NeuroWeb: An Internet-Based Neural Network Simulator

Erich Schikuta

Institut für Informatik und Wirtschaftsinformatik, University of Vienna
Rathausstr. 19/9, A-1010 Vienna, Austria
schiki@ifs.univie.ac.at

Abstract

The NeuroWeb project is an Internet-based framework for the simulation of neural networks. It aims for using the Internet as a transparent environment to allow users the exchange of information (neural network objects, neural network paradigms) and the exploit of available computing resources for neural network specific tasks (specifically training of neural networks). NeuroWeb’s design principles are acceptance, homogeneity, and efficiency.

1 Introduction

In the last few years many different artificial neural network systems were proposed of which only a few got accepted by the community. There are systems developed specifically for certain network paradigms, as Aspirin/MIGRAINES [6], SOM-PAK [9], and even more which try to deliver a comprehensive environment, as AXON [3], SNNS [11], NeurDS [10] to name only a few.

Basically all these systems, reaching from highly sophisticated interactive systems to programming language extensions, share the same common problems:

- A clumsy software environment, which mostly do not present an intuitive interface to the user. So the user faces the problem to cope with a new and/or complex tool.

- Most of these systems present a proprietary system, which is not capable to interconnect to other software systems and makes any information exchange difficult or impossible.

- All these systems lack of a generalized framework for handling data sets and neural networks homogenously. During the training phase and the evaluation phase of a neural net the user has to feed the net with large amounts of data. Conventionally data sets are mostly supported via sequential files only and the definition of the input stream, output or target stream of a neural net is often static and extremely complex.

These problems lead to the situation that a large number of rarely used simulation systems exists, because most scientists, scared of existing programs, develop their own systems for their specific neural network problem.

We believe that this situation is one of the reasons of an obstructed open information and data exchange within the scientific community.

We see a solution to this problem by the NeuroWeb system presented in this paper.

The layout of the paper is as follows. In the next section we give an overview of the NeuroWeb system, describe the design principles and justify them. Then we introduce the NeuroWeb interface on the basis of a simple example, followed by an explanation of the system extensibility. Finally we give a short conclusion, a description of the current application domain and a preview of the planned development of the system in the near future.
2 The NeuroWeb system

NeuroWeb is an artificial neural network simulator [5]. It provides basic neural network functions like creating, training and evaluating neural nets. The simulator includes eleven standard paradigms that can be classified into four groups:

- **Backpropagation**
  - Backpropagation in batch mode
  - Quickpropagation
  - Resilient propagation
- **Self organizing maps**
  - Counter propagation
  - Hopfield net
  - ART 1/2
  - Kohonen TSP
- **Recurrent nets**
  - Jordan net
  - Elman net
- **Cascade Correlation (with Quickpropagation)**

The interface is user-friendly so that the user is able to perform all tasks very comfortably and easily. The system is Internet-based in order to make it open to the growing Internet community. The simulator interacts with a database to store and retrieve all relevant data about the static and dynamic components of all neural networks. An important fact is that NeuroWeb is extensible. The system provides an interface, that makes it possible to integrate additional paradigms into the simulator. Due to the situation that the training phases require extended computational effort the system is able to harness free resources on the Internet by distributing the workload on available computing nodes. Therefore several training phases can be processed at the same time in parallel. Finally the NeuroWeb system is platform independent due to Java as implementation language.

2.1 Design principles

To get acceptance from the community the basis for the development of the NeuroWeb system had to be generally accepted standards only. Thus we made the main decision that only methodologies based on Internet technology are used for the design and development of the system. NeuroWeb uses the Internet on the one hand as a communication and collaboration platform for neural network scientists and users to exchange neural network specific knowledge (e.g. network models, data sets), on the other hand it provides access to unused resources (e.g. processors, main memories, disk space) for neural network specific simulation tasks.

Besides the "Internet" decision we adhered to the following design principles for development of the system:

- Acceptance,
- Homogeneity, and
- Efficiency
2.2 Acceptance by information accessibility

The system has to be accepted by the user by providing an intuitive and flexible interface. Intuitiveness of the working environment is reached by modelling neural networks simple but without omitting their specific characteristics, thus adhering to the famous words ”Make it simple, but don’t make it simpler”. Further the usage of neural networks has to be flexible for the user; on the one hand it must be simple to apply distinct neural networks of one paradigm for different problems, on the other hand different neural networks of different paradigms for one problem.

NeuroWeb is programmed in Java to make it portable on the Internet. It can be executed both as a stand-alone program and as a Java applet in a java-enabled Internet Browser. NeuroWeb is using the ODBC interface to allow the communication to databases on the Internet, which store the describing information of the neural networks in focus. Besides the the neural network objects, it also provides the neural network paradigms to the users. A typical problem in the neural network research community is to invented the wheel over and over again. Any researcher is programming his own paradigm and it is difficult to share program code due to the proprietary nature of available neural network simulators. We solved this problem by a generalized Internet based approach introducing so called NeuroWeb servers. There we apply a mechanism, which can be simplified seen as a specialized FTP Server, where user can download requested or upload their own developed paradigms. Thus by the standardized internal interface any NeuroWeb simulator instance on the Internet can be extended simply by choosing a new paradigm from a NeuroWeb server. This allows a simple information (paradigm) exchange between neural network researchers.

![NeuroWeb architecture](image)

**Figure 1. NeuroWeb architecture**

The NeuroWeb system architecture on the Internet is quite manifold and is depicted by figure 1. NeuroWeb supports basically two execution modes, application and applet mode. With the application mode a stand-alone version of the NeuroWeb system is executed on a NeuroWeb Application User node, where the NeuroWeb system can access its own Database system storing the data in focus, as neural networks, training, and evaluation data. This information can also be stored on a dedicated NeuroWeb Database Server, which itself is not executing the NeuroWeb system, but providing the interesting NeuroWeb data via the ODBC interface. The NeuroWeb Applet User node is executing the NeuroWeb environment as a Java applet within a standardized, Java enabled Internet browser, but can (until now) not access any database on the Internet. Thus this mode is only of interest for the casual user, who performs some test runs on the fly. Both execution modes allow to access specific NeuroWeb Paradigm Servers (NPS), which provide libraries of neural network paradigms. These paradigm servers are decentralized and hosted by different NeuroWeb users, who provide their collection of NeuroWeb paradigms to the community. This allows the arbitrary neural network developer to share and exchange its novel paradigm with other NeuroWeb users. These paradigm servers can be seen as a specialized ”FTP” depository, which can be searched and accessed via the Internet. Besides the paradigm servers, which provide “information”, also NeuroWeb Simulation Servers (NSS) are available on the
Internet, which provide "performance". These simulation servers allow to download and execute neural network training applets from users, who are allowed to access these servers. The distribution of training processes enables the users to execute these time consuming operations in parallel and delivers a high degree of coarse grain parallelism to speed-up the typical turn around time in neural network research and evaluation.

2.3 Homogeneity by database modelling

It is our objective to consider neural networks as conventional data in the database system. From the logical point of view a neural network is a complex data value and can be stored as a normal data object.

The usage of a database system as an environment for neural networks provides both quantitative and qualitative advantages.

- **Quantitative Advantages.** Modern database systems allow the administration of objects efficiently. This is provided by a 'smart' internal level of the system, which exploits well studied and well known data structures, access paths, etc. A whole bunch of further concepts is inherent to these systems, like models for transaction handling, recovery, multi-user capability, concurrent access etc. This places an unchallenged platform in speed and security for the definition and manipulation of large data sets at users disposal.

- **Qualitative Advantages.** The user has powerful tools and models at hand, like data definition and manipulation languages, report generators or transaction processing. These tools provide a unified framework for both handling neural networks and the input/output data streams of these networks. A homogeneous and comprehensive user interface is provided to the user. This spares awkward tricks to feed the user's data into and analyze with a proprietary network simulator system.

In our approach, the *embedment* of neural networks into data base systems, we follow the opposite direction to conventional approaches. In other words we move the neural networks to the database systems, and not the data to the neural network simulators (see [8][7]).

Neural networks have to be modelled as simple objects following the well known and accepted object-oriented paradigm. These objects can be used and administered by the conventional standardized tools of a database system. A neural network object is defined by its static and dynamic components. These components are dependent on the appropriate network paradigm. Different paradigms show different static and dynamic properties. The static neural network components comprise all information stored in relations, as neural network specific parameters, links, training objects, evaluation objects, etc. The dynamic components of the neural network object are the typical operations on neural network, the training and evaluation phase, which are defined by the respective paradigm of the neural network object.

A further advantage of the database approach is the possibility to specify the data set for the analysis by a functional data stream definition in a comfortable way. It is not necessary to specify the data values explicitly but by SQL statements. This allows to describe, for example, the input data for a typical XOR net training phase by a SQL statement of the form 'select xvalue, yvalue from XORNetData;'. where XORNetData is a relation with the relational scheme XORNetData(xvalue, yvalue, targetvalue) of the underlying database system holding tuples for the attributes (xvalue, yvalue) with values \(0\ 0,\ 0\ 1,\ 1\ 0,\ 1\ 1\). These values can be fed into the input layer of the neural network. In similar form the respective target values (0 1 1 0) can be defined by the SQL query 'select targetvalue from XORNetData;'.

These are only simple examples showing the functionality in principle. It is up to the reader to imagine how comfortable this framework can be to describe data volumes in the Gigabytes by simple SQL statements in a standardized way.

The well known apparatus of the SQL database manipulation language is at hand of the user not only for specifying training data sets, but also for all information specific tasks. Thus the same tool can both be used homogeneously for administration and analysis of the stored information, and of neural networks as well. So it is easily possible to use 'real world' data sets as training set for neural networks and to analyze other (or the same) data with trained networks.

2.4 Efficiency by workload distribution

The power hungry neural network specific tasks (specifically the training phase) have to be processed efficiently by using all available resources. Thus we implemented into NeuroWeb the possibility to exploit processing resources on the Internet by the use of the Java RMI (remote method invocation) interface. Thus neural network specific tasks, as e.g. training, can
be distributed in a coarse grain parallel approach among available computers. The user can control these parallel executing instances from within NeuroWeb and has automatically available the calculated results. The situation is depicted in figure 1.

The user can choose and register available *NeuroWeb simulation Server* within the NeuroWeb system. A training phase is then executed on this server by downloading the specific applet and the necessary training data to the server in focus via the Java RMI mechanism.

RMI enables the programmer to create distributed Java-to-Java applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. A Java program can make a call on a remote object once it obtains a reference to the remote object, either by looking up the remote object in the bootstrap-naming service provided by RMI, or by receiving the reference as an argument or a return value. A client can call a remote object on a server, and that server can also be a client of other remote objects. RMI uses object serialization to marshal and unmarshal parameters and does not truncate types, supporting true object-oriented polymorphism.

3 The *NeuroWeb Interface*

In order to reach the goals of the previous section we decided on the use of Java as programming language. We developed the simulator with the JDK 1.2. Furthermore Java gave us the possibility to develop the simulator both as applet which is publicly accessible via the Internet and as stand-alone application. With RMI (remote method invocation), Java provides a mechanism to distribute calculation tasks of a system to different computers available on the network. A further reason to use Java was that the JDBC-ODBC-Bridge in Java offers the possibility to connect to any ODBC data source. As basic data store for the built-in network paradigms we used an MS Access database, but all other ODBC enabled database systems are possible, e.g. Oracle.

3.1 System operation

The NeuroWeb system can be started in two different modes, the applet mode, loaded within a Java enabled Web browser from a NeuroWeb site, and the stand-alone mode, as a compiled Java program.

![Figure 2. NeuroWeb start window](image)

When the simulator is started up the *start window* is shown (see Figure 2). The upper left list in the window named "Basic Net-Types" shows the built-in net paradigms. These paradigms are those eleven neural network types that are always part of the simulator in the distributed standard version. In the upper right corner of the window the connected database is shown (if the system is not connected to the database the DB-Name field contains the message "not connected"). The server list gives the names of the available Internet NeuroWeb servers waiting for processing requests. The net type window allows to load new, additional network paradigms from connected databases on the Internet.
After choosing a network paradigm, the user can define the structure of the network according to the selected paradigm in the create net window (see Figure 3). In the shown example we build a classical XOR backpropagation network with 2 input, 5 hidden, and 1 output neuron, which is depicted in the graph section of the create net window.

After defining all necessary training parameters, the training process of the net can be started. The control training window (see Figure 5) allows to distribute this specific training operation onto an available computer on the Internet. However, it is also possible to execute the task on the local host (as done in the shown example). So it is possible to start an arbitrary number of training operations onto an arbitrary number of available computers. For each such task a window is presented on the screen so the user can control and administer the tasks. During execution and after completion of a training, the error curve is shown graphically (right section of window) and the training weights of the nets are shown and stored in a database.
system (if available) for further usage. If no database system is connected the results are stored and available temporarily until the system shuts down.

In this section we showed a small and simple example only. Besides the presented interface a number of additional windows are available for evaluation of trained networks, database server and NeuroWeb server control, loading of new paradigms via the Internet and so on. However we tried carefully to design an intuitive window interface for the user.

3.2 System extensibility

The NeuroWeb system offers an easy to use interface for neural network researchers to extend the set of neural network paradigms. This can be done by downloading paradigms from a NeuroWeb paradigm server, or by programming a new paradigm in Java. This new paradigm can be both easily integrated into the system and provided to other researcher on the Internet by storing it on a paradigm server.

With the creation of a new paradigm two different types of nets have to be distinguished, extensions of basic net types and advanced net types.

Basic nets are the types provided directly by the system. Basic types are organized in an inheritance hierarchy. Thus new basic types must be based on existing types and integrated into the hierarchy. The process of the generation of a new basic type consists of the following steps:

- **New net type database entry.** The new net type must be entered into the underlying NeuroWeb database. This step is supported by an own form in the system.

- **Database table creation.** For each new type four tables have to be created in the database: a net structure, a training operation, a training result, and an evaluation table.

- **Paradigm in Java.** A Java class has to be realized describing the paradigm, which must implement a specific net interface (a subclass of java.io.Serializable). This interface has to define two methods, a training step and a training result method.

- **User interface.** Finally a start class providing an arbitrary user interface has to be developed, which allows to trigger the net paradigm classes.

The process for new advanced net types is similar but with fewer specifics, because it has not to comply to the integrated basic net paradigms.

The programmed Java classes and objects have to follow a pre-defined naming convention. All class files have to be packed into a JAR-file. To make this JAR-file public accessible on the Internet it has to be stored in a globally accessible directory.
on a NeuroWeb paradigm server (with FTP access). A short, one-line, description of the new class (giving paradigm name, start class name, JAR file name, database creation batch file name) has to be inserted into a specific NeuroWeb directory file, which is a text file named NewNets.txt. This file allows an arbitrary user to inspect the contents provided by the NeuroWeb paradigm server in focus and to download requested paradigms via FTP. However the FTP functionality is hidden to the user by the NeuroWeb interface.

A detailed description of the extension process can be found in [4].

4 Conclusion

We presented the NeuroWeb artificial neural network simulator, an Internet based framework striving for user acceptance, system homogeneity, and operation efficiency.

NeuroWeb aims for anybody interested in neural networks, from researcher to student. In the simplest case NeuroWeb can be used locally within an Internet browser or as an application without Internet access. Then it is at least a viable and user-friendly neural network simulator without the Internet goodies. We are aiming at students and we use it in graduate courses on neural network for the last two years. But our users are also researchers implementing their own paradigms.

It is used within the curriculum of computer science and business oriented computer science at the University for graduate courses on knowledge representation.

Due to NeuroWeb’s inherent extensibility we are constantly improving the available paradigms and adding new ones. At the moment we are working on a new self organizing map module.

In the near future we plan to move the NeuroWeb system onto the Grid [1], which allows a simpler and more natural embedding of the system into the Internet’s hardware and software infrastructure by usage of standardized middleware, e.g. Globus [2].

Acknowledgement

On this occasion we would like to express our thanks to Ewald Hombauer and Siegfried Zach, who implemented the NeuroWeb system.

References