Proceedings of the VI Brazilian Conference on Neural Networks - VI Congresso Brasileiro de Redes Neurais
pp. 349–354, June 2–5, 2003 – Centro Universitário da FEI, São Paulo - SP - Brazil

A Services-Oriented Architecture applied to Evolutionary Artificial Neural Network

Gilson F. Laforga[1], Roseli F. Romero[2]
[1]UNIRP-Centro Universitário de Rio Preto
[2]ICMC-USP São Carlos
E-mails: laforga@unirpnet.com.br, rafrance@icmc.usp.br

Abstract

In this work, a services-oriented architecture that allows the identification and the training of the best Artificial Neural Network (ANN) to be used in a certain problem, through the use of Genetic algorithms (GA), is proposed. The Web Services and the Neural Network Markup Language constitute the technology used for the development of this approach. A main advantage of this approach is that any user can invoke this application programmatically over the Internet, without that it needs to have a computer with great capacity of processing.

1. Introdution

Services-Oriented Architecture (SOA) provide a standard programming model that allows software components to be published, discovered and invoked by each other. There are essentially three components to a SOA: a Service Requester is a software component in search of a service to invoke, a Service Broker is a central repository that facilitates service discovery and a Service Provider, in this case a Web Service [1].

A Web Service is a software application or component identified by a URI (Uniform Resource Identifiers) [2], whose Interfaces and Interface Bindings: (a) are capable of being described by standard XML (Extensible Markup Language) vocabularies [3] and (b) support direct interactions with other software applications or components through the exchange of information that is expressed in terms of an XML Info set via Internet-based protocols [4].

At a long time, researchers seek in the nature inspiration for the development of techniques for the solution of problems. One of the reasons for this search is due to the fact that in the nature, we found satisfactory solutions for problems highly complex, as it is the case of the survival of the species. The ANN appeared of the attempt of imitating the human brain, and the GA appeared of the attempt of imitating the natural evolution.

An ANN is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the biological neuron. The processing ability of the network is stored in the inter-unit connection weights, obtained by a process of adaptation to, or learning from, a set of training patterns. There are a variety of kinds of design and learning techniques that enrich the choices that a user can make.

ANN can be trained to solve problems that are difficult for conventional computers or human beings. ANN has been trained to perform complex functions in various fields of application including pattern recognition, identification, classification, speech, vision, and control systems [7].

Neural network topology, or the number of nodes, and the placement and number of connections between them has a significant impact on the performance of a neural network and its training ability. The problem of finding an optimal topology can be thought of as a search problem, where the search space is the space of all possible network topologies. This space is infinite, since there can be any number of nodes and connections in a neural network.

Genetic Algorithms refer to a population-based stochastic search algorithms that are developed from ideas and principles of natural evolution. One important feature of this algorithm is their population-based search strategy. Individuals in a population compete and exchange information with each other to perform certain tasks. Hundreds of articles that explore the relationship between GA and ANN have been presented [8].

An aspect constantly pointed out in these articles, is that a lot of times GA cannot be explored in your totality, because the search space should be limited. One of the reasons for this limitation is that GA with large search space is very computationally expensive.

In this work, an approach to minimize this problem is proposed that uses the Web Service and the Neural Network Markup Language [10]. These technologies are used to create an architecture that allows authorized users to use the service that allows the identification and training of the best ANN to be used in a certain problem through the Internet. Several modules compose this architecture: user control module, ANN-WS (ANN Web Service) control module, input control module and ANN-WS modules. This paper is organized as it follows. In the section 2 and 3, a brief introduction to Web Service and Evolutionary Artificial Neural Networks concepts are presented. In section 4 and 5, the operation of the system and the conclusions are presented.

2. Web Service

In a simplified way, a Web Service (WS) is an application that exposes a Web-accessible Application Program Interface (API). In other words, this application can be invoked over the Web. Applications invoking this Web service are referred as clients [11]. In a more precise way, Web services are a platform for building interoperable distributed applications.

The Web service platform is a set of standards that the applications have to follow in order to achieve Web interoperability. Since the standards are respected, Web service can be writing in whatever language and on some operational system.

The Web service platform needs a minimum set of features to enable the building distributed applications:

· To enable interoperability, the Web service platform must provide a standard type system that bridge among system types of different platforms, programming languages, and component models. The XML is the basic format for data representing on the Web service platform. In addition, for being simple to create and parse, XML is an open pattern. The W3C (World Wide Web Consortium) XML Schema (XSD) [12] is a standard that specifies some built-in types and language to define additional types.

· The Web service platform must provide a mean for describing a Web service and information others need to invoke the own Web service. The Web Service Description Language (WSDL) [1] is an XML grammar used to formally describe Web service interfaces and protocol bindings. Being XML-based, WSDL is both machine and human readable.

· There must be a mechanism for invoking Web service remotely, similar to a Remote Procedure Call (RPC) protocol [13]. To promote interoperability, this RPC protocol must be platform and programming language independent. The Simple Object Access Protocol (SOAP) [14][19] provides the standard RPC mechanism used for invoking Web Service. The SOAP specification provides standards for the format of a SOAP message and how SOAP should be used over HTTP (HyperText Transfer Protocol) [15]. SOAP also builds on XML and XSD for providing standard rules for encoding data.

SOAP is the key technology behind the Web Service technology. In a general way, SOAP specifies a wire protocol that is used for highly distributed architectures. SOAP specifies a very lightweight protocol from an administrative and use perspective. Users of SOAP-based applications need only be able to process HTTP requests and parse XML data. These requirements are relatively easy to manage, especially when compared with the requirements of other distributed processing architectures [14].

In Figure 1, it is shown Web Service high-level architecture, assuming the utilization of SOAP over HTTP for invoking the Web service. The architecture processing consists in the following steps:

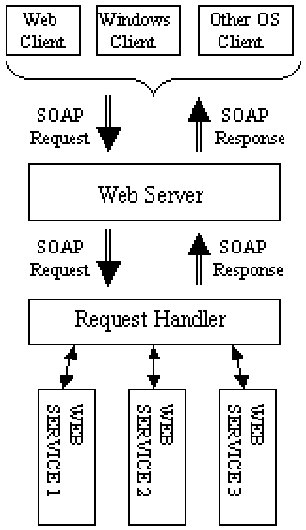

Figure 1 - Web Service Architecture

1. A client, written in any language and running on any platform, invoke the Web service by processing of the WSDL document that describes a Web Service. The client then formulates a SOAP request message based on the service description.
2. The Web server receives the SOAP request message as part of an HTTP POST request.
3. The Web server forwards to a Web service request handler these requests for processing.
4. The request handler is responsible for parsing the SOAP request, invoking your Web service, and creating the proper SOAP response.
5. The Web server then takes this SOAP response and sends it back to the client as part of HTTP response.

3. Evolutionary Artificial Neural Networks

The evolution in the nature is an excellent example of the operation of the process of natural adaptation. Thanks to this process, millions of species of plants and animals live in the Earth. Each one of these species survives, adapting yourself to the peculiar conditions of your environment.

The populations of organisms of each one of these species, cooperate and compete continually, in an evolutionary interaction with selection mechanisms and variation, what does with that the plan of creation of each new organism, codified in your genome, constantly vary. Like this, to each new generation, appear organisms that thanks to your characteristics and specific abilities are more capable to compete and to survive in your environmental conditions.

About the decade of 50, some researchers recognized that, starting from these beginnings adaptation evolutionary could be deduced countless concepts and strategies for the resolution of optimization problems. These researchers originated the call evolutionary computation, that is the generic name, given to computer methods, inspired in the theory of the evolution [9].

The algorithms used in evolutionary computation are known as evolutionary algorithms. They include Evolution Strategies (ES), Evolutionary Programming (EP) and Genetic Algorithm (GA). An important characteristic, common to all these algorithms, is the research strategy based on populations. The individuals in a population compete and exchange information, with the objective of accomplishing a certain task; this strategy is composed of the following steps:

1- An initial population is created of random form
2- It is determined the fitness degree, of each organism of the population
3- The parents of the next generation are selected, with base in your fitness degree
4- A new population is created, through the application of the genetic operators (crossing and mutation)
5- If the end condition be not satisfied, returns to the step 2

The evolutionary algorithms are very useful, to treat of optimization problems. For analogy, the resolution of optimization problems can be considered as the search by the highest mountain in an area covered by the fog. The number of individuals in the population is the number of climbers. After a certain time of exploration, the climbers communicate the base your altitude and position. The climbers that register the smallest altitudes are excluded of the expedition; new climbers substitute them, these are children of the climbers that registered the largest altitudes. This process is repeated until a certain altitude it is reached (stop condition).

The determination of the parameters of a neural network is an optimization process. In the case of an ANN Back-Propagation these parameters can be the learning rate, momentum, number of intermediate layers, number of elements in the intermediate layer and the activation function. The combination of all these parameters characterizes a space of multimodal search, with thousands of possible options. The observation of this fact took the creation of a new research area in neural networks, the Evolutionary Artificial Neural Network (EANN) [8].

In a general way, the evolution can be applied to neural networks in several ways: in the weights of the connections, in the architecture of the net, in the learning rules, among others.

4. A Services-Oriented Architecture applied to Evolutionary Artificial Neural Network (SOA-EANN)

A great number of researchers are working with ANN in the whole world. These researchers have been proposing a great amount of ANN models; these models have been applied in the resolution of hundreds of problems.

In the development of these applications, several types of ANN simulators can be used. These simulators usually belong to one of the three main existent approaches of ANN simulators: the parallel implementation techniques, hardware emulation and simulation tools.

One of the most popular approaches is the one that it uses simulation tools. The tools for development and simulation of ANN can be divided into three main categories: menu based/graphic oriented systems, module libraries and specific programming languages.

The systems based on module libraries are characterized by the provision of a module library written in a general purpose programming language. Such a library can be considered as a toolbox consisting of basic building blocks for the construction and execution of connectionist experiments. These basic building blocks may be complete network models, tools for training networks, tools for graphical representations and analysis [16].

The ANN Web Service (ANN-WS), proposed and implemented in this present work, belongs to the module libraries category of simulators. The Web Service is an application that exposes a Web-accessible Application Program Interface. That means that this application can be invoked programmatically over the Web.

The ANN-WS proposed is an independent module. It accepts as entrance a description of the characteristics of the ANN to be workout, as well as the necessary data for its training.

For the description of the characteristics of an ANN, exist some ANN specification languages, used in different simulators, such as ASPIRIN [17], CONNECT [18] and SNNS [5].

Recently, another type of ANN description language has been proposed; that is a language based on XML notation for the ANN description. The description of the ANN in XML format, allows an easy interpretation of the information, independent from operating system platform or programming language.

One of the first ones is proposed as a part of PMML (Predictive Model Markup Language) [6]. However, PMML presents a serious limitation, since it allows defining only Back-Propagation networks.

A more powerful approach is supplied by the NNML (Neural Network Markup Language) in [10]. NNML uses XML notation for full description of supervised learning neural networks, including data dictionary, pre-

processing and post-processing, details of structure and parameters of ANN and also various auxiliary information.

NNML allows the description of the ANN in an expressive and extensible way, easy to interpret and independent from individual platform or programming language. For these reasons, a similar NNML Language was chosen for ANN descriptions (Table 1) (Table 2).

The description of the ANN characteristics and necessary data is received in the XML description form. In the same way, the output of ANN-WS is also codified in XML .

Figure 2 shows SOA-ANN high-level architecture. The operation of SOA-ANN system occurs according to the following steps:

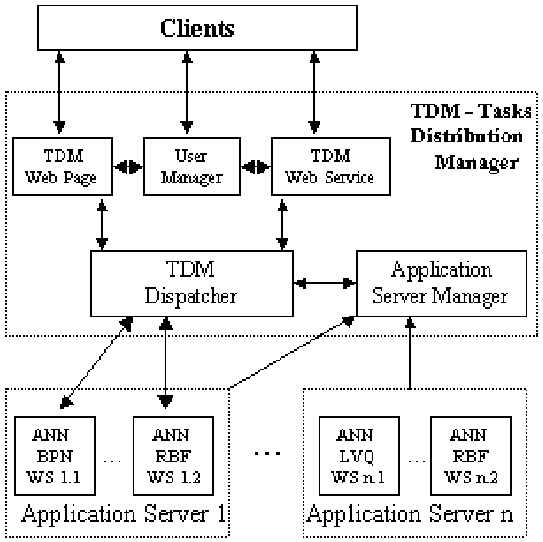

Figure 2 – Services-Oriented Architecture applied to Evolutionary Artificial Neural Network

1- Receives the information on ANN to be workout (type of ANN, number of inputs, number of outputs), as well as the group of data for training and test.
2- Initializes the genetic algorithm; create at random, the first generation of genotypes and phenotypes.
3- Sends simultaneously, each one of the phenotypes ANN and the group of training data and test for a WEB Service appropriate of an Application Server. Receive the trained phenotype, together with a fitness value.
4- Selects the best genotypes of the previous generation, makes the crossover and the mutation and creates the next genotypes generation. If is not the last generation, go back to previous step.
5- When finishing the last generation, chooses the best genotype, codifies your phenotype, and sends it for the customer.

For the operation of the system, many other procedures exist, for example, the maintenance of the Application Service Manager database and User Manager.

5. Conclusions and future works

Evolution can be introduced into ANN at many different ways: connection weights, architectures, learning rules, connection weights and architecture simultaneously, etc. As previously mentioned, this application consumes a great amount of computer resources. In this work it is presented an infrastructure that allows the exploration of the application of GA in ANN by users without that it needs to have a computer with great capacity of processing. SOA-EANN proposed is an initial model; certainly, many modifications are necessary for improvement the performance of the proposed system.

The number of ANN applications has been increasing in a considerable way in the last years, mainly in the industry and in the research. The possibility of the ANN utilization by users that don't have equipments with a good processing capacity, but that even so can work with complex training groups, can still enlarge the number of ANN users and applications.

At this time, ANN-WS implements only the Multi-Layer Perceptrons with the Back-Propagation algorithm (Figure 3) (Figure 4), but the incorporation of other ANN models into proposed system are also part of future works. The exploration in the several ways of application of GA to ANN will also be object of future works.

References

[1] Shohoud, Y. *Building XML Web Services.* Addison Wesley, 2002.

[2] Berners-Lee, T. *Uniform Resource Identifiers (URI): Generic Syntax.* RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, 1998.

[3] *Extensible Markup Language (XML) 1.0.* http://www.w3.org/TR/2000/REC-xml-20001006, 2001.

[4] *Web Services Description Language (WSDL) 1.1.* http://www.w3.org/TR/wsdl, 2001

[5] Zell, A. et al. *SNNS - User Manual, Version 4.2.* University of Stuttgart, 1995.

[6] The Data Mining Group (DMG). *PMML 2.0 - Neural Network.* http://www.dmg. org/v2-0/NeuralNetwork.html, 2001.

[7] Haykin, S. *Neural Networks, A Comprehensive Foundation.* Prentice Hall, 1999.

[8] Yao, X. *Evolving Artificial Neural Networks.* Proceedings of the IEEE, 87(9): 1423-1447, September 1999.

[9] Back, T. *Evolutionary Computation: comments on the history and current state.* IEEE Transactions on Evolutionary Computation, vol. 1, no.1, April 1997.

[10] Rubtsov D. *Development of technology of artificial neural networks application in applied information systems.* Ph.D. thesis, Altai State University, Barnaul, 2000.
[11] Coyle, F. XML, *Web Services, and the Data Revolution.* Addison Wesley, 2002.
[12] Holzner, S. *Inside XML.* New Riders Publishing, 2001.
[13] Tanenbaum, A. *Modern Operating Systems.* Prentice-Hall Inc, 1992.
[14] Scribner, K. *Understanding Soap.* Sams Publishing, 2000.
[15] Tittel, E. *Foundations of WWW.* IDG Books, 1996.
[16] Kock, G., Serbedzija, N. *Simulation of Artificial Neural Networks. Systems Analysis - Modelling - Simulation* (SAMS) 27(1):15-59, Gordon & Breach Science Publishers, 1996.
[17] Leighton, R. *The Aspirin/Migraines Software Tools.* MITRE Corporation, 1993.
[18] Kock, G. *Artificial neural networks: from compact descriptions to C++.* Proceedings of the International Conference on Artificial Neural Networks, Springer-Verlag, pp. 1372-1375, 1994.
[19] *Simple Object Access Protocol (SOAP) 1.1.* http://www.w3.org/TR/SOAP, 2001.

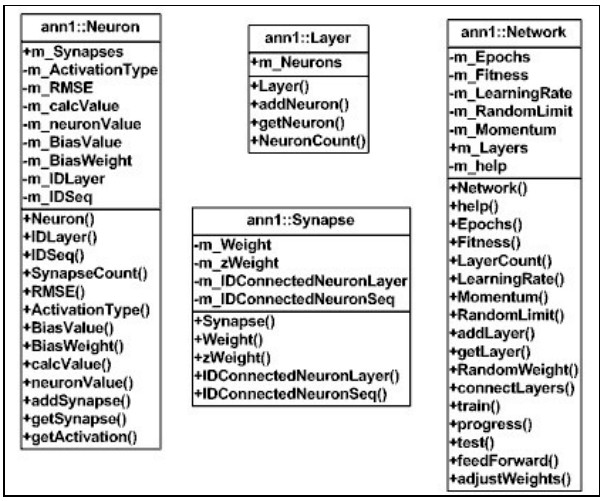

Figure 3: ANN Back Propagation Class

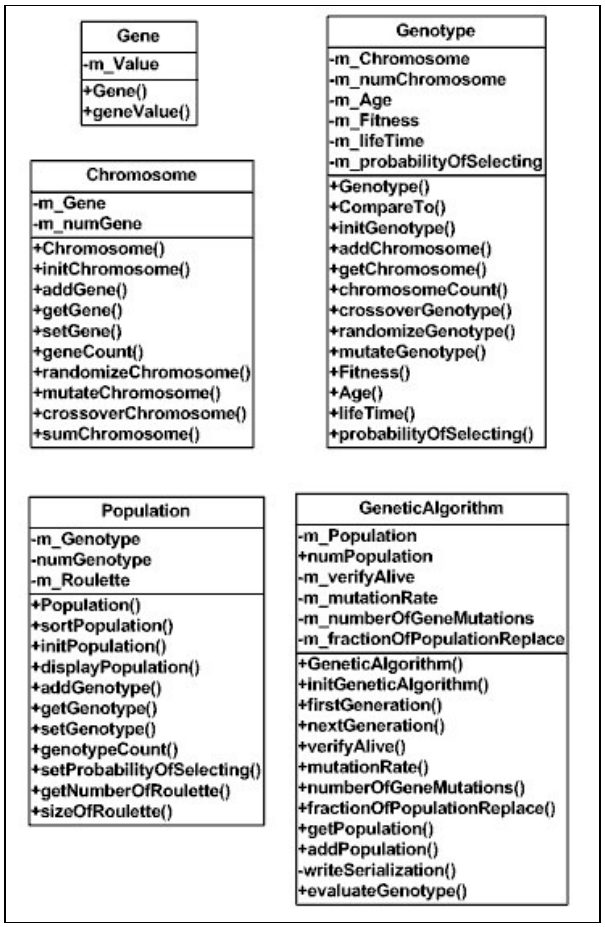

Figure 4: Genetic Algorithm Class

Table 1: ANN Patterns XML Schema

```
<?xml version="1.0"?>
<xs:schema
id="Patterns"
targetNamespace=http://tempuri.org/treina.xsd
xmlns=http://tempuri.org/treina.xsd
xmlns:xs="http://www.w3.org/2001/XMLSchema" >
  <xs:element name="Patterns">
      <xs:complexType>
        <xs:choice maxOccurs="unbounded">
        <xs:element name="P">
        <xs:complexType>
        <xs:sequence>
        <xs:element name="I" type="xs:string" minOccurs="0" />
        <xs:element name="O" type="xs:string" minOccurs="0" />
        </xs:sequence>
        </xs:complexType>
        </xs:element>
        </xs:choice>
      </xs:complexType>
</xs:element>
```

Table 2: ANN Back-Propagation XML Schema

```
<?xml version="1.0" ?>
<xs:schema id="ANN1BP" targetNamespace="http://tempuri.org/net0.xsd"
xmlns="http://tempuri.org/net0.xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema" >
<xs:element name="Network">
<xs:complexType>
<xs:sequence>
    <xs:element name="Epochs" type="xs:string" minOccurs="0" />
    <xs:element name="Fitness" type="xs:string" minOccurs="0" />
    <xs:element name="LearningRate" type="xs:string" minOccurs="0" />
    <xs:element name="Momentum" type="xs:string" minOccurs="0" />
    <xs:element name="RandomLimit" type="xs:string" minOccurs="0" />
    <xs:element name="m_Layers" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
      <xs:sequence>
      <xs:element name="m_Neurons" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
          <xs:sequence>
            <xs:element name="IDLayer" type="xs:string" minOccurs="0" />
            <xs:element name="IDSeq" type="xs:string" minOccurs="0" />
            <xs:element name="RMSE" type="xs:string" minOccurs="0" />
            <xs:element name="ActivationType" type="xs:string" minOccurs="0" />
            <xs:element name="BiasValue" type="xs:string" minOccurs="0" />
            <xs:element name="BiasWeight" type="xs:string" minOccurs="0" />
            <xs:element name="calcValue" type="xs:string" minOccurs="0" />
            <xs:element name="neuronValue" type="xs:string" minOccurs="0" />
            <xs:element name="m_Synapses" minOccurs="0" maxOccurs="unbounded">
                  <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Weight" type="xs:string" minOccurs="0" />
                    <xs:element name="IDConnectedNeuronLayer" type="xs:string" minOccurs="0" />
                    <xs:element name="IDConnectedNeuronSeq" type="xs:string" minOccurs="0" />
                   </xs:sequence>
                   </xs:complexType>
              </xs:element>
           </xs:sequence>
          </xs:complexType>
      </xs:element>
      </xs:sequence>
      </xs:complexType>
  </xs:element>
  </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```