the AMS automates the training sets generation process, and similarly, the test processes, which will be constantly updated. That is, they will always reflect the present situation of the EIA DB. The AMS will also be able to incorporate different preprocessing procedures through the logical and relational operators of the DB query language.

The entities NETWORK-ADDRESS, DATABASES, TABLES and QUERY-TRAINING-TEST of the AMS DB, according to Fig. 2, and their relationship with the ARCHITECTURE entity are the structures in charge of supporting the definitions of the training and test sets.

Within a supervised training context, the AMS will compose the information of the input and output layers, for this purpose using both the direct assignment of DB fields and the definition of relational or view sentences. Using non-supervised training, the process of training set generation is similar; that is, it disappears from the relational sentence to the desired output.

Figure 7 shows one of the multiple AMS forms, in particular, that of creating new ANNs or selecting an already existing one from the AMS connectionist DB.

This example shows the selection of one of the AMS's ANNs belonging to the EIA-specific domain. This ANN identifies a possible environmental impact indicator for the case studied. These indicators are the elements or parameters which provide the measure of each impact's magnitude, both qualitatively and quantitatively. Their selection is fundamental in order to carry out successful EIAs [24,25]. This ANN provides a single impact indicator for each case, which identifies a function integrating different subindicators.

The list displayed on the form (Fig. 7) labelled as *Network* shows the ANNs available in the AMS's DB related to the EIA application. The selection of one ANN will load all of its features into the AMS working memory for its later manipulation. Although the system has several ANNs, both in the EIA domain and in other application domains, that which identifies impact indicators will be the only one

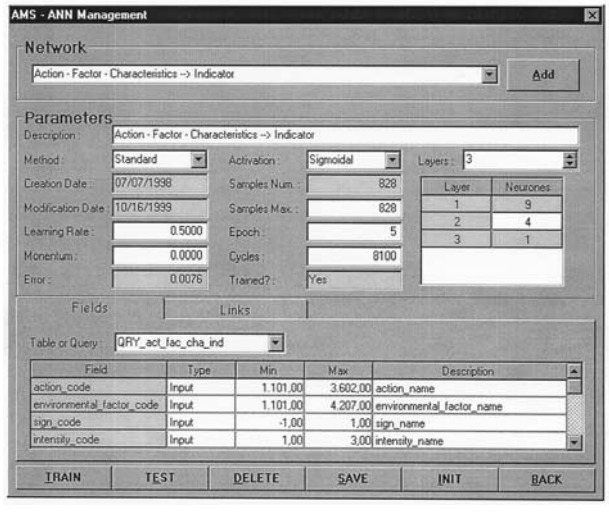

Fig. 7. ANN definition and manipulation form.

exposed due to its simplicity (a single output parameter). In this respect, the system integrates various ANNs to tackle the total or partial resolution, in series or in parallel [26], of some of the subproblems tackled by the EIA client-application. A second ANN for identifying impact crossovers, and a third for identifying corrective actions, have been developed among them. The enormous complexity of the composition queries of their training sets and a high number of output parameters warn us not to research these ANNs if we want to present the AMS functioning in a clear way.

At this stage, the AMS manages and handles all the aspects related to the architecture, learning method, training errors, sample number, creation and modification dates, etc.

The bottom section of the form in Fig. 7, labelled as *Fields*, is in charge of identifying the table or query used for definition of the ANN training set. The displayed *Table or Query* allows the selection of a table or query from the DB of the EIA domain as the basis for the training set definition. The AMS administrator assigns a label to each of the fields resulting from that query, identifying them as *input* or *output*. These labels define the fields that will form the input patterns and the desired outputs, the latter within a supervised learning framework. As one can imagine, the DB administrator of the EIA domain must include in its catalogue all those queries which may constitute training sets of any of the ANNs of interest to that field, so as to make them available to the AMS.

Another interesting aspect, which increases the usefulness of the AMS, and makes the system very friendly, is that of using descriptive fields of the variables used by the ANNs. The descriptive fields must also be part of the client-DB's tables or of the corresponding query which generates the training set. The form section labelled *Description* is in charge of selecting the semantic or descriptive field corresponding to each input and output field of the training set. The semantic or descriptive field assigns a meaning to the codes used by the ANN. The final user will work with these descriptive fields, while the ANN will work with the corresponding coded ones. In addition, the queries must include the whole preprocessing needed to carry out the conversion between information types, normally from qualitative to quantitative information. In this respect, a good design of the application domain's DBs will greatly facilitate this task, particularly taking into account beforehand all those tables which may be involved, totally or partially, in the solution of any subproblem which requires a connectionist approach [27,28].

Another possibility of the AMS manages the reuse of the different trained ANNs in any already existing or future client-application. To achieve this, the AMS relates the various client-applications forms to its ANNs by means of its own DB. The *Links* section in Fig. 8 manages the assignment of ANNs to the forms of client-applications. For this purpose, the active ANN is related to one or more forms belonging to the possible client-applications, which will have to be identified in the AMS's DB.

In this example, the ANN which identifies impact indicators is assigned as relevant to the form of the EIA application whose indicator bears the description *Environmental Impact Assessment* (*Action X Factor*). Thus, if the EIA client-application called the AMS within this form, this ANN would be assigned as relevant to its application among other possible ones. Another possible situation is that which assigns one ANN to several forms. For instance, the second impact crossover identification ANN is the one out of the three developed in the EIA domain which has been assigned three different forms from those of the client-application. Thus, a call to the AMS made by the application from any of these forms will identify that ANN as relevant. On the other hand, the assignment of an ANN to several forms, which could belong to different client-applications, is also contemplated. This is the case, for instance, of the AMS reusing those ANNs trained for image or signal processing.

Once the ANN's features, those of its learning process and those of the training set have been defined, learning is the next stage. The AMS visualises the procedural error using both a graphic and a text-based representation. Usually, the learning architecture handles great quantities of ANNs, carry-

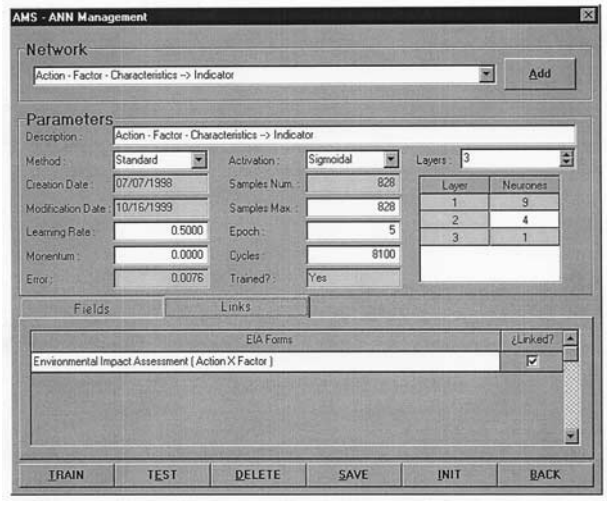

Fig. 8. System of ANN assignment to client-applications forms. Assignment to EIA application.

ing out, among other tasks, the ANN parameter adjustment and their relearning, considering the new registers which may have become part of the tables involved in that process, and above all, the convergence criteria desired for each case.

Once the learning stage is over, according to the convergence criterion required, the AMS tests the ANN. The test procedure requires a working pattern similar to that used at the learning stage. The AMS defines a new query which generates the test set and which, in particular, allows us to determine the generalisation capacity. A simple error rate, such as the number of errors per number of cases examined, or a sample confusion array (including sensitivity, specificity, predictive value and accuracy measurements), are obtained as outputs, depending on the case [27].

With regard to the third stage of the ANN lifecycle, the running stage, the AMS must implement a pattern which allows client-applications, and particularly its various forms, to reach those ANNs which are relevant in each case. In this respect, buttons or function keys which call the AMS in order to select those ANNs relevant to the form which has generated the call may be included into client-applications, separating the connectionist module and its resulting ANNs from potential client-applications.

For instance, if, during an EIA process, we include among one of its various impact crossovers or cause–effect links that formed by the action *Pruning and clearance* and the affected EF *Natural Vegetation of conservation value environment*, characterised by a *harmful* sign, *high* intensity, *medium* spread, *above 3 years* momentum, *temporary* duration, *impossible* reversibility, and finally, the possibility of corrective actions during the *building stage*, then the ANN which has already been trained for identifying possible indicators will conclude an impact indicator, assuming as input pattern the previously identified crossover. In this sense, the system becomes a perfect EIA advisor. Figure 9 shows the final unevaluated result of the previously seen crossover.

As may be seen, the form in Fig. 9, belonging to the EIA client-application, includes a *Suggest ANN* button for calling the AMS. The AMS consideration of that crossover, after running the corresponding ANN of the example, concludes (as shown in Fig. 10) that the system must take into account the possibility of using the indicator *Surface equivalent to vegetation of conservation value environment*. This indicator is defined as 'surface of vegetation units weighed up as their conservation value environment in relative magnitudes'. In this case, the ANN has selected what seems to be the most adequate indicator for the impact crossover studied.

Although in this case the ANN only identifies possible indicators, other developments, such as the second ANN of the three available in this domain, may get to identify whole new impact crossovers, characterising each and every one of the parameters involved. Usually, the access to the AMS running module from any form of a client-application brings about execution of a whole set of possible ANNs.

7. Conclusions

This research explains the foundations of how to model and implement the integration between Artificial Neural Networks (ANNs) and Databases (DBs) from a computer science approach.

The studies carried out on a set of tools for the design and development of commercial and freely distributed ANNs have exposed certain flaws with regard to their integration with DBs. None of them has rendered a complete integration with all of the ANN lifecycle levels. Therefore, this paper presents an analysis of the possibilities and advantages of the integration between ANNs and DBs, in connectionist, training, test and running architectures.

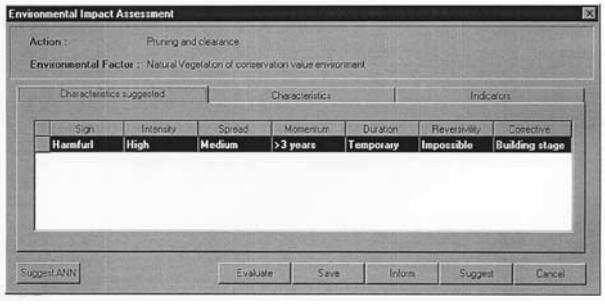

Fig. 9. Partial characterisation of an environmental impact crossover.

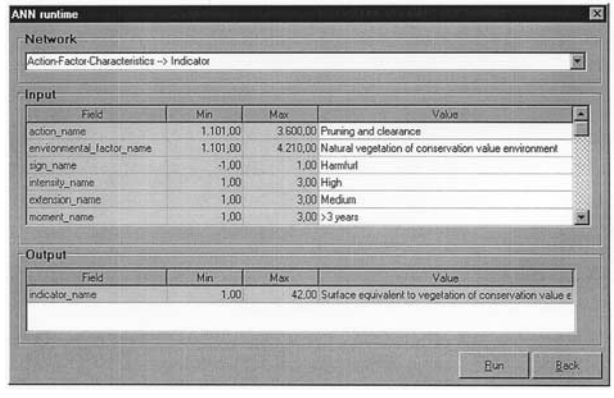

Fig. 10. AMS handling during the ANN running stage.

As a result of the research done, an AMS (ANNs Manipulation Server) which improves the handling and manipulation of multiple ANNs and their corresponding training and test sets is proposed and developed, thus facilitating its application on extended domains which may require great quantities of data and knowledge.

In addition, the AMS is constituted as an information system or core for future work on the study of the connectionist domain as application domain. In this case, both the connectionist DB of the AMS itself and the application domain DB would become a single DB, therefore the AMS would be suitable for being used for the study of different domains, such as that of the EIA (Environmental Impact Assessments) or the connectionist domain itself.

The advantages of the proposed approach are the following: improvement of the ANN's design, integration and experimentation possibilities; incorporation of integrity, consistency and security at the connectionist level; possibility of sharing the system's ANNs as reusable modules in client-server environments on data networks; support to their integration with heterogeneous systems; automation of the generation process of training and test sets, and improvement of the pattern-preprocessing tasks.

To check the AMS features and efficiency, its application with real cases of the EIA domain has been analysed. The integration architecture developed helps to solve some of the main problems posed by traditional EIA methods, such as the lack of robustness, depth and thoroughness in analyses. Thus, experts have an advanced tool which may help them when making decisions about different aspects, such as the optimal location of entrepreneurial projects, the study of alternatives for building infrastructures, the need to adopt corrective actions which decrease the impacts of any entrepreneurial activity, etc.

References

1. Ríos J, Brisaboa NR, Pazos A, Caridad S (1991) Estructura, Dinámica y Applicaciones de las Redes de Neuronas Artificiales. CE Ramon Areces, Madrid (in Spanish)
2. Codd EF (1972) Further normalisation of the data base relational model. In: Rustin R (ed). Data Base Systems, Prentice-Hall, MA
3. Codd EF (1982) Relational database: a practical foundation for productivity. Communications of ACM 25: 109–117
4. Ceri S (1983) Methodology and Tools for Data Base Design. North-Holland, Amsterdam
5. Codd EF (1972) Relational Completeness of Data Base Sublanguages. In: Rustin R (eds). Data Base Systems, Prentice-Hall, Mass
6. Cardenas AF (1985) Data Base Management Systems. Allyn and Bacon, Boston
7. Darken C, Moody J (1991) Note of learning rate schedules for stochastic optimisation. In: Lippmann and Moody (eds). Neural Information Processing Systems
8. Silva FM, Almeida LB (1990) Speeding up Backpropagation. In: Eckmiller (eds). Advanced Neural Computers
9. Salomon R (1991) Improved convergence rate of backpropagation with dynamic adaption of the learning rate. Lecture Notes in Computer Science 496: 269–273
10. Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial Intelligence through Simulated Evolution. Wiley, New York
11. Fogel DB (1995) Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press, NJ
12. Pazos A, Dorado J, Santos A, Rabuñal JR (1999) Optimisation of GA parameters to train recurrent ANN through weight adjustment and selection of activation functions. Proceedings Genetic and Evolutionary Computation Conf, Orlando, 2: 1793
13. Schiffmann M, Joost M, Werner R (1993) Application of Genetic Algorithms to the Construction of Topologies for Multilayer Perceptrons. Proceedings Conference on Artificial Neural Networks and Genetic Algorithms, Innsbruck, 1: 675–682
14. Holland JH (1975) Adaptation in Natural Artificial Systems. The University of Michigan Press, Michigan
15. Holland JH (1996) Sistemas Adaptativos Complejos. In: Pazos A (eds). Redes de Neuronas Artificiales y Algoritmos Genéticos. Universidad de A Coruña, A Coruña (in Spanish)
16. Goldberg DE (1989). Genetic Algorithms in Search, Optimisation and Machine Learning. Addison-Wesley, MA
17. Buckles BP, Petry FE (1992) Genetic Algorithms. IEEE Press
18. Leopold LB, Clarke E, Hanshaw BB, Balsley JB (1981) A procedure for evaluating environmental impact. USA Geological Survey Circular 645, USA Geological Survey, Washington DC
19. Morris P, Therivel R (1995) Methods of Environmental Impact Assessment. UCL Press, London
20. Pazos A, Santos A, Dorado J (1994) Linking of an artificial neural network with the EEIE expert system to identify environmental impacts. Proceedings World Congress on Neural Networks, San Diego, CA 2: 108–113
21. Pazos A, Santos A, Darado J (1994) The Evaluations of Environmental Impact: Cooperative Systems. Proceedings Conference on Artificial Neural Networks, Sorrento 1: 288–291
22. Santos A (1999) Desarrollo de una metodología e implementación de un sistema basado en el conocimiento de filosofía híbrida incluyendo Sistemas Expertos, Redes de Neuronas Artificiales y Bases de Datos. Una aplicación para la Evaluación del Impacto Ambiental. PhD Thesis, A Coruña University (in Spanish)
23. Date CJ (1994) A Guide to the SQL Standard, Addison Wesley, MA
24. Adriaanse A (1993) Environmental Policy Performance Indicators. Ministry of Housing Physical Planning and the Environment, Washington DC
25. McCracken RJ (1989) Impact Indicators for Measuring

Change in the Natural Resource Base. US Agency for International Development, Washington DC
26. Gutknecht M (1992) The ‘Postmodern Mind’: Hybrid Models of Cognition. Connection Science 4(3,4): 339–364
27. Weiss SM (1991) Computer Systems that learn. Classification and prediction methods from statistics, neural nets, machine learning, and expert systems. Morgan Kaufmann, San Francisco, CA
28. Lippmann RP (1987) An Introduction to Computing with Neural Nets. IEEE Trans Acoustics, Speech and Signal Processing 4: 4–22