A New XML-based Language for Neural Solution Interchange

Denis V. Rubtsov[1], **Sergei V. Butakov**[2]

[1]Altai State University
30 – 4 Gebler st., Barnaul 656099, Russia
rubtsov@math.dcn-asu.ru

[2]Altai State Technical University
49 – 5 Lenina st., Barnaul 656099, Russia
swb@agtu.secna.ru

Abstract

This article introduces a framework for interchange of trained neural network models. An XML-based language (Neural Network Markup Language) for the neural network model description is offered. It allows to write down all components of neural network model, which are necessary for its reproduction. We propose to use XML notation for full description of neural models, including data dictionary, properties of training sample, preprocessing methods, details of network structure and parameters, method for network output interpretation.

Introduction

Within the last decade artificial neural networks became a widely used technique for solving a variety of data analysis tasks. The number of neural-like models and schemas is increasing permanently as well as the number of their implementations in software and hardware simulators. But in our opinion there are a number of barriers, which makes practical application of neural networks quite problematic. One of them is determined by the lack of common approach to description and representation of neural models. At the moment there is no convenient way to distribute neural models between simulation systems.

As a possible solution of this problem we consider applying of an interchange electronic format that allows representing any neural models in a unified way. In this case each specific model would be exactly described by its creators in the terms of such a format for providing its reusing by other systems.

In the current work we have made an attempt to bring together most common ideas about neural network description and to develop rational and flexible framework for neural network exchanging format. The basic property of an interchange format is that it must contain all necessary information for unambiguous model reconstruction and its proper applying. Another important property is that it must be a portable and cross-platform language. Some discussion on the electronic neural network description may be found in (Kock and Serbedzija 1996, Smith 1996).

One of possible reasons for neural network interchange is exchange of neural solutions. In this work we treat a neural solution as a computational model, which is used for practical prediction and classification. Mapping of input signals to output ones in such a model is constructed according to the methods of neural network simulation. It is important that neural model doesn't consist of neural network only. It necessarily includes a set of input and output variables, which represents network environment, methods of its preprocessing and methods for sensible interpretation of network output signals.

It is significant that a *trained* neural network model is treated as an object for interchange. A neural net is considered as a static object with the fixed structure and parameters at the moment of interchange. In other words, a complete computational scheme for obtaining target (output) variables on the basis of independent (input) variables is transferred. The main purpose of this format is to represent all relevant information essential for unambiguous and exact reproduction of such a computational scheme by any neural simulator. That does not exclude further improving of this model by a simulator with its own training methods.

Our main goal is to develop a descriptive language that can provide a comprehensible way to write down any neural-like models with heterogeneous elements and arbitrary topology. Developing this format we aimed to make it extensible, easy to interpret and independent from hardware platforms or programming languages. As a result a language based on XML notation (World Wide Web Consortium 1998) has been developed. It may be named as Neural Network Markup Language (NNML). In further statement terms "format" and "language" are used as synonyms. In this article a following convention is accepted: real names of NNML tags are printed by `courier` font.

Related works

The problem of interchange format development is closely connected with neural network specification languages and investigation of formalization and standards for various neural network model components. Several neural network specification languages have been proposed, such as Aspirin (Leighton 1992), PlaNet v.5 (Miyata 1991), AXON (Hecht-Nilsen 1990), CuPit (Hopp and Prechelt 1997), EpsiloNN (Strey 1999), CONNECT (Kock and Serbedzija 1994), Nspec (Dorffner, Wiklicky and Prem 1993). They are based on notations, which are close to high level programming languages like C++ or Pascal and use similar methodology for neural system determination in terms of data structures, classes, variables and functions. Such a description can be used for code generation in programming languages or implemented on specialized neural simulator (possible, with parallel processing capabilities). These languages are aimed rather at description of simulation process than at exchange of networks. If one has intention to use such a language he must apply appropriate compiler and reconstruct his software to provide compatibility with underlying programming environment. Most of these languages are intended only for neural network definition with no taking into account specification of data dictionary, pre- and post processing etc.

Other type of language is presented at (The Data Mining Group 2000) as a part of Predictive Model Markup Language (PMML). Description with PMML provides extensive information about project, data dictionary and sample statistics. But it supports only backpropagation neural networks with few fixed types of neurons and uses a limited set of simple preprocessing techniques.

A few articles are devoted to formalization and standardization of neural systems. Fiesler et al (Fiesler and Caulfield 1994) propose hierarchical specification of known neural networks and introduce mathematical notation for their description. Smith (Smith 1996), Atencia et al (Atencia, Joya and Sandoval 2000) use more common mathematical constructions for definition of neural structures and dynamic. In (Strey 1999) an extended analysis of neural structures including biological neural networks is given. At great length all components of neural network simulation system are described in (Mirkes 1998). But analysis of those works has shown that the results they presented cannot be directly applied to neural network model exchanging.

Overview on NNML

The development of NNML is based on the following suppositions. Firstly we consider that significant parts of each neural model are:

- The problem and model purpose
- Data dictionary
- Data preprocessor
- Neural network
- Postprocessor
- Auxiliary information about model.

Information about all these components is significant for correct reconstruction of computational model of neural network. As shown in (Mirkes 1998), neural network can't be correctly reproduced without information about environment, in which it was created. In other words, the description of neural method for solving any practical problem, along with definition of neural network, must contain information about structure of data dictionary, preprocessing methods for each input data field, postprocessing methods for network output signals, extra information about current model.

Secondly, we treat a neural network in a broad sense, with minimum restrictions on topology and neuron's functionality. Suppose a neural network is a system of interconnected processing units, which may be represented as a directed graph. Nodes in such a graph designate neurons and branches mark connections between neurons. Directions of branches set signal transfer's directions in the net. Each unit performs a particular mathematical transformation on its inputs and may have its own parameters. It maps input vector of real numbers to output scalar real number. Weights of connections we consider as parameters of a neuron function.

Further we consider a preprocessor and a postprocessor also as sets of processing units, which have descriptions similar to neurons. The preprocessor receives input signals from an input data vector. Each preprocessing unit corresponds to particular neural net input. Each postprocessor unit describes a single target parameter of the problem neural model is purposed to. Thus the neural model has a fully modular structure where each object interacts with other objects only by passing his output signals to them (Fig.1).

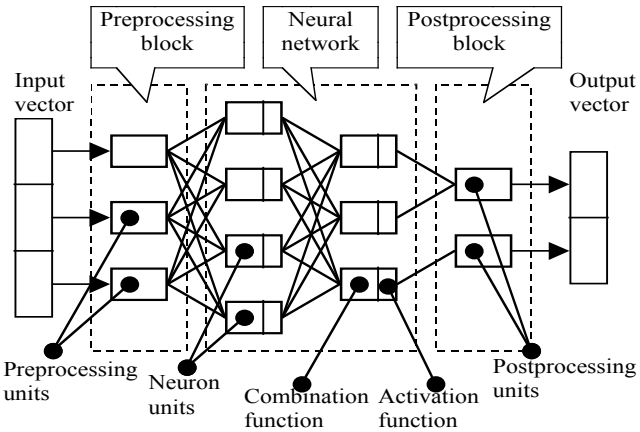

Figure 1: Representation of a neural model as a system of interconnected processing units

In essence, NNML description is a sufficiently structured document formed by hierarchy of nested sections with the

root element named `neuralmodel`. Each of high-level sections describes one basic aspect of neural solution. Neural model description on NNML has following main sections: **Header**, **Data**, **Preprocessing**, **Neural Network**, **Postprocessing**, **Training process description**.

In terms of XML section is an element, which is a basic unit of description. An element is formed by couple of named tags and may contain any data or child elements. Description of the element may be extended by a set of attributes.

Structure and content for each of sections have been developed on the basis of neural model decomposition with taking into account existing works in this field, in particular (Dorffner, Wiklicky and Prem 1993, Fiesler and Caulfield 1994, Mirkes 1998, Sarle 2001).

Initially, NNML was intended to description of the supervised learning neural networks. Now we work on its applying to unsupervised networks like self-organization maps. Further there is briefly describing content of the main NNML sections.

Content of the main NNML sections

Header

Element `header` mostly contains human-intended information. It includes three subsections: `problem`, `creator` and `project`. Subsection `project` describes task, which solving current neural model is intended for. Description fields are: name of domain, for example "financial time series prediction", name of specific task, for example "$/Euro rating forecast" and task description with explanation of essential detail. Subsection `creator` contains information about neural simulator, which produced current model (name and version), author's name, contact information, and copyright. Subsection `project` includes name of current model, identification field, date and time of model creation, comments of author.

Data

Section `data` contains a description of model's data dictionary. It is separated into data fields. A single data field is determined with following attributes:

- id – unique identifier for every data field,
- name of data field that can be displayed in the dialog with user, for example "$/DM rating twenty four hours ago",
- type of underlying variable: continuous, nominal, ordered,
- way of data field usage in the current model: as an input, an output or auxiliary field.

In addition to attributes data field contains `data_field_properties` tag. This tag describes underlying variable properties, for example permissible range of variable's values. For discrete (nominal or ordered) variables there is a description for each of its discrete states: id and name of discrete state, interval of numeric values assigned to discrete state.

Preprocessing

Applying of preliminary transformations to data sample is common practice for neural net training. Section `preprocessing` is intended to definition of methods, which are applied to raw input data to produce neural network input signals. Most of used preprocessing techniques are simple transformations of type "scalar-to-scalar" (various normalization and scaling transformations) or "scalar-to-vector" (various types of coding). In the latter case, several preprocessing output signals are corresponded to a single input data field. However more sophisticated preprocessing techniques may be used (Mirkes 1998).

To provide flexibility of the preprocessing description, we specify separate processing unit for each signal that preprocessing block passes to a neural network. Each preprocessing unit receives signals from one or more data fields and generates an output signal, which is passed to the neural network.

Neural network

Section `neural_network` contains description of neural network and its properties. Main purpose of this description is to simplify automatic recovering of underlying mathematical procedures. Because of the lack of standard terminology (and standard ways of decomposition) for neural networks, we have decided do not use terms like "synapse" or "axon". On the contrary, the network structure is recorded with common mathematical operations.

Neural network is defined as an object consists of connected processing units that exchange signals via links. There is a tag `layer`, which groups neurons (a fully connected network would have a one layer). Each neuron is represented as a couple of serially connected functions. First function combines inputs of neuron with the weights to yield a single value to which the activation function is applied. This is called a neuron combination function (in terms of (Sarle 2001)). Second function represents a neuron activation function and usually performs nonlinear transformation of combination function output. They are specified in the tags `combination_function` and `activation_function` respectively.

There are no special constraints on the structure of these functions. This means that a neuron may have any functioning, which may be defined with the combination of arithmetic operations or/and elementary classical functions. Varying these functions we can get different types of neurons. For example, using the scalar product of inputs and weights as a combination function and varying an activation function, we can describe some important types of neuron units, used in multilayer perceptrons, e.g. neurons with various sigmoid functions, with threshold, linear and piecewise-linear transfer functions. Application of various types of combination functions permits to

describe such kind of neurons, as radial basis functions (Euclidean distance), neurons of high degrees (high order polynomial) etc.

An output signal of the combination function is automatically passed to the activation function of the same neuron. Each neuron may be connected with other neurons (and with itself), preprocessing units and data fields. Weights and connections of neuron are set as arguments of combination function. Thus network structure is determined implicitly by references in combination function descriptions. This approach lets to describe networks with any topology. Neuron functioning may be specified with predefined NNML functions or user defined functions (see below). This clears the way to specification of heterogeneous neural networks.

The length of the network description may be reduced. If all neurons in a layer have the same type of combination or activation function, its description may be placed as a first tag within `layer` tag.

Along with neural network structure, tag `neural_network` contains rules for network dynamic.

Subsection `neural_network_properties` includes common information about current network: a type of structure (feedforward, feedback), neural paradigm (e.g. multilayer perceptron, radial basis function network), and comments of creator.

Postprocessing

Typically, a neural net produces a set of real numbers, which needs to be correctly interpreted. The `postprocessing` section is intended to provide information about methods of the substantial answer delivering on the basis of the network target signals. The methods of network output signal transformation are defined in the same way as preprocessing methods. Usually, denormalization (for regression), or rule "winner takes all" (for classification) are used, but more complex methods are also allowable.

A neural network could return several answers. Therefore methods of postprocessing are grouped by `output_field` tags, each of which corresponds to one of the model target parameters (not to one of the network output neurons!). The tag `output_field` contains the reference to appropriate target data field described in the data section. As input signals for postprocessing blocks, signals of network output neurons are used.

Each target field may include tag `verbalization`. It is intended for association text information with certain values (groups of values) of target parameter, for example: if target parameter is in [2.5,3.5] then the network answer is "satisfactorily".

A Framework for Description of Processing Units

Basic components of model – preprocessor, neural network, postprocessor – are described by processing units and representation of their underlying mathematical transformations is a key for NNML.

A proposed framework for representation of processing units consists of following. There are basic operands of two types: scalar number (constant) and reference (connection) to output of one of model objects. All transformations are applied to them.

1. The number is set by `number` tag, which has attribute `name` to designate argument semantic, for example "weight" or "bias". In general this tag contains a real number. All numerical parameters of model – weights of neurons, parameters of activation functions, parameters of pre- and postprocessing blocks – are written down by this tag.
2. The connection is set by empty `connect` tag. It has following attributes: `object` – contains the name of signal source type (a name of one of next tags: `preprocessing_field`, `neuron`, `data_field`), `id` – identification number of the particular object of specified type, `layer` – id of layer – for object of neuron type. `connect` tag serves for description of connections between processing units, which contain in pre- and postprocessing blocks and neural network.

There are three levels of function specification:

- Using basic functions. The set of elementary functions is introduced to define various mathematical transformations. It includes unary and n-ary arithmetic operations (sum, multiplication, raising to power, division, capture of absolute value etc.), trigonometric operations, logarithmic operations (logarithm, exponent), relations (more, less, equally etc), if-then condition, matrix operations (addition, scalar product etc) and piecewise function. The expression that has been written down by basic functions is contained in `explicit_record` tag. The basic operations may be performed on the elementary operands as well as on the results of basic operations. It means, that the tags of basic functions can contain other basic functions or elementary operands. The analysis of such expressions is carried out by construction of a computational tree. Terminal nodes of this tree always should contain one of elementary operands. Hierarchy of nesting tags determines priority of operations: all "internal" functions are calculated first without taking into account their type.
- Using the predefined functions. The set of predefined functions is intended to decrease the length of description. It includes most popular in neural simulation preprocessing functions (normalization, coding), neuron' combination functions (adaptive summator, square summator, euclidean distance etc.), activation functions (various kinds of sigmoids, the threshold function, linear and piecewise-linear function, exponent etc), methods of postprocessing (denormalization, winner takes all). Description of all the predefined functions may be found on the NNML site (Rubtsov 2001).
- Functions defined by user. User can introduce his own functions. In the body of the model description `user_function` tag is used with parameters: `name` – the name of user function, `source` – the Internet address,

which provides information about content of transformation. Within this tag only special tags `arg` is allowed. `arg` tags have attribute `id` and could contain basic operand tags as a `number` or `connect`. Semantics of transformations given by user is not determined in any way.

Applications of NNML

Now we suppose following applications of NNML:

- Use it as an interchange format in the large information systems. The technological approach to usage of neural network in applied information systems is preferred (Rubtsov 2000). We suppose that NNML allows specialization of the neural network simulation programs: separation of neural networks generators, interpreters, tools for visualization and knowledge extraction. It could facilitate effective documenting and storing models as well as introduction of neural network facilities into existing information systems.
- Exchange of neural models via Internet. NNML can be applied to distribution of neural network models on WWW. It is possible to implement usage of remote machines for producing of neural network models "by request", maintaining global archives to share neural network solutions for various problems. As an XML-based language, NNML could be easily used for introduction of neural network facilities on Web applications.

Conclusion

In this article we have presented an XML-based language for the description of neural network models – NNML. It allows to describe neural network model completely, including data dictionary, pre- and postprocessing, details of structure and parameters of neural network, and auxiliary information. Proposed framework provides features to write down a wide range of pre- and postprocessing procedures, and describe neural network of any topology with heterogeneous elements. Most widespread neural networks (such as multilayer perceptrons and radial basis function networks) can be described in a compact way. A user has an ability to use any functions for designing of neurons. NNML provides open and extensible description of neural models.

It is possible to find out the last version of NNML Document Type Definition and other related materials on the NNML site (Rubtsov 2001).

Acknowledgments. This research was partially supported by the Russian Foundation for Basic Research grant 01-01-01046

References

Kock, G. and Serbedzija, N.B. 1996. Simulation of Artificial Neural Networks, *Systems Analysis – Modelling – Simulation (SAMS)*, 27(1):15–59

Smith, L. 1996. Using a framework to specify a network of temporal neurons, Technical Report, University of Stirling,

Leighton, R.R. 1992. The Aspirin/MIGRANES neural network software, Users manual, MITRE Corp.

Miyata, Y. 1991. A User's Guide to PlaNet Version 5.6 – A Tool for Constructing, Running, and Looking in to a PDP Network, Computer Science Department, University of Colorado, Boulder

Hecht-Nilsen, R., 1990. *Neurocomputing.:* Addison-Wesley

Hopp H. and Prechelt, L. 1997. CuPit-2: A portable parallel programming language for artificial neural networks, In *Proc. 15th IMACS World Congress on Scientific Computation, Modelling, and Applied Mathematics*, 6:493–498, Wissenschaft & Technik Verlag, Berlin

Strey, A. 1999. EpsiloNN – A Tool for the Abstract Specification and Parallel Simulation of Neural Networks, *Systems Analysis – Modelling – Simulation (SAMS)*, 34-40, Gordon & Breach

Kock, G. and Serbedzija, N.B. 1994. Artificial neural networks: from compact descriptions to C++, In *M. Marinaro and P.G. Morasso, editors, Proc. of the International Conference on Artificial Neural Networks*, 1372–1375, Springer-Verlag

Dorffner, G., Wiklicky, H. and Prem, E. 1993. Formal neural network specification and its implications on standardization, Technical Report OFAI TR-93-24, Austrian Research Institute for Artificial Intelligence

The Data Mining Group, 2000. *PMML 1.1 Neural Network* www.dmg.org/html/neuralnetwork.html

Fiesler, E. and Caulfield, H.J. 1994. Neural network formalization, *Computer Standards and Interfaces*, 16(3): 231–239

Atencia, M. A., Joya, G. and Sandoval, F. 2000. A formal model for definition and simulation of generic neural networks, *Neural Processing Letters,* 11: 87–105 Kluwer Academic Publishers

Strey, A. 1999. A Unified Model for the Simulation of Artificial and Biology-Oriented Neural Networks, In *Proc. of the International Workshop on Artificial Neural Networks* 2:1–10

Mirkes, _._. 1998. *Neurocomputer. Project of standard.:* Nauka (in Russian)

World Wide Web Consortium 1998. Extensible Markup Language (XML) 1.0, W3C Recommendation, www.w3.org/TR/1998/REC-xml-19980210

Sarle, W. 2001. Frequent asked question on neural network, ftp.sas.com/pub/neural/FAQ.html

Rubtsov, D. 2000. Development of technology of artificial neural networks application in applied information systems, Ph.D. diss., Dept. of Computer Science, Altai State University (in Russian)

Rubtsov, D. 2001. Neural Network Markup Language Home Page, www.nnml.alt.ru