ITHEA

International Journal INFORMATION THEORIES & APPLICATIONS

2008 Volume 15 Number 1

International Journal INFORMATION THEORIES & APPLICATIONS

Volume 15 / 2008, Number 1

IJ ITA is official publisher of the scientific papers of the members of the ITHEA International Scientific Society, the Association of Developers and Users of Intellectualized Systems (ADUIS) and the Association for Development of the Information Society (ADIS)

IJ ITA welcomes scientific papers connected with any information theory or its application.
IJ ITA rules for preparing the manuscripts are compulsory.
The **rules for the papers** for IJ ITA as well as the **subscription fees** are given on *www.foibg.com/ijita*.
The camera-ready copy of the paper should be received by e-mail: *info@foibg.com***.**
Responsibility for papers published in IJ ITA belongs to authors.

General Sponsor of IJ ITA is the **Consortium FOI Bulgaria** (www.foibg.com).

International Journal "INFORMATION THEORIES & APPLICATIONS" Vol.15, Number 1, 2008

Printed in Bulgaria

Edited by the **Institute of Information Theories and Applications FOI ITHEA**, Bulgaria,
in collaboration with the V.M.Glushkov Institute of Cybernetics of NAS, Ukraine,
and the Institute of Mathematics and Informatics, BAS, Bulgaria.

Publisher: **Institute of Information Theories and Applications FOI ITHEA**
Sofia, 1000, P.O.B. 775, Bulgaria. www.ithea.org, www.foibg.com, e-mail: info@foibg.com

ISSN 1310-0513 (printed) **ISSN 1313-0463 (online)** **ISSN 1313-0498 (CD/DVD)**

XV^{-th} anniversary of IJ ITA !

PREFACE

Verba volant, scripta manent !

The "**International Journal on Information Theories and Applications**" (IJ ITA) has been established in 1993 as independent scientific printed and electronic media. IJ ITA is edited by the *Institute of Information Theories and Applications FOI ITHEA* in collaboration with the leading researchers from the Institute of Cybernetics "V.M.Glushkov", NASU (Ukraine) and Institute of Mathematics and Informatics, BAS (Bulgaria).

During the ***15*** years, IJ ITA became as well-known international journal. Till now, including this volume, ***686*** papers have been published. IJ ITA authors are widespread in ***41*** countries all over the world: *Armenia*, *Belarus*, Brazil, Belgium, *Bulgaria*, Canada, Czech Republic, Denmark, *Egypt*, Estonia, Finland, France, *Germany*, Greece, Hungary, *India*, *Iran*, *Ireland*, *Israel*, Italy, Japan, Kirghizia, Latvia, Lithuania, Malta, Mexico, *Moldova*, Netherlands, *Poland*, Portugal, Romania, *Russia*, Scotland, Senegal, Serbia and Montenegro, *Spain*, Sultanate of Oman, Turkey, *UK*, *Ukraine*, and USA.

The great success of IJ ITA belongs to the whole of the ITHEA International Scientific Society. We express our thanks to all authors, editors and collaborators who had developed and supported the International Journal on Information Theories and Applications.

Congratulations to Prof. ***Volodimir Donchenko*** *(Ukraine) who was awarded by the International Prize "**ITHEA**" for the year 2007. The "ITHEA" Prize has been established in 1995. It is aimed to mark the achievements in the field of the information theories and applications.*

Volume 15/2008 of the IJ ITA contains ***61*** papers written by ***118*** authors from ***15*** countries (marked in italics above), selected from several international conferences, seminars and workshops organized or supported by the Journal.
At the first place, the main source for selection were the ITA 2007 Joint International Events on Information Theories and Applications, (June 17- July 8, 2007, Varna, Bulgaria):

- XIII-th International Conference "Knowledge-Dialogue-Solution" (KDS 2007);
- V-th International Conference "Information Research and Applications" (i.TECH 2007);
- Second International Conference on Modern (e-) Learning (MeL 2007);
- International Conference on e-Management & Business Intelligence (eM&BI 2007);
- VI-th International Workshop on General Information Theory (GIT 2007);
- Second Int. Workshop on Cyber Security (CS 2007).

Several papers were selected from the pool of papers directly submitted to IJ ITA.

More information about the IJ ITA rules for preparing and submitting the papers as well as how to take out a subscription to the Journal may be obtained from www.foibg.com/ijita .

Krassimir Markov
IJ ITA Founder and Editor in chief

International Prize "ITHEA"

Awarded Scientists till 2007:

1995	*Sandansky*	*K. Bankov, P. Barnev, $\boxed{\text{G. Gargov}}$, V. Gladun, R. Kirkova, S. Lazarov, $\boxed{\text{S. Pironkov}}$, $\boxed{\text{V. Tomov}}$*
1996	*Sofia*	*T. Hinova, K. Ivanova, I. Mitov, $\boxed{\text{D. Shishkov}}$, N. Vashchenko*
1997	*Yalta*	*Z. Rabinovich, V. Sgurev, A. Timofeev, A. Voloshin*
1998	*Sofia*	*V. Jotsov*
1999	*Sofia*	*L. Zainutdinova*
2000	*Varna*	*I. Arefiev, A. Palagin*
2001	*St.Peterburg*	*N. Ivanova, $\boxed{\text{V. Koval}}$*
2002	*Primorsko*	*A. Milani, M. Mintchev*
2003	*Varna*	*T. Gavrilova, A. Eskenazi, V. Lozovskiy, P. Stanchev*
2004	*Varna*	*B. Kokinov, T. Vamos*
2005	*Varna*	*L.F. de Mingo, M. Dobreva*
2006	*Varna*	*J. Castellanos, G. Totkov*
2007	*Kiev*	*V. Donchenko*

IJ ITA major topics of interest include, but are not limited to:

INFORMATION THEORIES

Artificial Intelligence
- *Computer Intellectualization*
- *Intelligent Networks and Agents*
- *Intelligent Technologies*
- *Knowledge Discovery and Engineering*
- *Knowledge Acquisition and Formation*
- *Distributed Artificial Intelligence*
- *Models of Plausible Reasoning*
- *AI Planning and Scheduling*

Bioinformatics
Business Informatics
Cognitive Science
Decision Making
Education Informatics
General Information Theory
Hyper Technologies
Information Models
Intellectualization of Data Processing
Knowledge-based Society
Logical Inference
Natural language Processing
Neuroinformatics
Philosophy and Methodology of Informatics
Quality of the Programs
Software Engineering
Theory of Computation

APPLICATIONS

Business Information Systems
Communication Systems
Computer Art and Computer Music
Hyper Technologies
Intelligent Information Systems
Multimedia Systems
Programming Technologies
Program Systems with Artificial Intelligence
Pyramidal Information Systems
Very Large Information Spaces

SELFSTRUCTURIZED SYSTEMS

Victor Gladun, Vitalii Velychko, Yurii Ivaskiv

Abstract: The problems of constructing the selfsrtucturized systems of memory of intelligence information processing tools, allowing formation of associative links in the memory, hierarchical organization and classification, generating concepts in the process of the information input, are discussed. The principles and methods for realization of selfstructurized systems on basis of hierarchic network structures of some special class – growing pyramidal network are studied. The algorithms for building, learning and recognition on basis of such type network structures are proposed. The examples of practical application are demonstrated.

Keywords: knowledge discovery, classification, prediction, growing pyramidal networks, concept formation.

ACM Classification Keywords: I.2.4 Knowledge Representation Formalisms and Methods - Semantic networks, F.1.1 Models of Computation - Self-modifying machines (e.g., neural networks)

Introduction

The task of constructing the self structurized systems is considered in context with intellectualization of the information processing tools. Selfstructurization provides a possibility of changing the structure of data, stored in memory, in the process of the tools functioning as a result of interaction between the received and already stored information. Systems in which the perception of new information is accompanied by simultaneous structurization of the information stored in memory, we shall name hereinafter selfstructurized.

Development of principles and methods of constructing the selfstructurized systems in many respects defines a possibility of intellectualization of the information processing tools. Adaptability to the task, being solved, has to do with changing the structure of data. As a result, the possibility of searching in the memory focused on storage of complex data of the large volume occurs, which allows increasing productivity of the used tools, raise accuracy and reliability of received results.

Main processes of structurization of the perceived information consist in formation of semantic and syntactic links among objects by separation of crossings of their attributive representations, as well as the generalized logic attributive models of classes of objects - concepts. As a result of these processes realization, the semantic and syntactic similarity of the perceived information with the stored information is established. Detected associations are fixed as structural changes in memory.

Following basic requirements to data structures in the intellectual systems are set for the decision of such tasks as regularity discovery, classification, forecasting, diagnostics [1]:

The structure of the data should be the multiple-parameter model, reflecting significant properties of researched object. It should provide a possibility to account for the simultaneous influence on researched factor of various combinations of known properties of the researched object.

The model of the researched object should minimize scanning of large-scale data: along with growth of data size, the time of performing of the choice operations grows. It interferes with application of some analysis methods. The model also should be applicable for verification and interpretation.

It should be noted, that in solving tasks of diagnostics and forecasting the models characterized by higher level of generalization of models of classes of objects have advantage. The logic expressions describing such models turn out easier if the complexity is evaluated by number of variables. Simplification of logic expressions results in simple structure of memory and, therefore, simplifies the process of structurization.

In knowledge representation in intelligent systems, those network structures have advantages, which have some information units in vertices, and arches describing links among them. In similar systems, the elements of knowledge representation are combined in the hierarchical structure, realizing such functions, as formation of links among attributive presentations of researched object by allocation of their crossings, hierarchical ordering, classification, concepts formation. In selfstructurized systems, such functions should be performed in the process of the information perceiving.

Condition of an element formation of network structure, for example, unit or link between units, is some relation between determined structural elements of a network. The relations determining formation of structure elements of selfstrtucturized systems we call structurized.

There are two basic ways of objects representation in the information processing systems: by name (condensed) or by sets of attribute values (expanded). The memory structures in selfstrtucturized systems and the appropriate network structures should provide bidirectional conversion between such representations.

Building of selfstrtucturized systems is proposed to be realized on basis of network with hierarchical structures, named as growing pyramidal networks (GPN) [5].

The theory as well as practical application of GPN is expounded in a number of publications [3-6]. GPN realization has following stages:

- to construct the structure of a network for some initial set of objects, assigned by attributive descriptions,
- to train the structure, with a purpose to allocate its elements, allowing to classify all objects of the initial set,
- to recognize belonging to some class of objects of certain object, which is not belonging to initial set of objects.

The mechanisms, providing conversion between converged representation of objects and representation as a set of attributes values in human neurosystem, are discussed in the article [2]. The present work illustrates recent versions of algorithms for building and training GPN, as well as examples of their application.

Building of GPN

A *growing pyramidal network* is an acyclic oriented graph having no vertexes with a single incoming arc. Examples of the pyramidal networks are shown in Figs.1,2,3. Vertices having no incoming arcs are referred to as *receptors*. Other vertices are called *conceptors*. The subgraph of the pyramidal network that contains vertex *a* and all the vertices from which there are paths to vertex *a* is called the *pyramid* of vertex *a*. The set of vertices contained in the pyramid of vertex *a* is referred to as the *subset* of vertex *a*. The set of vertices reachable by paths from vertex *a* is called the *superset* of vertex *a*. The set of vertex, having paths from vertex *a*, is referred to its *superset.*

In *subset* and *superset* of the vertex, 0-*subset* and 0-*superset* are allocated, consisting of those vertices, which are connected to it directly. When the network is building, the input information is represented by sets of attributes values describing some objects (materials, states of the equipment, a situation, illness etc.). Receptors correspond to values of attributes. In various tasks, they can be represented by names of properties, relations, states, actions, objects or classes of objects. Conceptors correspond to descriptions of objects in general and to crossings of descriptions and represent GPN vertices.

Initially the network consists only of receptors. Conceptors are formed as a result of algorithm of construction of a network. After input of object attribute description, corresponding receptors switch to a *state of excitation*. The process of excitation propagates through the network. A conceptor switches into the state of excitation if all vertices of its 0-subset are excited. Receptors and conceptors retain their state of excitation during all operations of network building.

Let F_a be the subset of excited vertices of the 0-subset of vertex *a*; *G* be the set of excited vertices in the network that do not have other excited vertices in their supersets. New vertices are added to the network by the following two rules:

Rule A1. If vertex *a*, that is a conceptor, is not excited and the power of set F_a exceeds 1, then the arcs joining vertices of set F_a with the vertex *a* are liquidated and a new conceptor is added to the network which is joined with vertices of set F_a by incoming arcs and with the vertex *a* by an outgoing arc.

The new vertex is in the state of excitation. Rule A1 is illustrated in Fig.1 (a,b). According to the Rule A1, the condition for adding a new vertex to the network is a situation, when certain network vertex is not completely excited (at least two vertices of 0-subset are excited). Fig. 1.a shows a fragment of network in some initial state. Receptors 4,5 switch to a state of excitation, the network switches to state Fig. 1.b, a new vertex appears – a new conceptor. Receptors 2,3 switch to a state of excitation additionally. The network switches to state Fig. 1.c.

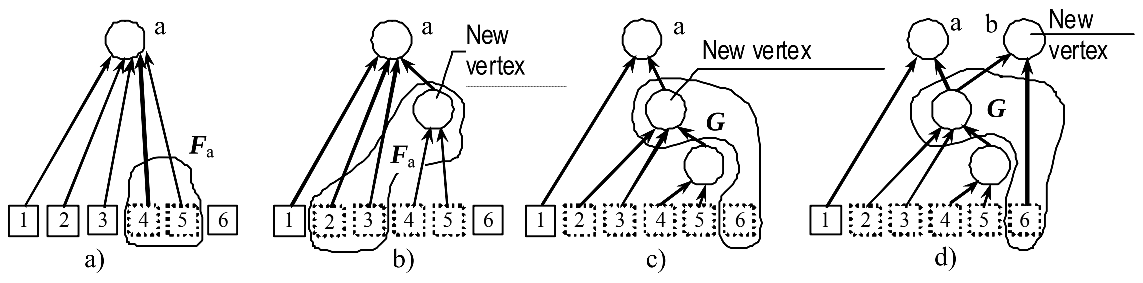


Fig. 1.

New vertices are inserted in 0-subset of vertices, which are not completely excited. New vertices correspond to intersection of object descriptions, represented by incoming arches. Once new vertices have been introduced into all network sections where the condition of rule A1 is satisfied, rule A2 is applied to the obtained network fragment, concluding the object pyramid building.

Rule A2. If the power of set *G* exceeds 1 element, a new conceptor is added to the network, which is joined with all vertices of set *G* by incoming arcs.

The new vertex is in the state of excitation. Rule A2 is illustrated in Fig.1c,d. Network Fig1.d was obtained after the excitation of receptors 2-6.

In applying the Rule A1 the main cross-linking relation is a relation of intersection of receptor set, excited by input of the object description and other sets of receptors included into pyramid, recently formed by conceptors. Rule A2 concludes the building of pyramid, which represents complete description of the introduced object.

Pyramidal networks are convenient for execution of various operations of associative search.

For example, it is possible to select all the objects that contain a given combination of attribute values by tracing the paths that outgo from the network vertex corresponding to this combination. To select all the objects whose descriptions intersect with the description of a given object it is necessary to trace the paths that outgo from vertices of its pyramid. Rules A1, A2 establishes associative proximity between objects having common combinations of attribute values.

Hierarchical organization is an important property of pyramidal networks. This provides a natural way for reflecting the structure of complex objects and generic-species interconnections.

Conceptors of the network correspond to combinations of attribute values that define separate objects and conjunctive classes of objects. By introducing the excited vertices into the object pyramid, the object is referred to classes, which descriptions are represented by these vertices. Thus, during the network building the conjunctive classes of objects are formed, the classification of objects is performed without a teacher. Classifying properties of pyramid network are vital for modeling environments and situations.

Conversion from converged representation of objects (conceptors) to expanded (sets of receptors) is performed by scanning pyramids in top-down and down-top directions.

Training GPN

Training GPN consists in formation of the structures representing concepts, on a basis of attributive descriptions of the objects incorporated into classes with known properties.

Concept is an element of knowledge system, representing generalized logic attributive model of a class of objects, by which processes of recognition of objects are realized. The set of objects generalized in concept is its *volume*.

Consider a task of inductive formation of concepts for not intersected sets of objects $V_1, V_2, ..., V_n$, each set represents some class of objects with known properties. Let L - be a set of objects used as training set. All the objects of set *L* are represented by sets of attribute values. Relations $L \cap V_i \neq \varnothing$ and $V_i \not\subset L(i = 1,2,...,n)$ are set. Each object from set *L* corresponds to one set V_i. It is necessary to generate *n* concepts by analysis *L*. The amount of these concepts must be sufficient for correct recognition of belongings of anyone $l \in L$ to one of sets V_i.

Each concept, generated on the basis of training set, is approximation to real concept, the proximity of concepts depends on representativeness of training set, i.e. on the detalization of peculiarities of the concept volume.

In forming the concept corresponding to set V_i, the objects of training set included in V_i**,** are considered as examples of set V_i, and the objects, not included in V_i, - as counterexamples of set V_i.

The combinations of attributes allocated in building of a pyramidal network, representing descriptions of objects of training set, are used as "a building material", a basis of further logic structure of concept.

Let *L* be the pyramidal network representing all of training set objects. For formation of concepts $A_1, A_2, ..., A_n$, corresponding to sets $V_1, V_2, ..., V_n$, pyramids of all objects of training set are scanned in order. The vertices of scanned pyramid during its scanning are considered excited. Special vertices in network are identified in order to recognize objects from the concept volume. They are referred to as *check vertices* of a certain concept. In selecting the check vertexes, two characteristics of network vertexes are used: $\{m_1, m_2, ..., m_n\}$, where $m_i\ (i = 1,2,...,n)$ is a number of objects of volume of concept A_i, which pyramids include the given vertex; and *k*, which is the number of receptors in the pyramid of this vertex. For receptors *k*=1. While scanning, the pyramid is transformed by the following rules.

Rule B1. If in the pyramid of an object from concept volume A_i, the vertex, having the largest *k* among all the vertices with the largest *m_i*, is not a check vertex of concept *A_i*, then it is marked as a check vertex of the concept *A_i*.

The rule allows existence several vertexes among the excited vertexes with identical m_i, exceeding m_i of other excited vertexes. If in group of the vertexes having largest m_i, values *k* of all vertexes are equal, any of vertexes can be marked as check vertex of concept A_i.

The rule B1 is illustrated in Fig. 2. In a situation demonstrated by Fig. 2, in excitation in pyramid of vertex 2 vertex 6 is selected as check vertex as having the largest *k* among vertices with the largest m_i (6, 13, 14). Values m_i are shown inside symbols of vertices.

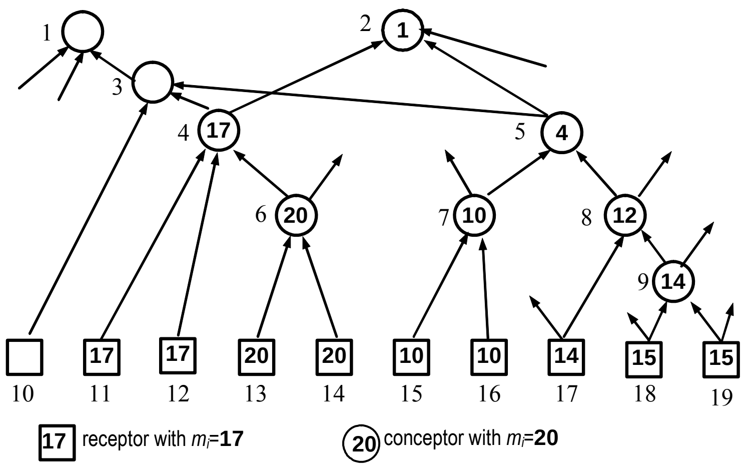


Fig. 2.

Rule B2. If the pyramid of an object from concept volume A_i contains check vertices of other concepts whose supersets do not contain excited check vertices of concept *A_i*, then in each of these supersets the vertex, having the largest *k* among all excited vertices with the largest *m_i*, is marked as a check vertex of concept *A_i*.

According to this rule the excitation of the pyramid of vertex 2 (Fig.3.a) on the condition, that it represents an objects from concept volume *A_i*, results in choosing vertex 5 as the check vertex of concept *A_i* (Fig. 3.b).

By check vertexes we select the most typical (having the largest m_i) combinations of attribute values, belonging to objects from concept volume. For example, selecting the vertex 8 (Fig 3a.) as a check vertex means selection of combination of value attributes, corresponding to receptors 17,18,19.

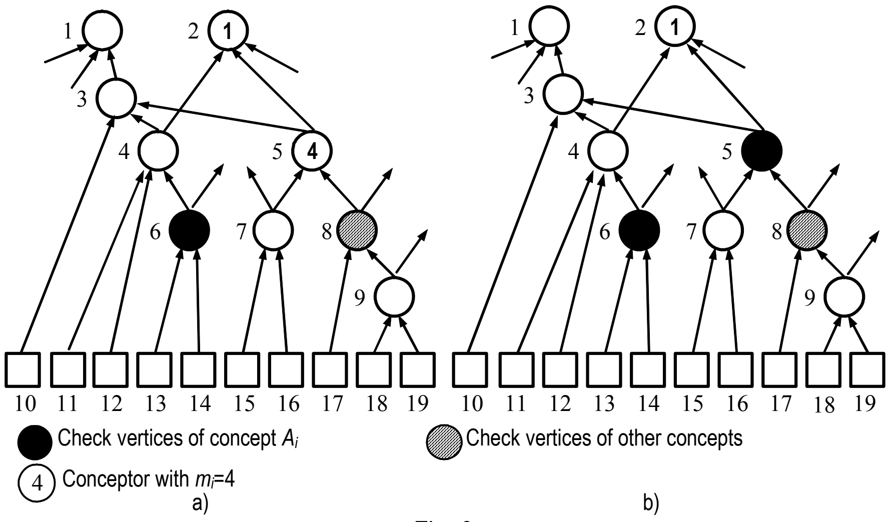


Fig. 3.

If at least one new check vertex appears while scanning objects of the training set, i.e. conditions of Rules B1 or B2 have been performed once at least, the training set is rescanned. The algorithm stops if during the scanning of the training set no new check vertex appears.

Recognition on basis of GPN

The task of recognition is based on the following rule.

Certain object belongs to the concept volume A_i if its pyramid has check vertexes A_i and does not contain check vertices of any other concept not having excited check vertices of concept A_i concept in their supersets. If this condition does not hold for any of the concepts, the object is referred to as unrecognized.

The execution time of the above algorithm is always finite. If the volumes of the formed concepts $V_1, V_2 ..., V_i, ..., V_n$ do not intersect, than after execution the algorithm the recognition rule completely divides the training set into subsets $L_i = V_i \cap L(i = 1...n)$.

The formed concepts are represented in the network as ensembles of check vertexes.

There is an algorithm of composing the logic descriptions of concepts, formed in the network as a result of the training process, described above. The formed logical expression contains logical relations, represented by allocation of check vertexes, describing the concepts in the network, defining different classes of objects.

The analytical tasks, such as diagnostics or prognosis, can be reduced to the task of classification, i.e. to belongings the research object to a class of objects, with a property characteristic or a set of properties significant for diagnostics of prognosis.

GPN Application

The following example illustrates the result of concepts formation on the basis of the analysis of a fragment of training set shown in the table 1.

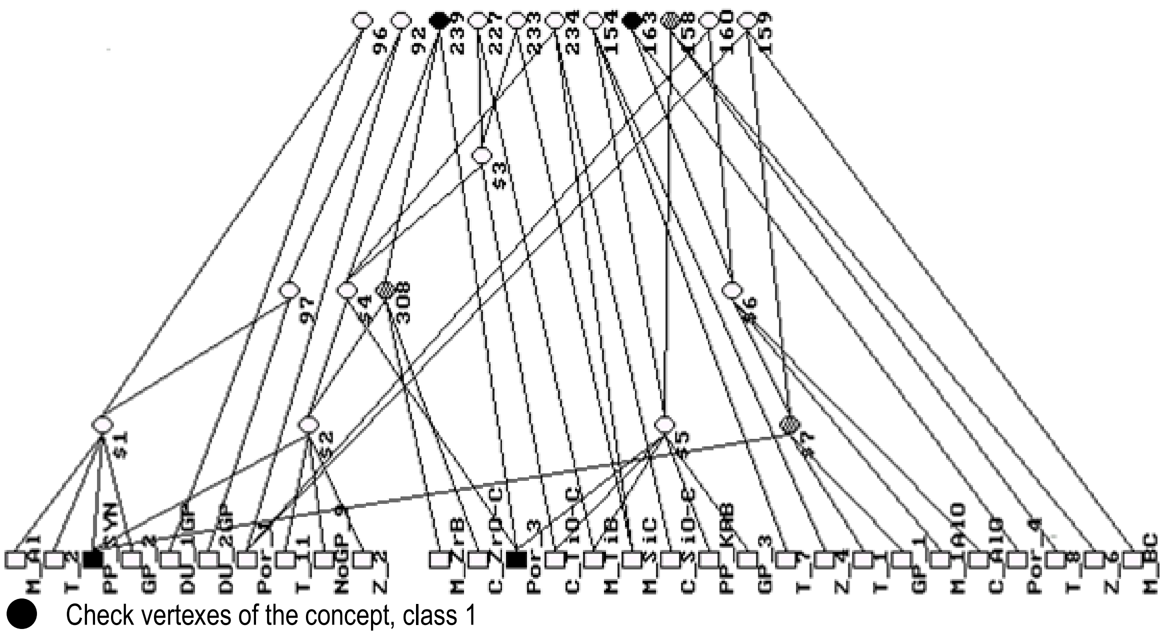
The table has descriptions of ceramic materials of two classes with the following attributes: M - material, T - fineness of powder, C - mix proportion, PP – powder manufacturing method, GP - conditions of obtaining the sample at hot pressing, NoGP - conditions of obtaining the sample without hot pressing, DU - special conditions of manufacturing of a sample, Por - porosity, Z - granularity.

Letters and figures in sections specify values of the appropriate attributes.

Fig. 4 demonstrates the appropriate pyramidal network with the formed concepts. Check vertices PP_SYN, Por_3, 239, 163 characterize class 1, check vertexes 158, 308 and $7 characterize class 2.

Table 1. Training set.

Object	Class	M	T	C	PP	GP	NoGP	DU	Por	Z
97	1	Al	2		SYN	2		2GP		
96	1	Al	2		SYN	2		1GP		
92	1	Al	2		SYN	2		2GP	1	
227	1	TiB	11	TiO-C	SYN		9		3	2
228	1	TiB	11	TiO-C	SYN		9		3	2
229	1	TiB	11	TiO-C	SYN		9		3	2
233	1	SiC	11	TiO-C	SYN		9		3	2
234	1	SiC	11	SiO-C	SYN		9		3	2
235	1	SiC	11	SiO-C	SYN		9		3	2
237	1	SiC	11	SiO-C	SYN		9		3	2
239	1	ZrB	11	ZrO-C	SYN		9		3	2
240	1	ZrB	11	ZrO-C	SYN		9		3	2
241	1	ZrB	11	ZrO-C	SYN		9		3	2
242	1	ZrB	11	ZrO-C	SYN		9		3	2
154	1	TiB	7	TiO-C	KRB	3			3	4
156	1	TiB	7	TiO-C	KRB	3			3	4
163	1	1AlO	1	AlO	SYN	1			4	
158	2	TiB	8	TiO-C	KRB	3			3	6
160	2	1AlO	1	AlO	SYN	1			1	
159	2	BC	1		SYN	1			1	
308	2	ZrB	11	ZrO-C	SYN		9			2



● Check vertexes of the concept, class 1

⊗ Check vertexes of the concept, class 2

○ Conceptor, which is not check vertex

Fig 4.

The class 1 is described by the following logical expression, where $\vee$, $\wedge$, $\neg$ - logical operations of a disjunction, conjunction and negation:

$$\begin{aligned}&PP\_SYN \wedge \neg\{T\_1 \wedge GP\_1\} \wedge \neg\{M\_ZrB \wedge C\_ZrO\text{-}C \wedge T\_11 \wedge NoGP\_9 \wedge Z\_2\} \vee \\ &Por\_3 \wedge \neg\{T\_8 \wedge Z\_6 \wedge M\_TiB \wedge C\_TiO\text{-}C \wedge PP\_KRB \wedge GP\_3\} \vee \\ &M\_ZrB \wedge C\_ZrO\text{-}C \wedge T\_11 \wedge PP\_SYN \wedge NoGP\_9 \wedge Por\_3 \wedge Z\_2 \vee \\ &M\_1AlO \wedge T\_1 \wedge C\_AlO \wedge PP\_SYN \wedge GP\_1 \wedge Por\_4.\end{aligned}$$

The logic expressions, defining various classes of objects, are united in *cluster databases* (CDB). CDB contain the information on groups of objects (clusters), specific to the area of study. On basis CDB problems of classification, diagnostics and forecasting are solved. After the concept for some class of objects has been formed, problems of forecasting and diagnostics are reduced to a problem of classification. Classification of new objects is performed by comparing the attribute descriptions with the concept, defining a class of predictable or diagnosing objects. Objects can be classified by evaluating the value of the logical expressions that represent corresponding concepts. The variables, corresponding to the attribute values of the recognized object, set 1, other variable set 0. If the entire expression takes the value 1, that means that the object is included into volume of concept.

The next geometric interpretation of the concept formation algorithm can be proposed.

Every network vertex, having *k* receptors in its subset, corresponds to $(s-k)$-dimensional plane in *s*-dimensional attribute space. The plane contains all the points corresponding to objects whose perceiving results in exiting of this vertex. $(s-k)$-dimensional planes corresponding to check vertices of concept A_i are referred to as *zones* of concept A_i.

The following statements are true for growing pyramidal networks.

Statement 1. The zone of any network vertex is totally included in zones of its subset vertices and totally includes all zones of its superset vertices.

Statement 2. The point corresponding to an object in the attribute space is located inside an intersection of zones of those check vertices, which are exited when the object is perceived.

Point *a* corresponding to the object in the attribute space is directly included in the zone *Z* of concept A_i if there is no other zones of this concept which include point *a* and totally are included in zone Z.

The geometric interpretation of the above-described rules for concept formation algorithm is as follows.

Rule B1. For every object of concept volume A_i $(s-k)$-dimensional plane of the exited vertex having the highest *k* among all the vertices with the highest m_i becomes the zone of concept A_i.

Rule B2. If the point, corresponding to an object of concept volume A_i in the attribute space, is directly included in zones of the other concepts, then a zone of concept A_i is created inside each of those zones.

The algorithm of concept formation stops, when during regular examination of the training set, points corresponding to objects from any class are not directly included in zones of the other concepts. When learning is finished, an object corresponds to concept volume A_i if the appropriate point in the attribute space is directly included in at least one zone of concept A_i and is not included in any zone of the other concepts.

Zones of concept A_i, directly inclusive points of objects, corresponding to objects from its volume, as well as points, corresponding to objects from different concepts, are referred to as *boundary zones* of concept A_i.

Statement 3. According to Rule B2 new zones can be created only directly inside boundary zones.

Formation of new zones inside boundary zones results in division of boundary zones.

Construction of approximating region for concept A_i consists of two processes: rough covering with concept A_i *i* zones the distribution domain of training set objects corresponding to concept A_i (Rule B1); and division of arising boundary zones (Rule B2).

On the basis of geometrical interpretation, algorithm convergence can have the following explanation.

For each concept the total covering by zones of allocation area of the training set objects, which are included in its volume, results in scanning of all objects, i.e. during single scanning of training set. The boundary zones include points of objects of training set, for which conditions of Rule B2 work. Therefore in every scanning of training set the division of all boundary zones, formed by previous scanning, occurs.

Process of division of boundary zones proceeds, as long as boundary zones exist, and can result in allocation of separate points of attribute space as zones. As number of the points corresponding to objects of training set is finite in each boundary zone, the process of division of boundary zones is finite too.

Absence of boundary zones after the termination of process of division means, that each of concepts in attribute space has area containing all points, corresponding objects of training set which are included in concept volume, and not including any point corresponding to other objects of training set. Thus, after the termination of division of boundary zones total division of training set into subsets $H_i = V_i \cap H(i = 1,2,...,n)$ occurs. As a result algorithm operation for each of the formed concepts, the area is composed of zones of attribute space. This area contains all points of objects of the appropriate class and does not contain any point corresponding to objects of other classes. This area approximates allocation area of objects of the corresponding class. As the approximating area consists of linear elementary areas (hyperplanes), its limiting surface is piecewise-linear. Therefore, the algorithm performs the piecewise-linear division of objects, which correspond to different concepts.

The described method provides decisions of analytical problems of classification, diagnostics and forecasting on the basis of logic models of objects classes. The model displays dependences of an investigated class on combinations of values of attributes, i.e. allows taking account for combined influence of several attributes.

An important distinction of a method of concepts formation in growing pyramidal networks is the possibility to introduce in concepts the so-called excluding attributes which do not correspond to objects of a researched class. As a result, the formed concepts have more compact logic structure, which allows increasing the accuracy of diagnosis or forecasting. In logic expression the excluding attributes are presented by variables with negation.

All search operations in growing pyramidal network are limited to rather small fragment of a network, which includes an object pyramid and vertices directly linked to it. As a result, we have a possibility solve practical analytical problems based on large-scale data.

In a pyramidal network the information is stored by its representation in structure of network. Rules A1-A2, B1-B2 define the rules of memory organization while new information perception. The information of objects and classes of objects is presented by ensembles of vertices (pyramids), allocated in all network. Incoming of the new information causes redistribution of links among vertices of network, i.e. modifying of its structure.

The advantages of growing pyramidal networks become obvious in implementation, which allows parallel distribution of signals in network. The important property of a network as means of information storage is that the possibility of parallel distribution of signals is combined with parallel reception of signals to receptors.

Despite of the certain similarity of the processes proceeding in GPN and neural networks there some distinctions in operating. Main distinction of GPN is that its structure is formed depending on the input data automatically. The adaptation of network structure to the structure of data results in optimization of the information representation. In addition, in contrast to neural networks, the adaptation does not require the introduction of aprioristic redundancy of a network, and training process does not depend on the predetermined configuration of a network. The weakness of neural networks comparing with GPN is that the allocated generalized knowledge cannot be explicitly represented as rules or logic expression. It complicates their understanding by person.

Various set-theoretic descriptions of GPN are given in [4,5]. The [5] considers the so-called β–pyramidal networks (β-PN) modification of GPN for the ranked data. β-PN are useful for data presentation in problems of management, taking and planning decisions (for example, in planning the actions of robots), and also in semantic analysis and synthesis of natural-language texts. In [3-5] the algorithm of formation of concepts in GPN for nondetermined learning process, i.e. for a case when crossing volumes of different concepts occurs, is considered.

The program complex used for experimentation and solving the applied tasks using GPN [8], includes systems CONFOR, realizing processes of building and training GPN, and DISCRET by which the attributes given in numerical scales, are transformed in nominal scales. Discretization of attributes is performed on numerical scales by analysis of distributions of training set objects belonging to different classes.

Typical application fields for GPN are as follows: forecasting of new chemical compounds and materials with the predefined properties[7-9], forecasting in genetics, geology, the solar activity forecasting, medical and technical diagnostics, the robot planning, forecasting of failures of complex units etc. As an example we offer tasks of inorganic compounds forecasting with predefined properties. The tables containing attribute descriptions of binary, ternary and quaternary systems of chemical elements, forming or not forming the chemical compounds were used as training set. Training sets for binary, ternary and quaternary systems included 1333, 4278 and 4963 descriptions, and test set - 692, 2156 and 2536 descriptions. Each chemical element was described by the set of 87 attribute values. Descriptions of binary, ternary and quaternary systems had 174, 261 and 348 attributes. The recognition furnished the 99% accuracy result.

Conclusion

The growing pyramidal network is the network memory self-adapting to the structure of incoming information. In selfstructurized systems the structure of data adapts to the task (classes of objects are allocated and defined) which results in optimization of the solution. In contrast to neural networks, the adaptation does not require the introduction of aprioristic redundancy of a network. In GPN various combinations of the assigned initial properties are formed, which increase the accuracy of analytical tasks solving. Selfstructurized systems allow not only to locate the dependences providing the diagnosis or the forecasting but also to create their logic descriptions.

The researches, operating on complex large-scale data, have shown high efficiency in applying the growing pyramidal networks for solving the analytical tasks. Such properties as simplicity of modification, combining the input of information with classification, generalization and allocation of essential attributes, high associativity, all make the growing pyramidal networks an indispensable component of intellectual systems.

Bibliography

1. Pospelov D.A. Logic-linguistic models in control systems. -Moscow: Energoizdat.-1981.
2. Voronkov G.V., Rabinovich Z.L. Natural environment of memory and thinking: modelling representation. Proceedings of international conference. "Knowledge - Dialogue-Solution"-2001.-SPb.-2001.
3. Gladun V.P. Partnership with computer. Man-Computer Purposeful Systems.-Kiev: Port-Royal. - 2000.
4. V.P.Gladun. Processes of New Knowledge Formation. Sofia: SD Pedagog, 1994, 192 p.
5. V.P.Gladun. Planning of Solutions. Kiev: Naukova Dumka. 1987. 168 p.
6. Gladun V.P. and Vashchenko N.D. Analytical processes in pyramidal networks // Intern. Journal on Information Theories and Applications. FOI-COMMERCE, Sofia.-2000.-Vol.7, - №3.
7. Kiselyova N., Gladun V., Vashchenko N. Computational Materials Design Using Artificial Intelligence Methods. Journal of Alloys and Compounds. 279 (1998), pp. 8-13.
8. www.aduis.com.ua <http: // www.aduis.com.ua>
9. Kiseleva N.N. [editor V.S.Zemskov] Computer designing of inorganic compounds: use of databases and methods of artificial intelligence; Institute metallurgy and sience of material named for A.A.Bajkov.-of M.: Nauka.-2005.

Authors' Information

Victor Gladun, Vitalii Velychko – V.M.Glushkov Institute of cybernetics of NAS of Ukraine, Prospekt akad. Glushkova 40, 03680 Kiev, Ukraine; e-mail: glad@aduis.kiev.ua

Yurii Ivaskiv – National Aviation University, Prospekt Kosmonavta Komarova 1, 03058, Kiev, Ukraine.

ALMOST SEPARABLE DATA AGGREGATION BY LAYERS OF FORMAL NEURONS[1]

Leon Bobrowski

Abstract: Information extraction or knowledge discovery from large data sets should be linked to data aggregation process. Data aggregation process can result in a new data representation with decreased number of objects of a given set. A deterministic approach to separable data aggregation means a lesser number of objects without mixing of objects from different categories. A statistical approach is less restrictive and allows for almost separable data aggregation with a low level of mixing of objects from different categories. Layers of formal neurons can be designed for the purpose of data aggregation both in the case of deterministic and statistical approach. The proposed designing method is based on minimization of the of the convex and piecewise linear (CPL) criterion functions.

Key words: data aggregation, layers of formal neurons, separability principles

1. Introduction

Data exploration or data mining tools should allow for efficient discovering of regularities (*patterns*) in large data sets. Data models can be designed on the basis of such patterns. Data exploration tools can be based on variety of methods of multivariate data analysis or pattern recognition [1], [2], [3]. In these approaches, each object or event is typically represented as a feature vector or as a point in a multidimensional feature space. Feature vectors are often divided by experts into categories in such a way that each vector belongs to no more than one category (*class*). In this way the reference (*learning*) set can be generated for each category. For example, teams of medical experts want to obtain such representative learning set for each important disease. Such a set should contain a large number of multidimensional vectors representing particular patients linked to this disease.

In the presented paper, data aggregation term means a reduction of numbers of different feature vectors in learning sets resulting from nonlinear transformation of these vectors. Such transformations may cause merging of a large number of different feature vectors into the same transformed vector. In the case of separable data aggregation, only some feature vectors belonging to the same category are merged. In the case of almost separable data aggregation, a low fraction of feature vectors from different categories can be merged.

Data aggregation can be performed by a layer of formal neurons [4]. The feature vectors are transformed by a layer of formal neurons into vectors with binary components. The dipolar and the ranked strategies of designing separable layers of formal neurons were proposed earlier [5]. The possibility of applying the dipolar and ranked strategies to designing almost separable layers is analyzed in the presented paper.

2. Separable learning sets

Let us assume that *m* objects O_j (j = 1,….,m) are represented as the so-called feature vectors $\mathbf{x}_j = [x_{j1},......,x_{jn}]^T$, or as points in the *n*-dimensional feature space $F[n]$ ($\mathbf{x}_j \in F[n]$). Components (*features*) x_i of the feature vector $\mathbf{x}$ represent numerical results of different measurements on a given object O ($x_i \in \{0,1\}$ or $x_i \in R$).

We assume that the feature vector $\mathbf{x}_j(k)$ (j = 1,......, m) has been labeled in accordance with the object $O_j(k)$ *category* (*class*) ω_k (k = 1,….,K). The learning set C_k contains m_k feature vectors $\mathbf{x}_j(k)$ assigned to the *k*-th category ω_k

$$C_k = \{\mathbf{x}_j(k)\} \quad (j \in I_k) \tag{1}$$

[1] This work was partially supported by the grants KBN 3T11F011 30, by the grant W/II/1/2007 from the Białystok University of Technology and by the grant 16/St/2007 from the Institute of Biocybernetics and Biomedical Engineering PAS.

where I_k is the set of indices j of the feature vectors $\mathbf{x}_j(k)$ assigned to the class ω_k.

Definition 1: The learning sets C_k (1) are *separable* in the feature space $F[n]$, if they are disjoined in this space ($C_k \cap C_{k'} = \varnothing$, if $k \neq k'$). It means that the feature vectors $\mathbf{x}_j(k)$ and $\mathbf{x}_{j'}(k')$ belonging to different learning sets C_k and $C_{k'}$ cannot be equal:

$$(k \neq k') \Rightarrow (\forall j \in I_k) \textit{ and } (\forall j' \in I_{k'}) \ \mathbf{x}_j(k) \neq \mathbf{x}_{j'}(k') \qquad (2)$$

We are also considering the separation of the sets C_k (1) by the hyper planes $H(\mathbf{w}_k,\theta_k)$ in the feature space $F[n]$:

$$H(\mathbf{w}_k,\theta_k) = \{\mathbf{x}: \mathbf{w}_k^T\mathbf{x} = \theta_k\}. \qquad (3)$$

where $\mathbf{w}_k = [w_{k1},....,w_{kn}]^T \in R^n$ is the weight vector, $\theta_k \in R^1$ is the threshold, and $(\mathbf{w}_k)^T\mathbf{x}$ is the inner product.

Definition 2: The feature vector $\mathbf{x}_j$ is situated on the *positive side* of the hyper plane $H(\mathbf{w}_k,\theta_k)$ (3) if and only if $(\mathbf{w}_k)^T\mathbf{x}_j > \theta_k$. Similarly, vector $\mathbf{x}_j$ is situated on the *negative side* of $H(\mathbf{w}_k,\theta_k)$ if and only if $(\mathbf{w}_k)^T\mathbf{x}_j < \theta_k$.

Definition 3: The learning sets (1) are *linearly separable* in the *n*-dimensional feature space $F[n]$ if each of the sets C_k can be fully separated from the sum of the remaining sets C_i by some hyper plane $H(\mathbf{w}_k,\theta_k)$ (3):

$$\begin{aligned}(\exists k \in \{1,...,K\}) \ (\exists \ \mathbf{w}_k,\theta_k) \ (\forall \mathbf{x}_j(k) \in C_k) \ \ \mathbf{w}_k^T\mathbf{x}_j(k) > \theta_k. \\ \textbf{and } \ (\forall \mathbf{x}_j(i) \in C_i, \ i \neq k) \ \ \mathbf{w}_k^T\mathbf{x}_j(i) < \theta_k\end{aligned} \qquad (4)$$

In accordance with the relation (4), all the vectors $\mathbf{x}_j(k)$ from the set C_k are situated on the positive side of the hyper plane $H(\mathbf{w}_k,\theta_k)$ (3) and all vectors $\mathbf{x}_j(i)$ from the remaining sets C_i are situated on the negative side of this hyper plane.

Linear independence of the feature vectors $\mathbf{x}_j(k)$ is a sufficient condition for linear separability of the learning sets C_k (1) [5]:

Remark 1: If the feature vectors $\mathbf{x}_j(k)$ constituting the learning sets C_k (1) are linearly independent in given feature space $F[n]$, then the sets C_k (1) are linearly separable (4) in this space.

3. Separable layers of formal neurons

The formal neuron NF($\mathbf{w}$,θ) can be defined by the threshold activation function $r_t(\mathbf{w},\theta; \mathbf{x})$

$$r = r(\mathbf{w}_k,\theta_k; \mathbf{x}) = \begin{cases} 1 & \textit{if} \quad \mathbf{w}_k^T\mathbf{x} \geq \theta_k \\ 0 & \textit{if} \quad \mathbf{w}_k^T\mathbf{x} < \theta_k \end{cases} \qquad (5)$$

where $w = [w_1,....,w_n]^T \in R^n$ is the weight vector, $\theta \in R^1$ is the threshold, and r is the output.

The layer of L formal neurons $NF(\mathbf{w}_k,\theta_k)$ transforms feature vectors $\mathbf{x}$ into output vectors $\mathbf{r} = [r_1,.....,r_L]^T$ with L binary components $r_i \in \{0,1\}$:

$$\mathbf{r} = \mathbf{r}(\mathbf{W}; \mathbf{x}) = [r_t(\mathbf{w}_1,\theta_1;\mathbf{x}),......., r_t(\mathbf{w}_L,\theta_L;\mathbf{x})]^T \qquad (6)$$

where $\mathbf{W} = [\mathbf{w}_1^T, \theta_1,......., \mathbf{w}_L^T, \theta_L]^T$ is the vector of the layer parameters.

The relation (6) determines the transformed vectors $\mathbf{r}_j(k)$ with binary components $r_t(\mathbf{w}_i,\theta_i;\mathbf{x})$.

$$(\forall k \in \{1,...,K\}) \ \ (\forall \mathbf{x}_j(k) \in C_k) \ \ \mathbf{r}_j(k) = \mathbf{r}(\mathbf{W};\mathbf{x}_j(k)) \qquad (7)$$

The transformed learning sets C'_k (1) constitute of the vectors $\mathbf{r}_j(k)$:

$$C'_k = \{\mathbf{r}_j(k)\} \ \ (j \in I_k) \qquad (8)$$

We are examining the properties of the transformation (7) which assure the separability $((k \neq k') \Rightarrow \mathbf{x}_{j,}(k) \neq \mathbf{x}_{j,}(k'))$ (2) of the transformed sets C'_k (8). Such a property can be based on the concept of the mixed dipoles separation [5].

Definition 3: A pair of feature vectors $(\mathbf{x}_j(k),\mathbf{x}_{j'}(k'))$ creates a *mixed dipole* if and only if these vectors belong to different classes ω_k $(k \neq k')$. Similarly, a pair of vectors from the same class ω_k create the *clear dipole* $(\mathbf{x}_j(k),\mathbf{x}_{j'}(k))$.

Definition 4: The formal neuron $NF(\mathbf{w}_k,\theta_k)$ (5) separates the dipole $(\mathbf{x}_j(k),\mathbf{x}_{j'}(k'))$ if <u>only one</u> vector $\mathbf{x}_j(k)$ or $\mathbf{x}_{j'}(k')$ is situated on the positive side of the hyper plane $H(\mathbf{w}_k,\theta_k)$ (3).

Definition 5: The layer of formal neurons $NF(\mathbf{w}_k,\theta_k)$ (5) is *separable* in respect to the learning sets C_k (1) if and only if the transformed sets C'_k (8) are separable (2) and each feature vector $\mathbf{x}_j(k)$ (1) is situated on the positive side of at least one of the hyper plane $H(\mathbf{w}_k,\theta_k)$ (3).

Lemma 1: The necessary condition for separability (*Def.* 5) of the layer of formal neurons $NF(\mathbf{w}_k,\theta_k)$ (5) is the separation (*Def.* 4) of each mixed dipole $(\mathbf{x}_j(k),\mathbf{x}_{j'}(k'))$ by at least one neuron $NF(\mathbf{w}_k,\theta_k)$ (9) of this layer. [5].

The formal neuron $NF(\mathbf{w}_k,\theta_k)$ (5) is *activated* by the vector $\mathbf{x}_j(k)$ $(r_I(\mathbf{w}_k,\theta_k;\ \mathbf{x}_j(k)) = 1)$ if and only if this vector is situated on the positive side of the hyper plane $H(\mathbf{w}_k,\theta_k)$ (3). In accordance with *Definition* 5, the separable layer should assure that each feature vector $\mathbf{x}_j(k)$ (1) activates at least one neuron $NF(\mathbf{w}_k,\theta_k)$ (5). The separable layer can be designed in a multistage procedure, when at one stage a successive neuron $NF(\mathbf{w}_k,\theta_k)$ (5) is added to the layer. In order to increase the generality of the designed neural layers, the following postulate has been introduced [6]:

Postulate of dipolar designing: The separating hyper plane $H(\mathbf{w}_k,\theta_k)$ (3) should divide the highest possible number of mixed and undivided yet dipoles $(\mathbf{x}_j(k),\mathbf{x}_{j'}(k'))$ and at the same time the lowest possible number of the clear dipoles $(\mathbf{x}_j(k),\mathbf{x}_{j'}(k))$ should be divided.

The linear separability (4) of the sets C'_l (11) could also be achieved in the layer of formal neurons $NF(\mathbf{w}_k,\theta_k)$ (5):

Definition 6: The layer of formal neurons $NF(\mathbf{w}_k,\theta_k)$ (5) is *linearly separable* in respect to the learning sets C_k (1) if and only if the transformed sets C'_k (8) are separated (4) by the hyper planes $H(\mathbf{w}_k,\theta_k)$ (3) and each feature vector $\mathbf{x}_j(k)$ activates at least one of these neurons.

The linear separability (4) of the sets C'_l (11) can be achieved by applying a multistage designing process consistent with the following postulate:

Postulate of ranked designing: The hyper plane $H(\mathbf{w}_k,\theta_k)$ (3) designed during the *l*-th stage should separate $((\mathbf{w}_k)^T\mathbf{x}_j(k) > \theta_k)$ as many as possible feature vectors $\mathbf{x}_j(k)$ from one set $C_k[l]$ under the condition that no vector $\mathbf{x}_{j'}(k')$ from the remaining sets $C_{k'}[l]$ $(k' \neq k)$ is separated.

The symbol $C_k[l]$ in the above postulate means that the learning set C_k (1) has been reduced as a result of neglecting feature vectors $\mathbf{x}_j(k)$ which have been separated by the hyper planes $H(\mathbf{w}_k,\theta_k)$ (3) during previous l - 1 steps. In the deterministic approach, the designing procedure is stopped during the L-th step if all the sets $C_k[L]$ become empty.

Lemma 2: The layer of L formal neurons $NF(\mathbf{w}_k,\theta_k)$ (5) designed in accordance with the above ranked postulate results in linearly separable (4) transformed sets C'_k (8) [5].

4. Convex and piecewise linear criterion functions (CPL)

The procedure of designing a separable layer can be based on a sequence of minimization of the convex and piecewise linear (*CPL*) criterion functions $\Psi_k(\mathbf{w},\theta)$ [3], [4]. The perceptron criterion function belongs to the *CPL* family. Let us define the function $\Psi_k(\mathbf{w},\theta)$ by using the positive G_k^+ and the negative G_k^- sets of the feature vectors $\mathbf{x}_j = [x_{j1},.....,x_{jn}]^T$ (1):

$$G_k^+ = \{\mathbf{x}_j\} \quad (j \in J_k^+) \quad \textit{and} \quad G_k^- = \{\mathbf{x}_j\} \quad (j \in J_k^-) \tag{9}$$

Each element $\mathbf{x}_j$ of the set G_k^+ defines the positive penalty function $\varphi_j^+(\mathbf{w},\theta)$

$$(\forall \mathbf{x}_j \in G_k^+) \quad \varphi_j^+(\mathbf{w},\theta) = \begin{cases} 1 - \mathbf{w}^T\mathbf{x}_j + \theta & \textit{if} \quad \mathbf{w}^T\mathbf{x}_j - \theta \leq 1 \\ 0 & \textit{if} \quad \mathbf{w}^T\mathbf{x}_j - \theta > 1 \end{cases} \tag{10}$$

Similarly, each element $\mathbf{x}_j$ of the set G_l^- defines the negative penalty function $\varphi_j^-(\mathbf{w},\theta)$

$$(\forall \mathbf{x}_j \in G_k^-) \quad \varphi_j^-(\mathbf{w},\theta) = \begin{cases} 1 + \mathbf{w}^T\mathbf{x}_j - \theta & \textit{if} \quad \mathbf{w}^T\mathbf{x}_j - \theta \geq -1 \\ 0 & \textit{if} \quad \mathbf{w}^T\mathbf{x}_j - \theta < -1 \end{cases} \tag{11}$$

The penalty function $\varphi_j^+(\mathbf{w},\theta)$ is aimed at positioning the vector $\mathbf{x}_j$ from the set G_k^+ ($\mathbf{x}_j \in G_k^+$) on the positive side of the hyper plane $H(\mathbf{w}_k,\theta_k)$ (3). Similarly, the function $\varphi_j^-(\mathbf{w},\theta)$ should set the vector $\mathbf{x}_j$ from the set G_k^- ($\mathbf{x}_j \in G_k^-$) on the negative side of this hyper plane.

The criterion function $\Psi_k(\mathbf{w},\theta)$ is the positively weighted sum of the penalty functions $\varphi_j^+(\mathbf{w},\theta)$ and $\varphi_j^+(\mathbf{w},\theta)$

$$\Psi_k(\mathbf{w},\theta) = \sum_{j \in J_k^+} \alpha_j \, \varphi_j^+(\mathbf{w},\theta) + \sum_{j \in J_k^-} \alpha_j \, \varphi_j^-(\mathbf{w},\theta) \tag{12}$$

where α_j ($\alpha_j > 0$) are the positive parameters (*prices*).

The criterion function $\Psi_k(\mathbf{w},\theta)$ belongs to the family of the convex and piecewise linear (*CPL*) criterion functions. Minimization of the function $\Psi_k(\mathbf{w},\theta)$ allows to find optimal parameters $(\mathbf{w}_k^*,\theta_k^*)$:

$$\Psi_k^* = \Psi_k(\mathbf{w}_k^*,\theta_k^*) = \min \Psi_k(\mathbf{w},\theta) \geq 0 \tag{13}$$

The basis exchange algorithms which are similar to the linear programming allow to find the minimum of the criterion function $\Psi_k(\mathbf{w},\theta)$ efficiently, even in the case of large, multidimensional data sets G_k^+ and G_k^- (29) [5].

The parameters $(\mathbf{w}_k^*, \theta_k^*)$ constituting the minimum of the function $\Psi_k(\mathbf{w},\theta)$ (12) define the *k*-th neuron $NF(\mathbf{w}_k,\theta_k)$ (5) of the layer and the separating hyper plane $H(\mathbf{w}_k,\theta_k)$ (3). The criterion functions $\Psi_k(\mathbf{w},\theta)$ can be specified both for the dipolar and for the ranked designing postulates. The specification of the criterion function $\Psi_k(\mathbf{w},\theta)$ (12) is performed through the choice of adequate sets G_k^+ and G_k^- (9) and the prices α_j related to particular vectors $\mathbf{x}_j$ from these sets.

It has been proved that the minimal value Ψ_k^* (13) of the criterion function $\Psi_k(\mathbf{w},\theta)$ (12) is equal to zero ($\Psi_k^* = 0$) if and only if the positive G_k^+ and the negative G_k^- sets (9) are linearly separable (4). In this case, all elements $\mathbf{x}_j$ of the set G_k^+ (9) are located on the positive side of the hyper plane $H(\mathbf{w}_k^*,\theta_k^*)$ (3) and all elements $\mathbf{x}_j$ of the set G_k^- are located on the negative side:

$$\begin{aligned} &(\forall \mathbf{x}_j \in G_k^+) \quad (\mathbf{w}_k^*)^T\mathbf{x}_j > \theta_k^* \\ \textit{and} \quad &(\forall \mathbf{x}_{j'} \in G_k^-) \quad (\mathbf{w}_k^*)^T\mathbf{x}_{j'} < \theta_k^* \end{aligned} \tag{14}$$

If the sets G_k^+ and G_k^- (9) are not linearly separable (4), then $\Psi_k^* > 0$ and the inequalities (14) are fulfilled only partly, not by all, but by a majority of elements $\mathbf{x}_j$ of the sets (9).

Minimization of the function $\Psi_k(\mathbf{w},\theta)$ (12) allows one to find the optimal parameters $(\mathbf{w}_k^*,\theta_k^*)$ defining such hyper plane $H(\mathbf{w}_k^*,\theta_k^*)$ (3) which relatively well separates two sets G_l^+ and G_l^- (9). The parameters $(\mathbf{w}_k^*,\theta_k^*)$ can be also used in the definition of the *l*-th element $NF(\mathbf{w}_k^*,\theta_k^*)$ (5) of a neural layer.

5. Almost separable layers

A layer of L formal neurons $NF(\mathbf{w}_k,\theta_k)$ (6) can transform a large number of feature vectors $\mathbf{x}j(k)$ ($k = 1,.....,K$) (1) into the same output vector $\mathbf{r}_l = [r_l1,.....,r_lL]^T$ (6) with L binary components $r_l \in \{0,1\}$, where different indexes l and l′ mean different vectors **r**l and **r**l′:

$$(l \neq l') \Rightarrow (\mathbf{r}l \neq \mathbf{r}l') \tag{15}$$

The set of such feature vectors $\mathbf{x}j(k)$ (1) which are transformed into the same output vector $\mathbf{r}_l$ is called the l-th *activation field* Sl of the layer of formal neurons:.

$$Sl \;=\; \{\, \mathbf{x}j(k): [r_t(\mathbf{w}_1,\theta_1;\mathbf{x}j(k)),........., r_t(\mathbf{w}L,\theta\, L;\mathbf{x}j(k))]T = \mathbf{r}_l\} \tag{16}$$

Definition 7: The set Sl (16) is the *clear activation field* if all feature vectors $\mathbf{x}j(k)$ (1) from this set ($\mathbf{x}j(k) \in Sl$) belong to the same learning set Ck (1). Similarly, the set Sl is the *mixed activation field* if it contains feature vectors $\mathbf{x}j(k)$ from different sets Ck.

Lemma 3: The layer of L formal neurons $NF(\mathbf{w}_k,\theta_k)$ (5) is separable in respect to the learning sets C_k (*Def.* 5) if and only if all the activation fields Sl (16) of this layer are clear.

It results from the above *Lemma* that the layer with all the clear activation fields Sl (16) is separable and preserves the learning sets Ck (1) separabilty during data aggregation. Even one mixed field Sl (16) results in a nonseparbilty of the layer. The concept of mixed activation fields Sl (16) is useful in the analysis of nearly separable layers.

Definition 8: The class ω_k is *dominant* in the active the set Sl (16) if and only if the <u>most</u> of the feature vectors $\mathbf{x}j(k)$ (1) from this set are assigned to the class ω_k.

The activation field $Sl(k)$ (16) and the output vector $\mathbf{r}l(k)$ is assigned to the dominant class ωk.

Definition 9: The set Sl (16) is the ε-*clear* (*almost clear*) activation field if the fraction fl of the feature vectors $\mathbf{x}j(k')$ (1) from nondominant classes $\omega k'$ in this set ($\mathbf{x}j(k') \in Sl(k)$) is less than ε.

$$fl(k) = ml' \,/\, (ml(k) + ml') < \varepsilon \tag{17}$$

where $ml(k)$ is the number of elements $\mathbf{x}j(k)$ of the set $Sl(k)$ (16) belonging to the dominant class ω_k and ml' is the number of elements $\mathbf{x}j(k')$ of the set $Sl(k)$ belonging to non-dominant classes $\omega_{k'}$ ($k' \neq k$).

All feature vectors $\mathbf{x}j(k)$ from the l-th activation field Sl are *aggregated* by the layer of formal neurons into one output vector **r**l. In other words, the vector **r**l. *generalizes* all feature vectors $\mathbf{x}j(k)$ from the field Sl (26). It can be expected that the layer of formal neurons with a large and ε-clear activation fields Sl (26) could have a great *generalization power*. Such layer could be also used as a classifier with the following decision rule:

$$\textit{if}\;\; (\mathbf{x}0 \in Sl(k)) \;\; \textit{then}\;\; \mathbf{x}0 \in \omega_k \tag{18}$$

The quality of a layer of formal neurons $NF(\mathbf{w}_k,\theta_k)$ (5) can be evaluated by the *error rate* of the decision rule (18). The error rate is often estimated through an *apparent error er* [1]:

$$\mathrm{er} = \mathrm{me} \,/\, \mathrm{m} \tag{19}$$

where me is the number of such feature vector $\mathbf{x}j(k)$ from the sets Ck (1) which are wrongly allocated by the decision rule (18).

Lemma 4: The error rate *er* (19) of the decision rule (18) is equal to zero if and only if the layer of formal neurons is separable (*Def.* 5).

The error rate evaluation (19) is positively biased (*optimistic bias*) [1]. The unbiased error rate *er* evaluations can be based on the technique of cross-validation [3].

6. Designing almost separable layers of formal neurons

The separable layer of formal neurons with the decision rule (18) assures the correct classification of all the feature vectors **x**j(*k*) from the learning sets *C*k (1) and the apparent error rate *er* (19) is equal to zero (*er* = 0). As it results from previous considerations, if the learning sets *C*k (1) are separable (2), the separable layers (*Def.* 5) can be designed in accordance with the dipolar strategy, and in accordance with the ranked strategy. Unfortunately, a separable layer can cause the *over-fitting problem* [3]. This problem is manifested in such a way that despite the fact that the apparent error rate *er* (19) of the rule (18) evaluated on the vectors **x**j(*k*) from learning sets *C*k (1) is equal to zero, the error rate of this rule on new feature vectors **x** that does not belong to the sets *C*k (1) is too large.

A layer of formal neurons with the apparent error rate *er* (19) greater than zero (*er* > 0) could have a larger *discriminative power* than a separable layer (*er* = 0). We shall take into considerations almost separable (ε-separable) layers of formal neurons.

Definition 10: The layer of formal neurons is ε-separable (*almost separable*) if and only if the apparent error rate *er* (19) is less than ε ($er < \varepsilon$).

Lemma 5: If all the activation fields *S*l(*k*) (16) of the layer of formal neurons are of the ε-*clear* type (*Def.* 9), then this layer is ε-separable

Proof: The apparent error rate *er* (19) can be represented in the below manner:

$$er = me \,/\, m = (m1' + \ldots\ldots + mM') \,/\, m \tag{20}$$

where *m*i′ (17) is the number of elements **x**j(*k*′) of the activation field *S*i(*k*) (16) belonging to non-dominant classes $\omega_{k'}$ ($k' \neq k$), and *M* is the number of activation fields. Thus

$$\begin{aligned} er = m1'/(ml(k(1)) + ml')\ (ml(k(1)) + ml') \,/\, m + \ldots \ldots \\ \ldots \ldots m1'/(ml(k(M)) + mM')\ (mM(k(M)) + mM') \,/\, m < \varepsilon \end{aligned} \tag{21}$$

and the thesis is proved. □

The *Lemma* 5 gives indications for designing almost separable layers of formal neurons $NF(\mathbf{w}_k,\theta_k)$ (6). Designing process can be based on a generation of such activation fields *S*l(*k*) (16) which are of the ε-*clear* type (*Def.* 9). As a consequence, the *Ranked designing postulate* can be modified in the below manner:

Postulate of almost ranked designing: The hyper plane $H(\mathbf{w}_k,\theta_k)$ (3) designed during the *l*-th stage should separate ($(\mathbf{w}_k)^T\mathbf{x}_j(k) > \theta_k$) as many as possible feature vectors $\mathbf{x}_j(k)$ from one set $C_k[l]$ under the condition that the fraction *f*l (17) of the separated vectors **x**j(*k*′) (1) from other sets C*k*′[*l*] ($k' \neq k$) is less than ε.

One can see that all the activation fields *S*l(*k*) (16) of the layer of formal neurons designed in accordance with the above postulate are of the ε-*clear* type (*Def.* 9).

The *Postulate of dipolar designing* can be also modified in a similar manner, for example by neglecting such mixed dipoles ($\mathbf{x}_j(k),\mathbf{x}_{j'}(k')$) that the feature vectors $\mathbf{x}_j(k)$ and $\mathbf{x}_{j'}(k')$ belong to the activation field *S*l(*k*) (16) of the ε-*clear* type (*Def.* 9).

The data aggregation process can be based on a layer of formal neurons $NF(\mathbf{w}_k,\theta_k)$ (6). Let us define the *aggregation coefficient* ηa of such layer in the following manner

$$\eta a = (m - m(\mathbf{rl}) \,/\, (m - K) \tag{22}$$

where, *m* is the number of the feature vectors **x**j(*k*) in the sets *C*k (1), *m*(**rl**) is he number of different output vectors **rl** (15) from a separable layer, and *K* is the number of the classes ω_k or the learning sets Ck(1).

The minimal number *m*(**rl**) of the output vectors **rl** (15) from a separable layer is equal to *K* (*m*(**rl**) = *K*). The aggregation coefficient ηa (22) takes the maximal value equal to one (ηa = 1) in this ideal situation. The aggregation coefficient ηa (22) of a layer of formal neurons *NF*(**w**i,θi) (5) can take the maximal value ηa = 1 if and only if the learning sets *C*k (1) are linearly separable. The maximal value of the number *m*(**rl**) is equal to *m.*

There is no aggregation in this case and the aggregation coefficient ηa (29) takes the minimal value equal to 0 ($\eta a = 0$). As a result:

$$0 \leq \eta a \leq 1 \tag{23}$$

It can be noted that a solution of the *Optimization problem* leads to the maximization of the aggregation coefficient ηa (22).

Optimization problem: To design such ε-separable (*Def.* 10) layer of formal neurons $NF(\mathbf{w}_k,\theta_k)$ (6) which has a minimal number M of activation fields $Sl(k)$ (16) or different output vectors **rl** (15).

The minimal number M of the activation fields Sl (16) can not be less than the number K of the classes ω_k ($M \geq K$).

7. Concluding remarks

Separable layer of formal neurons $NF(\mathbf{w}_k,\theta_k)$ (5) can be induced from the learning sets C_k (1) only when these sets are separable (2) in a given feature space. The dipolar strategy allows for preserving the separability of the sets C_k (1) during their transformation by the induced layer of formal neurons. The ranked strategy also allows achieving the linear separability (4) of the transformed sets C'_k (8).

Separable layers of formal neurons with the decision rule (18) secure correct classification of all the feature vectors $\mathbf{x}j(k)$ from the learning sets Ck (1) and the apparent error rate er (19) is equal to zero ($er = 0$) in this case. Unfortunately, such property of the designed layers is often linked with the occurrence of over-fitting [3]. Despite the fact that apparent error rate er (19) evaluated on the basis of elements $\mathbf{x}_j(k)$ of the learning sets Ck (1) is equal to zero, the decision rule (18) can be burdened with a too large error rate on new feature vectors $\mathbf{x}$ ($\mathbf{x} \notin Ck$). In this case, the generalization power of the designed network is too low.
Almost separable layer of formal neurons $NF(\mathbf{w}_k,\theta_k)$ (5) allow apparent error rate er (19) greater than zero. Designing almost separable layers should allow for achieving a larger generalization power and level of data aggregation in comparison to strictly separable layers. There are still many problems to resolve in the search for efficient strategy for designing almost separable layer.

Bibliography

[1]. Johnson R. A., Wichern D. W.: Applied Multivariate Statistical Analysis, Prentice-Hall, Inc., Englewood Cliffs, New York, 1991

[2]. Duda O. R. and Hart P. E., Stork D. G.: Pattern Classification, J. Wiley, New York, 2001

[3]. Fukunaga K., Introduction to Statistical Pattern Recognition, Academic Press 1990

[4]. Rosenblatt F.: Principles of neurodynamics, Spartan Books, Washington 1962

[5]. Bobrowski L.: Data mining based on convex and piecewise linear (CPL) criterion functions, (in Polish), Technical University Białystok, 2005

[6]. Bobrowski L.: “Piecewise-Linear Classifiers, Formal Neurons and separability of the Learning Sets”, Proceedings of ICPR’96, pp. 224-228, (13th International Conference on Pattern Recognition", August 25-29, 1996, Vienna)

[7]. Bobrowski L,: "Design of piecewise linear classifiers from formal neurons by some basis exchange technique"' Pattern Recognition, 24(9), pp. 863-870, 1991

[8]. Bobrowski L., 'The ranked neuronal networks", Biocybernetics and Biomedical Engineering, Vol. 12, No. 1-4, pp. 61-75, 1992

Author's Information

Leon Bobrowski – leon@ii.pb.bialystok.pl
Faculty of Computer Science, Bialystok Technical University
Institute of Biocybernetics and Biomedical Engineering, PAS, Warsaw, Poland

THE FUZZY-NEURO CLASSIFIER FOR DECISION SUPPORT

Galina Setlak

Abstract: This paper aims at development of procedures and algorithms for application of artificial intelligence tools to acquire process and analyze various types of knowledge. The proposed environment integrates techniques of knowledge and decision process modeling such as neural networks and fuzzy logic-based reasoning methods. The problem of an identification of complex processes with the use of neuro-fuzzy systems is solved. The proposed classifier has been successfully applied for building one decision support systems for solving managerial problem.

Keywords: artificial intelligence, artificial neural networks, fuzzy inference systems, classification, decision support.

ACM Classification Keywords: I. Computing Methodologies, I.2 Artificial Intelligence

Introduction

A managerial decision support system must harness the information embedded in corporate data and apply this information to problem-solving processes of managers. Information systems required by the factories of the future must be capable of managing, maintaining and processing all forms of information required in the factory. Information is considered as being a vital resource of the enterprise because it represents its mind, because it is the basis for decision making and communication and it forms the basis for new designs.

The traditional artificial intelligence systems, mainly based on the symbolic paradigm, are showed to be efficient tools for solving exactly and completely stated problems. However, they were ineffective for solving the real life problems that are described or represented by the imprecise, incomplete, uncertain and linguistic knowledge or by large amounts of numerical data collected in databases. The foregoing drawbacks of the symbolic paradigm based artificial intelligence systems have motivated many researches for creating new tools for designing intelligent decision support systems. As a result of those efforts the techniques named "computational intelligence" have been developed. They have been worked out as a joint of three methodologies: artificial neural networks, fuzzy logic and genetic algorithms. The artificial neural networks bring in the resulting system the ability for learning, generalizing and processing large amount of numerical data, the fuzzy logic allows the follow-on systems to represent and process inexact and uncertain information [Zadeh L.A., Kacprzyk J., 1992], and the genetic algorithm - as a global optimization tool - is used for strengthening the learning abilities of the resulting tool [Rutkowska D., M.Pilinski, L. Rutkowski, 1997]. As a result of joining the artificial neural networks (ANN), fuzzy logic, and genetic algorithms we get the system that is a synergistic combination of the three complementary technologies [Takagi H., 2000], [Rutkowska D., 2000].

Neural computing, genetic algorithms, and fuzzy systems are effective ways to deal with complex problems efficiently. Each method handles uncertainty and ambiguity differently, and these technologies can often be blended to utilize the features of each, achieving impressive results. A combination of artificial neural networks and fuzzy logic can result in synergy that improves speed, fault tolerance, and adaptiveness. Fusion of neural networks and fuzzy inference systems have attracted the growing interest of researchers in various scientific and engineering areas due to the growing need of intelligent decision support systems to solve the real world problems. There are many real-world applications of intelligent systems integration [Takagi H., 2000], [Li S., 2000], [F. Wong, 1992], [D.Nauck, F. Klawonn, R.Kruse, 1997]. Each intelligent system can be a valuable component in a decision support system in which each technology can be used in series or in parallel. For instance, the neural network can identify classes of membership function for the fuzzy system [Jang R., Sun C.T., Mizutani E., 1997], [Takagi H., 2000].The genetic learning method can perform rule discovery in large databases, with the rules fed into the conventional expert system [Takagi H., 2000].

There are two directions of researches on systems that are built as a combination of neural networks and fuzzy logic based systems. The first one gives as results so-called fuzzy neural networks build of fuzzy neurons [Rutkowska D., M.Pilinski, L.Rutkowski, 1997]. The results of the second research course are neuro-fuzzy systems that use the artificial neural networks within the fuzzy logic systems framework. The most advanced types of the neuro-fuzzy systems are hybrid ones [Li S., 2000], [Rutkowska D., 2000].

The paper presents the neuro-fuzzy technologies which can be used for designing the rule-based intelligent decision support systems. In the paper a connectionist neuro-fuzzy system designed for classification problems is presented. The proposed classifier has been successfully applied for building one decision support systems for solving managerial problem. Example of classification problems solved by means of this hybrid intelligent system is illustrated.

The Neuro-Fuzzy Method for Knowledge Modeling

Neural Networks can be used in constructing fuzzy inference systems in ways other than training. They can also be used for rule selection, membership function determination and in what we can refer to as hybrid intelligent systems.

Fuzzy systems that have several inputs suffer from the curse of dimensionality. In this paper we will investigate and apply the Takagi-Hayashi method [Takagi H., Hayashi I., 1991] for the construction and tuning of fuzzy rules, which is commonly referred to as neural network driven fuzzy reasoning – NDF – method (see Fig.1). The NDF method is an automatic procedure for extracting rules and can greatly reduce the number of rules in a high dimensional problem, thus making the problem tractable.

The NDF method performs three major functions:

- Partitions the decision hyperspace into a number of rules. It performs this with a clustering algorithm.
- Identifies a rule's antecedent values (left hand side - LHS membership function). It performs this with a neural network.
- Identifies a rule's consequent values (right hand side - RHS membership function) by using a neural network with supervised training. This part necessitates the existence of target outputs.

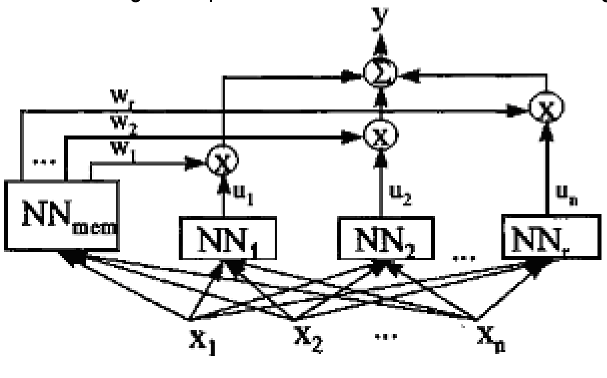


Fig.1. Neural Network Driven Fuzzy Reasoning [Takagi H., Hayashi I., 1991].

The above block diagram represents the NDF method of fuzzy rule extraction. This method uses a variation of the Sugeno fuzzy rule:

IF x_i is A_i AND x_2 is A2 AND ... AND x_n is A_n THEN $y=f(x_1, x_2,\dots, x_n)$, (1)

where f(.) is a neural network model rather than a mathematical function. This results in a rule of the form:

IF x_i is A_i AND x_2 is A_2 AND ... AND x_n is A_n THEN $y=NN(x_1, x_2,\dots, x_n)$. (2)

The NN$_{mem}$ calculates the membership of the input to the LHS membership functions and outputs the membership values. The other neural networks form the RHS of the rules. The LHS membership values weigh the RHS neural network outputs through a product function. The altered RHS membership values are aggregated to calculate the NDF system output. The neural networks are standard feed forward multilayer perceptron designs.

The Neuro-Fuzzy System for Classification

Neural networks are widely used as classifiers; see e.g. [Jang R., Sun C.T., Mizutani E., 1997], [Moon Y.B., Divers C.K., and H.-J.Kim, 1998], [Takagi H., 2000]. Classification and clustering problems has been addressed in many problems and by researchers in many disciplines like statistics, machine learning, and data bases. The basic algorithms of the classification methods are presented in [D.Nauck, F. Klawonn, R.Kruse, 1997], [Setlak G., 2004]. The application of the clustering procedure can be classified into one of the following techniques [Jang R., Sun C.T., Mizutani E., 1997] partition in which a set is divided into m subsets, when m is the input parameter:

- hierarchical form trees in which the leaves represent particular objects, and the nodes represent their groups. The higher level concentrations include the lower level concentrations. In terms of hierarchical methods, depending on the technique of creating hierarchy classes (agglomerative methods and divisive methods);
- graph-theoretic clustering,
- fuzzy clustering,
- methods based on evolutionary methods,
- methods based on artificial neural networks.

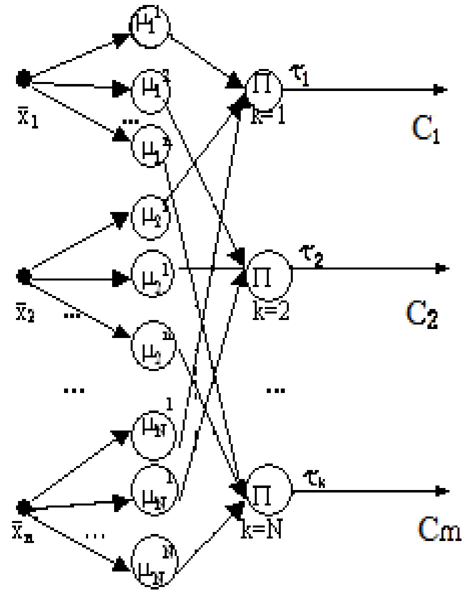


Fig.2. A neuro-fuzzy system for classyfication

In this work two approaches have been applied to solving of the classification and clustering problems. As basic method it was used Self Organizing Map (SOM) of Kohonen, a class of unsupervised learning neural networks, to perform direct clustering of parts families and assembly units. Self Organizing Maps are unsupervised learning neural networks which were introduced by T. Kohonen [Kohonen T., 1990] in the early '80s. This type of neural network is usually a two-dimensional lattice of neurons all of which have a reference model weight vector. SOM are very well suited to organize and visualize complex data in a two dimensional display, and by the same effect, to create abstractions or clusters of that data. Therefore neural networks of Kohonen are frequently used in data exploration applications [Kohonen T., 1990], [Takagi H., 2000]. SOM have been applied to classification of machine elements in group technology [Setlak G., 2004].

The other approach applies fuzzy logic and fuzzy neural systems for classification problems. However, neural networks work as a "black box", which means that they produce classification results but do not explain their performance. Thus, we do not know the rules of classification. Neural network weights have no physical interpretation. Fuzzy and fuzzy neural systems can be employed in order to solve classification problems [Setlak G., 2000]. The neural-fuzzy systems are rule-based systems that realize fuzzy IF-THEN rules. Some of the major works in this area are ANFIS [Jang, 1992], [Jang 1997], NEFCLASS [D.Nauck, F. Klawonn, R.Kruse, 1997], CANFIS [F. Wong, 1992].

ANFIS (Adaptive Neuro-Fuzzy Inference System) [Jang, 1992], [Jang R., Sun C.T., Mizutani E., 1997] (Fig.1) is a network-structured adaptive fuzzy inference system which has found various applications including control, system identification, time series prediction, and noise cancellation. A common version of ANFIS uses normalized input fuzzy membership functions, product fuzzification, product inference, sum composition and Sugeno-type linear output functions (and thus needs no defuzzification). The system parameters are tuned using stochastic gradient descent method for the premise parameters and recursive least square method for the consequent parameters.

The CANFIS (Co-Active Neuro-Fuzzy Inference System) model integrates fuzzy inputs with modular neural network to quickly solve poorly defined problems. Fuzzy inference systems are also valuable as they combine the explanatory nature of rules (membership functions) with the power of "black box" neural networks.

A hybrid intelligent system for classification can be presented and is shown in Fig.2. The Neuro-Fuzzy Classifier (NFC) is a neuro-fuzzy system that has a feed-forward network-like structure. The structure of this system expresses the fuzzy rules base that models the process of decision-making.

The classifier reflects the fuzzy classification rules, called the rule base, described as follows:

$$R^{(k)}: \textbf{ IF } x_1 \text{ is } G_1^k \textbf{ and } x_2 \text{ is } G_2^k \textbf{ and ... and } x_n \text{ is } G_n^k \textbf{ THEN } (x \in C_l) \tag{5}$$

where $x = [x_1, x_2 ..., x_n]^T$, and x_i, for $i = 1,2,..., n$, are linguistic variables, G_i^k is fuzzy sets for i-th input and k-th fuzzy rule, C_l, for l=1,2,...,m, are classes, N denotes the number of rules $R^{(k)}$, for $k = 1,..., N$.

The crisp input values, presented in Fig.2, constitute the input vector: $\bar{x} = [\bar{x_1}, \bar{x_2}, \ldots \bar{x_n}]^T$.

The output values, τ_k, for $k = 1, 2,...,$ N, represent degrees of rule activation [6], expressed as follows:

$$\tau_k = \prod_{i=1}^{n} \mu_i^k(\bar{x_i}), \tag{6}$$

where

$$\mu_i^k(x_i) = exp\left[-\left(\frac{x_i - \bar{X}_i^k}{\sigma_i^k}\right)^2\right] \tag{7}$$

is the Gaussian membership function, characterized by the center and width parameters, $\bar{x}_i^k$ and σ_i^k, respectively. The neuro-fuzzy network illustrated in Fig.2 performs a classification task based on the values of τ_k, for $k = 1,..., N$. Each input vector $\bar{x} = [\bar{x_1}, \bar{x_2}, \ldots, \bar{x_n}]^T$ is classified to the class C_l (where l = 1,2,…,m), which is associated with the maximal degree of rule activation, that is $\max\limits_{k} \{\tau_k\}$.

There are five phases of designing the NFC system:

- Each input attribute is described by a number of fuzzy sets;
- The initial fuzzy rules base is determined;
- System training;
- Testing the system against test data;
- Pruning the system – removing "weak", superfluous fuzzy rules in order to improve the system's transparency.

An example of implementing this neuro-fuzzy classifier is given below.

Example: international stock selection

The presented hybrid neuro-fuzzy system has been applied for building Intelligent Decision Support System (IDSS). As example of a hybrid neuro-fuzzy system we have chosen a method for deriving a stock portfolio plan.

An international investment company uses a hybrid neuro-fuzzy system to forecast the expected returns from stocks, cash, bonds, and other assets to determine the optimal allocation of assets. Because the company invests in global markets, it is first necessary to determine the creditworthiness of various countries, based on past and estimated performances of key socio-economic ratios, and then select specific stocks based on company, industry, and economic data. The final stock portfolio must be adjusted according to the forecast of foreign exchange rates, interest rates, and so forth, which are handled by a currency exposure analysis. The IDSS includes the following technologies:

- **Expert system.** The system provides the necessary knowledge for both country and stock selection (rule-based system).
- **Neural network.** The neural network conducts forecasting based on the data included in the database.
- **Fuzzy logic.** The fuzzy logic component supports the assessment of factors for which there are no reliable data. For example, the credibility of rules in the rule base is given only as a probability. Therefore, the conclusion of the rule can be expressed either as a probability or as a fuzzy membership degree.

The rule base feeds into IDSS along with data from the database. IDSS is composed of three modules: membership function generator, neuro-fuzzy inference system (NFIS), and neural network (NN). The modules are interconnected, and each performs a different task in the decision process.

Performance of an IDSS has been tested on the following input data:

- There are three input nodes (n = 3):

X_1 – risk of investment, it is defined by a linguistic term G_i^k, such as “high”, “medium”, “low”.

X_2 – clear profit, also it is defined by a linguistic term G_i^k, such as “high”, “medium” and “low”.

X_3 – period refund of investment, it is defined by a linguistic term G_i^k, such as “long”, “medium” and “short”.

- The output values: there are three C_l classes, where C_1 – is defined by a linguistic term “very good investment”, C_2 – “poor investment” and C_3 – “resign”.
- The fuzzy set is characterized by a membership function $\mu_i^k(x_i)$: R $\rightarrow$ [0,1]. The membership functions for the fuzzy set are expressed as (7).

Following the procedure described in section 2, the initial shapes of the fuzzy sets describing the input attributes were defined and the initial fuzzy rules base, containing 144 rules, was generated. The NFC method was used to classification tasks and results were compared with traditional agglomerative methods. Results were showed in Table 1.

Table 1. Results of the classification problem obtained using NFC and agglomerative methods

N	Price	Advertising	Volume of sales	Stimulus of sale	Neuro-fuzzy Classifier	Agglomerative methods
1	385,65	12000	227180	10000	C1	C1
2	397,24	10000	235090	6000	C1	C1
3	452,20	10000	217340	10000	C4	C4
4	478,92	12000	261280	8000	C3	C3
5	493,10	10000	184380	5000	C2,C3	C3
6	526,35	8000	147180	4000	C2	C2
7	583,24	5000	149300	3000	C2,C3	C2
8	594,93	5000	156520	4000	C1	C1
9	620,70	5000	121280	2000	C2,C3	C2
10	634,56	10000	116530	0	C3,C4	C4
11	663,20	2000	102160	0	C2,C3	C3
12	672,35	0	112510	0	C1,C2,C3	C3

Conclusions

The fuzzy neural network, used in this paper for classification, has the following features:

- each neuron represents one fuzzy IF-THEN rule,
- the number of neurons equals to the number of rules in the rule base,
- weights of the neurons have an interpretation concerning parameters of the membership functions of the corresponding neuro-fuzzy system.

Thus, in contrast to classical neural networks, this network does not work as a "black box", it is a rule-based neural network.

In the paper we have applied basic soft techniques for extracting rules and classification in a high dimensional managerial problem. The hybrid neuro-fuzzy system briefly presented in the paper was successfully applied for designing intelligent decision support system.

By using several advanced technologies (combination of fuzzy logic and neural networks) it is possible to handle a broader range of information and solve more complex problems. The research conducted proves that fuzzy neural networks are a very effective and useful instrument of implementation of intelligent decision support systems in management.

Bibliography

[Jang, 1992] Jang R.: Neuro-Fuzzy Modeling: Architectures, Analyses and Applications, PhD Thesis, University of California, Berkeley, 1992.

[Jang R., Sun C.T., Mizutani E., 1997] Jang S.R., Sun C.T., Mizutani E.: Neurofuzzy and Soft Computing, Prentice-Hall, Upper Saddle River 1997, p. 245.

[Kohonen T., 1990] Kohonen T.: Self-organizing Maps, Proc. IEEE, 1990, 78, NR.9, pp. 1464-1480.

[Li S., 2000] Li S.: The Development of a Hybrid Intelligent System for Developing Marketing Strategy, Decision Support Systems, 2000, Vol 27, N4,

[Moon Y.B., Divers C.K., and H.-J. Kim, 1998] Moon Y.B., Divers C.K., and H.-J. Kim: AEWS: An Integrated Knowledge-based System with Neural Network for Reliability Prediction // Computers in Industry, 1998, Vol.35, N2, pp.312-344.

[D.Nauck, F. Klawonn, R.Kruse, 1997] D.Nauck, F. Klawonn, R.Kruse: Foundations of Neuro-Fuzzy Systems, J.Wiley&Sons, Chichester, 1997.

[Rutkowska D., 2000] Rutkowska D.: Implication-based neuro-fuzzy architectures - Applied mathematics and computer science, V.10, N4, 2000, Technical Unieversity Press, Zielona Gora, 675-701.

[Rutkowska D., M.Pilinski, L. Rutkowski, 1997] Rutkowska D., M.Pilinski, L. Rutkowski: Sieci neuronowe, algorytmy genetyczne i systemy rozmyte, PWN, Warszawa, 1997 r., pp.411.

[Setlak G., 2004] Setlak G.: Intelligent Decision Support System, // LOGOS, Kiev, 2004, (in Rus.), pp. 250.

[Setlak G., 2000] Setlak G.: Neural networks in intelligent decision support systems for management // Journal of Automation and Information Sciences, Kiev, N1, 2000r., pp. 112-119.

[Takagi H., Hayashi I., 1991] Takagi H., Hayashi I.: NN-Driven Fuzzy Reasoning, //International Journal of Approximate Reasoning, 1991, Vol.3, p.1376-1389.

[Takagi H., 2000] Takagi H. Fusion technology of neural networks and fuzzy systems //International Journal of Applied mathematics and computer science, Zielona Gora, 2000, Vol.10, №4, pp.647-675.

[F. Wong, 1992] F. Wong: "Neural Networks, Genetic Algorithms, and Fuzzy Logic for Forecasting," Proceedings, International Conference on Advanced Trading Technologies, New York, July 1992, pp.504-532.

[Zadeh L.A., Kacprzyk J., 1992] Zadeh L.A., Kacprzyk J.(ed.): Fuzzy logic for the Management of Uncertainty, Wiley, New York, 1992, p. 492.

Author's Information

Galina Setlak – Ph.D., D.Sc, Eng., Associate Professor, Rzeszow University of Technology, Department of Computer Science, W. Pola 2 Rzeszow 35-959, Poland, Phone: (48-17)- 86-51-433, e-mail: gsetlak@prz.edu.pl

FORMING OF LEARNING SET FOR NEURAL NETWORKS IN PROBLEMS OF LOSSLESS DATA COMPRESSION

Yuriy Ivaskiv, Victor Levchenko

Abstract: questions of forming of learning sets for artificial neural networks in problems of lossless data compression are considered. Methods of construction and use of learning sets are studied. The way of forming of learning set during training an artificial neural network on the data stream is offered.

Keywords: neural network; modeling of data sources; learning set.

ACM Classification Keywords: C.1.3 Other Architecture Styles – Neural Nets. D.4.8 Performance – Modeling and predictions. E.4 Coding and Information Theory – Data compaction and compression.

Introduction

In problems of data compression wide application was found with artificial neural networks (ANN). The neural methods of lossless data compression have got wide spread occurrence [Jiang, 1999]. Applicability of lossy compression methods is limited to a class of multimedia data, and neural realization of such methods is specific to each concrete type of data that limits an opportunity of their use [Cottrell, 1987; Verma, 1999]. Methods of lossless data compression differ that at their use, as a rule, special restrictions on structure and type of data are not imposed that expands opportunities of their practical application for the decision of various classes of problems.

The most of modern methods of lossless data compression are entropic, that is such in which the length of a code necessary for representation of an element of data, contacts probability of occurrence of an element in a stream of compressed data [Ватолин, 2002]. Division of a problem of compression into two problems is standard: *coding* and *statistical modeling* of a data source [Rissanen, 1981]. Now various methods and algorithms of coding redundancy of which practically does not differ from theoretically minimal possible are developed and applied [Кричевский, 1989]. However, a problem of increase of efficiency of statistical modeling is insufficiently studied and an opportunity of increase of compression ratio in particular is connected with its decision.

The existing statistical models used for data compression, differ complexity and heterogeneity of structure, and also that they are sensitive to type of modeled data that complicates their practical realization and application [Bell, 1989; Witten, 1989; Ker-Chang Chang, 1993]. At the same time the models which are based on ANN, differ simplicity and uniformity of structure, an opportunity of use of the same algorithms of adjustment and functioning in various models, and also some other useful properties [Хайкин, 2006]. Therefore, prospects of development of statistical models are connected with use as modeling system at lossless compression of the artificial neural networks.

Target of Settings

The problem of statistical modeling of data sources during realization of lossless compression consists in definition of probabilities of occurrence of elements of compressed data on an input of coding system. One of methods of reception of estimations of these probabilities is forecasting occurrence of elements of data in an input stream. At use ANN such forecasting is carried out similarly to forecasting of time series [Ежов, 1998]. Distinction consists basically in principle of forming of learning set for ANN. At forecasting time series there are accessible all values that enables training of ANN with the given level of an error. At statistical modeling data come in an input of modeling system in the consecutive portions that demands permanent recustomizing of model during its functioning with the purpose of adaptation to changes of data [Schmidhuber, 1996]. In this connection there is a necessity of development and research of special methods and algorithms of formation of learning set for neural models which would consider features of its functioning on the given data stream.

Main Results

Description of neural model.

For carrying out of researches the two-layer back propagation neural network with adjustable quantity of neurons in an input layer (if the inputs of ANN are considered as the first layer such ANN is considered to be three-layer) was used. As one of possible variants of realization of transferring function the hyperbolic tangent – *th x* – was used. Other parameters ANN had following values:

- the maximal absolute error of training per one example: 0,001;

- a steepness of transfer function: 1,0 – for an input layer, 0,1 – for a target layer;

- factor of learning speed: 0,1 – for all layers.

Proceeding from convenience of realization neural models on the existing computer-based means dimension of 1 byte has been accepted for elements of the input data stream that is not basic restriction of parameters of considered model. Accordingly, the quantity ANN inputs was defined as 8N, where N – quantity of data elements on which forecasting (that is context length), and the quantity of outputs was fixed and was equal 8 (fig. 1). Coding of inputs and target signals of a network has great value for process of ANN learning because finally it defines finish of learning. Zero bits of inputs and target signals of a network were coded by value "-0,5", and set bits – "+0,5" as use of such values allows to consider process of learning completed only at concurrence of signs of the received and required values of a network without achievement of a concrete numerical result.

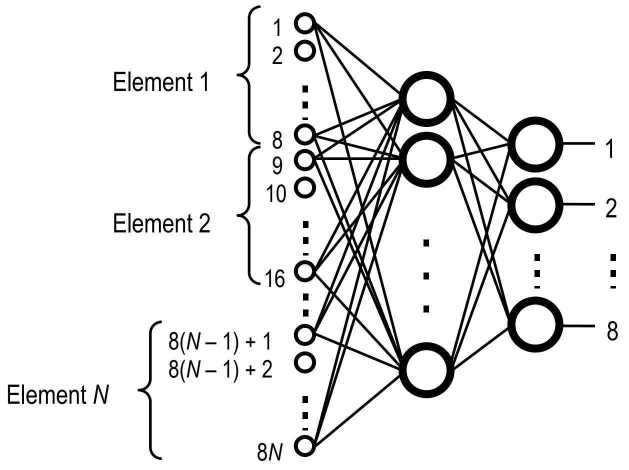


Fig. 1

Influence of context length.

During ANN functioning, it was learned of forecasting of the values appearing in an input data stream on contexts with length from 1 up to 15 elements. Here start value answers minimally possible length of a context and the range of its change is defined by known methods of contextual modeling [Ватолин, 2002]. It is established, that quality of forecasting in a greater measure depends on length of a context for the data, described by strongly pronounced internal structure (texts in natural languages and texts of programs) and in a smaller measure – for data of type "an analog signal" (the digitized images and a sound). For text data dependence of optimum length of a context on volume of data is revealed that is coordinated with experimental data. At use of contexts of optimum length the quantity of correct forecasts has increased on the average on (25 – 45) %.

Window learning mode.

As a possible way of increase of quantity of correct forecasts window mode of ANN learning is considered at which learning is carried out not on one context, but on set of the overlapped consecutive contexts that are directly previous of prediction value (fig. 2). Researches were spent for windows with a width from 2 up to 50 contexts. The range of change width of a window is chosen experimentally. The basic problem of formation of learning set for a window mode of learning was elimination of inconsistent contexts [Тарасенко, 2001], the presence of which cycles process of learning and decreases speed of functioning of model sharply. Elimination of inconsistent contexts was carried out as follows:

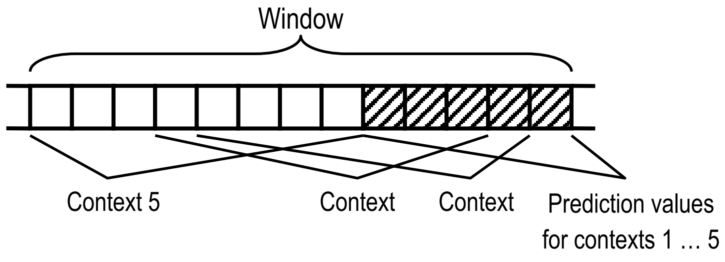
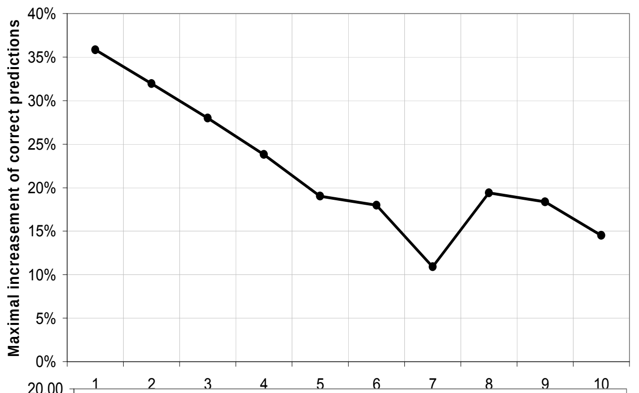


Fig. 2

if there were two or more conterminous contexts with various prediction values in a window the nearest to prediction value was left and all previous were removed.

As a result of test of neural model in a window mode of learning it has been established, that the increase in width of a window allows raising quantity of correct forecasts (on 40 % for text data), however it reduces speed of ANN learning, and speed of model as a whole. Besides the increase in length of a context, it leads to reduction of number of the correct forecasts connected with increase of width of a window (fig. 3). Thus the window mode of learning is favorable only at use of contexts of small length (1 … 4 elements) under condition of elimination of inconsistent contexts.



Selective learning

The analysis of change of quality of forecasting during model functioning has shown, that neural model functioning on all set of contexts with any width of a window is characterized by initial splash in quantity of correct forecasts and its following decrease to some level. It is supposed, that such behavior of quality of forecasting shows that only some subset from all contexts used for ANN learning represents laws, characteristic for all set of data, and the others contexts have casual character and lead to decrease in quality of training process. For check of the given assumption learning of a network has been led in two modes: 1) standard (on all set of contexts); 2) selective (on set of contexts after which forecasting was successful). As a result it was revealed, that at selective learning change of quantity of correct forecasts was absolutely different unlike a standard mode (fig. 4); thus the stable increase in quality of forecasting during all process of ANN training and functioning is observed.

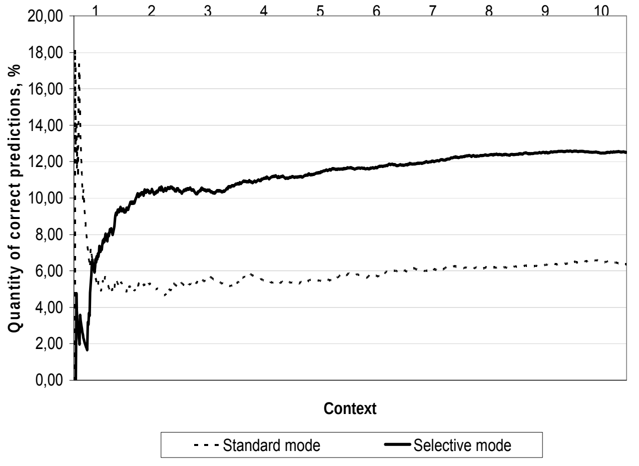


Fig. 4

Essential lack of a selective learning mode is dependence of quality of forecasting process on an initial condition of model, which is on a choice of initial values of ANN weight factors. The reason of such dependence is that at a selective mode training of a network begins with the first context for which the correct forecast has been executed, thus the probability of such forecast depends on initial values of ANN weight factors. In connection with that casual values are usually appropriated to ANN weight factors, there is a probability of their unsuccessful choice, as entails sharp decrease in speed, and quality of ANN learning also.

The account of influence of an initial condition of model on forecasting process is offered to be carried out by one of the next two ways. The first way assumes to carry out learning of a network on contexts for which the forecast has been executed with the error that is not exceeding given. Thus, in view of that modeled data are represented by binary codes, for an estimation of an error it is expedient to use Hemming distance. At a choice of Hemming distance it is important to consider, that when admissible error of forecasting increases, the quantity of the casual contexts increases also and it makes worse quality of forecasting, and when admissible error of forecasting decreases, the probability of the duly beginning of ANN learning decreases also.

The second way assumes application of the mixed mode of learning at which in the beginning of ANN functioning it is trained in a usual mode, which is on all meeting contexts (“start” of model), and after a time learning becomes selective. The choice of length of data on which “start” of model is carried out depends on type of data, and also their structure, defined by a solved problem and demands special studying.

Conclusion

The choice of a context length for learning set should be carried out in view of type of modelled data.

The choice of a context length without taking into account type of modeled data does not allow getting optimum quantity of correct forecasts that can be compensated by increase in width of a learning window.

Forming of training set from small contexts (1 … 4 elements), and also with use of windows in width from 7 up to 10 contexts is preferable to increase in number of correct forecasts at the solving of problems of losless compression for various types of data.

The greatest number of correct forecasts in the data stream can be achieved at the use of a selective mode of learning with preliminary “start” of model or with use of forecasts within the limits of Hemming distance no more than 2 bits.

Bibliography

[Jiang, 1999] Jiang J. Image compression with neural networks – A survey. Signal Processing: Image Commun. 1999. – V. 14. – P. 737 – 760.

[Cottrell, 1987] Cottrell G. V., Munro P., Zipser D. Image compression by back propagation: An example of extentional programming // Proc. 9th Annual Confer., Cognitive Soc. – 1987. – P. 461 – 473.

[Verma, 1999] B. Verma, M. Blumenstein, S. Kulkarni. A New Compression Technique Using an Artifical Neural Network // Journal of Intelligent Systems. – 1999. – Vol. 9. – P. 39 – 53.

[Ватолин, 2002] Ватолин Д., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М.: ДИАЛОГ-МИФИ, 2002. – 384 с.

[Rissanen, 1981] Jorma Rissanen, Glen G. Langdon. Universal modelling and coding. // IEEE transaction on information theory. 1981. V. IT – 27. № 1. – P. 12 – 33.

[Кричевский, 1989] Кричевский Р. Е. Сжатие и поиск информации. – М.: Радио и связь, 1989. – 168 с.

[Bell, 1989] Bell T. C., Moffat A. M. A note on the DMC data compression scheme. // Comput. J. – 1989. – V. 32. – P.16–20.

[Witten, 1989] Bell T. C., Witten I. H., Cleary J. G. Modeling for text compression // ACM Computer Surveys. – 1989. V. 24. – P. 555 – 591.

[Ker-Chang Chang, 1993] H. Ker-Chang Chang, Shing-Hong Chen. A New Locally Adaptive Data Compression Scheme Using Multilist Structure // Comput. J. – 1993. V. 36.

[Хайкин, 2006] Хайкин С. Нейронные сети: полный курс, 2-е издание.: Пер. с англ. – М.: Издательский дом “Вильямс”, 2006. – 1104 с.

[Ежов, 1998] Ежов А. А., Шумский С. А. Нейрокомпьютинг и его применение в экономике и бизнесе (серия “Учебники экономико-аналитического института МИФИ” под ред. проф. В. В. Харитонова). – М.: МИФИ, 1998. – 244 с.

[Schmidhuber, 1996] Schmidhuber J. Sequential neural text compression // IEEE transaction on Neural Networks. – 1996. – V. 7 (1). – P. 142 – 146.

[Тарасенко, 2001] Тарасенко Р. А., Крисилов В. А. Предварительная оценка качества обучающей выборки для нейронных сетей в задачах прогнозирования временных рядов. // Тр. Одес. политехн. ун-та. – Одесса, 2001. – Вып. 1. С. 25 – 28.

Authors' Information

Yu. L. Ivaskiv - National Aviation University, Doctor of Science; professor; P.O.Box: 03058, Kosmonavta Komarova Ave., bl. 1, Kiev, Ukraine; e-mail: post@nau.edu.ua.

V. V. Levchenko - Industrially-Economic College of National Aviation University, post-graduate student; P.O.Box: 03126, Geroev Sevastopolya St., bl. 44 - A, 53, Kiev, Ukraine; e-mail: vvl@voliacable.com

NETWORKS OF EVOLUTIONARY PROCESSORS (NEP) AS DECISION SUPPORT SYSTEMS[1]

Nuria Gómez Blas, Miguel Angel Díaz, Juan Castellanos, Francisco Serradilla

Abstract: This paper presents the application of Networks of Evolutionary Processors to Decision Support Systems, precisely Knowledge-Driven DSS. Symbolic information and rule-based behavior in Networks of Evolutionary Processors turn out to be a great tool to obtain decisions based on objects present in the network. The non-deterministic and massive parallel way of operation results in NP-problem solving in linear time. A working NEP example is shown.

Keywords: Natural Computing, Networks of Evolutionary Processors, Decision Support Systems.

ACM Classification Keywords: F.1.2 Modes of Computation, I.6.1 Simulation Theory, H.1.1 Systems and Information Theory.

Introduction

There are many approaches to decision-making and because of the wide range of domains in which decisions are made; the concept of decision support system (DSS) is very broad. A DSS can take many different forms. In general, we can say that a DSS is a computerized system for helping make decisions. A decision is a choice between alternatives based on estimates of the values of those alternatives. Supporting a decision means helping people working alone or in a group, to gather intelligence, generate alternatives and make choices. Supporting the choice making process involves supporting the estimation, the evaluation and/or the comparison of alternatives. In practice, references to DSS are usually references to computer applications that perform such a supporting role [Alter, 1980].

Abbreviated DSS, the term refers to an interactive computerized system that gathers and presents data from a wide range of sources, typically for business purposes. DSS applications are systems and subsystems that help people make decisions based on data that is culled from a wide range of sources. For example: a national on-line book seller wants to begin selling its products internationally but first needs to determine if that will be a wise business decision. The vendor can use a DSS to gather information from its own resources (using a tool such as OLAP) to determine if the company has the ability or potential ability to expand its business and also from external resources, such as industry data, to determine if there is indeed a demand to meet. The DSS will collect and analyze the data and then present it in a way that can be interpreted by humans. Some decision support systems come very close to acting as artificial intelligence agents.

DSS applications are not single information resources, such as a database or a program that graphically represents sales figures, but the combination of integrated resources working together.

Using the mode of assistance as the criterion [Power, 2002] differentiates communication-driven DSS, data-driven DSS, document-driven DSS, knowledge-driven DSS, and model-driven DSS.

- A model-driven DSS emphasizes access to and manipulation of a statistical, financial, optimization, or simulation model. Model-driven DSS use data and parameters provided by users to assist decision makers in analyzing a situation; they are not necessarily data intensive. Dicodess is an example of an open source model-driven DSS generator [Gachet, 2004].
- A communication-driven DSS supports more than one person working on a shared task; examples include integrated tools like Microsoft's NetMeeting or Groove [Stanhope, 2002].

[1] This work has been possible thanks to an Academic Stay at Department of Computer Science, King's College – University of London.

- A data-driven DSS or data-oriented DSS emphasizes access to and manipulation of a time series of internal company data and, sometimes, external data.
- A document-driven DSS manages, retrieves and manipulates unstructured information in a variety of electronic formats.
- A knowledge-driven DSS provides specialized problem solving expertise stored as facts, rules, procedures, or in similar structures.

By incorporating AI techniques in a decision support system, we make that DSS artificially intelligent - capable of displaying behavior that would be regarded as intelligent if observed in humans. Artificially intelligent DSSs are becoming increasingly common. Perhaps the most prominent of these are expert systems, which support decision making by giving advice comparable to what human experts would provide.

This paper is focused on knowledge-driven DSS using Artificial Intelligence models. Networks of Evolutionary Processors have rules, facts, and collaboration among processors to generate a final decision based on evolutionary steps that take place in processors. Next section describes the computational model of Networks of Evolutionary Processors. And finally, an example is shown.

Networks of Evolutionary Processors

A network of evolutionary processors of size n is a construct NEP = $(V, N_1, N_2,\dots, N_n, G)$, where V is an alphabet and for each $1 \leq i \leq n$, $N_i = (M_i, A_i, PI_i, PO_i)$ is the i-th evolutionary node processor of the network. The parameters of every processor are:

- M_i is a finite set of evolution rules of one of the following forms only:

  $a \prod b$, $a, b \in V$ (substitution rules)

  $a \prod \varepsilon$, $a \in V$ (deletion rules)

  $\varepsilon \prod a$, $a \in V$ (insertion rules)

  More clearly, the set of evolution rules of any processor contains either substitution or deletion or insertion rules.
- A_i is a finite set of strings over V. The set A_i is the set of initial strings in the i-th node. Actually, in what follows, we consider that each string appearing in any node at any step has an arbitrarily large number of copies in that node, so that we shall identify multisets by their supports.
- PI_i and PO_i are subsets of V^*representing the input and the output filter, respectively. These filters are defined by the membership condition, namely a string $w \in V^*$ can pass the input filter (the output filter) if $w \in PI_i$ ($w \in PO_i$).

$G = (N_1, N_2,\dots, N_n, E)$ is an undirected graph called the underlying graph of the network [Paun, 2002] [Paun, 2000]. The edges of G, that is the elements of E, are given in the form of sets of two nodes. Kn denotes the complete graph with n vertices. By a configuration (state) of an NEP as above we mean an n-tuple $C = (L_1, L_2,\dots, L_n)$\$, with $L_i \subseteq V^*$ for all $1 \leq i \leq n$. A configuration represents the sets of strings (remember that each string appears in an arbitrarily large number of copies) which are present in any node at a given moment; clearly the initial configuration of the network is $C_0 = (A_1, A_2,\dots, A_n)$.

A configuration can change either by an evolutionary step or by a communicating step. When changing by an evolutionary step, each component L_i of the configuration is changed in accordance with the evolutionary rules associated with the node i. When changing by a communication step, each node processor N_i sends all copies of the strings it has which are able to pass its output filter to all the node processors connected to N_i and receives all copies of the strings sent by any node processor connected with N_i providing that they can pass its input filter [Manea, 2006] [Martin, 2005] [Garey, 1979].

Theorem 1. A complete NEP of size 5 can generate each recursively enumerable language.

Theorem 2. A star NEP of size 5 can generate each recursively enumerable language.

Theorem 3. The bounded PCP can be solved by an NEP in size and time linearly bounded by the product of K and the length of the longest string of the two Post lists.

An Example

Opportunities for building business expert systems abound for both small and large problems. In each case, the expert system is built by developing its rule set. The planning that precedes rule set development is much like the planning that would precede any project of comparable magnitude within the organization. The development process itself follows an evolutionary spiral composed of development cycles.

Each cycle picks up where the last ended, building on the prior rule set. For a developer, the spiral represents a continuing education process in which more and more of an expert's reasoning knowledge is discovered and formalized in the rule set. Here, each development cycle was presented in terms of seven consecutive stages. Other characterizations of a development cycle (involving different stages or sequences) may be equally valuable. Many aspects of traditional systems analysis and project management can be applied to the development of expert systems.

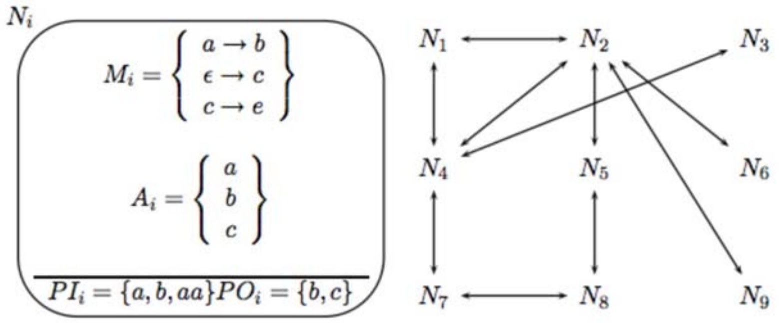

N_i
$M_i = \{a \to b, \epsilon \to c, c \to e\}$
$A_i = \{a, b, c\}$
$PI_i = \{a, b, aa\} PO_i = \{b, c\}$
N_1
N_2
N_3
N_4
N_5
N_6
N_7
N_8
N_9

Figure 1.- A sample Network of Evolutionary Processors

Rule set development is a process of discovery and documentation. Research continues in search of ways of automating various aspects of the process. It would not be surprising to eventually see expert systems that can assist in this process -- that is, an expert system that "picks the mind" of a human expert in order to build new expert systems. Until that time comes, the topics discussed in this chapter should serve as reminders to developers of expert decision support systems about issues to consider during the development process.

As it stands, rule-based systems are the most widely used and accepted AI in the world outside of games. The fields of medicine, finance and many others have benefited greatly by intelligent use of such systems. With the combination of rule-based systems and ID trees, there is great potential for most fields. Here it is an example of a rule-base expert system.

Table 1.- Rule-based decision support system applied to medical diagnosis (snapshoot)

Assertions A1: runny nose A2: temperature=101.7 A3: headache A4: cough Rules R1: if (nasal congestion) (viremia) then diagnose (influenza) exit R2: if (runny nose) then assert (nasal congestion)	R3: if (body- aches) then assert (achiness) R4: if (temp >100) then assert (fever) R5: if (headache) then assert (achiness) R6: if (fever) (achiness) (cough) then assert (viremia)

It is necessary to define the underlying graph In order to simulate previous example with a Network of Evolutionary Processors. First of all, an initial processor containing the assertions and basic rules will forward important information to a second processor, which is in charge of a given disease, and forward its result to a container processor. This process can be shown in figure 2.

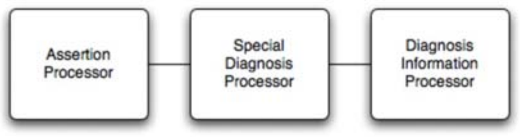


Figure 2.- Network of Evolutionary Processors Architecture sample

Previous simple example can be extended to a more general diagnosis system as detailed in figure 3. There exists one processor or even more processors in charge of a located diagnosis problem such as: influenza, migraines, heart diseases, etc… These local diagnosis processors can communicate each other to auto complete information diagnosis. Finally, each result of diagnosis processors is sent to an information processor that can combine multiple diagnoses or just show them.

Configuration of a Network of Evolutionary Processors

Table 2 shows an XML file with the NEP initial configuration corresponding to the medical diagnosis shown in table 1. There are three processors (see figure 2): assertion process (name = 0), specific diagnosis processor (name = 1) and diagnosis information processor (name = 2). Objects travel trough these processors until a final diagnosis is present in the last one. Obviously, this is a simple example, but according to figure 3 the NEP architecture could be complicated in order to obtain a more sophisticated diagnosis. Main idea is to put some assertions in one or more processors and then let them evolve using evolution steps (rules application) and communication steps.

This configuration file is parsed into JAVA objects and a separate thread for each processor is created; also each rule and filter are coded as threads in order to keep the massive parallelization defined in the theoretical model of Networks of Evolutionary Processors.

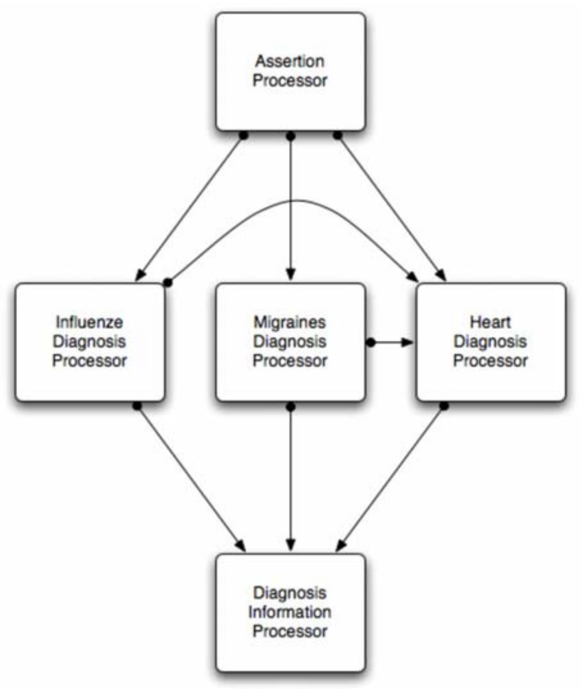

Assertion Processor
Influenze Diagnosis Processor
Migraines Diagnosis Processor
Heart Diagnosis Processor
Diagnosis Information Processor

Figure 3.- Network of Evolutionary Processors Architecture for medical diagnosis

Table 2.- NEP initial configuration corresponding to table 1

```
<?xml version="1.0"?>
<NEP>
   <processor>
      <name>0</name>
      <object>runny nose</object>
      <object>high temperature</object>
      <object>headache</object>
      <object>cough</object>
      <rule>
         <antecedent>
            <object>runny nose</object>
         </antecedent>
         <consequent>
            <object>nasal congestion</object>
         </consequent>
      </rule>
      <rule>
         <antecedent>
            <object>body-aches</object>
         </antecedent>
         <consequent>
            <object>achiness</object>
         </consequent>
      </rule>
      <rule>
         <antecedent>
            <object>high temperature</object>
         </antecedent>
         <consequent>
```

```
   <processor>
      <name>1</name>
      <rule>
         <antecedent>
            <object>nasal congestion</object>
            <object>viremia</object>
         </antecedent>
         <consequent>
            <object>influenza</object>
         </consequent>
      </rule>
      <rule>
         <antecedent>
            <object>fever</object>
            <object>achiness</object>
            <object>cough</object>
         </antecedent>
         <consequent>
            <object>viremia</object>
         </consequent>
      </rule>
      <inputfilter>
         <object>nasal congestion</object>
         <object>fever</object>
         <object>achiness</object>
         <object>cough</object>
      </inputfilter>
      <outputfilter>
         <object>influenza</object>
         <object>fever</object>
      </consequent>
   </rule>
   <rule>
      <antecedent>
         <object>headache</object>
      </antecedent>
      <consequent>
         <object>achiness</object>
      </consequent>
   </rule>
   <inputfilter>
   </inputfilter>
   <outputfilter>
      <object>nasal congestion</object>
      <object>fever</object>
      <object>achiness</object>
      <object>cough</object>
   </outputfilter>
</processor>
```

```
      </outputfilter>
   </processor>
   <processor>
      <name>2</name>
      <inputfilter>
         <object>influenza</object>
      </inputfilter>
      <outputfilter>
      </outputfilter>
   </processor>
   <conn>
      <from>0</from>
      <to>1</to>
   </conn>
   <conn>
      <from>1</from>
      <to>2</to>
   </conn>
</NEP>
```

Note that connections among processors are defined in a unidirectional way, but any type of connection can be expressed in the XML configuration file in order to have a more complex underlying graph in the network architecture.

Simulation Results

Results concerning simulation of NEP configuration in table 2 can be seen in table 3. `Processor 2:3`, which is the output processor, has the object `influenza`, desired result. This object is generated in `Processor 1:2` using rules inside it. NEP behavior is totally non-deterministic since rules, filters and processors run together in parallel. This example only uses substitution rules, neither insertion nor deletion rules are coded.

Table 3.- Final configuration of NEP corresponding to described configuration on table 2

```
[----------
Processor 0 : 1
Rules: [[runny nose] --> [nasal congestion], [body-aches] --> [achiness], [high temperature]
--> [fever], [headache] --> [achiness]]
Objects: [runny nose, high temperature, headache]
Output Filter: [nasal congestion, fever, achiness, cough]
Input Filter: []
----------
, ----------
Processor 1 : 2
Rules: [[nasal congestion, viremia] --> [influenza], [fever, achiness, cough] --> [viremia]]
Objects: [cough, fever, achiness, viremia, nasal congestion, influenza]
Output Filter: [influenza]
Input Filter: [nasal congestion, fever, achiness, cough]
----------
, ----------
Processor 2 : 3
Rules: []
Objects: [influenza]
Output Filter: []
Input Filter: [influenza]
----------
]
```

The great disadvantage is that a given NEP can only solve a given problem; if it is necessary to solve another problem (maybe a little variation) then another different NEP has to be implemented. The idea of learning tries to undertake such disadvantage proposing a model able to solve different kinds of problems (that is a general class of problems). Learning can be based on the self-organizing maps. There are a lot of open problems that need to be solved in order to show the computational power of this learning idea, but the possibility to compute NP-problems is promising apart from the massive parallelization and non-determinism of the model.

Conclusion

This paper has introduced the computational paradigm Networks of Evolutionary Processors. NEPs can be easily applied to Knowlede-driven Decision Support Systems due to the inherent rule-based behavior of NEPs. JAVA implementation of this model works as defined by the theoretical background of NEPs: massive parallelization and non-deterministic behavior.

Connectionists' models such as Neural Networks can be taken into account to develop NEP architecture in order to improve behavior. As a future research, learning concepts in neural networks can be adapted in a NEP architecture provided the numeric-symbolic difference in both models. NEPs can be considered universal models since they are able to solve NP-problems.

Bibliography

[Alter, 1980] Alter, S. L. Decision support systems: current practice and continuing challenges. Reading, Mass., Addison-Wesley Pub. (1980).

[Gachet, 2004] Gachet, A. Building Model-Driven Decision Support Systems with Dicodess. Zurich, VDF. (2004).

[Garey, 1979] M. Garey and D. Johnson. Computers and Intractability. A Guide to the Theory of NP-completeness. Freeman, San Francisco, CA, (1979).

[Manea, 2006] F. Manea, C. Martın-Vide, and V. Mitrana. All NP-problems can be solved in polynomial time by accepting networks of splicing processors of constant size. Proc. of DNA 12, in press. (2006).

[Martin, 2005] C. Martin-Vide and V. Mitrana. Networks of evolutionary processors: Results and perspectives. Molecular Computational Models: Unconventional Approaches. 78–114, Idea Group Publishing, Hershey. (2005).

[Paun, 2000] Paun G. Computing with Membranes. In: Journal of Computer and Systems Sciences, 61, 1. 108--143. (2000).

[Paun, 2002] Gh. Paun. Membrane Computing. An Introduction, Springer-Verlag, Berlin, (2002).

[Power, 2002] Power, D. J. Decision support systems: concepts and resources for managers. Westport, Conn., Quorum Books. (2002).

[Stanhope, 2002] Stanhope, P. Get in the Groove: building tools and peer-to-peer solutions with the Groove platform. New York, Hungry Minds. (2002).

Authors' Information

Nuria Gómez Blas – Dept. Organización y Estructura de la Información, Escuela Universitaria de Informática, Universidad Politécnica de Madrid, Crta. De Valencia km. 7, 28031 Madrid, Spain; e-mail: ngomez@dalum.eui.upm.es

Miguel Angel Díaz – Dept. Organización y Estructura de la Información, Escuela Universitaria de Informática, Universidad Politécnica de Madrid, Crta. De Valencia km. 7, 28031 Madrid, Spain; e-mail: mdiaz@eui.upm.es

Juan Castellanos – Dept. Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, 28660 Madrid, Spain; e-mail: jcastellanos@fi.upm.es

Francisco Serradilla – Dept. Sistemas Inteligentes Aplicados, Escuela Universitaria de Informática, Universidad Politécnica de Madrid, Crta. De Valencia km. 7, 28031 Madrid, Spain; e-mail: fserra@eui.upm.es

NETWORKS OF EVOLUTIONARY PROCESSORS: JAVA IMPLEMENTATION OF A THREADED PROCESSOR[1]

Miguel Angel Díaz, Luis Fernando de Mingo López, Nuria Gómez Blas

Abstract: This paper is focused on a parallel JAVA implementation of a processor defined in a Network of Evolutionary Processors. Processor description is based on JDom, which provides a complete, Java-based solution for accessing, manipulating, and outputting XML data from Java code. Communication among different processor to obtain a fully functional simulation of a Network of Evolutionary Processors will be treated in future. A safe-thread model of processors performs all parallel operations such as rules and filters. A non-deterministic behavior of processors is achieved with a thread for each rule and for each filter (input and output). Different results of a processor evolution are shown.

Keywords: Networks of Evolutionary Processors, Membrane Systems, Natural Computation.

ACM Classification Keywords: F.1.2 Modes of Computation, I.6.1 Simulation Theory, H.1.1 Systems and Information Theory

Introduction

A network of evolutionary processors of size n is a construct NEP = $(V, N_1, N_2, \ldots, N_n, G)$, where V is an alphabet and for each $1 \le i \le n$, $N_i = (M_i, A_i, PI_i, PO_i)$ is the i-th evolutionary node processor of the network. The parameters of every processor are:

- M_i is a finite set of evolution rules of one of the following forms only:
  $a \prod b$, $a, b \in V$ (substitution rules), $a \prod \varepsilon$, $a \in V$ (deletion rules), $\varepsilon \prod a$, $a \in V$ (insertion rules)
  More clearly, the set of evolution rules of any processor contains either substitution or deletion or insertion rules.
- A_i is a finite set of strings over V. The set A_i is the set of initial strings in the i-th node. Actually, in what follows, we consider that each string appearing in any node at any step has an arbitrarily large number of copies in that node, so that we shall identify multisets by their supports.
- PI_i and PO_i are subsets of V^*representing the input and the output filter, respectively. These filters are defined by the membership condition, namely a string $w \in V^*$ can pass the input filter (the output filter) if $w \in PI_i$ ($w \in PO_i$).

$G = (N_1, N_2, \ldots, N_n, E)$ is an undirected graph called the underlying graph of the network [Paun, 2002] [Paun, 2000]. The edges of G, that is the elements of E, are given in the form of sets of two nodes. Kn denotes the complete graph with n vertices. By a configuration (state) of an NEP as above we mean an n-tuple $C = (L_1, L_2, \ldots., L_n)$\$, with $L_i \subseteq V^*$ for all $1 \le i \le n$. A configuration represents the sets of strings (remember that each string appears in an arbitrarily large number of copies) which are present in any node at a given moment; clearly the initial configuration of the network is $C_0 = (A_1, A_2, \ldots, A_n)$.

A configuration can change either by an evolutionary step or by a communicating step. When changing by an evolutionary step, each component L_i of the configuration is changed in accordance with the evolutionary rules associated with the node i. When changing by a communication step, each node processor N_i sends all copies of the strings it has which are able to pass its output filter to all the node processors connected to N_i and receives all copies of the strings sent by any node processor connected with N_i providing that they can pass its input filter [Manea, 2006] [Martin, 2005] [Garey, 1979].

[1] This work has been possible thanks to an Academic Stay at Department of Computer Science, King's College – University of London.

Theorem 1. A complete NEP of size 5 can generate each recursively enumerable language.

Theorem 2. A star NEP of size 5 can generate each recursively enumerable language.

Theorem 3. The bounded PCP can be solved by an NEP in size and time linearly bounded by the product of K and the length of the longest string of the two Post lists.

Next sections deal with the JAVA implementation of a processor as the first step to achieve a fully functional simulation of NEPs. The non-deterministic behavior of NEPs must be taken into account, that is, a massive parallel implementation is reached having each rule in a thread and each filter in a thread. Objects in processor are locked to avoid mutual exclusion problems due to concurrent programming. [Fahlman, 1983] [Errico, 1994]

JAVA Implementation

NEP processors must behave in a non-deterministic way. Configuration changes are outcome by a communication step or by an evolutionary step, but these two steps are accomplished with no order at all, that is, evolution or communication is chosen depending on the thread model of processor [Diaz, 2007]. Rules and filters (input and output) are implemented as threads extending `Runnable` interface. Therefore a processor is the parent of a set of threads, which use objects in processor in a mutual exclusion region.

Figure 1 shows an UML class diagram corresponding to all classes involved in the definition of a NEP processor. Rules, filters and objects are part of a processor. Filters can be either input filters or output filters, depending on their behavior, controlling how objects are sent or are received by different processors. Substitution rules have an antecedent and a consequent implemented as an object set. When a processor is run through the `start` method, it starts in a cascade way the rule threads and filter threads. The whole system is prepared to a NEP implementation; only the communication classes must be coded in order to add the communication step to NEP since there exists methods to send a to receive objects in the processor class.

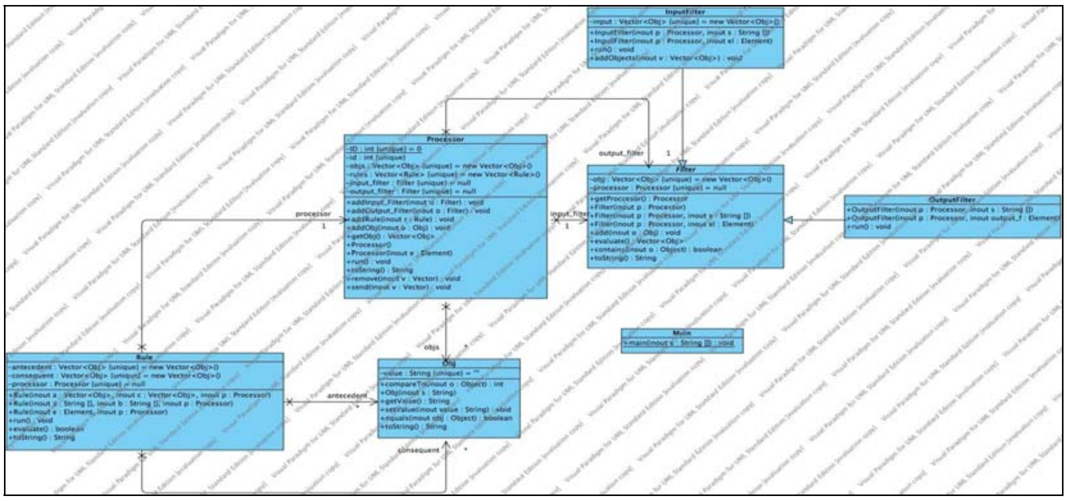

InputFilter
-input : Vector<Obj> {unique} = new Vector<Obj>()
+InputFilter(inout p : Processor, inout s : String [])
+InputFilter(inout p : Processor, inout el : Element)
+run() : void
+addObjects(inout v : Vector<Obj>) : void
Processor
-ID : int {unique} = 0
-id : int {unique}
-objs : Vector<Obj> {unique} = new Vector<Obj>()
-rules : Vector<Rule> {unique} = new Vector<Rule>()
-input_filter : Filter {unique} = null
-output_filter : Filter {unique} = null
+addInput_Filter(inout u : Filter) : void
+addOutput_Filter(inout o : Filter) : void
+addRule(inout r : Rule) : void
+addObj(inout o : Obj) : void
+getObj() : Vector<Obj>
+Processor()
+Processor(inout e : Element)
+run() : void
+toString() : String
+remove(inout v : Vector) : void
+send(inout v : Vector) : void
output_filter
1
Filter
-obj : Vector<Obj> {unique} = new Vector<Obj>()
-processor : Processor {unique} = null
+getProccesor() : Processor
+Filter(inout p : Processor)
+Filter(inout p : Processor, inout s : String [])
+Filter(inout p : Processor, inout el : Element)
+add(inout o : Obj) : void
+evaluate() : Vector<Obj>
+contains(inout o : Object) : boolean
+toString() : String
OutputFilter
+OutputFilter(inout p : Processor, inout s : String [])
+OutputFilter(inout p : Processor, inout output_f : Element)
+run() : void
processor
1
input_filter
1
Main
+main(inout s : String []) : void
objs
*
Rule
-antecedent : Vector<Obj> {unique} = new Vector<Obj>()
-consequent : Vector<Obj> {unique} = new Vector<Obj>()
-processor : Processor {unique} = null
+Rule(inout a : Vector<Obj>, inout c : Vector<Obj>, inout p : Processor)
+Rule(inout a : String [], inout b : String [], inout p : Processor)
+Rule(inout e : Element, inout p : Processor)
+run() : void
+evaluate() : boolean
+toString() : String
antecedent
*
Obj
-value : String {unique} = ""
+compareTo(inout o : Object) : int
+Obj(inout s : String)
+getValue() : String
+setValue(inout value : String) : void
+equals(inout obj : Object) : boolean
+toString() : String
consequent
*

Figure 1.- UML Class diagram of a NEP processor.

This is the basic composition of an evolutionary processor; nevertheless, there exist NEP architectures that have forbidden filters in the input and in the output. Differences in the implementation for the resolution of problems will be defined as types of the generic model for such given kind of problems.

Processors

According to figure 1, each processor has a number of rules, objects and one input filter and one output filter. When the processor thread starts all rule threads are started and input/output threads to, see figure 2 and listing 1. Objects in processor are store using the `Vector` class that is thread-safe, so synchronization is guaranteed.

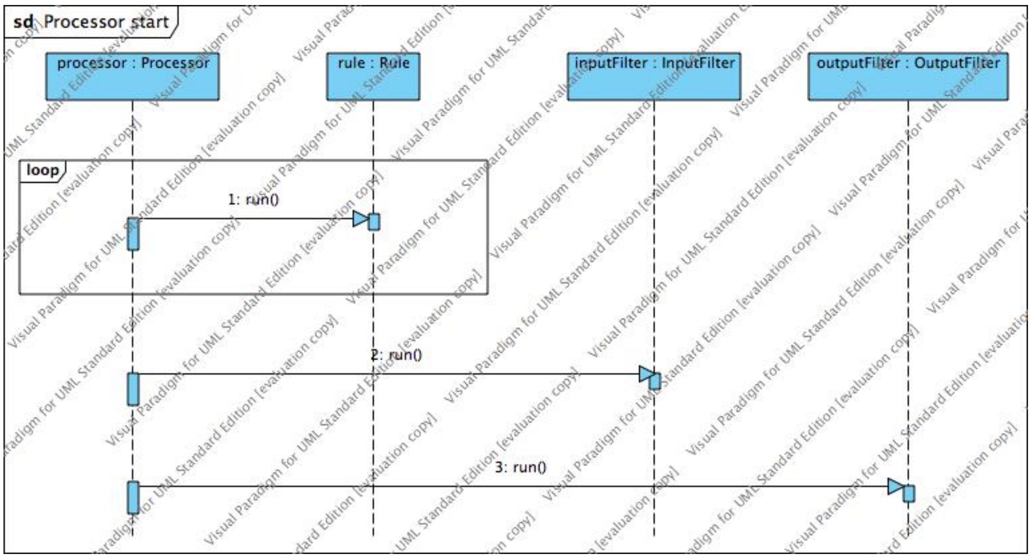


Figure 2.- UML Sequence diagram of processor.

```
public void run() {
       for (int i=0; i<this.rules.size(); i++)
              new Thread(this.rules.get(i)).start();
       new Thread((OutputFilter) this.output_filter).start();
       new Thread((InputFilter) this.input_filter).start();
       return;    }
```

Listing 1.- Processor behavior schema.

Processors can send and receive objects provided filter constraints are satisfied. This communication is perform in the following way:

- Sending objects. Output filter will check constraints and all objects that satisfy them will be removed from the object pool. In future they will be sent to processors connected to this one.

```
public void run() {
       Vector v = null;
       while (true) {
              v = super.evaluate();
              super.getProccesor().remove(v);
              super.send(v);
       }
}
```

- Receiving objects. When the method `send` is invoked from another processor, some object will be located in the input filter pool to be checked, if it pass the constraints then it will be added to the processor if it is not present.

```
synchronized public void send(Vector v) {
       ((InputFilter)this.input_filter).addObjects(v);
}
```

Please note that all rules and filters are threaded, so the non-deterministic behavior is guaranteed.

Rules

Rule threads are quite simple, they only check if the antecedent objects are in the processor object pool, if so objects in consequent are incorporated in processor pool, see figure 3. There is no order when applying rules, they all in separated threads, therefore they all check object pool at the “same time” and they will be applied at the “same time”. Some random delay has been incorporated in order to make a more realistic simulation.

There are no insertion and deletion rules in our system, but they can be easily added just defining a `null` object in the configuration file `config.xml` and the system will work in such NEP schema.

Listing 2 shows an outline of the rule behavior according to figure 3. It can be noted that if the antecedent is satisfactory evaluated then all consequent objects will be added to the processor pool.

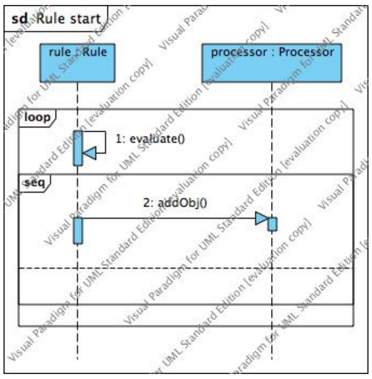


Figure 3.
UML Sequence diagram of rules.

```
while(true) {
  if (this.evaluate()) {
    Enumeration<Obj> e = this.consequent.elements();
    while (e.hasMoreElements()) {
      Obj o = e.nextElement();
      if (!processor.getObj().contains(o))
        this.processor.addObj(o);
    }
  }
}
```

Listing 2.- Rule behavior schema.

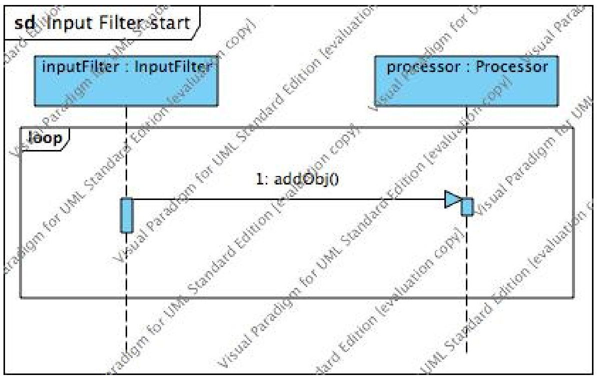
Filters

Filters are implemented as threads. Therefore they run in parallel with rules. When a filter is satisfied then it will remove or add some objects to the processor pool. Main difference between input and output filters is that, see figure 4:

- Input filter just add objects to processor if they pass the constraints.
- Output filter evaluate object pool to guess which objects must be sent out.

Both of them use the functionality of a `Filter` class, which provides the evaluation of objects, see listing 3.

Several filters can be implemented in the evolutionary processors. A filter is a system that allows a symbol to go from one processor to another. Normally, the detection system is to compare a symbol with another one. Among possible filters that an evolutionary processor can have, most common filters are the PI or input filters, and the PO or output filters. A processor can have several filters of each type. This is the basic composition of an evolutionary processor, nevertheless, there exist NEP architectures that have forbidden filters in the input and in the output. Differences in the implementation for the resolution of problems will be defined as types of the generic model for such given kind of problems.



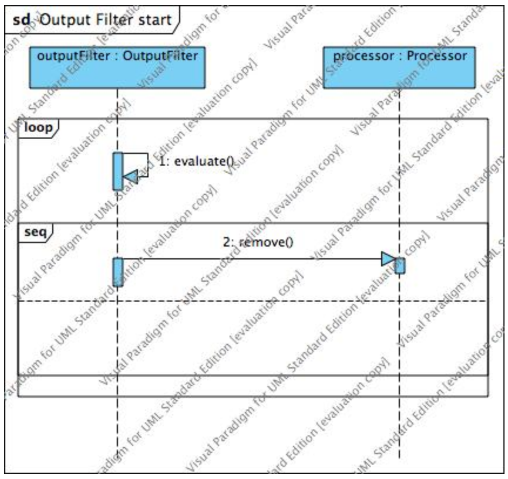


Figure 4.- UML Sequence diagram of Input and Output filters.

Listing 3 shows evaluation method corresponding to output filter to check if objects in pool must be sent out or not to connected processors, if so they will be removed from processor (see sending objects in processor subsection).

```java
public Vector<Obj> evaluate() {
     Vector<Obj> v = this.processor.getObj();
     Enumeration<Obj> e = v.elements();
     v = new Vector<Obj>();
     while(e.hasMoreElements()) {
       Obj o = e.nextElement();
       if (this.obj.contains(o)) if (!v.contains(o)) v.add(o);
     }
     return v;
}
```

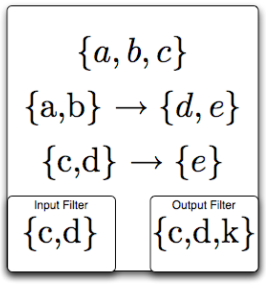
Listing 3.- Filter evaluation using object pool.

Configuration

Configuration of a given processor is done with an XML file. In order to parse such configuration JDOM technology is used. There is no compelling reason for a Java API to manipulate XML to be complex, tricky, unintuitive, or a pain in the neck. JDOM is both Java-centric and Java-optimized. It behaves like Java, it uses Java collections, it is completely natural API for current Java developers, and it provides a low-cost entry point for using XML. While JDOM interoperates well with existing standards such as the Simple API for XML (SAX) and the Document Object Model (DOM), it is not an abstraction layer or enhancement to those APIs. Rather, it provides a robust, lightweight means of reading and writing XML data without the complex and memory-consumptive options that current API offerings provide.

Table 1 shows a processor with its corresponding XML configuration. Elements in such XML are: `processor`, `object`, `rule`, `antecedent`, `consequent`, `inputfilter` and `outputfilter`.

Table 1.- XML Configuration corresponding to sample processor in figure.



```xml
<NEP>
   <processor>
      <object>a</object>
      <object>b</object>
      <object>c</object>
      <rule>
         <antecedent>
            <object>a</object>
            <object>b</object>
         </antecedent>
         <consequent>
            <object>d</object>
            <object>e</object>
         </consequent>
      </rule>
      <rule>
                         <antecedent>

        <object>c</object>
            <object>d</object>
         </antecedent>
         <consequent>
            <object>e</object>
         </consequent>
      </rule>
      <inputfilter>
         <object>c</object>
         <object>d</object>
      </inputfilter>
      <outputfilter>
         <object>c</object>
         <object>d</object>
         <object>k</object>
      </outputfilter>
   </processor>
</NEP>
```

This configuration file is extensible to a NEP system just adding as many processors as needed and adding a XML communication element to define the underlying graph in NEP architecture. As mentioned before, processors have methods to send and to receive objects and only the finalization condition must be taken into account to obtain a fully functional NEP system.

Results: Non-deterministic behavior

This section shows results of evolution corresponding to processor in table 1. When the system starts some threads are create:

- One thread of rule `[a, b] --> [d, e]`
- One thread of rule `[c, d] --> [e]`
- One thread for output filter `[c, d, k]`
- One thread for input filter `[c, d]`

All four threads have access to objects `[a, b, c]` in processor. Depending on which thread access the object set it will be modified with new objects (rules) or some objects will be deleted (output filter) or some objects will be added (input filter). The input filter thread controls when new objects are sent from other processor to this one, this case is not yet implemented but it will be in future, when the NEP system will work as a whole not only isolated processors.

Table 2.- Initial and Final configuration of processor in table 1 (one possible evolution).

Initial Configuration	Final Configuration

`Processor 1`
`Rules: [[a, b] --> [d, e],`
`        [c, d] --> [e]]`
`Objects: [a, b, c]`
`Output Filter: [c, d, k]`
`Input Filter: [c, d]`
	`Processor 1`
	`Rules: [[a, b] --> [d, e],`
	`        [c, d] --> [e]]`
	`Objects:` **`[a, b, e]`**
	`Output Filter: [c, d, k]`
	`Input Filter: [c, d]`

Another execution of same processor outputs results in table 3. Please note objects in final configuration, they are different to those in table 2. This is due to the fact that first rule produces objects d and e but output filter was not activated (the processor stops). In table 2, object d is sent out by output filter.

Table 3.- Initial and Final configuration of processor in table 1 (another possible evolution).

Initial Configuration	Final Configuration

`Processor 1`
`Rules: [[a, b] --> [d, e],`
`        [c, d] --> [e]]`
`Objects: [a, b, c]`
`Output Filter: [c, d, k]`
`Input Filter: [c, d]`
	`Processor 1`
	`Rules: [[a, b] --> [d, e],`
	`        [c, d] --> [e]]`
	`Objects:` **`[a, b, e, d]`**
	`Output Filter: [c, d, k]`
	`Input Filter: [c, d]`

If this processor receives an object it will take part of object set. With this implementation the communication and evolution steps have a non-deterministic way, that is, both steps have no priority. In some cases the communication will take place before the evolution and in other cases evolution will take place before communication.

Conclusion and Future Work

This paper has introduced the novel computational paradigm Networks of Evolutionary Processors that is able to solve NP-problems in linear time. The implementation of such model in a traditional computer is being performed and this paper shows an UML architecture. This architecture is a generic representation of a NEP processor behavior. The non-deterministic behavior is performed using JAVA threads accessing the object pool in processor; depending on the Java Virtual Machine a thread will run faster than another. Tables 2 and 3 show such non-deterministic behavior on a given processor.

Future work includes the simulation of a NEP system; only communication must be added to presented model. Such communication will be expressed in the XML configuration file. A separate thread for each processor will be created in final model together with a communication matrix to open communication channels with other processors.

Bibliography

[Diaz, 2007] Miguel Angel Diaz, Miguel Angel Peña, and Luis F. de Mingo: Simulation of Networks of Evolutionary Processors with Filtered Connections. WESAS Transactions on Information, Science and Applications. Issue 3, Vol. 4. ISSN: 1709-0832. Pp.: 608-616. (2007).

[Errico, 1994] L. Errico and C. Jesshope. Towards a new architecture for symbolic processing. Artificial Intelligence and Information-Control Systems of Robots 94, 31–40, World Scientific, Singapore. (1994).

[Fahlman, 1983] S. Fahlman, G. Hinton, and T. Seijnowski. Massively parallel architectures for AI: NETL, THISTLE and Boltzmann machines. Proc. AAAI National Conf. on AI, 1983:109–113, William Kaufman, Los Altos. (1983).

[Garey, 1979] M. Garey and D. Johnson. Computers and Intractability. A Guide to the Theory of NP-completeness. Freeman, San Francisco, CA, (1979).

[Hillis, 1985] W. Hillis. The Connection Machine. MIT Press, Cambridge, (1985).

[Manea, 2006] F. Manea, C. Martın-Vide, and V. Mitrana. All NP-problems can be solved in polynomial time by accepting networks of splicing processors of constant size. Proc. of DNA 12, in press. (2006).

[Martin, 2005] C. Martin-Vide and V. Mitrana. Networks of evolutionary processors: Results and perspectives. Molecular Computational Models: Unconventional Approaches. 78–114, Idea Group Publishing, Hershey. (2005).

[Paun, 2000] Paun G. Computing with Membranes. In: Journal of Computer and Systems Sciences, 61, 1. 108--143. (2000).

[Paun, 2002] Gh. Paun. Membrane Computing. An Introduction, Springer-Verlag, Berlin, (2002).

Authors' Information

Miguel Angel Díaz – e-mail: mdiaz@eui.upm.es

Luis Fernando de Mingo López – e-mail: lfmingo@eui.upm.es

Nuria Gómez Blas – e-mail: ngomez@dalum.eui.upm.es

Dept. Organización y Estructura de la Información, Escuela Universitaria de Informática, Universidad Politécnica de Madrid, Crta. De Valencia km. 7, 28031 Madrid, Spain.

HIERARCHICAL LOGICAL DESCRIPTION AND NEURAL RECOGNITION OF COMPLEX PATTERNS

Tatiana Kosovskaya, Adil Timofeev

Abstract: Authors suggested earlier hierarchical method for definition of class description at pattern recognition problems solution. In this paper development and use of such hierarchical descriptions for parallel representation of complex patterns on the base of multi-core computers or neural networks is proposed.

Keywords: complex patterns, logical class description, neural network.

Introduction

In papers [1–4] the authors suggested logic-axiomatic approach to solution of a series of pattern recognition problems. In papers [1, 2] in the framework of this approach hierarchical method of definition for class description, intended firstly for reducing of complexity of described problems, has been proposed.

In current work it is proposed to use such hierarchical descriptions for classes (DC) for their parallel representation on the base of neural networks. Neural representation of logic descriptions of classes and resolution rules provides mass parallelism and high speed at complex pattern recognition.

For building of logic DC it has been proposed to use the following methods and technologies:

– word (linguistic) class description from special study, scientific or reference literature [2–4];
– direct description of learning set in terms of defined predicates [4];
– logic-frequency method [3, 4];

– test tables [5];
– diophantine neural networks [6–8];
– multi-agent technologies of solution making by neural networks group [9].

1. Problem set for complex pattern logic recognition

Let on parts (fragments) *x* which are included in plants of arbitrary nature ω from the set Ω, the set of predicates, characterizing features and relations between elements of fragment *x* of the plant $\omega \in \Omega$, is defined.

Let be defined expansion of the set Ω on M (uncrossing or probably crossing) classes (patterns) of the kind

$$\Omega = U_{k=1}^{M} \Omega_k .$$

Description S(ω) of the plant ω is called the set of atomic formulas of the kind $p_i(x)$ or $\neg p_i(x)$, true for *x*, written for all possible parts (fragments) *x* of the plant ω.

Description of the class (OK) Ω_k is called such logic formula $A_k(\omega)$, that

1. $A_k(\omega)$ contains as atomic ones only the formulas of the kind $p_i(x)$, where $x \subseteq \omega$;
2. $A_k(\omega)$ does not contain quantors;
3. if the formula $A_k(\omega)$ is true, then $\omega \in \Omega_k$

DC may be written always in the kind of logic formula

$$\mathbf{V}_{j=1}^{J_k} \&_{i \in I_k^j} \&_{x \subseteq \omega_{i,j}} p_i^{\alpha_{ijx}}(x) , \tag{1}$$

where J_k – natural number, $I_k^j \subseteq \{1,...,n\}, \omega_{i,j} \subseteq \omega, \alpha_{ijx}$ – logic constants. Expression p^{α} is used as cancellation for writing of *p* or $\neg p$ in dependence of that whether α be the constant of И (1) or Л (0) accordingly.

By the help of built logic DC it is proposed tosolve the following problems of recognition for simple or complex patterns:

1. Identification problem. It is necessary to determine, whether the plant ω or its part *x* belongs to the class Ω or its part belongs to the class Ω_k.

This problem in the works [3, 4] is reduced to proof of formula derivation ability

$$\exists y(y \subseteq \omega \,\&\, A_k(y)) \tag{2}$$

from description of recognized plant S(ω).

2. Classification problem. It is necessary to find all such numbers of the classes *k*, that $\omega \in \Omega_k$.

This problem in the works [3, 4] is reduced to proof of formula derivation ability

$$\mathbf{V}_{j=1}^{M} A_k(\omega) \tag{3}$$

from description of recognized plant S(ω) with notification of all such numbers *k*, for which according disjunctive memberis true on ω.

3. Complex object analysis problem. It is necessary to find and classify all parts *x* of the plant ω, for which $x \in \Omega$.

This problem in the works [3, 4] is reduced to proof of formula derivation ability

$$\mathbf{V}_{j=1}^{M} \exists x(x \subseteq \omega \,\&\, A_k(y)) \tag{4}$$

from description of recognized complex plant S(ω) with notification (localization) and identification of all parts of complex plant ω, that may be classified, i.e to determine belonging to one or another class every selected fragment.

Number of steps for algorithm work, building derivation of mentioned formulas, especially in the case of solution of problem for complex plant analysis, may be sufficiently great. For reducing of steps number of solution for recognition problems in [2] more economical hierarchical description of classes, permitting unparallelizing at realization on multi-core computers and neural networks, has been proposed.

2. Hierarchical logic class description

There is a description of plants, which structure permits select their simpler parts, i.e. fragments, and give description of plant in terns of properties of these parts (fragments) and relations between them. Particularly it may be done by selection of "frequently" found sub-formulas of the formulas $A_k(\omega)$ of "small complexity". In this case the system of equivalences of the kind

$$p_i^l(x) \leftrightarrow P_i^l(x), \tag{5}$$

is formed where p_i^l – predicates of *l*-th level, P_i^l – sub-formulas of the formula $A_k(\omega)$.

More promptly, let say that the system of initial predicates $p_1,\ldots,p_n$ determines properties and relations of first level and write sometimes p_i^1 instead of p_i.

Let fix integer positive numbers *r* and N. They will characterize such fuzzy conception as "small complexity" of sub-formula (number of variables in sub-formula is less than *r*) and "frequently" (number of enterings, with precision to variables names, of given sub-formula in already available one is more than N).

Let select all frequently found sub-formulas of small complexity of the formulas $A_k(\omega)$ and designate them $P_i^2(x)$ (where *x* – list of variables, entering in sub-formula).

Let designate predicates, defined by these sub-formulas by mean of $p_i^2\,(i = 1,\ldots,n_2)$ and let name them parts of predicates of second level. These predicates are determined by the correlations

$$p_i^2(x) \leftrightarrow P_i^2(x) \tag{6}$$

Let designate formulas, obtained from $A_k(\omega)$ by means of substitution of all enterings of formulas of the kind $P_i^2(x)$ on atomic formulas $p_i^2(x)$ (at $x \subseteq \omega$) through $A_k^2(\omega)$. Such formulas may be described as logic DC in terms of first and second level.

Procedure of selecting of frequently found (typical) sub-formulas of small complexity may be repeated with the formulas $A_k^2(\omega)$.

Let be component predicates of 1-st, 2-nd, ..., *l*-1-th levels.

Let select all frequently found (typical) sub-formulas of small complexity of the formulas $A_k^{l-1}(\omega)$ and let designate them $P_i^2(x)$. Here *x* – list of variables, entering in sub-formula).

Let designate predicates, defined by these sub-formulas by means of $p_i^l\,(i = 1,\ldots,n_l)$ and let name them component (complex) predicates of *l*-th level.

These predicated are determined by the correlations

$$p_i^l(x) \leftrightarrow P_i^l(x) \tag{7}$$

Formulas, obtained from $A_k^{l-1}(\omega)$ by means of substitution of all enterings of the formulas of the kind $P_i^l(x)$ on atomic formulas $p_i^l(x)$, are designated through $A_k^l(\omega)$. Such formulas may be considered as logic descriptions of classes in terms of predicates of 1-st, 2-nd, ..., *l*-th levels.

To stop creation of component (complex) predicates of next level is possible in every moment but no later than current situation, when whether lengths of all formulas $A_k^l(\omega)$ are less than *r*, or among these formulas it is not found *N* sub-formulas of the same kind. This is rule of stopping of proposed initial-converged algorithm of hierarchical DC.

In result of building of component (complex) predicates and multi-level logic DC initial system of DC may be written by the means of equivalent multi-level logic system of DC of the kind

$$\begin{array}{ccc} & A_k^L(\omega), & \\ p_1^2(\omega) & \Leftrightarrow & P_1^2(\omega), \\ & \vdots & \\ p_{n_2}^2(\omega) & \Leftrightarrow & P_{n_2}^2(\omega), \\ & \vdots & \\ p_i^l(\omega) & \Leftrightarrow & P_i^l(\omega), \\ & \vdots & \\ p_{n_L}^L(\omega) & \Leftrightarrow & P_{n_L}^L(\omega). \end{array} \qquad (8)$$

It is possible to show that at sufficiently successful choice of parameters *r* and N time of work of algorithms solving different problems of recognition will be decreased [1].

3. Neural representation of hierarchical class description

At building of neural networks for unparalleling of representation of logic DC and recognition of complex patterns solution of values of predicates of one or another level may be done by corresponding neural level or core of multi-core computer. For example, at use of only predicates of first (initial) and second (synthesized) levels architecture of multi-level neural network will be such as it is presented in the fig.1.

Thus, in the first layer (sensor neurons) all possible values of initial predicates of recognized plant are calculated. In the second layer all possible values of second level predicates are calculated. In the third level formulas, defining logic DC, are checked.

In process of recognition of patterns complex plant or its fragment will be given to that class with number *k* for which the formula $A_k^2(\omega)$ is true.

It is necessary to mind that process of logic recognition for complex patterns on neural network may be organized by such means that to remember those fragments of initial plant ω, for which even if one of the formulas defining belonging to a class, will be true. By this mean problem of full logic analysis of complex plant may be solved automatically.

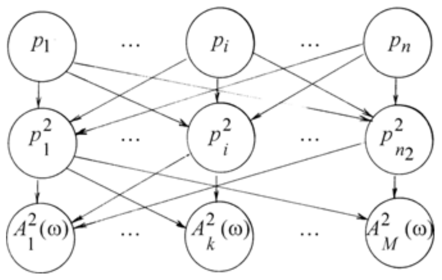


Fig.1. Architecture of multi-level neural network

Conclusion

Thus, for complex patterns (complex patterns and scenes, mixed signals etc.) methods for hierarchical description of classes (patterns) and evaluation of their effectiveness in problems of logis analysis and pattern recognition have been proposed. Informational technology for neural or multi-core representation of hierarchical descriptions of classes (patterns), providing mass parallelism at data processing and significant acceleration of processes for solution making in the process of pattern recognition has been described.

Results have been obtained at support of RFBR grant № 06-08-01612-a and Program “Innovations Support” of Presidium of RAS.

Bibliography

[1]. Kosovskaya T.M. Evaluations of Complexity for Solution of Pattern Recognition Problems at Hierarchical Class Description. – Proceedings of XI International conference on problems of theoretical cybernetics. Ulyanovsk, 1996. (In Russian).

[2]. Kosovskaya T.M. Multi-level Descriptions of Classes for Solution Making in – Proceedings of III International conference "Discrete models in theory of control systems". M., Dialogue — MSU, 1998. (In Russian).

[3]. Timofeev A.V. Robots and Artifiial Intelligence. – M., "Nauka", 1978. 191 pp. (In Russian).

[4]. Kosovskaya T.M., Timofeev A.M. About One New Approach to Forming of Logic Resolution Rules – LSU Informer, 1985, №8.P. 22–29. No 1, P. 160–176. (In Russian).

[5]. Yablonskiy S.V. Test – Mathematical Encyclopedy, v.5. M., 1985. P. 342–346. Publication. John Wiley \& Sons, Inc. 2000. 491 pp. (In Russian).

[6]. Timofeev A.M., Kalyaev A.V. Methods for Synthesis and Minimization of Complexity of Cognitive Neuromodules of Super-Macro-Neurocomputers with Programmed Architecture. – Papers of Russian Academy of Sciences, 1994, v. 273, № 2, pp. 180–183 (In Russian and English).

[7]. Timofeev A.V. M Methods for Synthesis of Diophantine Neural Networks of Minimal Complexity. – Papers of Russian Academy of Sciences, 1995, т. 301, № 3, pp. 1106–1109 (In Russian and English).

[8]. Timofeev A.V., Semyonov A.V. Genetic Algorithms of Database Control and Knowledge Base Synthesis and Their Applications. – International Journal of Information Theories & Applications, Sofia, 1996, v.4, N1, pp. 17–22.

[9]. Timofeev A.V., Sheozhev A.M., Shibzukhov Z.M. Multi-Agent Diophantine Neural Networks in Problems of Recognition and Diagnostics. – Neurocomputers: Development and Application, 2005, № 10–11, pp. 69–74. (In Russian).

Authors' Information

Timofeev Adil V. – Dr.Sc., Professor, Head of the Laboratory of Information Technologies in Control and Robotics of Saint-Petersburg Institute for Informatics and Automation of Russian Academy of Sciences, tav@iias.spb.su

Kosovskaya Tatiana M. – Doctor's Degree Student, Laboratory of Information Technologies in Control and Robotics of Saint-Petersburg Institute for Informatics and Automation of Russian Academy of Sciences, Russia;

Ph.D., Assistant Professor, Department of Mathematics and Mechanics, Saint-Petersburg State University, Russia, e-mail: kosov@nk1022.spb.edu

ADAPTIVE WAVELET-NEURO-FUZZY NETWORK IN THE FORECASTING AND EMULATION TASKS

Yevgeniy Bodyanskiy, Iryna Pliss, Olena Vynokurova

Abstract: The architecture of adaptive wavelet-neuro-fuzzy-network and its learning algorithm for the solving of nonstationary processes forecasting and emulation tasks are proposed. The learning algorithm is optimal on rate of convergence and allows tuning both the synaptic weights and dilations and translations parameters of wavelet activation functions. The simulation of developed wavelet-neuro-fuzzy network architecture and its learning algorithm justifies the effectiveness of proposed approach.

Keywords: wavelet, adaptive wavelet-neuro-fuzzy network, recurrent learning algorithm, forecasting, emulation.

ACM Classification Keywords: I.2.6 Learning – Connectionism and neural nets

Introduction

At present time the neuro-fuzzy systems have been an increasingly popular technique of soft computing [1-4] successfully applied for the processing of information containing complex nonlinear regularities and distortions of all kinds. These systems combine the linguistic interpretability and the approximation properties of the fuzzy inference systems [5, 6] with the learning and universal approximation capabilities of artificial neural networks [7, 8]. This means, that they can be used in forecasting and emulation of the stochastic and chaotic signals and sequences with complex nonlinear trends and nonstationary parameters, described by the difference nonlinear autoregression equations (NAR) in the form

$$x(k) = F(X(k)) + \xi(k),$$

where $X(k) = (x(k-1), x(k-2), \dots, x(k-n))^T$ is $(n \times 1)$ the prehistory vector, which determines present state $x(k)$, $x(k)$ is a signal value in k-th instant of discrete time $k = 0,1,2,\dots$, $F(\bullet)$ is an arbitrary nonlinear function, unknown in the general case, $\xi(k)$ is a stochastic disturbance with unknown characteristics but with bounded second moment.

Along with the neuro-fuzzy systems for processing the signals of all kinds, the wavelet transform has been an increasingly popular technique [9-11] which provides a compact local signal representation in both time and frequency domain. At the turn of the artificial neural network and wavelets theories the wavelet neural networks have evolved for the analysis of nonstationary processes with considerably nonlinear trends [12-18].

The natural step is to combine the transparency and the interpretability of fuzzy inference systems, powerful approximation and learning capabilities of artificial neural networks and compact description and the flexibility of wavelet transform in the context of hybrid systems of computational intelligence, which further we shall call as the adaptive wavelet-neuro-fuzzy networks (AWNFN).

The key point, defining effectiveness of such systems, is the choice of learning algorithm, which is usually based on the gradient procedures of the accepted criterion minimization. Combination of the gradient optimization with the error backpropagation essentially reduces the rate of learning hybrid systems [19] and leads to necessity of using rather large training samples. In the case when the data processing has to be carried out in real time, forecasted or emulated sequence is nonstationary and distorted, conventional gradient descent learning algorithms (let alone genetic algorithms) appeared to be ineffective.

The paper is devoted to the tasks of synthesis of adaptive wavelet-neuro-fuzzy network for the forecasting and emulation tasks. This network has higher rate of learning in comparison with systems using conventional backpropagation gradient algorithm.

Architecture of the adaptive wavelet-neuro-fuzzy network

Let us introduce into consideration the five-layers architecture, shown on fig. 1, someway similar to the well-known ANFIS [2] which is in turn the learning system of Takagi-Sugeno-Kang fuzzy inference [20,21].

The input layer of the architecture is formed of the time-delay elements z^{-1} ($z^{-1}x(k) = x(k-1)$) and under the input of current signal value $x(k)$ the prehistory vector $X(k) = (x(k-1), x(k-2), \dots, x(k-n))^T$ is formed as an output of this layer.

The first hidden layer unlike the neuro-fuzzy systems is formed not of conventional non-negative membership functions, but of hn wavelets (h wavelets for each input) $\varphi_{ji}(x(k-i)) = \varphi_{ji}(x(k-i), c_{ji}, \sigma_{ji}) = \varphi_{ji}(k)$ with $2hn$ tuning parameters of dilation (center) c_{ji} and translation (width) σ_{ji}.

Various kinds of analytical wavelets can be used as the activation functions in adaptive wavelet-neuro-fuzzy network, for example: Morlet wavelets, “Mexican hat” wavelets, Polywog wavelets, Rasp wavelets [12], the generator of analytic wavelets [22], the triangular wavelets [23].

Here it can be noticed, that the oscillation character of wavelet function doesn’t contradict the unipolarity of membership functions as negative values φ_{ji} can be interpreted in terms of the small membership or nonmembership levels [24, 25].

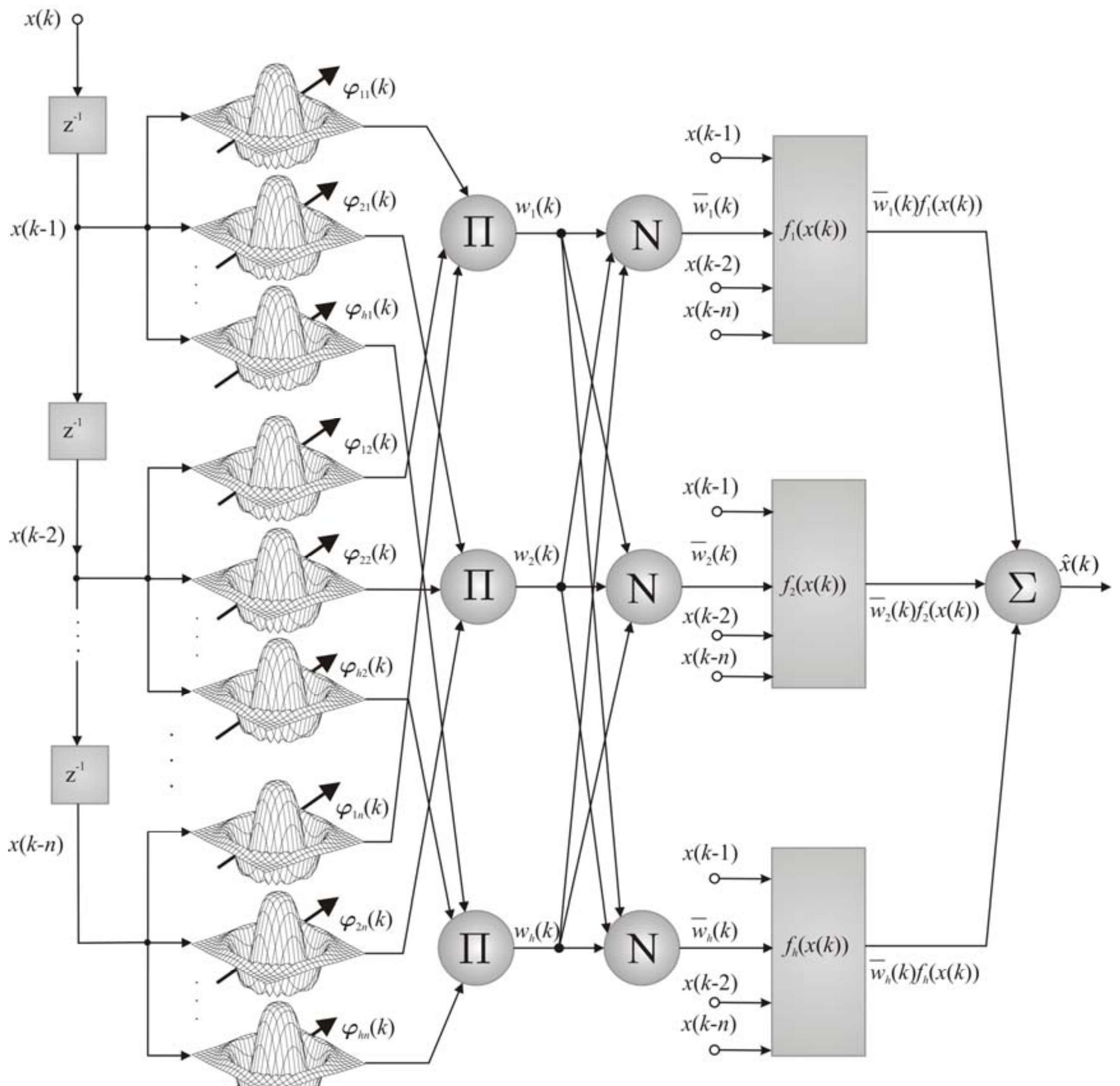


Fig. 1 – Adaptive wavelet-neuro-fuzzy network for the forecasting and emulation tasks

The second hidden layer performs the operation similar to computing of fuzzy T-norm

$$w_j(k) = \prod_{i=1}^{n} \varphi_{ji}(x(k-i)),\;\; j = 1, 2, \ldots, h,$$

after that the normalization is performed in the third hidden layer

$$\bar{w}_j(k) = \frac{w_j(k)}{\sum_{j=1}^{h} w_j(k)} = \frac{\prod_{i=1}^{n} \varphi_{ji}(x(k-i))}{\sum_{j=1}^{h} \prod_{i=1}^{n} \varphi_{ji}(x_i(k-i))},$$

providing fulfillment of the condition

$$\sum_{j=1}^{h} \bar{w}_j(k) = 1.$$

The fourth hidden layer performs an operation similar to computing of the consequent in the fuzzy inference systems. The most often used function $f_j(x(k))$ in fuzzy inference systems is linear form (in our case local autoregression model):

$$f_j(X(k)) = p_{j0} + \sum_{i=1}^{n} p_{ji} x(k-i) .$$

In this case in the fourth layer signal values are computed

$$\overline{w}_j(k)(p_{j0} + \sum_{i=1}^{n} p_{ji} x(k-i)) = \overline{w}_j(k) p_j^T \overline{X}(k),$$

where $\overline{X}(k) = (1, X^T(k))^T$, $p_j = (p_{j0}, p_{j1}, \ldots, p_{jn})^T$, and $h(n+1)$ parameters p_{ji} , $j = 1,2,\ldots,h$, $i = 0,1,2,\ldots,n$ are to be determined.

And at last output signal (forecast $\hat{x}(k)$) of network is computed in the fifth output layer

$$\hat{x}(k) = \sum_{j=1}^{h} \overline{w}_j(k) f_j(X(k)) = \sum_{j=1}^{h} \frac{w_j(k)}{\sum_{j=1}^{h} w_j(k)} f_j(X(k)) = \sum_{j=1}^{h} \frac{\prod_{i=1}^{n} \varphi_{ji}(x_i(k-i), c_{ji}, \sigma_{ji})}{\sum_{j=1}^{h} \prod_{i=1}^{n} \varphi_{ji}(x_i(k-i), c_{ji}, \sigma_{ji})} f_j(X(k)),$$

which, introducing the variables vectors $f(X(k)) = (\overline{w}_1(k), \overline{w}_1(k)x(k-1), \ldots, \overline{w}_1(k)x(k-n), \overline{w}_2(k),$ $\overline{w}_2(k)x(k-1), \ldots, \overline{w}_2(k)x(k-n), \ldots, \overline{w}_h(k), \overline{w}_h(k)x(k-1), \ldots, \overline{w}_h(k)x(k-n))^T$, $p = (p_{10}, p_{11}, \ldots,$ $p_{1n}, p_{20}, p_{21}, \ldots, p_{2n}, p_{h0}, p_{h1}, \ldots, p_{hn})^T$ of dimensionality $h(n+1)$, can be rewritten in the compact form

$$\hat{x}(k) = p^T f(X(k)) .$$

The tunable parameters of this network are located only in the first and fourth hidden layers. These are $2hn$ wavelets parameters c_{ji} and σ_{ji} , and $h(n+1)$ parameters of the linear local autoregression models p_{ji} . Namely they must be determined during the learning process.

The learning of adaptive wavelet-neuro-fuzzy network

As far as tunable vector of parameters p is contained in the network description linearly, for its refinement any of the algorithms used in adaptive identification [26] will operate, primarily the exponentially weighted recurrent least squares method (this method is the second order optimization procedure and has both filtering and following properties) in the form

$$\begin{cases} p(k+1) = p(k) + \dfrac{P(k)(x(k) - p^T(k) f(X(k)))}{\alpha + f^T(X(k)) P(k) f(X(k))} f(X(k)), \\ P(k+1) = \dfrac{1}{\alpha}\left(P(k) - \dfrac{P(k) f(X(k+1)) f^T(X(k+1)) P(k)}{\alpha + f^T(X(k+1)) P(k) f(X(k+1))} \right) \end{cases} \tag{1}$$

where $x(k) - p^T(k) f(x(k)) = x(k) - \hat{x}(k) = e(k)$ is the forecasting (emulation) error, $0 < \alpha \le 1$ is the out-dated information forgetting factor; optimal on operation rate one-step gradient Kaczmarz algorithm [27, 28], having the following properties

$$p(k+1) = p(k) + \frac{x(k) - p^T(k) f(X(k))}{f^T(X(k)) f(X(k))} f(X(k)) , \tag{2}$$

or Goodwin-Ramadge-Caines algorithm [29]

$$\begin{cases} p(k+1) = p(k) + r^{-1}(k)(x(k) - p^T f(X(k)))f(X(k)), \\ r(k+1) = r(k) + \|f(X(k+1))\|^2, \end{cases} \tag{3}$$

which is the stochastic approximation procedure.

Here it should be mentioned, that exponentially weighted recurrent least squares method (1), having filtering and following properties, can be unstable under small values of parameter α ; convergence of the algorithm (2) under the intensive disturbance ξ is disrupted, and stochastic approximation procedures, including (3), operate only in the stationary conditions.

For tuning of the first hidden layer parameters in AWNFN backpropagation learning algorithm based on the chain rule of differentiation and gradient descend optimization of local criterion

$$E(k) = \frac{1}{2}e^2(k) = \frac{1}{2}(x(k) - \hat{x}(k))^2$$

is used.

In the general case learning procedure in this layer has the form

$$\begin{cases} c_{ji}(k+1) = c_{ji}(k) - \eta_c(k)\dfrac{\partial E(k)}{\partial c_{ji}(k)}, \\ \sigma_{ji}(k+1) = \sigma_{ji}(k) - \eta_\sigma(k)\dfrac{\partial E(k)}{\partial \sigma_{ji}(k)}, \end{cases}$$

and its properties are completely determined by the learning rate parameter $\eta_c(k)$, $\eta_\sigma(k)$, selected according to the empirical reasons. It should be noticed that if the parameters of the fourth layer can be tuning most rapidly, the operation rate is lost in the first layer.

Increasing of the convergence rate can be achieved with more complex than gradient procedures, such as Hartley [30] or Marquardt [31] algorithms which can be written in general form [32]

$$\Phi(k+1) = \Phi(k) + \lambda(J(k)J^T(k) + \eta I)^{-1}J(k)e(k), \tag{4}$$

where $\Phi(k) = (c_{11}(k), \sigma_{11}^{-1}(k), c_{21}(k), \sigma_{21}^{-1}(k), \ldots, c_{ji}(k), \ \sigma_{ji}^{-1}(k), \ldots, c_{hn}(k), \sigma_{hn}^{-1}(k))^T$ is the $(2hn \times 1)$ tunable parameter vector (at that for the computation complexity reduction it includes not the width parameter σ_{ji}, but its inverse value σ_{ji}^{-1}), $J(k)$ is the $(2hn \times 1)$ gradient vector of output signal $\hat{x}(k)$ on the tunable parameters, I is the $(2hn \times 2hn)$ identity matrix, η is a scalar regularizing parameter, λ is the positive scalar gain.

To compute elements of gradient vector

$$J(k) = \left(\frac{\partial \hat{x}(k)}{\partial c_{11}}, \frac{\partial \hat{x}(k)}{\partial \sigma_{11}^{-1}}, \frac{\partial \hat{x}(k)}{\partial c_{21}}, \frac{\partial \hat{x}(k)}{\partial \sigma_{21}^{-1}}, \ldots, \frac{\partial \hat{x}(k)}{\partial c_{ji}}, \frac{\partial \hat{x}(k)}{\partial \sigma_{ji}^{-1}}, \ldots, \frac{\partial \hat{x}(k)}{\partial c_{hn}}, \frac{\partial \hat{x}(k)}{\partial \sigma_{hn}^{-1}} \right)^T$$

the chain rule can be used, at that

$$\begin{cases} \dfrac{\partial \hat{x}(k)}{\partial c_{ji}} = \dfrac{\partial \hat{x}(k)}{\partial \overline{w}_j} \cdot \dfrac{\partial \overline{w}_j}{\partial w_j} \cdot \dfrac{\partial w_j}{\partial \varphi_{ji}} \cdot \dfrac{\partial \varphi_{ji}}{\partial c_{ji}} = f_j(X(k))\overline{w}_j(k)(1 - \overline{w}_j(k)) \dfrac{1}{\varphi_{ji}(x_i(k), c_{ji}, \sigma_{ji}^{-1})} \cdot \dfrac{\partial \varphi_{ji}}{\partial c_{ji}}, \\ \dfrac{\partial \hat{x}(k)}{\partial \sigma_{ji}^{-1}} = \dfrac{\partial \hat{x}(k)}{\partial \overline{w}_j} \cdot \dfrac{\partial \overline{w}_j}{\partial w_j} \cdot \dfrac{\partial w_j}{\partial \varphi_{ji}} \cdot \dfrac{\partial \varphi_{ji}}{\partial \sigma_{ji}^{-1}} = f_j(X(k))\overline{w}_j(k)(1 - \overline{w}_j(k)) \dfrac{1}{\varphi_{ji}(x_i(k), c_{ji}, \sigma_{ji}^{-1})} \cdot \dfrac{\partial \varphi_{ji}}{\partial \sigma_{ji}^{-1}}. \end{cases}$$

where $\partial \varphi_{ji} / \partial c_{ji}$, $\partial \varphi_{ji} / \partial \sigma_{ji}^{-1}$ is partial derivatives of concrete wavelet activation function.

To reduce the computational complexity of the learning algorithm we can use the matrix inversion lemma in the form

$$(JJ^T + \eta I)^{-1} = \eta^{-1} I - \frac{\eta^{-1} I J J^T \eta^{-1} I}{1 + J^T \eta^{-1} I J},$$

using which it is easy to obtain the relation

$$\lambda (JJ^T + \eta I)^{-1} J = \lambda \frac{J}{\eta + \|J\|^2}.$$

Substituting this relation to the algorithm (4), we obtain first hidden layer parameters learning algorithm in the form

$$\Phi(k+1) = \Phi(k) + \lambda \frac{J(k)e(k)}{\eta + \|J(k)\|^2}. \tag{5}$$

It is easy to see, that algorithm (5) is the nonlinear additive-multiplicative modification of Kaczmarz algorithm, and under $\lambda = 1,\ \eta = 0$ coincides with it structurally.

To provide the filtering properties to the learning algorithm (5) let us introduce additional tuning procedure of the regularizing parameter η in the form [33, 34, 35]

$$\begin{cases} \Phi(k+1) = \Phi(k) + \lambda \dfrac{J(k)e(k)}{\eta(k)}, \\ \eta(k+1) = \alpha \eta(k) + \|J(k+1)\|^2. \end{cases} \tag{6}$$

If $\alpha = 0$, then this procedure coincides with (5) and has the highest rate of convergence, and if $\alpha = 1$, then this procedure obtains properties of stochastic approximation, and serves as generalization of procedure (3) in the nonlinear case.

Here it should be noticed, that the algorithm (6) is stable at any value of forgetting factor α, what favorably differs it from the exponentially weighted recurrent least squares method (1). As a result this procedure can be used too in the form

$$\begin{cases} p(k+1) = p(k) + \lambda_p \eta_p^{-1}(k)(x(k) - p^T(k) f(X(k))) f(X(k)), \\ \eta_p(k+1) = \alpha \eta_p(k) + \|f(x(k))\|^2 \end{cases} \tag{7}$$

for the fourth layer parameters tuning. One can notice close relation of the algorithms (1) and (7), as

$$\eta^{-1}(k) = TrP(k).$$

However algorithm (7) is much simpler in the computing implementation and easily reconstructs its properties from the most following to the most filtering ones.

Simulation results

To demonstrate the effectiveness of the proposed adaptive wavelet-neuro-fuzzy-network and its learning algorithm (6), (7), AWNFN was trained to emulate the nonlinear dynamical system which proposed in [36]. Emulation of the Narendra's dynamical system is a standard test, widely used to evaluate and compare the performance of neural and neuro-fuzzy systems for nonlinear system modeling and time series forecasting. The nonlinear dynamical system is generated by equation in form [36]

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + f(u(k)), \tag{8}$$

where $f(u(k)) = 0.6\sin(u(k)) + 0.3\sin(3u(k)) + 0.1\sin(5u(k))$ and $u(k) = \sin(2k/250)$, k is discret time.

The values $x(t-4), x(t-3), x(t-2), x(t-1)$ were used to emulate $x(t+1)$. In the online mode of learning, AWNFN was trained with procedure (6), (7) using signal $u(k)=\sin 2k/250$ for $k=1...1000$. The parameters of the learning algorithm were $\alpha=0.9$, $\lambda_p=2$, $\lambda=1$. Initial values were $\eta(0)=1$ and $\eta_p(0)=10000$. After 1000 iterations the training was stopped, and the next 800 points for the signals $u(k)=\sin 2k/250$, $k=1...300$ and $u(k)=0.5\sin 2k/250+0.5\sin 2k/25$, $k=501...1000$ were used as the testing data set to emulate dynamical system. As the activation function "Mexican hat" wavelet is used. Initial values of synaptic weights were generated in a random way from -0.1 to $+0.1$.

The root mean-square error (RMSE) was used as criterion for the quality of emulation

$$RMSE=\frac{1}{N}\sum_{k=1}^{N}(x(k)-\hat{x}(k))^2 .$$

Fig. 5 shows the results of nonlinear dynamical system emulation. The two curves, representing the actual (dot line) and emulation (solid line) values, are almost indistinguishable.

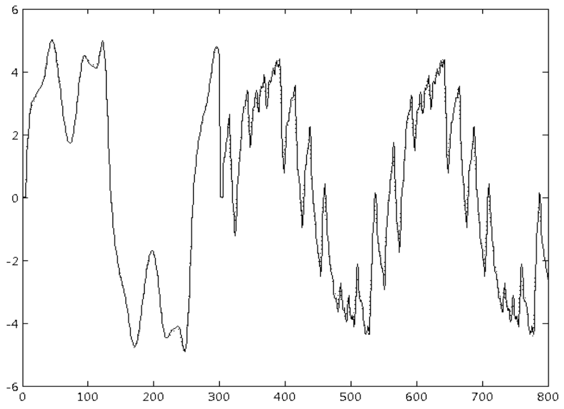


Fig. 5 – Emulation of the nonlinear dynamical system using adaptive wavelet-neuro-fuzzy network

Table 1 shows the results of the emulation process on the basis of the adaptive wavelet-neuro-fuzzy-network compared the results of emulation process on the basis of standard ANFIS with the backpropagation learning algorithm.

Table 1: The results of nonlinear dynamical system emulation

Neural network/ Learning algorithm	RMSE
Adaptive wavelet-neuro-fuzzy-network / Proposed learning algorithm (6), (7)	0.025
Backpropagation ANFIS	0.110

Thus as it can be seen from experimental results the proposed adaptive wavelet-neuro-fuzzy-network with the learning algorithm (6), (7) having the same number of adjustable parameters ensures the best quality of emulationt and high learning rate in comparison with conventional ANFIS architecture.

Conclusions

Computationally simple learning algorithms for the adaptive wavelet-neuro-fuzzy network in the forecasting and emulation of the nonlinear nonstationary signals tasks are proposed. The simulation of developed approach justifies the effectiveness of AWNFN using for solving wide category of emulation, forecasting and diagnostics problems.

Bibliography

[1]. Jang J.-S. R. Neuro-Fuzzy Modeling: Architectures, Analyses and Applications. Berkeley, CA: University of California. 1992, 155 p.

[2]. Jang J.-S. R. ANFIS: Adaptive network-based fuzzy inference systems. IEEE Trans. on Systems, Man, and Cybernetics. 23, 1993, P. 665-685.

[3]. Jang J.-S. R., Sun C.-T. Neuro-fuzzy modeling and control. Proc. IEEE. 83 (3), 1995, P. 378-406.

[4]. Jang J.-S. R., Sun C.-T., Muzutani E. Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence. Upper Saddle River, N.J.: Prentice Hall, Inc., 1997, 614 p.

[5]. Kosko B. Fuzzy systems as universal approximators. Proc. 1-st IEEE Int. Conf. on Fuzzy Systems. San Diego, CA. 1992, P. 1153-1162.

[6]. Kreinovich V., Mouzouris Y. C., Nguen H. T. Fuzzy rule based modeling as a universal approximation tool. In “Fuzzy Systems: Modeling and Control”. Eds.: H. T. Nguen, M. Sugeno. Boston: Kluwer Academic Publishers. 1998, P. 135-195.

[7]. Hornik K., Stinchcombe M., White H. Multilayer feedforward networks are universal approximators. Neural Networks. 2, 1982, P. 359-366.

[8]. Scarcelli F., Tsoi A. S. Universal approximation using feedforward neural networks: a survey of some existing methods and some new results. Neural Networks. 11, 1998, P. 15-37.

[9]. Chui C. K. An Introduction to Wavelets. New York: Academic. 1992, 264 p.

[10]. Daubechies I. Ten Lectures on Wavelets. Philadelphia, PA: SIAM. 1992, 228 p.

[11]. Meyer Y. Wavelets: Algorithms and Applications. Philadelphia, PA: SIAM. 1993, 133 p.

[12]. Lekutai G., VanLandingham H. F. Self-tuning control of nonlinear systems using neural network adaptive frame wavelets. Proc. IEEE Int. Conf. on Systems, Man and Cybernetics. Piscataway, N.J. 2, 1997, P. 1017-1022.

[13]. Billings S. A., Wei H.-L. A new class of wavelet networks for nonlinear system identification. IEEE Trans. on Neural Networks. 16 (4), 2005, P. 862-874.

[14]. Zhang Q. H., Benveniste A. Wavelet networks. IEEE Trans. on Neural Networks. 3 (6), 1992, P. 889–898.

[15]. Oussar Y., Dreyfus G. Initialization by selection for wavelet network training. Neurocomputing. 34, 2000, P. 131–143.

[16]. Zhang J., Walter G. G., Miao Y., Lee W. N. W. Wavelet neural networks for function learning. IEEE Trans. on Signal Process. 43(6), 1995, P. 1485–1497.

[17]. Zhang Q. H. Using wavelet network in nonparametric estimation. IEEE Trans. on Neural Networks. 8 (2), 1997, P. 227–236.

[18]. Bodyanskiy Ye., Lamonova N., Pliss I., Vynokurova O. An adaptive learning algorithm for a wavelet neural network. Blackwell Synergy: Expert Systems. 22 (5), 2005, P. 235-240.

[19]. Avci E., Turkoglu I., Poyraz M. Intelligent target recognition based on wavelet adaptive network based fuzzy inference system. Lecture Notes on Computer Science. Berlin-Heidelberg: Springer-Verlag, 3522, 2005, P. 594-603.

[20]. Takagi T., Sugeno M. Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. on Systems, Man, and Cybernetics. 15, 1985, P. 115-132.

[21]. Sugeno M., Kang G. T. Structure identification of fuzzy model. Fuzzy Sets and Systems, 28, 1998, P. 15-33.

[22]. Vynokurova O., Lamonova N., Pliss I. Analytic wavelets generator. Problemy Bioniki, 60, 2004, P. 104-109. (in Russian)

[23]. Bodyanskiy Ye., Vynokurova O. Variable form of triangular wavelet in the wavelet neural networks. Proc. of 13th International Conference on Automatic Control “Automatics-2006”, Vinnitsa, 2006, P. 393. (in Russian)

[24]. Mitaim S., Kosko B. What is the best shape for a fuzzy set in function approximation? Proc. 5th IEEE Int. Conf on Fuzzy Systems “Fuzz-96”, 2, 1996, P. 1237-1213.

[25]. Mitaim S., Kosko B. Adaptive joint fuzzy sets for function approximation. Proc. Int. Conf. on Neural Networks “ICNN-97”, 1997, P. 537-542.

[26]. Ljung L. System Identification: Theory for the User. PTR Prentice Hall, Upper Saddle River, N.J., 1999

[27]. Kaczmarz S. Angenaeherte Ausloesung von Systemen linearer Gleichungen. Bull. Int. Acad. Polon. Sci., Let. A, 1973, S. 355-357.

[28]. Kaczmarz S. Approximate solution of systems of linear equations. Int. J. Control, 53, 1993, P. 1269-1271.

[29]. Goodwin Y. C., Ramadge P. J., Caines P. E. A globally convergent adaptive predictor. Automatica, 17, 1981, P. 135-140.

[30]. Hartley H. The modified Gauss-Newton method for the fitting of nonlinear regression functions of least squares. Technometrics, 3, 1961, P. 269-280.

[31]. Marquardt D. An algorithm for least squares estimation of nonlinear parameters. SIAM J. Appl. Math., 11, 1963, P. 431-441.

[32]. Bodyanskiy Ye. Adaptive identification algorithm of nonlinear control object. Automatic Control System and Devices of Automatic, Kharkiv: Vyshcha Shk., 81, 1987, P. 43-46. (in Russian)

[33]. Bodyanskiy Ye., Pliss I., Solovyova T. Multistage optimal predictors of multidimensional nonstationary stochastic processes. Docl. AN USSR. 12, 1986, P. 41-49.

[34]. Bodyanskiy Ye., Kolodyazhniy V., Stephan A. An adaptive learning algorithm for a neuro-fuzzy network. Ed. by B. Reusch "Computational Intelligence. Theory and Applications." Berlin-Heidelberg-New York: Springer-Verlag, 2001, P. 68-75.

[35]. Bodyanskiy Ye., Kolodyazhniy V., Otto P. A new learning algorithm for a forecasting neuro-fuzzy network. Integrated Computer-Aided Engineering. Amsterdam: IOS Press, 10 (4), 2003, P. 399-409.

[36]. Narendra K.S., Parthasarathy K. Identification and control of dynamic systems using neural networks. IEEE Trans. on Neural Networks, 1990, 1(1), P. 4-26.

Authors' Information

Bodyanskiy Yevgeniy - Doctor of Technical Sciences, Professor of Artificial Intelligence Department and Scientific Head of the Control Systems Research Laboratory, Kharkiv National University of Radio Electronic, Lenina av. 14, Kharkiv, Ukraine 61166, Tel +380577021890, bodya@kture.kharkov.ua

Pliss Iryna - Candidate of Technical Sciences (equivalent Ph.D.), Senior Researcher, Leading Researcher of the Control Systems Research Laboratory, Kharkiv National University of Radio Electronic, Lenina av. 14, Kharkiv, Ukraine, 61166, Tel +380577021890, pliss@kture.kharkov.ua

Vynokurova Olena - Candidate of Technical Sciences (equivalent Ph.D.), Senior Researcher of the Control Systems Research Laboratory, Kharkiv National University of Radio Electronic, Lenina av. 14, Kharkiv, Ukraine, 61166, Tel +380577021890, vinokurova@kture.kharkov.ua

MULTIALGEBRAIC SYSTEMS IN INFORMATION GRANULATION

Alexander Kagramanyan, Vladimir Mashtalir, Vladislav Shlyakhov

Abstract: In different fields a conception of granules is applied both as a group of elements defined by internal properties and as something inseparable whole reflecting external properties. Granular computing may be interpreted in terms of abstraction, generalization, clustering, levels of abstraction, levels of detail, and so on. We have proposed to use multialgebraic systems as a mathematical tool for synthesis and analysis of granules and granule structures. The theorem of necessary and sufficient conditions for multialgebraic systems existence has been proved.

Keywords: granular computing, multirelations, multioperations.

ACM Classification Keywords: I.2.4 Knowledge representation formalisms and methods: relation systems.

Introduction

Granular computing explores knowledge from different standpoints to reveal various types of structures and information embedded in the data [Zadeh, 1997, Bargiela, Pedrycz, 2002]. A paradigm of granular computing consists in grouping elements together (in a granule) by indistinguishability, similarity, proximity or functionality in arbitrary feature or signal spaces. Taking into account a semantic interpretation of why two objects are put into the same granule and how two objects are related with each other it provides one of a general methodology for intelligent data analysis on different levels of roughening or detailing [Pal et al., 2005, Yao, Yao, 2002].

Internal, external and contextual properties of granules, collective structure of a family of granules and hierarchical structure of granules represent a possible foundation for qualitative/quantitative characterization of levels of abstraction, detail, control, explanation, difficulty, organization and so on. Focusing on high conceptual level issues by ignoring much irrelevant details, granular computing are actively used in computational intelligence [Doherty et al., 2003], information granulation based on rough sets [Yao, 2001, Pal et al., 2005], data mining [Yao, 2006], interval analysis, cluster analysis, machine learning and many others [Yager, 2002, Lin, 2003, etc.]. The integration of multiple views on different types of granulation and granular structure may provide more useful data analysis tools [Lin, 2003, Yao, 2005]. One of a number of possible approaches is to use multialgebraic systems [Mashtalir, Shlyakhov, 2003] as mathematical apparatus for synthesis and analysis of granules and granule structures.

Thus, we need tools providing a granular linkage, i.e. formal operations and relations determined on granules. Furthermore, this linkage has to be induced either by information embedded in the data or by given close coupling with field of application. These questions are at present far from being solved. But the important point to note here is the search of necessary and sufficient conditions for existence of multialgebraic systems as enough general tool of granular computing.

Motivation of granular computing modeling by multialgebraic systems

If we choose any natural number $p \in \mathbb{N}$ then we can consider a ternary relation

$$\mathrm{E}(n_1,n_2,n_3)=\begin{cases}1, & (n_1+n_2)\ mod\ p = n_3\ mod\ p;\\ 0, & (n_1+n_2)\ mod\ p \neq n_3\ mod\ p\end{cases} \tag{1}$$

where $a\ mod\ b$ defines a as modulo b residue, i.e. $a\ mod\ b \triangleq a-\lfloor a/b\rfloor\times b$ and $\lfloor\circ\rfloor$ is a floor function. It is easily seen, if we hold fixed $k\in\{1,2,3\}$ then we get an equivalence relation $\mathrm{P}_k^{\mathrm{E}}$, e.g. for $k=1$

$$\mathrm{P}_1^{\mathrm{E}}(n_1,n_1')=1 \Leftrightarrow \mathrm{E}(n_1,n_2,n_3)\equiv \mathrm{E}(n_1',n_2,n_3).$$

This equivalence partitions set of natural numbers into residue classes modulo p. Indeed, if remainders in division n_1 and n_1' by p are the same then for arbitrary n_2 and n_3 continued equality is

$$\begin{aligned}(n_1+n_2)\ mod\ p &= (ps_1+r_1+ps_2+r_2)\ mod\ p=(r_1+r_2)\ mod\ p=\\ &=(ps_1'+r_1+ps_2+r_2)\ mod\ p=(n_1'+n_2)\ mod\ p.\end{aligned} \tag{2}$$

Here it is implied that $n_1=ps_1+r_1, n_1'=ps_1'+r_1, n_2=ps_2+r_2$, and $s_1,s_1',s_2\in\mathbb{N}$. From (2) it follows that $\mathrm{E}(n_1,n_2,n_3)\equiv\mathrm{E}(n_1',n_2,n_3)$. The converse proposition is the valid one also. If $\mathrm{E}(n_1,n_2,n_3)\equiv\mathrm{E}(n_1',n_2,n_3)$ then remainders in division n_1 and n_1' by p have to be equal, if not when $n_1=ps_1+r_1$, $n_1'=ps_1+r_1'$ and $r_1\neq r_1'$ under $n_2=0$, $n_3=r_1$ we obtain, on the one hand,

$$(n_1+n_2)\ mod\ p=(ps_1+r_1)\ mod\ p=r_1\ mod\ p=n_3\ mod\ p,$$

i.e. $\mathrm{E}(n_1,0,r_1)=1$. On the other hand,

$$(n_1'+n_2)\ mod\ p=(ps_1'+r_1)\ mod\ p=r_1'\ mod\ p\neq n_3\ mod\ p.$$

Since $r_1',r_1\leq p$ и $r_1'\neq r_1$ then $\mathrm{E}(n_1',0,r_1)=0$, which contradicts the original assumption. Notice, from (1) we get $\mathrm{P}_1^{\mathrm{E}}=\mathrm{P}_2^{\mathrm{E}}=\mathrm{P}_3^{\mathrm{E}}$.

Let us sum up. The carrier of original relation is the set of natural numbers $\mathbb{N}$ but the induced equivalence demonstrates the significant carrier change: we have got a finite set $\Pi=\{0,1,\ldots,p-1\}$.

New relation, which will be named a multirelation and denoted by E^{M} in the sequel, is generated on new carrier. As before it is ternary relation but the domain is Π^3 instead of $\mathbb{N}^3$ and multirelation E^{M} acquires the new property that can be expressed as an operation

$$r_1 \oplus r_2 = r_3 \Leftrightarrow \mathrm{E}_m(r_1, r_2, r_3) = 1 \Leftrightarrow \mathrm{E}(n_1, n_2, n_3) = 1$$

where sign "$\oplus$" denotes p congruence addition and $n_i = ps_i + r_i$, $i = 1,2,3$, $r_i \in \{0,1,\ldots,p-1\}$. Operations on equivalence classes here and subsequently will be denoted by F^{M}.

If they follow terminology of algebraic system a triplet $\langle \mathrm{A}, \mathrm{R}, \mathrm{Q} \rangle$ (here A is arbitrary set (carrier), R is a relation suite, Q is a set of operations) is called a model on conditions that $\mathrm{Q} = \varnothing$ and it is said to be an algebra if $\mathrm{R} = \varnothing$. Consequently, from the model $\langle \mathbb{N}, \mathrm{E}, \varnothing \rangle$ we pass on to the algebra $\langle \Pi, \varnothing, \oplus \rangle$ whose carrier is well-known algebraic structure viz a cyclic Abelian group of p-th order.

It is necessary to understand that original carrier can represent a set and carrier of induced relation on equivalence classes can be Cartesian product of different sets. Let us consider one more example

$$\mathrm{E}(n_1, n_2, n_3) = \begin{cases} 1, & (n_1 + n_2) \; mod \; p_1 = n_3 \; mod \; p_2; \\ 0, & (n_1 + n_2) \; mod \; p_1 \neq n_3 \; mod \; p_2 \end{cases} \quad (3)$$

where $p_1 \neq p_2$. It should be noted that $\mathrm{P}_1^{\mathrm{E}} = \mathrm{P}_2^{\mathrm{E}} \neq \mathrm{P}_3^{\mathrm{E}}$, i.e. E_m is Cartesian product $\mathrm{A} \times \mathrm{B}$ where $\mathrm{A} = \{0,1,\ldots,p_1 - 1\}$, $\mathrm{B} = \{0,1,\ldots,p_2 - 1\}$. As may be seen from (3) the multirelation E^{M} as a ternary relation is defined on $\mathrm{A}^2 \times \mathrm{B}$ and represents an operation from A^2 into B. There is no difficulty in understanding that under certain p_1 and p_2 not only an equivalence inequality appears but a level of partition detail and equivalence nesting are changed. For instance, if $p_1 = 4$, $p_2 = 2$ then $\mathrm{P}_1^{\mathrm{E}} = \mathrm{P}_2^{\mathrm{E}} \subseteq \mathrm{P}_3^{\mathrm{E}}$ as $\mathbb{N}$ is partitioned into 4 classes corresponding to residues of division $\mathrm{A} = \{0,1,2,3\}$ at the expense of $\mathrm{P}_1^{\mathrm{E}} = \mathrm{P}_2^{\mathrm{E}}$. Equivalence $\mathrm{P}_3^{\mathrm{E}}$ partitions original set $\mathbb{N}$ into 2 classes from even and odd numbers, i.e. $\mathrm{B} = \{0,1\}$. In this connection classes $\{0,2\}$ belong to the set of even numbers, classes $\{1,3\}$ form part of odd numbers set respectively.

In analyzed examples the original relation E induces the multirelation (more precisely the multioperation), i.e. an operation with ranges of definition as equivalence classes. It may seem that a similar situation is observed all the time, however this is by no means always the case. Consider the binary relation E (tab. 1) which is defined on the Cartesian product $\{1,2,3,4,5\} \times \{a_1, a_2, b_1, c_1, c_2, c_3, c_4, c_5, c_6\}$.

Table 1

		A		B	C					
		a_1	a_2	b_1	c_1	c_2	c_3	c_4	c_5	c_6
Π_{I}	1	0	0	1	1	1	1	1	1	1
	2	0	0	1	1	1	1	1	1	1
Π_{II}	3	1	1	1	0	0	0	0	0	0
	4	1	1	1	0	0	0	0	0	0
	5	1	1	1	0	0	0	0	0	0

It should be clear that the induced equivalences $\mathrm{P}_1^{\mathrm{E}}$ and $\mathrm{P}_2^{\mathrm{E}}$ dissect the first set $\mathrm{A}_1 = \{1,2,3,4,5\}$ into 2 classes: $\Pi_{\mathrm{I}} = \{1,2\}$, $\Pi_{\mathrm{II}} = \{3,4,5\}$ and the second one $\mathrm{A}_2 = \{a_1, a_2, b_1, c_1, c_2, c_3, c_4, c_5, c_6\}$ into 3 classes: $\mathrm{A} = \{a_1, a_2\}$, $\mathrm{B} = \{b_1\}$, $\mathrm{C} = \{c_1, c_2, c_3, c_4, c_5, c_6\}$. Thus, the multirelation E_m is defined on the Cartesian product of induced equivalence classes, i.e. $\{\Pi_{\mathrm{I}}, \Pi_{\mathrm{II}}\} \times \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$ (tab. 2).

Table 2

	A	B	C
Π_{I}	0	1	1
Π_{II}	1	1	0

This multirelation can be represented as two explicit mappings associating $\{\Pi_{\mathrm{I}}, \Pi_{\mathrm{II}}\}$ with $\{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$. Denote induced mappings as F^{EM} and $(\mathrm{F}^{\mathrm{EM}})^{-1}$ in both directions then $\mathrm{F}^{\mathrm{EM}}(\Pi_{\mathrm{I}}) = \{\mathrm{B}, \mathrm{C}\}$, $\mathrm{F}^{\mathrm{EM}}(\Pi_{\mathrm{II}}) = \{\mathrm{A}, \mathrm{B}\}$ and

$(F^{EM})^{-1}(A)=\{\Pi_{II}\}$, $(F^{EM})^{-1}(B)=\{\Pi_{I},\Pi_{II}\}$, $(F^{EM})^{-1}(C)=\{\Pi_{I}\}$. Single-valuedness is lacking in both cases therefore we are not able to indicate multioperation.

Thus, algebraic model can lead either to multimodels or to multialgebra and there arises an important question: when do two relations with different arities generate one carrier?

Necessary and sufficient conditions for multirelations carriers equality

Let $A_1, A_2, \ldots, A_n$ be any given sets. Consider n-arity relation $E(x_1,\ldots,x_n)$ on Cartesian product of arbitrary carriers $A_1\times\ldots\times A_n$. A trivial verification shows that

$$P_k^E(x_k, x'_k)=1 \Leftrightarrow E(x_1,\ldots,x_k,\ldots,x_n)\equiv E(x_1,\ldots,x'_k,\ldots,x_n) \qquad (4)$$

constitutes an equivalence relation and partitions may be regarded on each A_k. The understanding of the appearance mechanism of P_k^E awaits further investigation.

If $A_k=A_l$ then relations P_k^E and P_l^E can be compared. For instance,

$$[P_k^E(x_k, x'_k)=1 \Rightarrow P_l^E(x_k, x'_k)=1] \Leftrightarrow P_k^E \subseteq P_l^E,$$

i.e. P_k^E fulfills more detail partition than P_l^E and information can be analyzed with greater exactness. Using terminology of relation $E(x_1,\ldots,x_n)$ we get in that case

$$\begin{aligned}&E(x_1,\ldots,x_{k-1},x_k,x_{k+1},\ldots,x_n)\equiv E(x_1,\ldots,x_{k-1},x'_k,x_{k+1},\ldots,x_n)\Rightarrow\\ \Rightarrow\ &E(x_1,\ldots,x_{l-1},x_k,x_{l+1},\ldots,x_n)\equiv E(x_1,\ldots,x_{l-1},x'_k,x_{l+1},\ldots,x_n).\end{aligned}$$

Generally, on $\{1,2,\ldots,n\}$ the relation $E(x_1,\ldots,x_n)$ produces the four-valued indicator function

$$f(k,l)=\begin{cases}-1,\ A_k=A_l,\ P_k^E\subseteq P_l^E;\\ 0,\ A_k\neq A_l,\ P_k^E \not\parallel P_l^E;\\ 1,\ A_k=A_l,\ P_k^E=P_l^E;\\ 2,\ A_k=A_l,\ P_l^E\subseteq P_k^E\end{cases}$$

where symbol "$\not\parallel$" denotes relation incomparability. Let us introduce notations

$$\begin{aligned}X&=E(x_1,\ldots,x_{k-1},x_k,x_{k+1},\ldots,x_n)\equiv E(x_1,\ldots,x_{k-1},x'_k,x_{k+1},\ldots,x_n),\\ Y&=E(x_1,\ldots,x_{l-1},x_k,x_{l+1},\ldots,x_n)\equiv E(x_1,\ldots,x_{l-1},x'_k,x_{l+1},\ldots,x_n)\end{aligned}$$

then it leds to the following sufficiently clear statement.

Proposition 1. For arbitrary n-arity relation $E(x_1,\ldots,x_n)$ values of the indicator function $f(k,l)$ are specified by conditions

if $X\Leftrightarrow Y$ then $f(k,l)=1$,
if $X\Rightarrow Y$ then $f(k,l)=-1$,
if $X\Leftarrow Y$ then $f(k,l)=2$,
if $X\not\parallel Y$ then $f(k,l)=0$.

Definition 1. Arbitrary n-arity relation $E(x_1,\ldots,x_n)$ is said to be internally (*k*,*l*)-coherent if and only if $A_k=A_l$ and $P_k^E=P_l^E$.

It is reasonable to mention that equivalence relation

$$V_E(k,l)=1\Leftrightarrow A_k=A_l,\ P_k^E=P_l^E\quad (f(k,l)=1)$$

is induced on $\{1,2,\ldots,n\}$. This relation can be expressed as matrix of internal coherence $\Phi(E)=(V_E(k,l))$.

Proposition 2. Under corresponding renumbering of n-arity relation $\mathrm{E}(x_1,\ldots,x_n)$ arguments, the matrix of internal coherence $\Phi(\mathrm{E})$ can be represented as block-diagonal matrix

$$\Phi(\mathrm{E})=\left(\begin{array}{cccccccc} 1 & \cdots & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \vdots & & & & & \vdots \\ 1 & \cdots & 1 & & & & & \vdots \\ 0 & & & \ddots & & & & \vdots \\ \vdots & & & & \ddots & & & \vdots \\ \vdots & & & & & 1 & \cdots & 1 \\ \vdots & & & & & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 1 & \cdots & 1 \end{array}\right)\begin{array}{l} \left.\vphantom{\begin{array}{c}1\\ \vdots\\ 1\end{array}}\right\}r_1 \\ \vdots \\ \left.\vphantom{\begin{array}{c}1\\ \vdots\\ 1\end{array}}\right\}r_s \end{array} \tag{5}$$

where $r_1+r_2+\ldots+r_s=n$, $\begin{cases} \mathrm{A}_1=\ldots=\mathrm{A}_{r_1}=\mathrm{B}_1, \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \mathrm{A}_{n-r_s+1}=\ldots=\mathrm{A}_n=\mathrm{B}_n, \end{cases}$ $\begin{cases} \mathrm{P}_1^{\mathrm{E}}=\ldots=\mathrm{P}_{r_1}^{\mathrm{E}}=\mathrm{L}_1^{\mathrm{E}}, \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \mathrm{P}_{n-r_s+1}^{\mathrm{E}}=\ldots=\mathrm{P}_n^{\mathrm{E}}=\mathrm{L}_s^{\mathrm{E}}. \end{cases}$

Proposition 2 yields information that $\mathrm{E}(x_1,\ldots,x_n)$ has the carrier $\mathrm{B}_1^{r_1}\times\ldots\times\mathrm{B}_s^{r_s}$ and establishes s different equivalences $\mathrm{L}_i^{\mathrm{E}}$ on $\mathrm{B}_i\times\mathrm{B}_i$. From now on, $\mathrm{B}_i/\mathrm{L}_i^{\mathrm{E}}$ stands for cosets and $[\mathrm{B}_i/\mathrm{L}_i^{\mathrm{E}}]^{r_i}$ denotes the direct product of equal cosets, i.e. we can conclude that desired relation forms on $[\mathrm{B}_1/\mathrm{L}_1^{\mathrm{E}}]^{r_1}\times\ldots\times[\mathrm{B}_s/\mathrm{L}_s^{\mathrm{E}}]^{r_s}$ or actually on $\mathrm{B}_i/\mathrm{L}_i^{\mathrm{E}}\times\ldots\times\mathrm{B}_i/\mathrm{L}_i^{\mathrm{E}}$.

Definition 2. Arbitrary n-arity relation $\mathrm{E}(x_1,\ldots,x_n)$ induces on $\mathrm{B}_i/\mathrm{L}_i^{\mathrm{E}}\times\ldots\times\mathrm{B}_i/\mathrm{L}_i^{\mathrm{E}}$ a relation E^{M} which will be referred to as a multirelation.

As it has already been stated above, it is important to understand that significance should be assigned to the simultaneous application of relations.

Definition 3. Two arbitrary relations n-arity $\mathrm{E}_1(x_1,\ldots,x_n)$ on $\mathrm{A}_1\times\ldots\times\mathrm{A}_n$ and m-arity $\mathrm{E}_2(x_1,\ldots,x_m)$ on $\mathrm{C}_1\times\ldots\times\mathrm{C}_m$ are externally (i,j)-coherent if and only if $\mathrm{A}_i=\mathrm{A}_j$ and $\mathrm{P}_i^{\mathrm{E}1}=\mathrm{P}_j^{\mathrm{E}2}$.

Obviously, on $\{1,2,\ldots,n\}\times\{1,2,\ldots,m\}$ an equivalence relation $\mathrm{V}_{\mathrm{E}_1,\mathrm{E}_2}$ and $(n\times m)$ matrix of external coherence $\Phi(\mathrm{E}_1,\mathrm{E}_2)$ are introduced similarly to the one relation case. More precisely, elements $l_{ij}, i=\overline{1,n}, i=\overline{1,m}$ of matrix $\Phi(\mathrm{E}_1,\mathrm{E}_2)$ are specified by expression

$$l_{ij}=\begin{cases} 1, \mathrm{A}_i=\mathrm{A}_j, \mathrm{P}_i^{\mathrm{E}1}=\mathrm{P}_j^{\mathrm{E}2}, \\ 0, \mathit{otherwise}. \end{cases}$$

Proposition 3. Two arbitrary relations $\mathrm{E}_1(x_1,\ldots,x_n)$ on $\mathrm{A}_1\times\ldots\times\mathrm{A}_n$ and $\mathrm{E}_2(x_1,\ldots,x_m)$ on $\mathrm{C}_1\times\ldots\times\mathrm{C}_m$ induce two multirelations $\mathrm{E}_1^{\mathrm{M}},\mathrm{E}_2^{\mathrm{M}}$ with the same carrier if and only if by rows (column) transpositions the matrix of external coherence $\Phi(\mathrm{E}_1,\mathrm{E}_2)$ is reduced to the block-diagonal form

$$\Phi^*(\mathrm{E}_1,\mathrm{E}_2)=\begin{array}{c} \begin{array}{cccc} \overbrace{\quad\quad}^{\beta_1} & \overbrace{\quad\quad}^{\beta_2} & \overbrace{\quad\quad}^{\beta_i} & \overbrace{\quad\quad}^{\beta_s} \end{array} \\ \left(\begin{array}{cccccccccccccc} 1 & \cdots & 1 & 0 & \cdots & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots & & & & & & & & \vdots \\ 1 & \cdots & 1 & 0 & \cdots & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots & & & & & & & & \vdots \\ \vdots & & 0 & 1 & \cdots & 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & & \vdots & 0 & & 0 & \ddots & & & & & & & \vdots \\ \vdots & & \vdots & \vdots & & \vdots & 0 & 1 & \cdots & 1 & 0 & \cdots & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots & & & & \vdots \\ \vdots & & \vdots & \vdots & & \vdots & 0 & 1 & \cdots & 1 & 0 & \cdots & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & & & \ddots & & & \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & & & 0 & 1 & \cdots & 1 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & \cdots & \cdots & 0 & 1 & \cdots & 1 \end{array}\right) \end{array} \begin{array}{l} \left.\vphantom{\begin{array}{c}1\\ \vdots\\ 1\end{array}}\right\}\alpha_1 \\ \left.\vphantom{\begin{array}{c}1\\ \vdots\\ 1\end{array}}\right\}\alpha_2 \\ \\ \left.\vphantom{\begin{array}{c}1\\ \vdots\\ 1\end{array}}\right\}\alpha_i \\ \\ \left.\vphantom{\begin{array}{c}1\\ \vdots\\ 1\end{array}}\right\}\alpha_s \end{array}.$$

It should be emphasized that proposition 3 can be reformulated in terms of difunctional relations. Let us recall that a binary relation is difunctional if for all $i, i', j, j' \in \{1, 2, ..., max(n,m)\}$ the implication

$$\mathrm{V}_{\mathrm{E}_1,\mathrm{E}_2}(i, j') = 1,\ \mathrm{V}_{\mathrm{E}_1,\mathrm{E}_2}(i', j') = 1, \mathrm{V}_{\mathrm{E}_1,\mathrm{E}_2}(i', j) = 1 \ \Rightarrow\ \mathrm{V}_{\mathrm{E}_1,\mathrm{E}_2}(i, j) = 1.$$

holds.

Proposition 3*. Two arbitrary relations $\mathrm{E}_1(x_1,\ldots,x_n)$ on $\mathrm{A}_1 \times \ldots \times \mathrm{A}_n$ and $\mathrm{E}_2(x_1,\ldots,x_m)$ on $\mathrm{C}_1 \times \ldots \times \mathrm{C}_m$ induce two multirelations $\mathrm{E}_1^{\mathrm{M}}, \mathrm{E}_2^{\mathrm{M}}$ with the same carrier if and only if $\mathrm{V}_{\mathrm{E}_1,\mathrm{E}_2}$ is a difunctional relation.

Now it is seems quite logical to assert that interpretation of multirelations may have two-valued nature. On the one hand, we have seen that a multirelation is induced by embedded properties of original information. On the other hand, data of arbitrary nature can be analyzed jointly with given equivalence relation associated with an object-oriented problems.

Necessary and sufficient conditions of multialgebraic systems existence

We have seen necessary and sufficient properties for multirelations carrier equality, however, we have not yet argued general existence conditions of multialgebraic systems. At this point it will be useful to introduce some terminology.

Let $\mathrm{A}_1,\ldots,\mathrm{A}_n,\ldots$ be arbitrary sets and let $\mathrm{P}_1,\ldots,\mathrm{P}_n,\ldots$ $(dom\,\mathrm{P}_i = \mathrm{A}_i)$ be corresponding equivalence relations then if $\mathrm{A}_i = \{x_1^i,\ldots,x_\alpha^i,\ldots\}$ we have

$$\begin{aligned} &\mathrm{A}(n) = \textstyle\prod_{i=1}^{n} \mathrm{A}_i, \\ &\mathrm{P}(n) = \textstyle\prod_{i=1}^{n} \mathrm{P}_i, \\ &\alpha(n) = \{\alpha_1,\ldots,\alpha_n\}, \\ &x_{\alpha(n)} = \{x_{\alpha_1}^1,\ldots,x_{\alpha_n}^n\} \in \mathrm{A}(n) \end{aligned}$$

and $\mathrm{P}(n)$ is the equivalence on $\mathrm{A}(n)$ in the sense that

$$\mathrm{P}(n)[x_{\alpha(n)}, x_{\alpha'(n)}] = 1 \Leftrightarrow \mathrm{P}_i(x_{\alpha_i}^i, x_{\alpha_i}^i),\ i = \overline{1, n}\,.$$

Definition 4. An equivalence relation $\mathrm{P}(n)$ on $\mathrm{A}(n)$ will be referred to as partial factor with the notation $\mathrm{P}(n) = h - fac\,\mathrm{S}$ if and only if

$$\forall x_{\alpha(n)}, x_{\alpha'(n)} \in \mathrm{A}(n) : \mathrm{P}(n)[x_{\alpha(n)}, x_{\alpha'(n)}] = 1, \mathrm{S}(x_{\alpha(n)}) = 1 \Rightarrow \mathrm{S}(x_{\alpha'(n)}) = 1\,.$$

It is obvious that the equivalence induces cosets $\mathrm{A}(n)/\mathrm{P}(n)$. If $[x_{\alpha(n)}]_{\mathrm{P}(n)} \in \mathrm{A}(n)/\mathrm{P}(n)$ is certain coset then pair $\mathrm{P}(n)$ and S defines n-arity multirelation

$$\mathrm{S}^{\mathrm{M}}([x_{\alpha(n)}]_{\mathrm{P}(n)}) = 1 \Leftrightarrow \mathrm{S}(x_{\alpha(n)}) = 1\,. \tag{6}$$

Under condition (6) a multirelation S^{M} is congruent dependent on $\mathrm{P}(n)$ and S what we denote by $\mathrm{S}^{\mathrm{M}} = con\,(\mathrm{P}(n),\mathrm{S})$ for brevity.

Remark 1. It is easy enough to understand that the definition of congruent dependence is correct if and only if $\mathrm{P}(n) = h - fac\,\mathrm{S}$.

Remark 2. It is a simple matter to show that $\mathrm{P}(n) = h - fac\,\mathrm{S}$ if and only if $\mathrm{P}(n)$ is induced by n-arity relation S satisfying (4).

Definition 5. Partial factor $\mathrm{P}(n)$ will be named factor (full factor) with notation $\mathrm{P}(n) = fac\,\mathrm{S}$ if and only if

$$\forall x_{\alpha(n-1)} \in \mathrm{A}(n-1), \forall x_{\alpha_n}^n, x_{\alpha'_n}^n \in \mathrm{A}(n):\ \mathrm{S}(x_{\alpha(n)}) = \mathrm{S}(x_{\alpha'(n)}) = 1 \Rightarrow \mathrm{P}_n[x_{\alpha_n}^n, x_{\alpha'_n}^n] = 1$$

where $x_{\alpha(n)} = (x_{\alpha(n-1)}, x_{\alpha_n}^n)$, $x_{\alpha'(n)} = (x_{\alpha(n-1)}, x_{\alpha'_n}^n)$.

Consequently, we get a multioperation $\mathrm{F^M}$

$$\mathrm{F^M}([x_{\alpha(n)}]_{\mathrm{P}(n)}) = [x^n_{\alpha_n}]_{\mathrm{P}_n} \Leftrightarrow \mathrm{S}(x_{\alpha(n)}) = 1$$

where $[x^n_{\alpha_n}]_{\mathrm{P}_n}$ is the coset of the set A_n in regard to the equivalence P_n and the element $x^n_{\alpha_n}$ belongs to this coset.

Remark 3. It is easy enough to see that $[x^n_{\alpha_n}]_{\mathrm{P}_n}$ is unique coset. In this connection $\mathrm{F^M} = con\,(\mathrm{P}(n),\mathrm{S})$ and $\mathrm{P}(n) = fac\,\mathrm{S}$ if and only if $\mathrm{P}(n)$ is induced by n-arity relation S satisfying (4).

The theorem of a multialgebraic system existence under given external equivalence and the same carrier had been proved [Mashtalir et al., 2003] and with mentioned notations it can be represented as follows.

Theorem 1. Suppose that A is arbitrary carrier, P is given equivalence on A^2 and $\Sigma_\mathrm{S} = \{\mathrm{P},\mathrm{S}_1,\ldots,\mathrm{S}_\beta,\ldots\}$ is a family of n-arity relations then a model $\langle \mathrm{A},\Sigma_\mathrm{S}\rangle$ generates multialgebraic system $\langle \mathrm{A/P},\{\mathrm{F^M_\xi}\},\{\mathrm{S^M_\eta}\}\rangle$ where $\mathrm{F^M_\xi} = con\,(\mathrm{P}^n,\mathrm{S}_\xi)$, $\mathrm{S^M_\eta} = con\,(\mathrm{P}^n,\mathrm{S}_\eta)$ and

$$\exists \Sigma_{1\mathrm{S}},\Sigma_{2\mathrm{S}} \subset \Sigma_\mathrm{S} \;:\; \Sigma_{1\mathrm{S}} \cap \Sigma_{2\mathrm{S}} = \varnothing,\; \Sigma_{1\mathrm{S}} \cup \Sigma_{2\mathrm{S}} = \Sigma_\mathrm{S} \setminus \mathrm{P},\quad \mathrm{S}_\xi \in \Sigma_{1\mathrm{S}},\;\; \mathrm{S}_\eta \in \Sigma_{2\mathrm{S}}$$

if and only if

$$\forall \mathrm{S}_\beta \in \Sigma_\mathrm{S} \setminus \mathrm{P} \Rightarrow \mathrm{P}^n = \begin{cases} h - fac\,\mathrm{S}_\beta, & \mathrm{S}_\beta \in \Sigma_{1\mathrm{S}} \setminus \mathrm{P}, \\ fac\,\mathrm{S}_\beta, & \mathrm{S}_\beta \in \Sigma_{2\mathrm{S}} \setminus \mathrm{P}. \end{cases}$$

It should be emphasized that any n-arity relation E forms its equivalence $\mathrm{L^E} = \prod_{i=1}^{s} \mathrm{L}_i^\mathrm{E}$ on the carrier $\mathrm{B}_1 \times \ldots \times \mathrm{B}_s$ (see the explication of expression (5)) which is determined by the matrix of internal coherence. Further, the carrier structure is direct product of matrix blocks. Hence, the consideration of relations and matrices of external coherence by pairs gives possibilities to establish conditions that due to proposition 3* all pairs $(\mathrm{S}_{\beta'},\mathrm{S}_{\beta''})$ from this collection represent difunctional relations $\mathrm{V}_{\mathrm{S}_{\beta'},\mathrm{S}_{\beta''}}$. Granting remarks 1–3, we can restate theorem 1 and give more strong assertion of necessary and sufficient conditions for multialgebraic systems existence.

Theorem 2. Let $\{\mathrm{S}_1,\ldots,\mathrm{S}_\beta,\ldots\}$ be a family of arbitrary arity relations whose carriers may be different then multialgebraic system is induced if and only if

i) $\Sigma_\mathrm{S} = \{\mathrm{L^E},\mathrm{S}_1,\ldots,\mathrm{S}_\beta,\ldots\}$, $\mathrm{L^E}$ is an equivalence induced by $\mathrm{S}_1,\ldots,\mathrm{S}_\beta,\ldots$,

ii) $\exists \Sigma_{1\mathrm{S}},\Sigma_{2\mathrm{S}} \subset \Sigma_\mathrm{S} : \Sigma_{1\mathrm{S}} \cap \Sigma_{2\mathrm{S}} = \varnothing, \Sigma_{1\mathrm{S}} \cup \Sigma_{2\mathrm{S}} = \Sigma_\mathrm{S} \setminus \mathrm{L^E}$,

iii) $\forall \mathrm{S}_\beta \in \Sigma_\mathrm{S} \setminus \mathrm{L^E} \Rightarrow \underbrace{\mathrm{L^E} \times \ldots \times \mathrm{L^E}}_{n} = \begin{cases} h - fac\,\mathrm{S}_\beta, & \mathrm{S}_\beta \in \Sigma_{1\mathrm{S}} \setminus \mathrm{L^E}, \\ fac\,\mathrm{S}_\beta, & \mathrm{S}_\beta \in \Sigma_{2\mathrm{S}} \setminus \mathrm{L^E}, \end{cases}$

iv) $\forall \mathrm{S}_{\beta'},\mathrm{S}_{\beta''} \in \Sigma_\mathrm{S} \setminus \mathrm{L^E}$ and $\mathrm{V}_{\mathrm{S}_{\beta'},\mathrm{S}_{\beta''}}$ is difunctional relation.

Thus, a factorization of information in any feature space conceptually is one of the basic methods providing an interpretation of data. On the one hand, identification can be required up to given or explored equivalence relations set. With another, construction of equivalence classes often represents an essence and a purpose of data processing. We have introduced and proved conditions describing interdependence of different levels information representations.

Conclusion

Different types of granulation represent different aspects of data and provide different types of knowledge embedded in data. An intelligent data analysis based on granular computing deals with theories, methodologies, techniques and tools that provide consideration what is relevant and permit to ignore irrelevant details. Granular computing involves two-way communications upward and downward in a hierarchy of different abstraction levels that represent different granulated views of problems understanding. It is reasonable to assume that granules relations satisfying various axiomatics and operations with operands corresponding to granules offer advantages for formalization of transformations and interpretations in multilevel processing of arbitrary nature data.

There exist two distinct varieties of relations concerning data to be analyzed. First of all, we should emphasized internal (embedded) interrelationships of original data. Thus, latent information that induces relationships between granules has to be explored. In the second place, a relation concerned with applications can be introduced on the original data. In both cases joint analysis has to be carried into effect.

Usually there are possibilities of empirical verification of the properties only at the lower level of abstractions, i.e. with the use of original data. Partitions and coverings can be normally valid models of granulation, and properties of relations along with operations in conformity with equivalence or tolerance classes generate a basic interest. In other words, the problem consists in an examination of original data to know properties of granule families. In our opinion multialgebraic systems can be sufficiently adequate tools in order to formalize elements of detailing or roughening such as granule, granulated view, granularity and hierarchy in the framework at least formal mathematical structures. We have established necessary and sufficient conditions of producing relations (with the same carrier) on granules induced by relations associated with original data. Furthermore, we have found conditions of multialgebraic systems existence. As development of these results it should be indicated the investigation of specific algebraic structures on original data such as semigroup, group, ring, different vector spaces etc. There arise several problems (it seems that peculiar but, vice versa, very important). Among them it should be noted comparisons of granule families for which there are no two ways about an introduction of an admissible metric on granule structures, e.g. on set partitions [Bobrowsky et al., 2006, Mashtalir et al., 2006] since it is often necessary to have dealings with a whole family of partitions and we have to be able to compare these partitions.

Bibliography

[Bargiela, Pedrycz, 2002] A. Bargiela, W. Pedrycz Granular computing: an introduction. Boston, Kluwer Academic Publishers, The Kluwer International Series in Engineering and Computer Science. Vol. 717. 2002. 478 p.

[Bobrowsky et al., 2006] L. Bobrowsky, V. Mashtalir, V. Shlyakhov Metrics on arbitrary partitions of measurable sets. Global Information Systems. Problems and Tendencies of Progress. Kharkov, KnNURE. 2006. pp. 70–71.

[Doherty et al., 2003] P. Doherty, W. Lukaszewicz, A. Szalas Information granules for intelligent knowledge structures. Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing / G. Wang, Q. Liu, Y. Yao, A. Skowron (Eds.) Lecture Notes in Artificial Intelligence. Spinger-Verlag Berlin Heidelberg. 2003. Vol. 2639. pp. 405-412.

[Lin, 2003] T.Y. Lin Granular computing (Structures, Representations, and Applications). Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing. G. Wang, Q. Liu, Y. Yao, A. Skowron (Eds.) Lecture Notes in Artificial Intelligence. Spinger-Verlag Berlin Heidelberg. 2003. Vol. 2639. pp. 16-24.

[Mashtalir et al., 2006] Mashtalir V., Mikhnova E., Shlyakhov V., Yegorova E. A novel metric on partitions for image segmentation. IEEE International Conference on Video and Signal Based Surveillance, avss, p. 18, 2006.

[Mashtalir, Shlyakhov, 2003] Mashtalir V.P., Shlyakhov V.V. Properties of multialgebraic systems in problems of comparative recognition. Cybernetics and Systems Analysis. Vol.39, No.6. 2003. pp. 790–804.

[Pal et al., 2005] S.K. Pal, B.U. Shankar, P. Mitra Granular computing, rough entropy and object extraction. Pattern Recognition Letters. 2005. Vol. 26. pp. 2509-2517.

[Yager, 2002] R.R. Yager Using granular objects in multi-source data fusion // Rough Sets and Current Trends in Computing. J.J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.). Lecture Notes in Artificial Intelligence. Spinger-Verlag Berlin Heidelberg. 2002. Vol. 2475. pp. 324-330.

[Yao, 2001] Y.Y. Yao Information granulation and rough set approximation. International Journal of Intelligent Systems. 2001. Vol. 16, No. 1, pp. 87-104.

[Yao, Yao, 2002] J.T. Yao, Y.Y. Yao Induction of classification rules by granular computing. Rough Sets and Current Trends in Computing / J.J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.). Lecture Notes in Artificial Intelligence. Spinger-Verlag Berlin Heidelberg. 2002. Vol. 2475. pp. 331-338.

[Yao, 2005] Y.Y. Yao Perspectives of granular computing. Proceedings of 2005 IEEE International Conference on Granular Computing. 2005. Vol. 1. pp. 85-90.

[Yao, 2006] Y.Y. Yao Granular computing for data mining // Proceedings of SPIE Conference on Data Mining, Intrusion Detection, Information Assurance and Data Networks Security / B.V. Dasarathy (Ed.), Kissimmee, Florida, USA. 2006. pp. 1-12 (624105).

[Zadeh, 1997] L.A. Zadeh Towards a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic // Fuzzy Sets Systems. 1997. Vol. 19. pp. 111-127.

Authors' Information

Kagramanyan Alexander – Kharkov National University, Svobody sq., 4, Kharkov, Ukraine, 61077

Mashtalir Vladimir – Kharkov National University of Radio Electronics, Lenina ave., 14, Kharkov, Ukraine, 61166, mashtalir@kture.kharkov.ua

Shlyakhov Vladislav – Kharkov National University of Radio Electronics, Lenina ave., 14, Kharkov, Ukraine 61166

SOLVING A DIRECT MARKETING PROBLEM BY THREE TYPES OF ARTMAP NEURAL NETWORKS

Anatoli Nachev

Abstract: An important task for a direct mailing company is to detect potential customers in order to avoid unnecessary and unwanted mailing. This paper describes a non-linear method to predict profiles of potential customers using dARTMAP, ARTMAP-IC, and Fuzzy ARTMAP neural networks. The paper discusses advantages of the proposed approaches over similar techniques based on MLP neural networks.

Keywords: ARTMAP, neural networks, data mining

ACM Classification Keywords: F.1.1 Models of Computation - neural networks, H.2.8 Database Applications - data mining

Introduction

Many companies use direct mailing to potential customers, or ‘junk mail’, to market a product or service. This can be an effective marketing approach, however much of this junk mail is really of no interest to the majority of people that receive it. The task how to predict the profiles of potential customers for a product, given information about the clients and a test sample of customers possessing the particular product is a well-known data mining problem from the world of direct marketing. The prediction task discussed in this paper, or the underlying problem, is to find a subset of customers with a probability of having a caravan insurance policy above some boundary probability. Those customers can be targeted by mailing promotional materials. The boundary of the targeted group depends on the cost and benefits such as of the costs of mailing and benefits of selling insurance policies.

The dataset used to test the proposed approach is based on real world business data [Van Der Putten, 2000]. It is a block of very detailed survey information on the people, some of whom bought and plan to buy a caravan insurance policy. The people were asked to answer 85 questions, each of which can be regarded as one feature in the classification. The block of data consists of 3 parts. The first is training data, which contains a number of survey responses, some of which come from caravan policy holders. The second part is testing data, and it contains answers from potential caravan insurance policy buyers. The last part is the true data that shows who of those potential buyers actually bought the policy at last. The maximum number of policy owners that could be found is 238. If a random selection is applied, average results provide 42 policy owners, or a hit rate (percentage of real policy buyers out of all predictions made) is about 6%.

A lot of techniques have been used to predict which customers are likely to respond or purchase a product, both linear and non-linear. Methods include: standard statistics [Van Der Putten, 2000], backpropagation MLP neural networks [Brierly, 2000], [Crocoll, 2000], [Shtovba et al., 2000], self-organizing maps (SOMs) [Vesanto et al., 2000], genetic programming, C4.5, CART, and other decision tree induction algorithms, fuzzy clustering and rule discovery, support vector machines (SVMs), logistic regression, boosting and bagging, all described in [Van Der Putten, 2000]. The best predictive technique reported in [Elkan, 2001] and [Van Der Putten, 2000] is the Naive Bayesian learning. It has been tested on 800 predictions and gives a hit rate about 15.2%. Predictors based on the backpropagation MLP networks show accuracy rate about 71% and hit rate about 13% as reported in [Brierly, 2000], [Candocia, 2004], [Crocoll, 2000], and [Van Der Putten, 2000].

This paper discusses three non-linear approaches based on dARTMAP, ARTMAP-IC, and Fuzzy ARTMAP neural networks.

Section 1 outlines the prediction task and introduces the reader to the discussed domain.

Section 2 outlines the main characteristics and function of predictors based on the three ARTMAP models.

Section 3 discusses the preprocessing steps needed to prepare an input dataset for the three ARTMAP network.

Section 4 describes experiments conducted in order to solve the prediction task and results from those experiments.

Predictors Based on dARTMAP, ATRMAP-IC, and Fuzzy ARTMAP Neural Networks

Adaptive Resonance Theory (ART) began with an analysis of human cognitive information processing [Grossberg, 1976]. Fundamental computational design goals have always included memory stability with fast or slow learning in an open and evolving input environment. As a real-time model of dynamic processes, an ART network is characterized by a system of ordinary differential equations, which are approximated by an algorithm for implementation purposes [Grossberg, 1980].

ART is a family of neural networks for fast learning, pattern recognition, and prediction, including both unsupervised: ART1, ART2, ART2-A, ART3, Fuzzy ART, Distributed ART; and supervised: ARTMAP, ARTMAP-IC, Fuzzy ARTMAP, ART-EMAP, ARTMAP-FTR, dARTMAP, and Default ARTMAP systems.

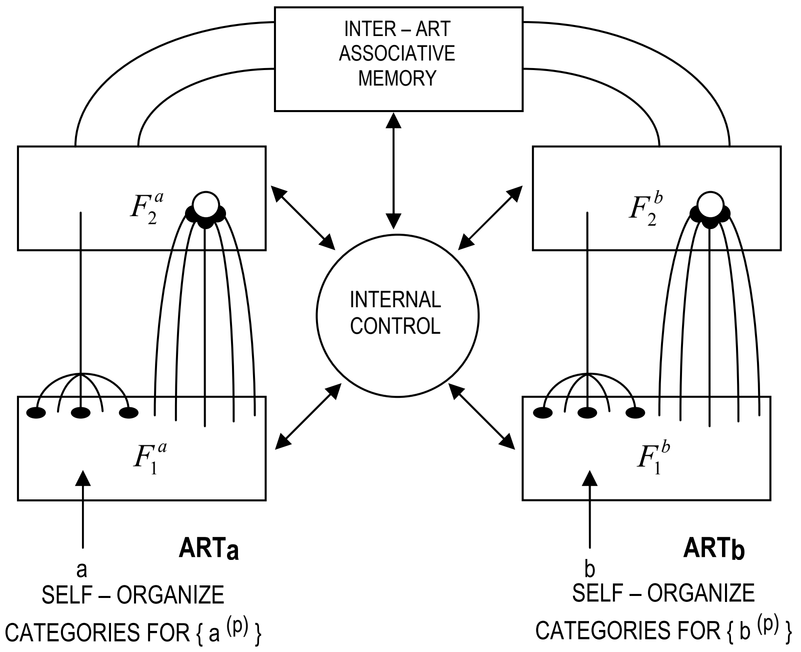

INTER – ART ASSOCIATIVE MEMORY
F_2^a
F_2^b
INTERNAL CONTROL
F_1^a
F_1^b
ARTa
ARTb
a
b
SELF – ORGANIZE CATEGORIES FOR { a (p) }
SELF – ORGANIZE CATEGORIES FOR { b (p) }

Figure 1. Components of an ARTMAP system.

ARTMAP neural networks develop stable recognition codes in real time in response to arbitrary sequences of input patterns. They were designed to solve the stability-plasticity dilemma that every intelligent machine learning system has to face: how to keep learning from new events without forgetting previously learned information.

ARTMAP networks consist of two ART1 networks, ARTa and ARTb, bridged via an inter-ART module, as shown on Figure 1. An ART module has three layers: the input layer (F0), the comparison layer (F1), and the recognition layer (F2) with m , m and n neurons, respectively. The neurons, or nodes, in the F2 layer represent input categories. The F1 and F2 layers interact with each other through weighted bottom-up and top-down connections, which are modified when the network learns. There are additional gain control signals in the network that regulate its operation.

Distributed ARTMAP (dARTMAP)

A key feature of the dARTMAP system is that it contains distributed ART (dART) modules instead of ART1 [Carpenter et al., 1998a]. A dART module combines the computational advantages of ART1 and MLP systems [Carpenter et al., 1996]. Properties include code stability when learning is fast and on-line, memory compression when inputs are noisy and unconstrained. The coding field of a dARTMAP supervised system is analogous to the hidden layer of a multi-layer perceptron (MLP), where distributed activation helps the network achieve memory compression and generalization. Each dARTMAP input first activates a distributed code. If this code produces a correct prediction, learning proceeds in the distributed coding mode. If the prediction is incorrect, the network resets the active code via match tracking feedback. In dARTMAP networks, the reset process triggers a search for a category node that can successfully code the current input. It also places the system in a winner-takes-all (WTA) coding mode for the duration of the search. In WTA mode, dARTMAP can add nodes incrementally as needed. When a coding node is added to the network, it becomes permanently associated with the output class that is active at the time. From then on, the network predicts this class whenever the same coding node is chosen in WTA mode.

ARTMAP-IC

ARTMAP- IC adds to the basic ARTMAP system new capabilities designed to solve the problem with inconsistent cases, which arises in prediction, where identical input vectors correspond to cases with different outcomes [Carpenter et al., 1998b]. It modifies the ARTMAP search algorithm to allow the network to encode inconsistent cases (IC) by involving an instance counting procedure and a new match tracking algorithm that consistently improve both predictive accuracy and code compression, compared to the basic ARTMAP networks. These added capabilities also allow ARTMAP-IC to encode predictions of inconsistent cases in the training set, giving good test set performance on various problems.

Fuzzy ARTMAP

Fuzzy ARTMAP, introduced in [Carpenter et al., 1992], is a natural extension to ARTMAP that uses Fuzzy ART instead of ART1 modules. Fuzzy ARTMAP is completely equivalent to ARTMAP, when the input domain is the Hamming cube {0,1}. It is capable of forming associative maps between clusters of its input and output domains in a supervised manner. Each module features its own set of parameters, with values assigned independently. ARTa is responsible clustering the input feature space; ARTb – for the output feature space. The inter-ART's role is to establish the correct association between input and output categories (cluster associations). The Fuzzy ARTMAP networks have been found useful in pattern recognition, because classification may be viewed as a many-to-one mapping task that entails clustering of the input space and then association of the produced clusters with a limited number of class labels (output clusters that encode a single class label).

The three ARTMAP modifications discussed above are designed to guarantee stable memories even with fast online learning. In contrast, when multi-layer perceptron (MLP) neural networks are used for classification problems, they employ slow off-line learning in order to avoid catastrophic forgetting in an open input environment, which limits adaptation for each input and so requires multiple presentations (epochs) of the training set. With fast learning, MLP memories suffer catastrophic forgetting.

Input Data

Input data for the three ARTMAP neural network simulators are available from the data mining company Sentient Machine Research [Van Der Putten, 2000].

The train dataset contains 5822 customer records. Each record consists of 86 attributes containing socio-demographic data represented by attributes 1-43 and product ownership attributes 44-86. The socio-demographic data is derived from zip codes. All customers living in areas with the same zip code have the same socio-demographic attributes. Attribute 86, "CARAVAN: Number of mobile home policies", is the target variable.

Evaluation dataset for validation of the prediction model consists of 4000 customer records. It has the same format as the training dataset, only the target is missing. Targets for the evaluation set have been provided by a separate file.

An important feature of the datasets is that the values of all numerical attributes were made discrete in advance. For example, instead of a real-valued feature giving the precise monetary amount that a customer pays for car insurance, the datasets include only a discrete-valued feature that categorizes this amount into one of seven different discrete levels.

Each of the three ARTMAP neural networks discussed here requires data samples, or input patterns, to be presented as M-component vectors of floating point numbers in the range [0, 1]. Therefore both train and evaluation datasets require normalization or mapping the original values into that range. For the purposes of normalization, each attribute was treated as an independent variable and submitted to a linear transformation:

$$\tilde{x}_i = \frac{(x_i - x_i^{\min})}{(x_i^{\max} - x_i^{\min})},$$

where $\tilde{x}_i$ is the normalized variable component; x_i is the original variable component; $x_i^{\max}$ is the max variable component; and $x_i^{\min}$ is the min variable component..

An effective solution of the prediction task should involve a selected subset of attributes, rather than the whole set, as the discriminatory power of a selection would be higher than one of the whole set. The selection itself is critical for a successful prediction. [Van Der Putten, 2000] reports a variety of techniques that rank importance and sensitivity of the attributes in the light of the prediction task, such as greedy feature selection algorithm, statistics, stepwise procedures, evolutionary algorithms, chi-analysis, etc.. For the purposes of the experiments reported here, those rank results were taken into consideration. In addition to that many empirical experiments and simulations were conducted in order to explore how different subsets of attributes influence the predictiveness of the three ARTMAP models. Experimental results show that the highest predictive rate for each of the tree models can be achieved by a set of the following attributes (sequence numbers correspond to the original notation):

- dARTMAP: {43, 47, 59}
- ARTMAP-IC: {43, 47, 59}
- Fuzzy ARTMAP: {1, 5, 12, 16, 18, 25, 30, 32, 34, 37, 42, 43, 44, 47, 59, 61, 65, 68, 80, 82, 85}.

Table 1 describes the attribute meanings.

The three attributes with maximal discriminatory power are:

- Purchasing power class (attribute #43, MKOOPKLA).
- Contribution car policies (attribute #47, PPERSAUT).
- Contribution fire policies (attribute #59, PBRAND).

No	*Attribute Name and Description*	*No*	*Attribute Name and Description*
1	*MOSTYPE Customer Subtype*	*43*	*MKOOPKLA Purchasing power class*
5	*MOSHOOFD Customer main type*	*44*	*PWAPART Contribution private third party insurance*
12	*MRELOV Other relation*	*47*	*PPERSAUT Contribution car policies*
16	*MOPLHOOG High level education*	*59*	*PBRAND Contribution fire policies*
18	*MOPLLAAG Lower level education*	*65*	*AWAPART Number of private third party insurance 1 - 12*
25	*MSKA Social class A*	*68*	*APERSAUT Number of car policies*
30	*MHHUUR Rented house*	*80*	*ABRAND Number of fire policies*
34	*MAUT0 No car*	*82*	*APLEZIER Number of boat policies*
37	*MINKM30 Income < 30.000*	*85*	*ABYSTAND Number of social security*
42	*MINKGEM Average income*		

Table 1. Selected attributes from train and evaluation datasets.

These three attributes describe people who are very likely to hold or possibly be in the market for a caravan insurance policy because such people are likely to be:

- People having a high level of purchasing power. Apart from 'Purchasing Power Class', all socio-demographic attributes, including customer segmentations by lifestyle, income, etc., often do not add any predictive power when behavioral data is available. People with high purchasing power are not necessarily enthusiastic about insuring their property, but they do have quite enough wealth to own a caravan, even if using it were not their prime hobby. Typical customers have high, or at least medium, education, status, social class, and income levels. For the feature selection, all demographic attributes were discarded, except attribute 43, 'MKOOPKLA Purchasing power class'
- Car owners with high contribution to car policy purchases. Those who do not have a car are unlikely to own a caravan, as they generally require to be towed. Car owners can be readily identified as those having existing car insurance policies. The amount spent on policies is also important. People who spend more on car insurance are most likely to be caravan policy buyers, and the more they spend, the more likely a buyer they are.
- People having fire policy with high level of contribution. This may indicate that the fire insurance is for a caravan. The level of the fire insurance cover that is most likely to be indicative of a caravan policy is level 4.

Intuitively, these three predictors identify customers who have a car and are wealthier than average, and who in general carry more insurance coverage than average. It is not surprising that these are the people who are most likely to have caravan insurance.

Experimental results however show that the Fuzzy ARTMAP model requires 13 more attributes to achieve its best predictiveness, as mentioned above. These extra attributes have less discriminatory power, but in combination with the major three, make the Fuzzy ARTMAP the best performer among the three neural network models.

Experimental Results

A number of experiments were conducted using simulators of dARTMAP, ARTMAP-IC, and Fuzzy ARTMAP neural networks. The goal of the experiments was to identify how order in which attributes and input patters are submitted influences the predictiveness; what the optimal network parameters are; and how network parameters affect the train and test time and memory consumption.

To maximize use of the datasets and to avoid bias in the selection of the training and test sets, a cross-validation technique was applied. Cross-validation creates N (N=5) copies of a classifier and tests each on 1/N of the evaluation dataset, after training it on 1/N-th of the training set. In other words, each classifier makes predictions for its 1/N-th of the data, yielding predictions for the whole set.

Results from the experiments showed that both dARTMAP and ARTMAP-IC models are sensitive to the order in which attributes appear in the training set. Out of six permutations of the attributes 43, 47, and 59, dARTMAP gave a satisfactory level of prediction for three permutations: {43, 47, 59}, {43, 59, 47}, and {59, 43, 47}; ARTMAP-IC gave satisfactory results for the four permutations: {43, 47, 59}, {43, 59, 47}, {59, 43, 47}, and {47, 43, 59}. Experiments showed that the Fuzzy ARTMAP is not sensitive to the order in which attributes appear to the input. To see how sequence of input patters affect the predictability, the networks was trained by eight different orders of the training sets: original order; real buyer entries shifted to the beginning; to the end; and five different randomly chosen orders. Experiments showed that the dARTMAP and ARTMAP-IC yield unsatisfactory prediction outputs for the shifted orders and do not distinguish the rest of the orders, yielding satisfactory results. The Fuzzy ARTMAP shows satisfactory results for all the orders.

To see how the network parameters influence the predictiveness, simulations with a full range of parameter values were conducted. Results show that an acceptable level of predictiveness for the three models can be achieved by the following values of network parameters: $\rho_{test} = 0$, $\alpha = 0.01$, $\varepsilon = -0.001$, and $p = 1.0$. The parameter $\beta = 1.0$ ensures best performance for the ARTMAP-IC and dARTMAP models, but Fuzzy ARTMAP performs best when $\beta = 0.968$.

The vigilance parameter ρ (Rhobar) affects the performance of the three models by tuning the details and granularity of the clusters, thus changing accuracy of predictions and hit rate. The parameter was set to various values between $0.915 \le \rho \le 0.955$ with step of increment 0.005. Figure 2 shows accuracy rate for each of the networks. Figures 3 shows total positive predictions, both correct and incorrect, which determines the boundary of the targeted group of customers, yet the expenses that should be made for direct marketing. Fuzzy ARTMAP has best performance when the size of the targeted group of is about 160; ARTMAP-IC outputs a group of 63 customers, whilst dARTMAP achieves a good prediction rate with a relatively small group of about 30 customers. Figure 4 shows the number of correct predictions. Given that the three models output different number of potential buyers, as per Figure 3, the number of correct predictions is influenced by the number of the total number of potential buyers, rather than a better prediction. Figure 5 shows how the vigilance parameter affects the hit rate. Maximal hit rates are: 17.96% for the Fuzzy ARTMAP; 14.29% for the ARTMAP-IC, and 30% for the dARTMAP. These hit rates exceed the reported 13% of the MLP networks used for the same prediction task. This result exceeds the best hit rate of 15% reported in [Elkan, 2001] and [Van Der Putten, 2000], but direct comparison of results would not be accurate, as the two cases are based on different size of the targeted group. For boundaries where a scale of a direct marketing is comparable to the scale adopted by the experiments described above, the three neural network models could be considered as good performing and applicable.

Training time is another advantage of the three ARTMAP models, which should be pointed out. With the dataset discussed here, the three neural networks get trained for about 5 seconds in contrast to the MPL networks that require about 35 minutes [Shtovba et al., 2000]. MPL networks, however, outperform ARTMAP in the test time, but both ARTMAP and MLP respond for less that a second, which is response in a real time.

Another advantage of the three ARTMAP models is that for implementation of the long-term memory (LTM) the simulators consume a little RAM - about 4.9 KB.

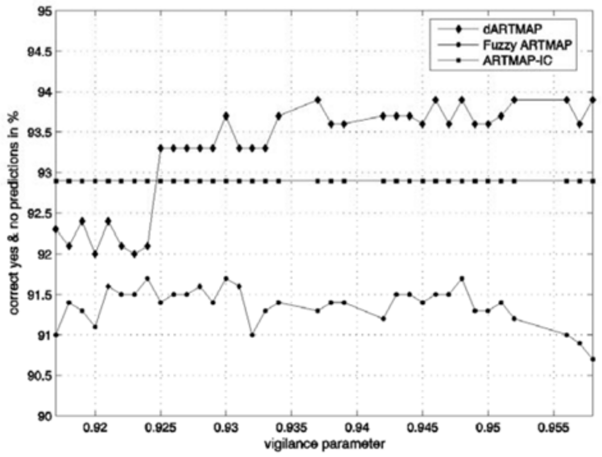


Figure 2. Correct yes and no predictions.

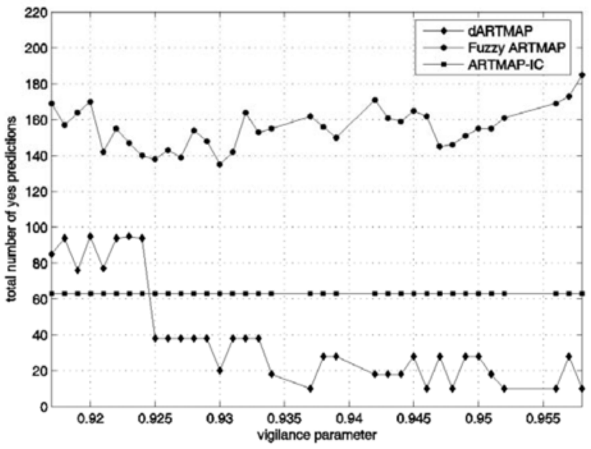


Figure 3. Total yes predictions (both correct and wrong).

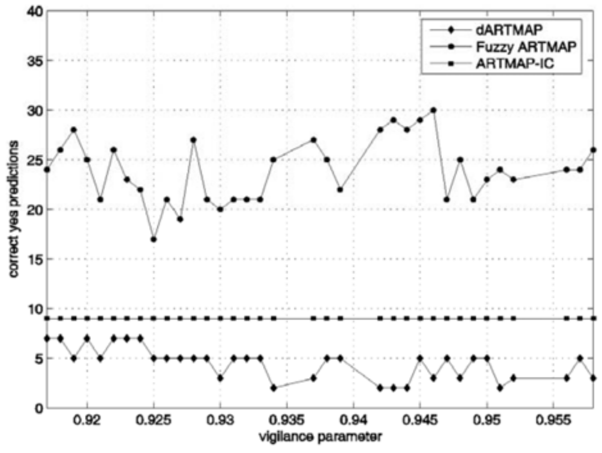


Figure 4. Correct yes predictions.

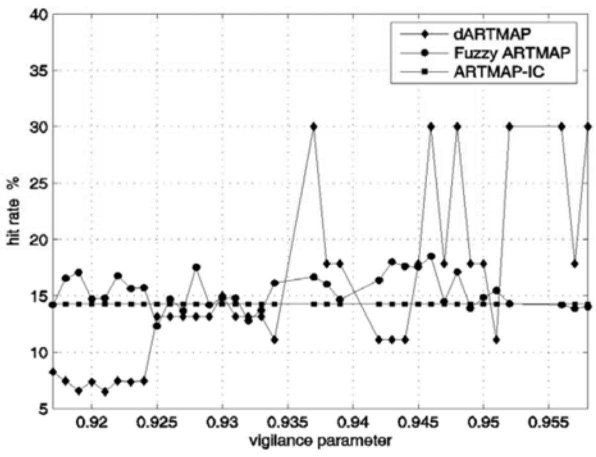


Figure 5. Hit rate.

Conclusion

This paper describes three non-linear methods to predict profiles of potential customers of a product using dARTMAP, ARTMAP-IC, and Fuzzy ARTMAP neural networks, given a test sample of customers possessing the particular product. Such a prediction can be used for direct marketing purposes. The proposed methods require a pre-processing of the test sample data in order to prepare specific input for simulators based on the ARTMAP paradigm. The theoretical concepts and conducted experiments lead to the following conclusions:

- The three ARTMAP models discussed in the paper show higher hit rates than those obtained by other non-linear approaches based on MLP neural networks.
- The Fuzzy ARTMAP model provides stable prediction, not affected by the order of attributes presented to the model, nor the order of input patterns. In contrast, the dARTMAP and ARTMAP-IC are sensitive to both orders and provide satisfactory results in some cases only.
- Predictors based on the three models discussed here have a very short training period, in contrast to the MPL neural networks. The three models also consume a negligible small amount of memory, which makes them applicable in large scale prediction tasks.

All conclusions above feature the Fuzzy ARTMAP model as most stable and suitable for business applications like those discussed here.

Bibliography

Brierly, P. (2000) 'Characteristics of caravan insurance policy owners', [online] available at http://www.liacs.nl/~putten/library/cc2000/ brierl~1.pdf, 2000

Candocia, F. (2004) 'EEL 6825 Pattern Recognition', [online] available at http://www.cise.ufl. edu/~bfeng/eel6825/ 6825_pattern.htm, 2004

Carpenter, G.A. (1996). 'Distributed ART networks for learning, recognition, and prediction.' Proceedings of the World Congress on Neural Networks (WCNN'96), pp. 333–344.

Carpenter, G.A., Milenova, B., & Noeske, B. (1998a). 'dARTMAP: A neural network for fast distributed supervised learning.' Neural Networks, 11, 793-813. Technical Report CAS/CNS TR-97-026, Boston, MA: Boston University.

Carpenter, G. A. and Markuzon, N. (1998b), 'ARTMAP-IC and Medical Diagnosis: Instance Counting and Inconsistent Cases', Neural Networks, 11:2, 323-336.

Crocoll, W. (2000) 'Artificial Neural Network Portion of Coil Study', [online] available at http://www.liacs.nl/~putten/library/cc2000/crocol~1.pdf

Elkan, C. (2001) 'Magical Thinking in Data Mining: Lessons From CoIL Challenge 2000.' Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining (KDD'01), pp. 426-431.

Grossberg, S. (1976) Adaptive pattern classification and universal recoding. II: Feedback, expectation, olfaction, and illusions. Biological Cybernetics, 23, 187–202, 1976.

Grossberg S. (1980). 'How does a brain build a cognitive code?', Psychological Review, 87, 1–51., 1980

Shtovba, S. and Mashnitskiy, Y. (2000) 'The Backpropagation Multilayer Feedforward Neural Network Based Competition Task Solution', [online] available at http://www.liacs.nl/ ~putten/library/cc2000/shtob~1.pdf, 2000

Van Der Putten, P. and Van Someren M. (eds).(2000) CoIL Challenge 2000: The Insurance Company Case. Published by Sentient Machine Research, Amsterdam. Also a Leiden Institute of Advanced Computer Science Technical Report 2000-09.

Vesanto, J. and Sinkonen, J. (2000) Submission for the Coil Chalange 2000, [online] available at http://www.liacs.nl/~putten/ library/cc2000/vesant~1.pdf, 2000

Author's Information

Anatoli Nachev – Information Systems, Dept. of A&F, NUI Galway, Ireland; e-mail: anatoli.nachev@nuigalway.ie

INTELLIGENT MODEL OF USER BEHAVIOR IN DISTRIBUTED SYSTEMS

Andrii Shelestov, Serhiy Skakun, Olga Kussul

Abstract: We present a complex neural network model of user behavior in distributed systems. The model reflects both dynamical and statistical features of user behavior and consists of three components: on-line and off-line models and change detection module. On-line model reflects dynamical features by predicting user actions on the basis of previous ones. Off-line model is based on the analysis of statistical parameters of user behavior. In both cases neural networks are used to reveal uncharacteristic activity of users. Change detection module is intended for trends analysis in user behavior. The efficiency of complex model is verified on real data of users of Space Research Institute of NASU-NSAU.

Keywords: distributed systems, user behavior model, neural networks.

Introduction

At present the solution of complex large-scale problems arising in the areas of Earth observations from space [Shelestov, *et al.,* 2006], [Fusco, 2006], [Fusco, *et al.*, 2003], high-energy physics [Holtman, 2001], bioinformatics [Peltier, *et al.,* 2002], astronomy [Annis, *et al.,* 2002] is impossible without use of distributed computer systems. (e.g. Grid systems). Many tasks, such as computing and data resources sharing, distributed data processing, data storage, archiving and transfer are relied on them. One of the important challenges in the development of heterogeneous distributed infrastructure is the security provision. For this purpose many problems must be solved such as user authentication, authorization, rights delegation, etc. This can be done by using, for example, Globus Grid Security Infrastructure (GSI) [Foster, *et al.,* 1998] which is an extension of the Public Key Infrastructure [IETF], [Adams and Lloyd, 2002]. On the other hand, there are many monitoring tools intended for resources state and jobs monitoring (e.g. GridICE (http://gridice.forge.cnaf.infn.it/), MOGAS (http://ntu-cg.ntu.edu.sg/pragma/index.jsp)), but the do not provide monitoring of users' activities in order to detect anomalies and potential intrusions. Though, many sources report that the majority (80%) of information security incidents is perpetrated by insiders [Tulloch, 2003]. Hence, the problem of monitoring and detection of malicious user activity in distributed computer system is an important issue.

Related Works

Nowadays different methods and approaches are applied for the analysis of user activity. They are mostly based on the analysis and exposure of regularities and common actions in user behavior for automation, prediction, anomaly detection, etc. Data that are used for model construction possess individual features that define user behavior. For data processing different approaches can be applied such as history-matching methods and machine-learning methods.

In general, creation of user behavior model involves the following steps: data collection and data pre-processing, when useful information about user activity is collected from log-files; data processing, when feature extraction is done to represent data, and dimension reduction methods are used to reduce the size of the data; application of different techniques to obtain interesting characteristics of user behavior; interpretation of the results.

Among the existing approaches to user activity analysis we may consider so called Personal Security Programs that are used by commercial companies to monitor the activity of their employees. The results of such monitoring can be used to reveal malicious users in the case of information leakage, or to find out whether users use computers for their personal purposes. For example, such programs as PC Spy (www.softdd.com/pcspy/index.htm), Inlook Express (www.jungle-monkey.com), Paparazzi (www.industar.net) allow to capture and save screen images (screenshots) showing exactly what was being viewed by users. All screens can be captured, including Web pages, chat windows, email windows, and anything else shown on the monitor. However, these programs have some disadvantages; among them are high volume of stored information and manual configuration of snapshots frequency. That is, if the frequency is low it would be rather difficult to find out something abnormal in user activity. Otherwise, a lot of the data should be stored.

Another example refers to Intrusion Detection Systems (IDS), particularly anomaly detection in computer systems. Usually, a model of normal user behavior is firstly created, so during monitoring any abnormal activity can be regarded as potential intrusion. Different approaches are applied to the development of anomaly detection systems: statistical methods [Javitz and Valdes, 1991], expert systems [Dowell and Ramstedt, 1990], finite automata [Kussul and Sokolov, 2003], neural networks [Cannady and Mahaffey, 1998], [Reznik, *et al.*, 1999], [Skakun, *et al.*, 2005], agent-based systems [Skakun, *et al.*, 2005], [Gorodetski, *et al.*, 2001], rule-based networks, genetic algorithms, etc.

It is worth mentioning that existing approaches do not provide adequate description of user behavior. There exist methods that exhibit only dynamical features of user behavior, and do not consider statistical properties, and vice versa. This paper describes a complex model of user behavior in distributed systems. The model consists of three components: on-line model, off-line model and change detection module. The use of on-line and off-line models allows the reflection of both dynamical and statistical features of user's activity. In order to provide adaptive and robust approach for the analysis and generalization of data obtained from user activity neural networks are applied. The proposed approach is verified on real data gathered during the work of users on the resources of Space Research Institute of NASU-NSAU.

Complex Model of Users Behavior in Distributed Systems

For adequate description of user behavior in distributed systems we propose complex model that consists of the following components:

- on-line model that describes user's activity during its work by predicting his actions;
- off-line model that is based on the analysis of statistical data acquired during user's work;
- change detection module intended for detection of trends in user's activity.

The proposed structure of complex model is depicted in Fig. 1.

For prediction of user actions neural network is used. The use of neural network is motivated by the fact that user behavior represents a complex non-linear process and by the need to reveal regularities in it. As a neural network paradigm, we use feed-forward neural network trained by means of back-propagation algorithm [Haykin, 1999]. Therefore, for each user a neural network is built, and trained in such way to predict user actions. The result of neural network work after completion of *i*-1 actions by user during session s_t is given by the following equation:

$$\widehat{c_i^{s_t}} = F(\mathbf{x}_i),\ \mathbf{x}_i = \left(c_{i-1}^{s_t}, c_{i-2}^{s_t}, \ldots, c_{i-m}^{s_t}\right),$$

where F — non-linear transformation performed by neural network; $\mathbf{x}_i$, $\widehat{c_i^{s_t}}$ — neural network input and output respectively; $c_i^{s_t}$ — the number of *i*-th user action during session s_t; m — number of previous actions used to predict the next one.

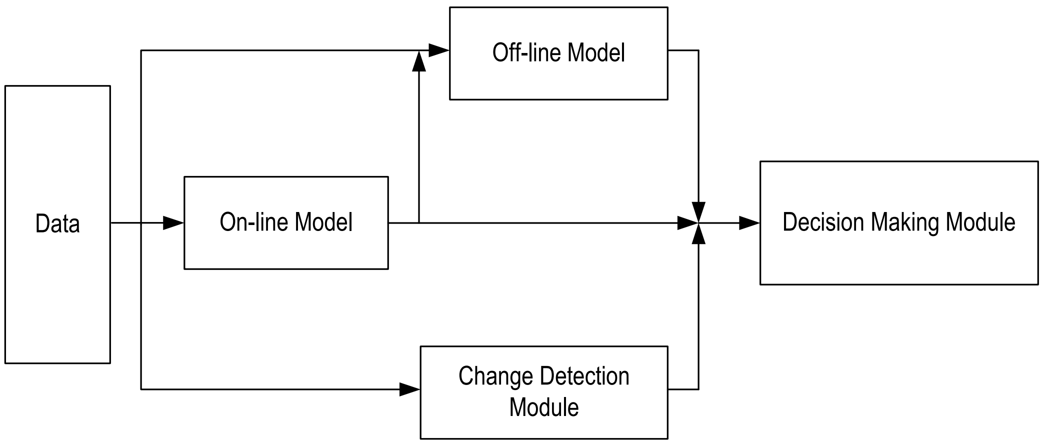


Fig. 1. Structure of complex neural network model of user behavior

On-line model describes dynamical features of user behavior by predicting user actions based on previous ones. The decision about user behavior (normal/abnormal) is based on the number of correctly predicted user actions by neural network (i.e. when $\widehat{c_i^{s_t}}$ is equal to $c_i^{s_t}$). Construction of on-line model must also take into account the possibility that user behavior will be changing in the course of time. In order to provide adaptation of model to these changes (i.e. retraining neural network) change detection module is used.

On contrast, *off-line model* is based on the use of statistical (integral) parameters obtained during user behavior. The following set of characteristics about user behavior was taken:

$$\left\{n_{s_t}, o_{s_t}, h_{s_t}, d_{s_t}, s_{s_t}\right\}, \tag{1}$$

where n_{s_t} — number of actions performed by user; o_{s_t} — results of on-line model use, i.e. the number of correctly predicted user actions; h_{s_t} — user login host; d_{s_t} — user session duration time; s_{s_t} — the time of user session start.

This set is used as input feature to neural network for detection of normal/abnormal user activity. As in the case of on-line model, for each user feed-forward neural network is trained with back-propagation algorithm in order to distinguish normal and abnormal user behavior. The expected output of neural network during training is binary, i.e. 1 corresponds to normal behavior and 0 corresponds to anomaly. The neural network output is defined as follows:

$$\Delta_{s_t} = F(x_{s_t}), \quad \mathbf{x}_{s_t} = (n_{s_t}, o_{s_t}, h_{s_t}, d_{s_t}, s_{s_t}),$$

where *F* — non-linear transformation performed by neural network; $\mathbf{x}_{s_t}$, Δ_{s_t} — neural network input and output respectively; n_{s_t} , o_{s_t} , h_{s_t} , d_{s_t} s_{s_t} are defined by (1).

If an input to neural network is an independent sample (that was not used during training process), the corresponding output Δ_{s_t} will lie in the range [0; 1], and provide probability of user normal activity (higher values correspond to normal user behavior).

The user behavior does not represent stationary process, and it will be changing in the course of time (as a rule during 2-3 months). This can be caused by different reasons, e.g. due to software version changes, new tasks accomplishment. That is why, complex model is required to include *change detection module* that will detect trends in user behavior.

Let *A* be the alphabet of user actions, i.e. the set of all actions performed by user during set of sessions $\left\{s_t\right\}_{t=1}^{T}$. We assume that user behavior has not been changed during this time of work. Let *N* be the number of actions in alphabet *A*, and each action has a number from 1 to *N*. The number *N*+1 will be reserved to new actions that were not performed during sessions $\left\{s_t\right\}_{t=1}^{T}$. Let s_t (t>T) be the current session of user work with the following actions performed $\mathbf{c}^{s_t} = \left(c_1^{s_t},\ c_2^{s_t},\ \ldots,\ c_{N_{s_t}}^{s_t}\right)$. Then $c_i^{s_t} = N+1$, if $c_i^{s_t} \notin \{1,...,N\}$. In order to detect changes in user behavior after session s_t we construct vector **g**(s_t) with the following components:

$$g_j(s_t) = \begin{cases} 1, \text{ if exists such } k = \overline{1,\ N_{s_t}}, \text{ that } \ c_k^{s_t} = j,\ j \in A \\ 0, \text{otherwise} \end{cases} .$$

That is, if an action was performed during session s_t, then corresponding component of vector **g**(s_t) is equal to 1, otherwise is equal to 0. Then the obtained vector **g**(s_t) is compared in pairs with vectors obtained during previous sessions s_{t-1}, s_{t-2}, …, s_{t-l}. As a measure of comparison Hamming distance is applied:

$$\aleph(\mathbf{g}(s_{t'}),\ \mathbf{g}(s_{t''})) = \sum_{j=1}^{N} \chi(g_j(s_{t'}),\ g_j(s_{t''})),$$

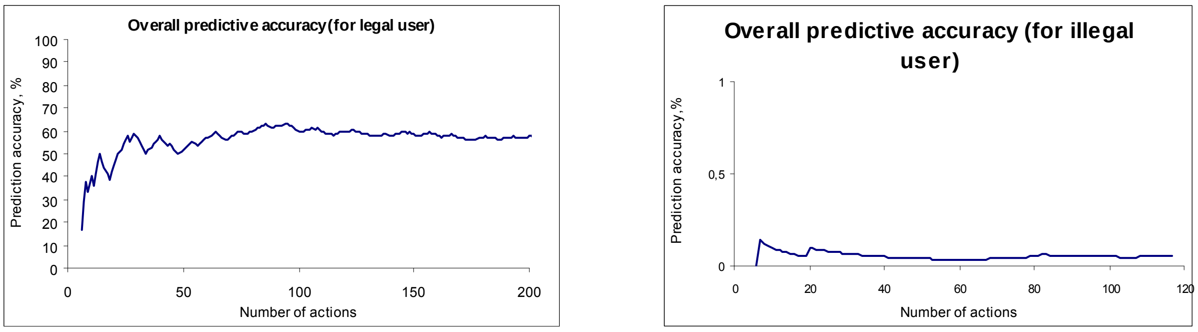
where $\chi(g_j(s_{t'}),\ g_j(s_{t''})) = \begin{cases} 1, & \text{if} \quad g_j(s_{t'}) \neq g_j(s_{t''}) \\ 0, & \text{otherwise} \end{cases}$. That is, $\aleph$ corresponds to number of components of two vectors that are different. As a result of comparisons we will obtain l values, following which we average and normalize on N: $H_t = \frac{1}{N}\left(\frac{1}{l}\sum_{k=1}^{l}\aleph(\mathbf{g}(s_t),\ \mathbf{g}(s_{t-k}))\right)$. If user behavior has not changed, then vector $\mathbf{g}(s_t)$ would not differ considerably from vectors for previous sessions. Hence, H_t would be below some threshold H_*: $H_t < H_*$. And vice versa, if anomaly occurred, vector $\mathbf{g}(s_t)$ would differ considerably from vectors for previous sessions and H_t would be under some threshold H^*: $H_t > H^*$. If $H_t \in (H^*; H^*)$, then a natural changes in user behavior took place.

Description of Experiments

Different experiments were run to demonstrate the efficiency of both on-line and off-line models and change detection module. For this purpose data needed for neural network training were acquired during a real work of users on the resources of Space Research Institute of NASU-NSAU.

Experiments for on-line model. For on-line model log files were transformed into format suitable for neural network. That is, for each user an alphabet of user actions was created, and each action was assigned an identifier (decimal number). For neural network input a binary coding was applied (7 bits per action). Feed-forward neural network was used to predict user actions based on 5 previous ones (the value of 5 previous actions was derived using autocorrelation function for sequence of user actions). Thus, the dimension of input data space was 35. In turn, for output data decimal coding was applied, and the dimension of output data space was 1. As to neural network architecture, we used neural network with 3 layers: input layer with 35 neurons, hidden layer with 35 neurons, and output layer with 1 neuron.

Then all data were randomly mixed and divided into train and test sets (70% for training and 30% for testing). Results of neural network work on test data showed that overall predictive accuracy (that is, the number of correctly predicted actions divided by total number) for different users varied from 33% to 59% (an example of overall predictive accuracy variations within number of actions is depicted in Fig. 2,a). To demonstrate that neural network was able to distinguish one user from another we run so called cross experiments. It was done in two ways. First one consisted in the following: data obtained during the work of one user (name him illegal user) were put to neural network trained for another (legal user). In such a case, overall predictive accuracy of neural network hardly exceeded 5% (it is shown on Fig. 2,b an example where overall predictive accuracy was 0,05%). Such experiment modeled the situation when illegal user logged on and begun to work under account of another user.



(a) (b)

Fig. 2. Predictive accuracy for: (a) legal user; (b) illegal user

The second method of cross experiments was done by inserting data of illegal user into data of legal one. This experiment modeled the situation when intruder begun to work under account of another user already logged on. In such a case, we used short-time predictive accuracy to measure the number of correctly predicted actions (this measure takes into account only last actions performed by user, for example, twenty last actions). Variations of short-time predictive accuracy for both legal and illegal user are depicted in Fig. 3. It is evident that short-time predictive accuracy for illegal user is considerably less then for legal one.

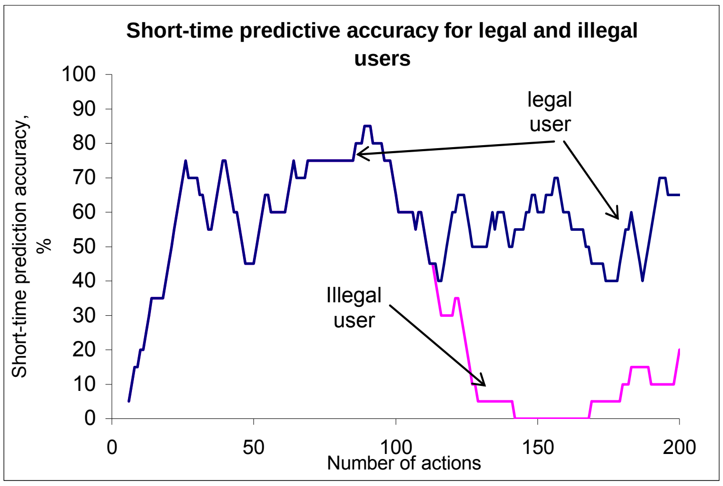


Fig. 3. Short-time predictive accuracy

Experiments for off-line model. Considering off-line model, all needed statistical data were obtained from log files. Then they were encoded, divided into training and test sets, and input to neural network. Results of neural network work on test data gave 80% accuracy of correct user behavior identification. That is, experiments showed that off-line model was able to distinguish normal and abnormal (anomalous) user behavior.

Experiments for change detection module. In order to verify change detection module first H_t was estimated for those user sessions when the use behavior was normal. Derived values of H_t did not differ considerably and lied in the range (0; H_*). (For each user different values of threshold H_* were obtained and varied from 0,1 to 0,17). For anomaly behaviour modelling cross-experiments were run. That is, vector $\mathbf{g}(s_t)$ was calculated for sessions of another user and compared with vectors obtained for a given user. Obtained values of H_t considerably increased (average in 2,5 times).

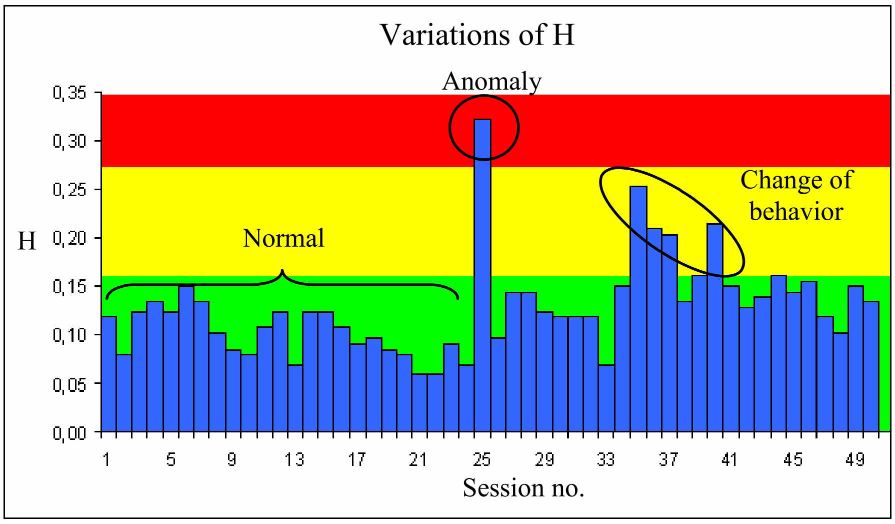


Fig. 4. Variations of value H_t depending on user behavior

In order to model natural changes in user behavior (that do not correspond to anomaly) for each user components of vector $\mathbf{g}(s_t)$ were randomly changed (with probability 0,25). In this case the value H_t increased in 2 times, and then decreased to ordinary level.

The variations of value H_t for typical user depending on user behavior are depicted in Fig. 4. Sessions #1-24 correspond to normal behavior, and H_t lie in the range (0; 0,15). When data of another user were inserted (session #25) H_t increased up to 0,32. When we modeled natural change of user behavior (sessions #35-50) H_t increased up to 0,25, and then decreased to ordinary level.

Therefore, experimental results showed the possibility of proposed complex neural network model to distinguish with confidence normal and abnormal (anomalous) user behavior.

Conclusions

In this paper we proposed a complex model of user behavior in distributed system. In order to adequately describe different features of user behavior the model consists of three components: on-line model considers dynamics of user behavior by predicting user actions; off-line model is based on the analysis of statistical parameters, and change detection module that is intended for detection of trends in user's activity. In contrast to existing methods the proposed model enables complex analysis of user behavior both during its work (in real-time) and after user's work completion (in off-line mode). The use of neural network provides intelligent approach to analysis and generalization of data acquired during user activity. In order to demonstrate efficiency of complex model different experiments were run on real data obtained during user work on resources of Space Research Institute of NASU-NSAU. The results of experiments showed applicability of the proposed approach.

Acknowledgement

This research is supported by INTAS-CNES-NSAU project "Data Fusion Grid Infrastructure", Ref. No 06-1000024-9154.

Bibliography

[Adams and Lloyd, 2002] Adams C., Lloyd S. "Understanding PKI: Concepts, Standards, and Deployment Considerations", 2nd ed. Addison-Wesley, 2002.

[Annis, *et al.*, 2002] Annis J., Zhao Y., et al., "Applying Chimera Virtual Data Concepts to Cluster Finding in the Sloan Sky Survey," Technical Report GriPhyN-2002-05, 2002.

[Cannady and Mahaffey, 1998] Cannady J., Mahaffey J. "The Application of Artificial Neural Networks to Misuse Detection: Initial Results", In Proc. of the 1998 National Information Systems Security Conf. (NISSC'98), Arlington, VA, 1998.

[Dowell and Ramstedt, 1990] Dowell C., Ramstedt P. "The ComputerWatch data reduction tool", In Proc. 13th National Computer Security Conf., 1990, pp. 99–108.

[Foster, *et al.*, 1998] Foster I., Kesselman C., Tsudik G., Tuecke S. "A Security Architecture for Computational Grids", In ACM Conf. on Computers and Security, 1998, pp. 83-91.

[Fusco, 2006] Fusco L. "Earth Science GRID on Demand", Presented on CEOS WGISS-21 GRID Task Team meeting, Budapest, Hungary, May 2006.

[Fusco, *et al.*, 2003] Fusco L., Goncalves P., Linford J., Fulcoli M., Terracina A., D'Acunzo G. "Putting Earth-Observation on the Grid", ESA Bulletin, 2003, 114, pp. 86-91.

[Gorodetski, *et al.*, 2001] Gorodetski V., Karsaev O., Khabalov A., Kotenko I., Popyack L., Skormin V. "Agent-based model of Computer Network Security System: A Case Study", In Proc. of the Int. Workshop 'Mathematical Methods, Models and Architectures for Computer Network Security', Lecture Notes in Computer Science, 2052, Springer Verlag, 2001, pp. 39-50.

[Haykin, 1999] Haykin S. "Neural Networks: a comprehensive foundation", Upper Saddle River, New Jersey: Prentice Hall, 1999, 842 p.

[Holtman, 2001] Holtman K., "CMS Requirements for the Grid", In Proc. of the Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP2001), 2001.

[IETF] IETF, Public-Key Infrastructure (pkix) Charter.

[Javitz and Valdes, 1991] Javitz H., Valdes A. "The SRI IDES statistical anomaly detector", In: Proc. IEEE Symp. on Research in Security and Privacy, 1991, pp. 316–326.

[Kussul and Sokolov, 2003] Kussul N., Sokolov A. "Adaptive anomaly detection of user behaviour using Markov chains with variable order", J. of Automation and Control, Vol. 4, pp. 83-88. (in Russian)

[Peltier, *et al.*, 2002] Peltier S.T., et al. "The Telescience Portal for Advanced Tomography Applications", J. of Parallel and Distributed Computing: Computational Grid, 2002, 63(5), pp. 539-550.

[Reznik, *et al.*, 1999] Reznik A., Kussul N., Sokolov A. “Identification of user activity using neural networks”, J. of Cybernetics and Computer Science, 1999, No. 123, pp. 70–79. (in Russian)

[Ryan, *et al.*, 1998] Ryan J., Lin M-J., Miikkulainen R. “Intrusion Detection with Neural Networks”, Advances in Neural Information Processing Systems, Cambridge, MA: MIT Press, 1998, pp. 943–949.

[Shelestov, et al., 2006] Shelestov A.Yu., Kussul N.N., Skakun S.V. “Grid Technologies in Monitoring Systems Based on Satellite Data”, J. of Automation and Information Science, 2006, Vol. 38, Issue 3, pp. 69-80.

[Skakun, *et al.*, 2005] Skakun S.V., Kussul N.N., Lobunets A.G. “Implementation of the Neural Network Model of Users of Computer Systems on the Basis of Agent Technology”, J. of Automation and Information Sciences, 2005, Vol. 37, Issue 4, pp. 11-18.

[Tulloch, 2003] Tulloch M. “Microsoft Encyclopedia of Security”, Redmond, Washington: Microsoft Press, 2003, 414 p.

Authors' Information

Andrii Yu. Shelestov – PhD, Senior Researcher, Department of Space Information Technologies and Systems, Space Research Institute of NASU-NSAU, Glushkov Ave 40, Kyiv-187, 03650 Ukraine, e-mail: inform@ikd.kiev.ua.

Serhiy V. Skakun – PhD, Research Assistant, Department of Space Information Technologies and Systems, Space Research Institute of NASU-NSAU, Glushkov Ave 40, Kyiv-187, 03650 Ukraine, e-mail: inform@ikd.kiev.ua.

Olga M. Kussul – BSc, Physics and Technology Institute, National Technical University “KPI”, Peremoga Ave 37, Kyiv-056, 03056 Ukraine, e-mail: olgakussul@gmail.com.

DATA ASSIMILATION TECHNIQUE FOR FLOOD MONITORING AND PREDICTION

Natalia Kussul, Andrii Shelestov, Serhiy Skakun, Oleksii Kravchenko

Abstract. This paper focuses on the development of methods and cascade of models for flood monitoring and forecasting and its implementation in Grid environment. The processing of satellite data for flood extent mapping is done using neural networks. For flood forecasting we use cascade of models: regional numerical weather prediction (NWP) model, hydrological model and hydraulic model. Implementation of developed methods and models in the Grid infrastructure and related projects are discussed.

Keywords: Grid computing, remote sensing, modeling, international Grid projects.

Introduction

Nowadays Grid represents a powerful technology for solution of complex large-scale problems arising in such areas as high-energy physics [1], gravitational-wave physics [2], astronomy [3], bioinformatics [4], Earth Observations (EO) [5, 6, 7], etc. To this end the efficiency of application of Grids in different domains was demonstrated in numerous projects: EGEE [8], GriPhyN [9], DataGrid [10], CrossGrid [11] etc. The advantages of Grids come from its ability to integrate heterogeneous computational and informational resources managed by different distributed organizations [12]. It is particularly important for EO domain where one needs to manage with large amounts of data acquired from different satellites in different spectral bands that need to be integrated with aerial and in-situ components and maps; complex workflows; distributed archives and so on [5, 7].

Today remote sensing data from space are widely used for natural hazards and environmental monitoring, land use management, agriculture, etc. Floods are among the most devastating natural hazards in the world, affecting more people and causing more property damage than any other natural phenomena [13]. The dramatic floods of Central and Eastern Europe in August 2002 and Spring 2001 and 2006 emphasize the extreme in climatic variations. Ukraine is also vulnerable to floods as, in particular in the Carpathian region where it occurs almost every year. During the floods in 2001, 9 people were killed and 12000 citizens were evacuated, more than 1500 buildings were destroyed, and more than 30000 buildings were flooded. That is why, flood monitoring and forecasting is a task of great importance. This task is also included as priority one to GEO 2007-2009 Work Plan in the framework of development of Global Earth Observation System of Systems (GEOSS) [14]. Efficient monitoring and prediction of floods and risk management is impossible without the use of EO data from space. Satellite observations enable acquisition of data for large and hard-to-reach territories, as well as continuous measurements.

One of the important problems associated with flood monitoring is flood extent extraction, since it is impractical to acquire the flood area through field observations. Flood extent can be used for hydraulic models to reconstruct what happened during the flood and determine what caused the water to go where it did, for damage assessment and risk management, and can benefit to rescuers during flooding.

In this paper a new neural network approach for flood extent extraction from space-borne Synthetic Aperture Radar (SAR) data is proposed. First, image segmentation is done by means of the self-organizing Kohonen maps. The further classification of the image is done by assigning each pixel of the image flood extent/not flood extent class using additional data from Landsat-7 ETM+ and Corine 2000 Land Cover.

As to flood forecasting, we use cascade of models: regional numerical weather prediction (NWP) model implemented with the Weather Research and Forecast (WRF) model, hydrological model (in particular, Hydrological Simulation Program (HSPF) and Noah Land Surface Model (LSM)) and hydraulic model (in particular, FESWMS).

We finish the paper with the description of implementation of our methods and models in Grid environment and related international projects.

Application of SAR data from space for flood extent extraction

SAR (synthetic aperture radar) measurements from space can provide valuable information to monitoring of flood events [15]. This is mainly due to the fact that SAR image acquisition is independent of daytime and weather conditions. The use of SAR data for flood extent mapping is motivated by the fact that smooth water surface provides no return to antenna in microwave spectrum and appears black in SAR imagery. Existing methods for flood extent mapping are primary based on the use of multitemporal technique (Fig. 1) [16], and pixel processing methods with threshold [17]. Though these methods are rather simple and quick, they posses some disadvantages: need of manual threshold selection and image segmentation, require expertise in visual interpretation of SAR images, require the use of complex models for speckle reduction, spatial connections between pixels not concerned.

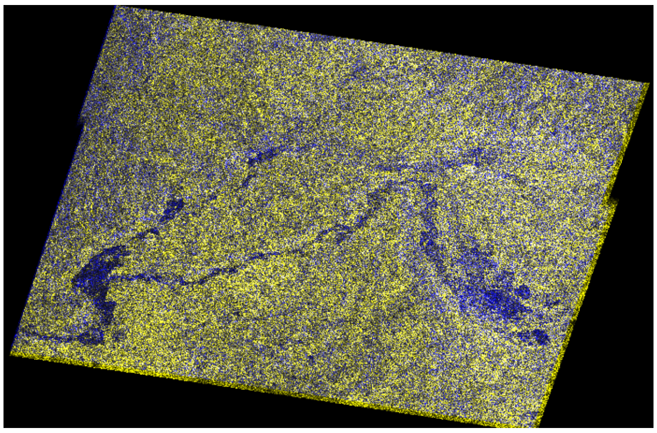


Fig. 1. Multitemporal image composed of two ERS-2 images acquired during flooding on Tisza River (10 March, 2001) and after flooding (14 April, 2001). R, G: March, 10, 2001; B: April,14,2001. Blue areas indicate flood extent. © ESA 2007.

To overcome these difficulties intelligent approach for image segmentation using artificial neural networks (NN)— self-organizing Kohonen's maps (SOMs) [18] — has been developed. This approach enables adaptive weights adjustment and considers spatial properties of pixels using sliding window. In turn, SOMs provide effective software tool for the visualization of high-dimensional data, automatically discover of statistically salient features of pattern vectors in data set, and can find clusters in training data pattern space which can be used to classify new patterns. The workflow for flood extent extraction based on SAR data consists of the following tasks:

1. Transformation of raw data to lat/long projection.
2. Image calibration.
3. Co-registration (to available Landsat-7 ETM+ data).
4. Image segmentation using SOMs.
5. Flood extent extraction: each neuron (cluster) is assigned flood extent/not flood extent class using additional data from Landsat-7 ETM+ and Corine 2000 Land Cover.

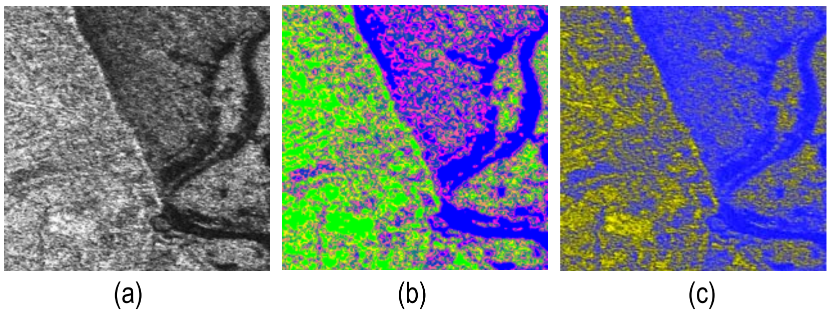


Fig. 2. Flood extent extraction process: (a) Raw ERS-2 image (b) Image segmentation using SOMs (c) Flood extent (marked with blue) © ESA 2007.

The results of application of the proposed approach to flood extent extraction using SOMs are depicted in Fig. 2. Figure 2(a) shows original ESA ERS-2 satellite image with flooded areas that absorb radar signal (these areas marked with black). Figure 2(b) shows results of image segmentation using SOMs which clearly reveals the existence of different clusters (marked with different colors). Landasat-7 ETM+ and Corine 2000 Land Cover were used to assign each neuron one of the classes: flood extent, not flood extent. The results of assignment are depicted in Fig. 2(c): blue color shows flooded areas or areas with strong presence of water, while non flooded area are marked with yellow color.

Data for this work were provided by the European Space Agency (ESA) within Category-1 project “Wide Area Grid Testbed for Flood Monitoring using Spaceborne SAR and Optical Data” (no. 4181).

Flood prediction methodology

Most flood events are driven by rapid development of runoff caused by intense precipitation or/and snowmelt. Therefore quantitative estimation of precipitation and snowmelt is crucial point in flood forecasting. To assess these phenomena hydrologists traditionally rely on conventional in-situ observations that were later expanded by meteorological radars and satellite-based precipitation estimates. The use of meteorological models was very limited due to their coarse resolution that allowed it to be used only for macroscale hydrological studies. However, with the progress in regional weather modeling and rapid increase of computation power it is become possible to utilize such models as additional source of precipitation.

To predict flooding parameters such as rivers stage/discharge and extents of flooded areas we use cascade of simulation models: regional numerical weather prediction (NWP) model, hydrological model and hydraulic model (fig. 3). This approach was modeled after successful work of Hluchy et. al. [19] considering Slovakian watersheds.

To obtain quantitative estimates of precipitation and other meteorological forcing in the Space Research Institute WRF (Weather Research&Forecasting) regional NWP model is used. WRF model is a joint development of a number of USA agencies and universities (http://wrf-model.org). This model was configured and adapted to the territory of Ukraine to run with spatial resolution of 10 km. Currently we routinely produce 72-hours weather forecasts every 6 hours. To drive regional model the additional weather forecasts from global NWP model are used. These data are required to specify external meteorological forcing as boundary conditions for regional weather model. Currently we use forecast frames produced by GFS (Global Forecast System) model operated by NCEP.

The one run of WRF model consists from the following steps (fig. 4):
— data download;
— data preprocessing, computation of forecast using WRF model and data postprocessing;
— visualization of results using mapping server.

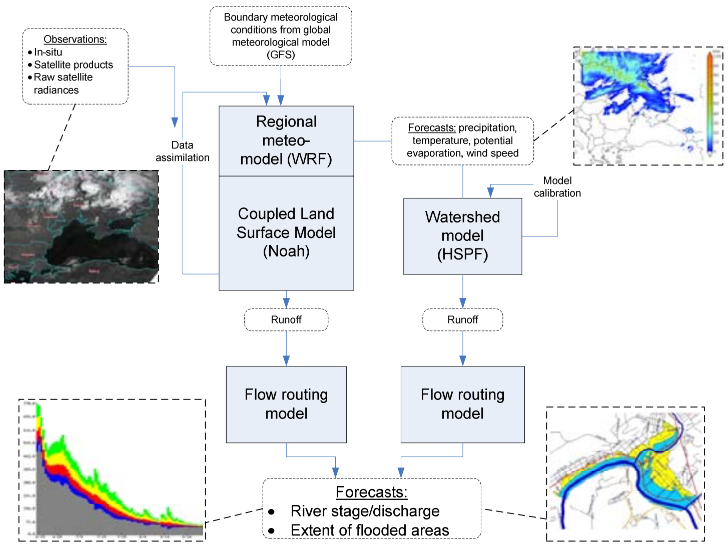

Observations:
• In-situ
• Satellite products
• Raw satellite radiances
Boundary meteorological conditions from global meteorological model (GFS)
Data assimilation
Regional meteo-model (WRF)
Coupled Land Surface Model (Noah)
Forecasts: precipitation, temperature, potential evaporation, wind speed
Model calibration
Watershed model (HSPF)
Runoff
Runoff
Flow routing model
Flow routing model
Forecasts:
• River stage/discharge
• Extent of flooded areas

Fig. 3. Simulation cascade to predict flood events

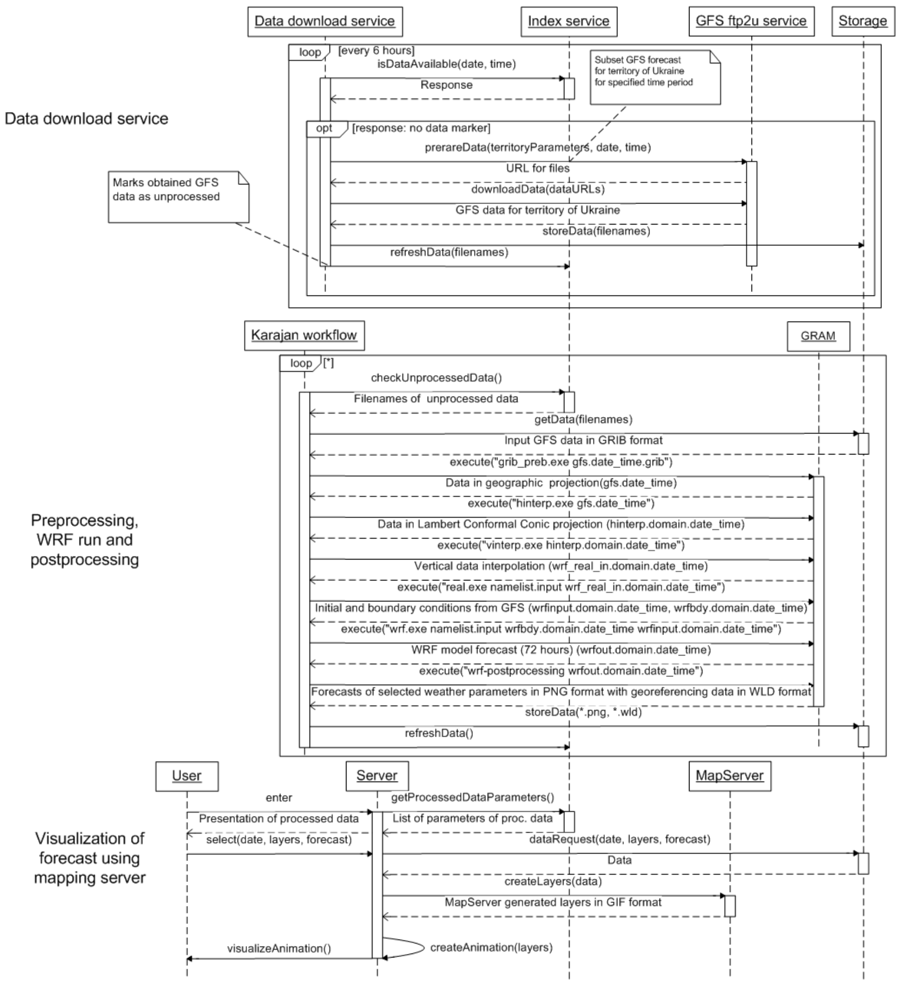

Data download service
Index service
GFS ftp2u service
Storage
loop [every 6 hours]
isDataAvailable(date, time)
Response
Subset GFS forecast for territory of Ukraine for specified time period
Data download service
opt [response: no data marker]
prerareData(territoryParameters, date, time)
URL for files
Marks obtained GFS data as unprocessed
downloadData(dataURLs)
GFS data for territory of Ukraine
storeData(filenames)
refreshData(filenames)
Karajan workflow
GRAM
loop [*]
checkUnprocessedData()
Filenames of unprocessed data
getData(filenames)
Input GFS data in GRIB format
execute("grib_preb.exe gfs.date_time.grib")
Data in geographic projection(gfs.date_time)
execute("hinterp.exe gfs.date_time")
Data in Lambert Conformal Conic projection (hinterp.domain.date_time)
Preprocessing, WRF run and postprocessing
execute("vinterp.exe hinterp.domain.date_time")
Vertical data interpolation (wrf_real_in.domain.date_time)
execute("real.exe namelist.input wrf_real_in.domain.date_time")
Initial and boundary conditions from GFS (wrfinput.domain.date_time, wrfbdy.domain.date_time)
execute("wrf.exe namelist.input wrfbdy.domain.date_time wrfinput.domain.date_time")
WRF model forecast (72 hours) (wrfout.domain.date_time)
execute("wrf-postprocessing wrfout.domain.date_time")
Forecasts of selected weather parameters in PNG format with georeferencing data in WLD format
storeData(*.png, *.wld)
refreshData()
User
Server
MapServer
enter
getProcessedDataParameters()
Presentation of processed data
List of parameters of proc. data
Visualization of forecast using mapping server
select(date, layers, forecast)
dataRequest(date, layers, forecast)
Data
createLayers(data)
MapServer generated layers in GIF format
visualizeAnimation()
createAnimation(layers)

Fig. 4. Sequence diagram for Numerical Weather Prediction service

Data download. To run WRF model it is required to obtain boundary and initial conditions for territory of Ukraine. This data can be extracted from GFS (Global Forecast System) model forecasts. To get required data the dedicated script was developed. This script downloads global forecasts every 6 hours from NOMADS servers (http://nomads.ncdc.noaa.gov/data.php). To decrease the data volume our script uses special web-service available to subset GFS data for territory of Ukraine. Obtained data is transferred to storage subsystem and marked as unprocessed (i.e. it has to processed by WRF model).

After new (unprocessed) GFS data has been obtained the Karajan script we have created initiates workflow for data preprocessing, running WRF and data postprocessing. GRAM service is used to execute jobs on Grid resources.

Data preprocessing is intended to transform downloaded data into format required to run WRF model. Input GFS data is delivered in GRIB format in geographical projection. This data is transformed into internal WRF format by grib_prep.exe command, warped into Lambert Conformal Conic projection (by hinterp.exe command) and vertically interpolated using vinderp.exe command. The result of transformations is stored in netCDF format. (grib_prep.exe, hinterp.exe and vinterp.exe commands are tools from WRF Standard Initialization (SI) package.) After that the real.exe command is used to produce initial and boundary conditions for WRF model run. real.exe inputs are GFS data in netCDF format and WRF configuration file (namelist.input).

Data processing step consists in performing WRF run using wrf.exe command to obtain meteorological forecast. This is the most computationally intensive task. At present we use model domain of 200x200 gridpoints with horizontal spatial resolution 10 km and produce 72 hours forecasts.

After WRF model run data postprocessing step is performed. Along this step for selected weather parameters and for each forecast frame (each 3 hours) a graphic representation (in PNG format) of spatial parameter distribution is created/ Additionally special files containing georeferencing information are created (files have *.wld extensions). The results of postprocessing are used to visualize WRF forecasts via mapping service. This service is available on http://dos.ikd.kiev.ua web page and provides to user animations of weather forecasts. To run service user could select it in services submenu of main menu on webpage. Service provides means to select a forecast time, forecast frames (up to 72 hours ahead) and weather parameters to display. Selected information packed into request to server. To process request

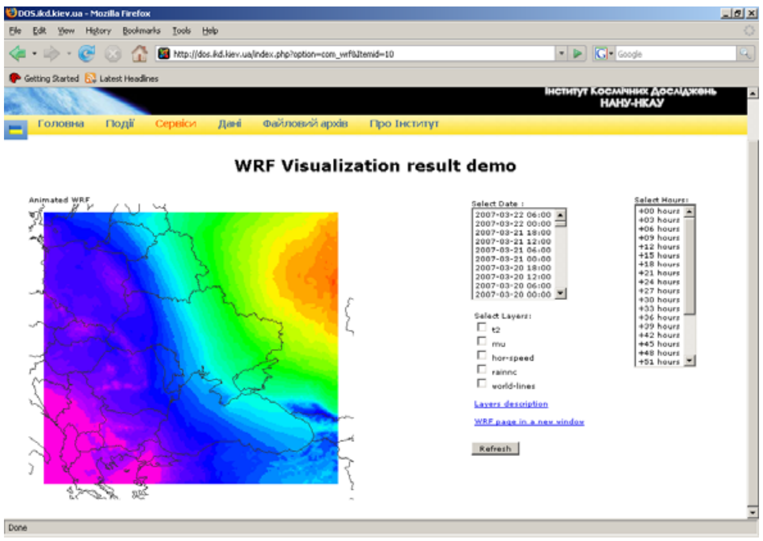


Fig. 5. Pressure forecast using WRF model

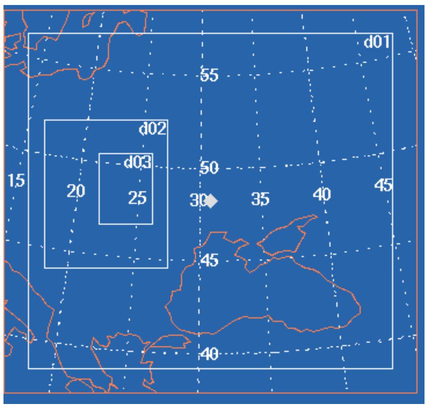


Fig. 6. Nested domains for Carpathian region to support high-resolution WRF forecasts

all required data (in PNG and WLD formats) is retrieved from storage subsystem and passed to mapping server to create maps. Maps are further processed by script to generate weather animation in GIF format. Finally this animation is presented at user side (fig. 5).

10 km ground spatial resolution is generally enough to support regional applications while it is clearly insufficient to represent small-scale features like precipitation or temperature distributions in mountainous of the Carpathian region. Ground resolution of about 1 km is required to cope with such life-threaten events such as flash floods. Regional models such as WRF are designed to operate at such resolutions, however direct increase of spatial resolution results in enormous increase of required computation performance. To solve this antinomy problem nesting approach is used. Within this method a set of nested model domains is used, each subsequent domain “zooms in” parent domain thus providing better resolution. Example of such approach applied to the Carpathians is shown in fig. 6.

Nesting approach can dramatically decrease required computation power to run model at high resolution however the power requirements remains enlarged when compared to the regular run. For instance to run model with configuration depicted at fig. 7 it is required 20 times more computational recourses compared to 10 km run. To run model with high spatial resolution we use HPC clusters SCIT-1 and SCIT-3 created in Institute of Cybernetics NAS of Ukraine.

SCIT-1 cluster consists of 24 dual processor nodes (total 48 Intel Xeon 2.66GHz processors) while SCIT-3 cluster consists of 75 nodes (total 300 Intel Xeon 3.0GHz cores). Use of clusters allowed to decrease computation time significantly while observing almost linear productivity growth within increasing number of computation nodes. For instance 8 nodes of SCIT-3 cluster give performance increase in 7.09 times (of 8.0 theoretically possible) when compared to one node. 64 nodes increase performance in 43.6 times (table 1, fig. 7).

Table 1. Computation time for 1 iteration of WRF model run using SCIT-3 cluster

Number of nodes	Number of processing cores	One iteration's computation time, s
1	4	4.36
2	8	2.49
4	16	1.22
8	32	0.615
16	64	0.35
32	128	0.18
64	256	0.1

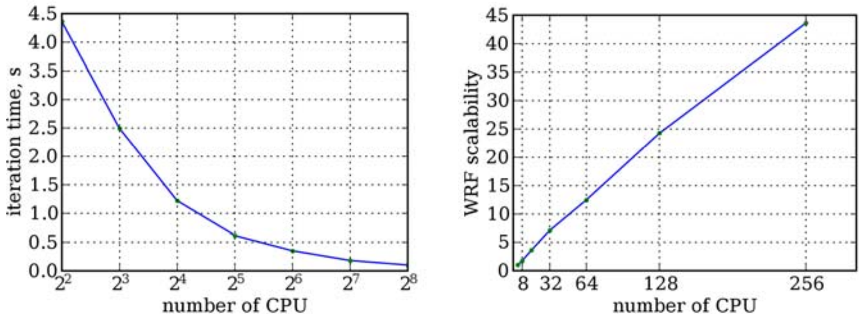


Fig. 8. WRF performance on SCIT-3 cluster: a) one iteration's computation time, b) acceleration of WRF model with respect to number of nodes

To simulate hydrological processes in the Carpathians watersheds two approaches are used: more traditional one using dedicated hydrological model and the second using coupled modeling. As dedicated hydrological model we use robust industry proven HSPF (Hydrological Simulation Program – Fortran, http://water.usgs.gov/software/hspf.html). HSPF is lumped model, i.e. within this type of models the whole watershed is divided into a number of sub-catchments that are considered hydrologically homogeneous. To run HSPF model several meteorological forcing parameters are required (at least precipitation and estimation of potential evapotranspiration). These data are obtained from high-resolution weather forecasts. Another approach we utilize is a coupled modeling. Within this approach we use Noah Land Surface Model (LSM) that is fully coupled with WRF model. Noah LSM is a distributed physically-based model that works with the same resolution as WRF model does.

Both hydrological model (HSPF and Noah) are able to compute quick response runoff to precipitation events. To route computed runoff and derive river stages and extent of flooded areas appropriate hydraulics models are involved.

Implementation in International Grid Projects

Space Research Institute of NASU-NSAU currently participate in two international Grid projects: Wide Area Grid (WAG) and INTAS-CNES-NSAU project “Data Fusion Grid Infrastructure”.

WAG project was initiated by French Space Agency CNES and CEOS (Committee on Earth Observing Satellites) Working Group on Information Systems and Services (WGISS). This project aims the development of “horizontal” infrastructure in order to integrate computational, human, intellectual, and informational resources of space agencies within large distributed system. Implementation of geospatial-related services and Grid-enable EO data archives are among the priority tasks of this project. To this end the following organizations participate in the WAG project: ESA, CNES (France), RSGS (China), SRI NASU-NSAU (Ukraine).

WAG infrastructure is also considered as CEOS WGISS contribution to GEOSS architecture implementation. Main objectives of the project, approaches and the state-of-the-art will be considered at the presentation.

The joint project of INTAS, CNES and NSAU “Data Fusion Grid Infrastructure” is targeting on solution of two tasks: flooding risk management and yield prediction. Grid approach is used for building large-scale computational system tuned for High Throughput Computing. Computational resources are used for automated adaptation of hydrometeorological model for different territories using evolutionary computations approach. The “Data Fusion Grid Infrastructure” consortium consists of CNES (France), Institute of Informatics of SAS (Slovak), Space Research Institute of RAS (Russia) and Space Research Institute of NASU-NSAU. CNES agency is developing technologies for Inter-Grid implementation. Institute of Informatics have developed cascade of models for flood monitoring for the territory of Slovakia and Space Research Institute of RAS have developed method for yield prediction. The role of Space Research Institute of NASU-NSAU is to develop Grid infrastructure to assimilate remote sensing data into modeling process and to perform model adaptation for the territory of Ukraine.

Conclusions

In this paper we proposed methods and models that exploit different kind of data to flood monitoring and forecasting. Information on flood extent extraction is derived from radar images by using neural network approach. The use of SOMs provide effective tool for automatically discover of statistically salient features of pattern vectors in data set. The advantages of our approach consist in the exploitation of textural information of the images (non-pixel based technique) and adaptive weights adjustment based on self-organization principle. The application of our approach on ERS-2 images during the flooding on Tisza river in spring of 2001 has demonstrated its efficiency.

Considering flood forecasting we propose to use cascade of NWP, hydrological and hydraulics models. To run these models high-performance computation need to be applied. The most advances were made on running WRF model to predict weather parameters and Noah model to assess soil moisture – one of the key inputs to hydrological and hydraulics models. Further studied will be carried out on hydrological and hydraulics models.

Acknowledgments

This work was supported by joint INTAS-CNES-NSAU project “Data Fusion Grid Infrastructure” (Ref Nr. 06-1000024-9154) and European Space Agency (ESA) Category-1 project “Wide Area Grid Testbed for Flood Monitoring using Spaceborne SAR and Optical Data” (#4181).

Bibliography

1. Holtman K.: CMS Requirements for the Grid. In: Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP2001) (2001).
2. Deelman E., Blackburn K., et al.: GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists. Presented at 11th Intl. Symposium on High Performance Distributed Computing (2002).
3. Annis J., Y. Zhao et al. : Applying Chimera Virtual Data Concepts to Cluster Finding in the Sloan Sky Survey. Technical Report GriPhyN-2002-05 (2002).
4. Peltier, S.T., et al.: The Telescience Portal for Advanced Tomography Applications. J. of Parallel and Distributed Computing: Computational Grid, 63(5), 539-550 (2002).
5. Fusco L.: Earth Science GRID on Demand. Presented at CEOS WGISS-21 GRID Task Team Meeting, Budapest, May (2006).
6. Fusco L., et al.: Putting Earth-Observation on the Grid. ESA Bulletin, 114, 86-91 (2003).
7. Shelestov A.Yu., Kussul N.N., Skakun S.V.: Grid Technologies in Monitoring Systems Based on Satellite Data. J. of Automation and Information Science, Vol. 38, Issue 3, 69-80 (2006).
8. EGEE, http://www.eu-egee.org.
9. GriPhyN, http://www.griphyn.org.
10. DataGrid, http://eu-datagrid.web.cern.ch/eu-datagrid.
11. CrossGrid, http://www.crossgrid.org.
12. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Int. J. of High Performance Computing Applications, 15 (3). 200-222 (2001).
13. Committee on Earth Observation Satellites Disaster Management Support Group: The Use of Earth Observing Satellites for Hazard Support: Assessments & Scenarios, Final Report; National Oceanic & Atmospheric Administration, Department of Commerce, USA (2001).
14. GEO Work Plan 2007-2009, “Toward Convergence”.
15. Solheim, I., Solbo, S., Indregard, M., Lauknes, I.: User Requirements and SAR-Solutions for Flood Mapping. In: 4th International Symposium on Retrieval of Bio- and Geophysical Parameters from SAR Data for Land Applications, Innsbruck, Austria (2001).
16. ESA Earth Watch, http://earth.esa.int/ew/floods/.
17. Yang, C., Zhou, C., Wan, Q.: Deciding the Flood Extent with RADARSAT SAR Data and Image Fusion. In: 20th Asian Conference on Remote Sensing (1999).
18. Haykin S.: Neural Networks: A Comprehensive Foundation. Upper Saddle River, New Jersey: Prentice Hall (1999).
19. Hluchy L., et al.: Collaborative environment for Grid-based flood prediction. Computing and Informatics, Vol. 24, 1001-1022 (2005).

Authors' Information

Natalia M. Kussul – Prof., Dr., Head of Department of Space Information Technologies and Systems, Space Research Institute of NASU-NSAU, Glushkov Ave 40, Kyiv-187, 03680 Ukraine, e-mail: inform@ikd.kiev.ua.

Andrii Yu. Shelestov – PhD, Senior Researcher, Department of Space Information Technologies and Systems, Space Research Institute of NASU-NSAU, Glushkov Ave 40, Kyiv-187, 03680 Ukraine, e-mail: inform@ikd.kiev.ua.

Serhiy V. Skakun – PhD, Research Assistant, Department of Space Information Technologies and Systems, Space Research Institute of NASU-NSAU, Glushkov Ave 40, Kyiv-187, 03680 Ukraine, e-mail: inform@ikd.kiev.ua.

Oleksii M. Kravchenko – Junior Research Assistant, Department of Space Information Technologies and Systems, Space Research Institute of NASU-NSAU, Glushkov Ave 40, Kyiv-187, 03680 Ukraine, e-mail: inform@ikd.kiev.ua.

A STATISTICAL CONVERGENCE APLICATION FOR THE HOPFIELD NETWORKS

Víctor Giménez-Martínez, Gloria Sánchez–Torrubia, Carmen Torres–Blanc

Abstract: When Recurrent Neural Networks (RNN) are going to be used as Pattern Recognition systems, the problem to be considered is how to impose prescribed prototype vectors $\xi^1, \xi^2, ..., \xi^p$, as fixed points. The synaptic matrix W should be interpreted as a sort of sign correlation matrix of the prototypes, In the classical approach. The weak point in this approach, comes from the fact that it does not have the appropriate tools to deal efficiently with the correlation between the state vectors and the prototype vectors The capacity of the net is very poor because one can only know if one given vector is adequately correlated with the prototypes or not and we are not able to know what its exact correlation degree. The interest of our approach lies precisely in the fact that it provides these tools. In this paper, a geometrical vision of the dynamic of states is explained. A fixed point is viewed as a point in the Euclidean plane R^2. The retrieving procedure is analyzed trough statistical frequency distribution of the prototypes. The capacity of the net is improved and the spurious states are reduced. In order to clarify and corroborate the theoretical results, together with the formal theory, an application is presented

Keywords: Learning Systems, Pattern Recognition, Graph Theory, Recurrent Neural Networks.

1. Introduction

As is well known, a RNN is a discrete time, discrete-valued dynamic system which at any given instant of time *t* is characterized by a binary *state vector* $x(t) = [x_1(t), ..., x_i(t), ..., x_n(t)] \in \{1, -1\}^n$. The behavior of the system is described by a *dynamic equation* of the type

$$x_i(t+1) = Sgn\left[\sum_{j=1}^{n} w_{ij} x_j(t) - \theta_i\right] \quad i = 1,2,...,n \tag{1}$$

A point x is a fixed point if all its components remain unchanged when (1) is applied. The aim is to get the network parameters, namely the synaptic matrix W and the threshold vector θ, for which the prototype vectors $\xi^1, \xi^2, ..., \xi^p$, are fixed points. In our approach $x(t) = \{0,1\}^n$ and, as the components of $x(t)$ may only be zero or one, we will refer to them as the null and unit components in $x(t)$. Associated to the network there will be a complete graph G, with n vertices $\{v_1, ..., v_n\}$, and *one* bi-directional *edge* a_{ij} for every possible pair of different vertices (1). Initially, at the *training stage* a null value w_{ij} is assigned to every edge a_{ij} in the graph; afterwards, when ξ^μ is presented to the net; the weight w_{ij} is updated by:

$$\Delta w_{ij} = \begin{cases} +1 & if \quad \xi_i^\mu = \xi_j^\mu = 1, \ i \neq j, \\ -1 & if \quad \xi_i^\mu = \xi_j^\mu = 0, \ i \neq j, \\ 0 & \qquad otherwise. \end{cases} \tag{2}$$

This idea may be much more easily understood using the next *graphical interpretation* of the training algorithm: At the first step, a null value is assigned to all the edges a_{ij}, then, when a learning pattern ξ^μ is acquired by the net, it is superposed over the graph *G.* The components $\{\xi_1^\mu, ..., \xi_n^\mu\}$, are going to be mapped over the vertices $\{v_1, ..., v_n\}$ of *G.* This mapping may be interpreted as a *coloring* of the edges in *G*, in such a way that, if $\xi_i^\mu = \xi_j^\mu = 1$, the edge a_{ij} (whose ending vertices are v_i and v_j) will be colored with a certain color, for example *red*. On the other hand, if $\xi_i^\mu = \xi_j^\mu = 0$, then a_{ij} will be colored with a different color, as for example *blue*. The rest of the edges in *G* remain uncolored. Once this coloring has been done, the value assigned over the, also *complete*, graph of *red* edges are positively reinforced and the value assigned over the edges of the *blue* graph are negatively reinforced. The value over the rest of the edges remains unchanged. Once the pattern ξ^μ is acquired, the colors are erased and we repeat the same color assignation with the next pattern to be acquired by the net, and so on. When every vector in the training pattern set has been integrated in the net, the training stage is finished, the *resulting graph G* has become edge-valued and its weight matrix is the *synaptic matrix W* of the net.

2. Parameters of the Net

According with the theorem proved in [1], If any set $\xi^1, \xi^2, \ldots, \xi^p$ of prototype vectors are acquired by the net, then for any possible four different components $"i", "j", "r"$ y $"s"$, then the relation: $w_{ij} + w_{rs} = w_{is} + w_{rj}$ is satisfied, and solving the system

$$\{w_{ij} = p_i + p_j,\ i \neq j \tag{3}$$

(in *n* unknown $p_1, p_2, \ldots, p_n$) a solution and only a solution is obtained [1].. The *training algorithm* could be revisited in order to obtain the *weight vector* $\vec{p}$ without the necessity of obtaining the *weight matrix W* first and then solving the system (3). In the *graphical interpretation* of the training algorithm, we may consider $\{p_i, p_j\}$ as the ending vertices of the generic edge a_{ij}. Just when the pattern ξ^μ has been acquired, if a_{ij} has been colored by red, its weight w_{ij} has been incremented by one. As $w_{ij} = p_i + p_j$ *j*, we may consider that p_i and p_j have both been incremented by "½". If n_1 and n_2 are the number of *unit* and *null* components of ξ^μ and if $\xi_i^\mu = 1$, then obviously the number of red edges with one end in p_i is equal to $(n_1 - 1)$. Consequently just when the pattern ξ^μ has been acquired p_i has been incremented by $½(n_1 - 1)$. In the same way, it could be proved that if $\xi_i^\mu = 0$, then p_i is incremented by $-½(n_0 - 1)$. The training algorithm in (3) could then be designed as follows:

$$\Delta p_i = \begin{cases} \frac{1}{2}(n_1 - 1) & if\ \ \xi_i^\mu = 1 \\ -\frac{1}{2}(n_0 - 1) & if\ \ \xi_i^\mu = 0 \end{cases} \tag{4}$$

(n_1 and n_0 are the number of *unit* and *null* components of ξ^μ). When all the learning patterns have been acquired, the training is finished. Considering ½ a scale factor and since the inner product $\xi^\mu . \xi^\mu$ is equal to n_1 and the inner product $\bar{\xi}^\mu . \bar{\xi}^\mu$ is equal to n_0,, it can be interpreted that when the learning pattern ξ^μ is acquired the *weight vector* $\vec{p}$ is modified as follows:

$$\vec{p} \leftarrow \vec{p} + (\xi^\mu\ .\ \xi^\mu\ - 1)\ \cdot I - (\bar{\xi}^\mu\ .\ \bar{\xi}^\mu\ - 1)\ \cdot I\text{, where } I \text{ is the unitary vector } (1,1,.., 1) \tag{5}$$

The above expression realizes the *updating* of the weights p_i, for *i* from *1* to *n*, when ξ^μ is acquired. The computational *time* of the training algorithm, is then highly optimized.

2.1. Energy

The state vector *x* at time *t* could also be interpreted as a *coloring* of the edges in *G,* but now this coloring is going to be used to retrieve the stored data. If the graph *G* is colored with the coloring associated with the pattern $x(t)$, it is easy to understand (taking into account how the training algorithm was designed), that the bigger the summation of all the edges in the *red graph* and the lower the summation of all the edges in the *blue graph* are, then the more correlated the pattern $x(t)$ must be, with those that were used during the training stage. So, if *W* is the weight matrix of *G*, the *energy point EP* of the net is defined as a pair of numbers. The first of them represents the summation of all the values on the edges of the *red* graph, and the second one represents the same summation, but on the *blue* ones. So if *G* is colored with the color associated to *x(t)*, then *{I (t), O(t)}* may be defined as the pair of quadratic forms:

$$\begin{cases} I(t) = \frac{1}{2} x(t) \cdot W \cdot x(t)^t \\ O(t) = \frac{1}{2} \bar{x}(t) \cdot W \cdot \bar{x}(t)^t \end{cases} \tag{6}$$

If n_1 is the number of *unit* components of $x(t)$ and n_0 is the number of the *null* ones (in other words n_1 is the *Hamming distance* from *x(t)* to the *zero* vector). By other hand, it is obvious [1], that if $\{I_i(t), O_i(t)\}$, is the *EP*, when $W = (w_{ij})$, is the matrix with all its values equal to zero except those in file or the row *I*, then

$$I_i(t) \begin{cases} n_1 - 1 & if\ x_i(t) = 1 \\ 0 & if\ x_i(t) = 0 \end{cases}, \quad and \quad O\ (t) \begin{cases} n_0 - 1 & if\ x_i(t) = 0 \\ 0 & if\ x_i(t) = 1 \end{cases} \tag{7}$$

So, if $i_1, i_2, \dots, i_{n_1}$ and $j_1, j_2, \dots, j_{n_0}$ are the places where the *unit* and *null* components of *x(t)* are respectively located, the equations (13) could be written as

$$I(t) = (n_1 - 1)\left(p_{i_1} + \dots + p_{i_{n1}}\right) \quad \text{and} \quad O(t) = (n_0 - 1)\left(p_{j_1} + \dots + p_{j_{n0}}\right) \tag{8}$$

Which means that

$$\frac{I(t)}{(n_1 - 1)} = \left(p_{i_1} + \dots + p_{i_{n1}}\right) \quad and \quad \frac{O(t)}{(n_0 - 1)} = \left(p_{j_1} + \dots + p_{j_{n0}}\right) \text{ and} \tag{9}$$

in other words

$$\frac{I(t)}{(n_1 - 1)} + \frac{O(t)}{(n_0 - 1)} = K \qquad being\ K = (p_1 + \dots + p_n) \tag{10}$$

As all *state vectors* $x(t)$ with the same *Hamming distance "i"* to the zero vector contains the same numbers n_1 and n_0 of *unit* and *null* components, the *energy points* $\{I(t), O(t)\}$, associated to all of them, will be placed in the same line r_i r_i, whose equation expressed in *(x,y)* is

$$r_i \equiv (n_0 - 1)x + (n_1 - 1)y - (n_0 - 1) \cdot (n_1 - 1) \cdot K = 0 \tag{11}$$

In other words, all the *EP´s* associated with state vectors with the same number of *unit* components are placed in the same line of the *energy field*, and the equation of this line is the one represented in (11). In this way the *state vector space* is classified in as many classes as the dimension *n* of the space.

2.2. Dynamics

On the other hand, and as we said in the introduction, the nature of the algorithm here proposed let to know how the value of $x(t)$ affects the whole energy of the state $x(t)$. We may define the relative weight of the neuron i when the net is in state x(t) as the contribution of this neuron to the component $I(t)$, if $x_i(t) = 1$; or as the contribution of this neuron to the component $O(t)$, if $x_i(t) = 0$. So, if $x_i(t) = 1$, we define the relative weight $w(t)$ of the neuron i when the net is in state $x(t)$ as:

$$w_i(t)=\frac{1}{n_1-1}+\frac{n_1-2}{n_1-1}\cdot\frac{p_i}{p.x(t)}=\frac{1}{x(t)\cdot x(t)-1}+\frac{x(t)\cdot x(t)-2}{x(t)\cdot x(t)-1}\cdot\frac{p_i}{p.xx(t)} \tag{12}$$

If in time *t* the state vector $x(t)$ is in class $[j]$, then for any *i* from *1* to *n*, the dynamic equation is defined as

$$x_i(t+1)=f_h\left[f_b(x_i(t))\ .\ (w_i(t)-\theta_j)\right] \tag{13}$$

where f_h is the Heaviside step function and f_b is the function defined as $f_b(x)=2x-1$, which achieves the transformation from the domain $\{0,1\}$ to the domain $\{1,-1\}$

It can also be stated that the sum of the relative weights $w_i(t)$ for the unit components of $x(t)$ is equal to *2*. The same could be proved for the null components. We have then that the relative weight vector $w(t)$ associated to any state vector $x(t)$ may also be interpreted as a sort of frequency distribution of probabilities [2]. The reason is that

$$\sum_{i=1}^{n}w_i(t)=4\ \Rightarrow\ \sum_{i=1}^{n}\tfrac{1}{4}w_i(t)=1 \tag{14}$$

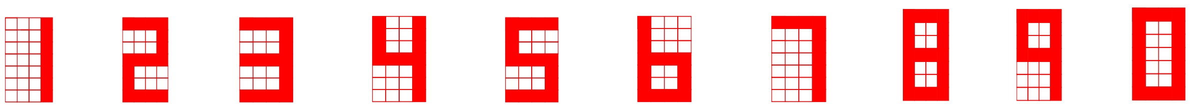
For any relative weight vector $w(t)$ The “uniform distribution vector” would be the one with all its components equal to $4/n$. For any state $x(t)$ we could then define its deviation $D(x(t))$ as

$$D(x(t))=\sqrt{\sum_{i=1}^{n}\left[x_i(t)-\frac{4}{n}\right]^2} \tag{15}$$

The deviation of a given vector to the prototypes has been used for avoiding the parasite fixed points.

3. Application

We take, as an example for validating the performance of the algorithm we propose, the problem of the recognition of the Arabian digits as the prototype vectors:



Where the dimension *n*, of the pattern space is *28*, and

$$\begin{cases}\xi^1=[0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1]\\ \xi^2=[1,1,1,1,0,0,0,1,0,0,0,1,1,1,1,1,1,0,0,0,1,0,0,0,1,1,1,1]\\ .\\ .\\ \xi^{10}=[1,1,1,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,,1,1,1,1]\end{cases}$$

and $p=1/14\ \{53, 25, 25, 53, 11, -73, -73, 39, 11, -73, -73, 39, 39, 25, 25, 67, -17, -73, -73, 53, -17, -73, -73, 53, 11, 11, 11, 67\}$. In figure 1 the reader may see the energy lines and theirs associated PE´s. The Arabian digits are in this way placed on the lines: r_7, r_{16}, r_{16}, r_{13}, r_{16}, r_{15}, r_{10}, r_{20}, r_{15}, r_{18}. The associated PE´s are $1/7\{1113,-3710\}$, $1/7\{3420,-2508\}$, $1/7\{4470,-3278\}$, $1/7\{3210,-3745\}$, $1/7\{4050,-2970\}$, $1/7\{2821,-2418\}$, $1/7\{2133,-4029\}$, $1/7\{5548,-2044\}$, $1/7\{4095,-3510\}$, $1/7\{4539,-2403\}$,

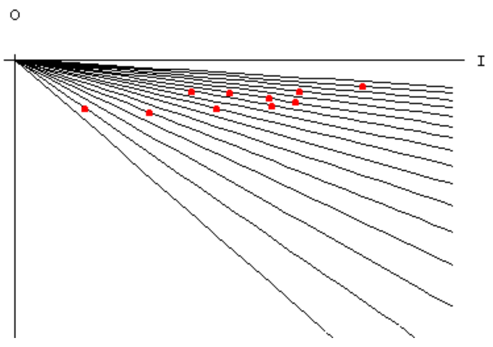


Figure 1 Arabian Digits Projections

The problem now is how to obtain in an adaptive way the capacity parameters $\theta_1, \theta_2, ..., \theta_{28}$, in order to obtain the Arabian digits as fixed points with the least number of parasitic points as possible.

When the dynamic equation in (13) is considered, a point $x(t)$ whose energy projection belongs to the r_j line, is a fixed point if, and only if, the (capacity) parameter θ_j is an upper bound for all the relative weights $w_i(t)$ associated to the components of $x(t)$. Once the training has finished, the relative weight vector of the prototypes could then be calculated. If the energy projection of the prototype ξ^μ belongs to r_j and the largest of the components of $w_i(t)$ is taken as θ_j: it is clear that the prototype ξ^μ will be a fixed point. But the problem is how to avoid that points with high degree of correlation with a prototype but with all its relative weights components lower than the capacity parameter to skip away from this prototype. The idea proposes in this paper, made use of the deviation defined in (15). When, in time *t*, the dynamic equation is applied to a component of the vector $x(t)$, this component will change its state not only if the relative weight $w_i(t)$ is lower that the capacity parameter of its class. The deviation of the new state, in the case of change of sate, must be similar to the deviation of the prototypes in the new class. The degree of similarity may be measured by a coefficient . The coefficient is handled in a dynamical way (the more is the time the higher is the coefficient).

Besides the weight vector, there are other set of parameters of the net. For every one class r_i, the capacity parameter i and the deviation of the prototypes in this class are obtained. So the algorithm control not only if the new state is strongly correlate with some prototype in its class, the algorithm also control that the components in the new state must, with a high degree of probability, be placed in similar places as some prototype of the class. We have applied with to our example, obtaining that almost all the points inside a neighborhood of radius 1, of the prototypes, are attracted by these prototypes. The 10 Arabian digits are fixed points of the system, and almost all the 28 neighbor of any one of them were attracted by its attractor prototype. In figure 2, the number of points inside a neighborhood of radius 1, of the prototypes are expressed.

$$\begin{cases} \boxed{24 \to 1} \; \boxed{23 \to 2} \; \boxed{25 \to 3} \boxed{22 \to 4} \boxed{27 \to 5} \\ \boxed{22 \to 6} \; \boxed{25 \to 7} \; \boxed{25 \to 8} \; \boxed{22 \to 9} \; \boxed{21 \to 0} \end{cases}$$

Figure 2. Prototypes belonging also to r_{15}

4. Conclusion

The weight parameters in the *Hopfield* network are not a free set of variables. They must fulfill a set of constrains which have been deduced trough a new re-interpretation of the net as *Graph Formalisms*. Making use of this constrains the *state-vector* has been classified in *n* classes according to the *n* different possible distances from any of the state-vectors to the *zero* vector. The *$(n \times n)$* matrix of weights may also be reduced to a *n*-vector of weights. In this way the computational time and the memory space, required for obtaining the weights, is optimized and simplified. The degree of correlation from a pattern with the prototypes may be controlled by the dynamical value of two parameters: the capacity parameter θ which is used for controlling the capacity of the net (it may be proved that the bigger is the θ_j component of θ, the lower is the number of fixed points located in the r_j energy line) and the parameter μ which measures the deviation to the prototypes. A typical example has been exposed, the obtained results have proved to improve the obtained when the classical algorithm is applied.

Bibliography

[1] V. Giménez-Martínez, *A Modified Algoritm Hopfield Auto-Associative Memory with Improved Capacity*, IEEE Transactions on Neural Networks, (in press), 2000.

[2] N. K. Bose and P. Liang. *Neural Network Fundamentals with Graphs, algorithms and Applications.* McGraw Series in Electrical and Computer Engineering, 1996.

[3] V. Giménez-Martínez, P. Gómez-Vilda, E. Torrano and M. Pérez-Castellanos, *A New Algorithm for Implementing a Recursive Neural Network,* Proc. of the IWANN´95 Torremolinos, Málaga, Spain, June, pp. 252-259, 1995.

[4] V. Giménez-Martínez, P. Gómez, M. Pérez- Castellanos, and E. Torrano, *A new approach for controlling the capacity of a Recursive Neural Network,* Proc of AMS´94. IASTED, Lugano, Suise, June, pp 90-93, 1994

Authors' Information

Víctor Giménez-Martínez – Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo s.n., 28660 Boadilla del Monte, Madrid, Spain; e-mail: vgimenez@fi.upm.es

M Gloria Sánchez–Torrubia – Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo s.n., 28660 Boadilla del Monte, Madrid, Spain; e-mail: gsanchez@fi.upm.es

Carmen Torres–Blanc – Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo s.n., 28660 Boadilla del Monte, Madrid, Spain; e-mail: ctorres@fi.upm.es

NEURAL NETWORKS DIAGNOSTICS IN HOMEOPATH SYSTEM

Larysa Katerynych, Alexander Provotar

Abstract: We suppose the neural networks for solution the problem of the diagnostic in Homeopath System and consider the algorithms of the training.

Keywords: artificial intelligence, neural networks, training of neural networks, information granules.

Introduction

As a rule, as a consequence of the cerebrum study and mechanisms of its functioning there have been created new computer models, namely artificial neural networks (NN). The tasks of the office automation processes based upon the research in the sphere of the artificial intelligence (AI) are of current importance to present day. NN permit to solve applications such as pattern recognition, modeling, fast data conversion (parallel computational processes), identifications, management, and expert systems creation [Терехов, 2002, Барский, 2004].

Theoretically, NN can solve a wide frame of tasks in the specific data domain. (as it is the human brain model prototype), but it is still not practically possible to create the integrated universal NN for the specific data domain at present, since there is no integrated construction algorithm (functioning) of the NN. The moment to date the specific structure NN and with the defined learning algorithms are used for the solution of the concrete group of tasks out of the fixed data domain.

As it is well-known, each neuron has a number of qualitative characteristics, such as condition (excited or dormant), input and output connections. The one-way only connections, mated with the inputs of the other neurons are called synapses and the output connection of the given neuron from which the signal (actuating or dormancy) comes on the synapses of the other neurons are called axon. [Терехов, 2002] The neuron overview is presented on figure 1.

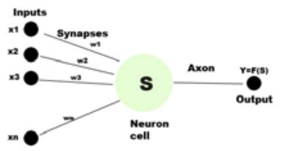


Figure 1. General view of neuron

Per se, the functioning of every neuron is relatively simple. As a rule, the set of the X=[x1,x2,x3,…xn] signals come to the neuron input. Each of the signals may be the output of the other neuron or source. Every input signal is multiplied on the corresponding angular coefficient W=[w1,w2,w3,…wn]. It complies with the force of the synapse of the biological neuron. The products of the wi*xi are summarized and come on the adding element. To initialize the networks, the input x0 (x0=+1) and the weighting factors of the synaptic ties w0 are specially entered. The neuron condition in the current moment is defined as the weighted total of its inputs:

$$S = \sum_{i=1}^{n} x_i w_i + x_0 w_0$$

The neuron output is the output of its condition:

Y=F(S)

The F function is the function of activation. It is monotonous, contiguously differentiable on the interval either (-1,1), or (0,+1).

In the multilayer neuronal networks (MNN) the basic elements outputs of each layer come to the inputs of all the basic elements of the next layer. The activation function F(S) is chosen as being the same for all the neurons of the network. In [1] the MNN it is determined in such a symbol form $N^{K}_{n_0,n_1,\ldots,n_R}$, where K – the number of the layers in the network, n_0 - the number of the network inputs; $n_i (i = 1, K-1)$ - the number of the basic elements in the i-x interlayers, n_K - the number of the basic elements in the output К layer and simultaneously the number of the outputs $q_1,\ldots,q_{n_K}$ of the MNN. The intermediary a layer has n_a neurons. There are no connections between the basic elements in the layer. The layer basic elements outputs come to the neurons inputs of only the next (a+1) layer. The output for any neuron is determined as being

$$q_i^{(a)} = f(\sum_{j=1}^{n_{a-1}} w_{i,j}^{(a)} q_1^{(a-1)} + w_{i,0}^{a} q_0^{(a-1)}) = f(s_i^{(a)})$$

Specialized NN in HOMEOPATH system

The scientists have been into the development of the mathematical methods solutions of medicinal tasks for many years already. The effectiveness of the similar mathematical methods may be followed by the set of the medical diagnostics systems, developed in the last time. The general trait of these systems is their dependence on the specific methods of the group data processing, poorly applied to the unit objects and, also, on the features of the medical information [Горбань, 1998].

The neuronal networks (NN) are easy-to-use instruments of the information models presentation. In the general case, the network receives some input signal from the outer world and passes it through itself with the conversion in each of the neurons. Hence, the signal processing is being made in the process of its passage through the network connections, the result of which is the specific output signal. For the purposes of the neuronal network designing in the system of HOMEOPATH there has been chosen the mostly spread structure of the neuronal networks – multilayer one. This structure imports that every neuron of the arbitrary layer is connected with all the outputs and inputs (axons) of the preceding layer or with all the NN inputs in the case of the first layer. In other words, the network has the following structure of the layers: the input, the intermediate (latent) and output. Such neuronal networks are also called fully connected [Барский, 2004].

For the solution of the diagnostics task in the system of HOMEOPATH the NN of the following architecture is being used (Figure 2).

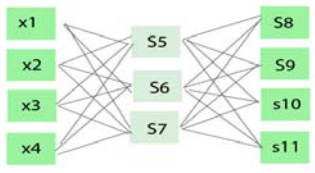


Figure 2. Architecture neuron network

The task of the habituation of the MNN in classical form could be presented as following. Let there is specified some series of x^* input data. It is requested to find such solution x, with which it is possible to classify the newly presented input data. The criterion $R(x,x^*)$ determines the quality of solution. The variety of solutions x is determined by the choice of the weighting factors $w_i^{(a)}$ adjustment algorithm. Under such problem definition, the training process is comes to the receipt of the best solution out of the series of possible ones. In other words, the MNN training – is the process of data x^* accumulation and, concurrently, the process of the solution x choice.

The algorithm of training

The NN of the HOMEOPATH system uses the algorithm of the inverse distribution the gist of which is in the distribution of the error signals from the NN outputs to its inputs in the direction, back to the direct signals distribution in the usual mode of operation (identification regime). In other words, we use the technologies of the series tuning of neurons starting with the last output layer and finishing with the tuning of the first layer elements. The NN habituation may be done the necessary number of times. For the habituation the so called δ -rule is used, which lies in the realization of the training strategy with the "teacher". Let us label as y^* - the required neuron output, y – the real output. The error of the training is calculated according to the following formula $\delta = y^* - y$ in the algorithm of the gradient descent (the weighted factor) $w_i(k+1) = w_i(k) - \gamma\delta x_i$, $\gamma > 0$, where γ -is the "strengthening of the algorithm" factor, x_i - the i input of the neuron synaptic connection.

The algorithm of the NN training for the determination of the diagnostics tasks in the HOMEOPATH system is in the following order of steps:

Algorithm:

1. In the context of the data domain the input signals vector is made: $x = \langle x_1, x_2, \ldots x_n \rangle$, where $x_1, x_{2,} \ldots, x_n$ are the patient symptoms.
2. The vector corresponding to the correct definitions (required) $y^* = \langle y_1, y_2, \ldots, y_n \rangle$, formed by an expert in the data domain.
3. The algorithm of the direct spread of the signal x through the network is executed. As a result of the algorithm execution the weighted sums S_{jn} are determined and the activators for each cell.
4. The algorithm of the inverse signal distribution through the cells of the output and intermediary layer is executed. The errors δ_0 calculation is performed for the output cell and δ_i for the encapsulated cells.
5. The neurons weights renovation is performed in the network, where S_{jn} – is the weighted sum of the output signals of the n layer n (the activation function argument).

The NN habituation is in the presentation of the training examples out of the little group of the desired actions. This is performed by the way of the algorithm of the inverse distribution performance with the account of the desired result and real result.

The network functions in two regimes: in the regime of habituation and in the regime of identification. In the regime of habituation the so called logical chains formation is made. In the regime of identification according to the specific input signals with the high range of validity the NN determines, which actions to undertake.

The habituation of the neuronal network is performed upon the limited number of examples, then, they permit it to independently generate the behavior in other situations. The ability to generate correct reaction on various symptoms, which are not in the training set, is the key factor in the creation of the NN. The data for testing is some scenarios with the set of possible actions. In the result, the network has to calculate the reaction on the inputs and perform an action, which will be similar to the training scenarios.

For the network habituation the following examples were used (Table 1).

Sympt1	*Sympt2*	*Sympt3*	*Sympt4*	*Preparation*
1	*1*	*0*	*1*	*P1*
1	*1*	*0*	*1*	*P2*
0	*0*	*1*	*0*	*P3*
1	*0*	*1*	*1*	*P4*
0	*0*	*1*	*1*	*P5*

Table 1. Input data

Sympt1	*Sympt2*	*Sympt3*	*Sympt4*	*Preparation*
1	*1*	*0*	*1*	*P1*
1	*2*	*0*	*1*	*P2*
0	*0*	*1*	*1*	*P3*
1	*1*	*0*	*1*	*P4*
0	*1*	*0*	*1*	*P5*

Table 2. Output data

In order to test the NN, the networks should be presented with the new examples. This allows determining how the network will react upon the scenarios, of which nothing is known. The similar tests permit to know hoe qualitatively the NN can react on the unforeseen situations and perform the necessary actions. To test the NN it is necessary to present new examples (Table 2).

The distinguishing property of the HOMEOPAT system is it's possibility to follow the strategy of construction (and following check of differentiated diagnosis), used by a clinicist. Such a diagnostic model includes the two-level procedure, which builds hypothesis of illness, based on input patient data, and processes the data estimation based on additional symptoms, which should be common for the possible illnesses. The search for the needed information to form the image of illness is performed through the knowledge base using the mentioned specialized NN.

Features of NN design in the HOMEOPAT system

The processes of defining, extraction and structurization of information are known to be the most important in artificial intelligence system building. Implementation of each process has a great distinction, depending on the information storage environment, organization of information and the concepts of NN design for searching and analyzing the required data. In HOMEOPAT system main informational blocks consist of general patient's state, symptoms and recommended drugs. Taking into consideration the fact that system uses the concept of granular NN design [Bargiela, 2003], based on the information granules creation process with their following processing. Information granules are defined as the map

$$A : X \rightarrow \delta(x)$$

where A - illness, X – the space on possible symptoms, δ – the space of homeopathic drugs.

The detail level of artificial information granules is defined be the number of elements in the granule. The level of precision is defined through the integral

$$card(A) = \int_X A(x)dx$$

where – A – examined granule. The more powerful granule is, the more precise will be the result of illness distinction using the given symptoms and less specific.

Let $A = \{A_1, A_2, ... A_c\}$ - where "c" is the precision level in naturally common illnesses in the form of granule set. The initial data can have a different level of precision. For example, $B = \{B_1, B_2, ... B_p\}$, where "p" is the level of precision, which is much higher, then of a granule with level “c. Such a transition from the granule of the selected level of precision to the granule with another level of precision brings up the mechanism of dealing with so-called dynamical information granules, or the mechanism of new granular space building.

For example, the evolution of granular space $< X, G, A' >$ into $< X, G, A'' >$, where A'' is more precise than A', in the HOMEOPAT system can mean the gathering of some new information about the illness, and consequently a dynamic change of corresponding information granule. Depending on the problem, granules in the system are grouped into layers. The given information network in the basis for the NN design in the HOMEOPAT system. In addition we should point out that hidden layers of NN in such a case represent the agents of granular environment, responsible for interaction between different objects.

Formally the granular environment with the interaction layer can be defined the following way $G =< X, G, A, C >$. Let's examine the general structure of data-model in terms of information granulation on example of simplified model of NN functioning. The upper generalized level of granular space could be represented in the form of the following information granules: structure of input data for the NN, which is represented as an array of illnesses symptoms, is the first element of informational granulation, which has a corresponding array of drugs, which could be used to cure the current symptoms, which is the second element of informational granulation. Formally this can be described so $G_{net} = \{X_{net}, G_A, In_{G1}, In_{G2} ...\}$, where G_{net} - neural network of granulation, G_A – agent of granulated environment, X_{net} - artificially represented space of informational granules, In_{G1} - informational granule containing an array of symptoms, In_{G2} - informational granule containing an array of drugs.

As conclusion we note that the simultaneous usage of main terms of informational granulation and neural networks gives a bunch of new possibilities in designing of the mentioned specialized neural network. Exactly: more precisely design the architecture of the network, number of hidden layers and the method of representation of learning set and the following work with the real data.

Bibliography

[Терехов, 2002] В.А. Терехов, Д.В. Ефимов, И.Ю. Тюкин. Нейросетевые системы управления. – Москва: «Радиотехника», 2002. – 467с.

[Барский, 2004] А.Б. Барский. Нейронные сети: распознавание, управление, принятие решений. – Москва: «Финансы и статистика», 2004. – 398с.

[Горбань, 1998] А.Н.Горбань, В.Л.Дунин-Барковский и др. Нейроинформатика.–Новосибирск: «Наука», 1998.

[Провотар, 2000] Провотар А.И., Дудка Т.Н., Гошко Б.М. Применение метода резолюций на семантических сетях //Проблемы программирования. –2000. –№ 1–2.

[Bargiela, Pedrycz, 2003] Andrzej Bargiela, Witold Pedrycz. Granular Computing. – 2003. -5-50 p.

Authors' Information

***Provotar Alexander**, professor, chief of the information systems department of the faculty of cybernetics Taras Schevchenko Kiev National University, aprowata@unicyb.kiev.ua*

***Katerynych Larysa**, lecture assistant of the faculty of cybernetics Taras Schevchenko Kiev National University, katerinich@rambler.ru*

GENETIC ALGORITHM FOR FINDING THE KEY'S LENGTH AND CRYPTANALYSIS OF THE PERMUTATION CIPHER

Aleksey Gorodilov, Vladimir Morozenko

Abstract: In this article we discuss a possibility to use genetic algorithms in cryptanalysis. We developed and described the genetic algorithm for finding the secret key of a block permutation cipher. In this case key is a permutation of some first natural numbers. Our algorithm finds the exact key's length and the key with controlled accuracy. Evaluation of conducted experiment's results shows that the almost automatic cryptanalysis is possible.

Keywords: cryptography, cryptanalysis, block permutation cipher, genetic algorithm, data encryption

ACM Classification Keywords: I.2 Artificial Intelligence: I.2.8 Problem Solving, Control Methods, and Search - Heuristic methods, E.3 DATA ENCRYPTION Code breaking.

Introduction

The main problems in cryptography are the development of reliable cryptographic schemes (a cryptography problem) and the search for new effective methods of deciphering existing schemes (a cryptanalysis problem). A cryptographic approach to secure information implies its transformation which enables it to be read only by the owner of the secret key. The reliability of a cryptographic method of securing data depends on cryptanalysis stability of the used scheme. When we talk about cryptanalysis we assume that we know the cryptographic scheme, but we don't know the key and/or its length. In other words, a cipher breaking problem (a cryptanalysis problem) is a problem of finding only one true secret key among all possible secret keys, i.e. it is a search problem. The search space is large and the criterion of found solution's "quality" is not usually purely formalized.

In this article we focus on the cryptanalysis problem of the block permutation cipher. The secret key of this cipher is a permutation of some first natural numbers with unknown length. To solve the cryptanalysis problem we use genetic algorithms. It seems reasonable to apply genetic algorithms, because they are successfully used in search problems and optimization problems. You can read about solving cryptanalysis problems by genetic algorithms in [Delman, 2005, Lebedev, 2005, Jakobsen, 1995]. But authors of these articles assume that the key's length is known and they use standard methods which do not consider peculiarities of the task or they don't investigate how algorithm's parameters can affect its convergence speed.

Before developing any genetic algorithm, you must choose the proper way to present possible solution as a symbol string and also choose proper main operators – selection, crossover and mutation. The quality of the genetic algorithm should be adjustable. We should have a possibility to change algorithm's parameters such as the size of the population, the number of generations and probabilistic characteristics of main operators.

A block permutation cipher

In this article we consider the particular cryptographic scheme – the symmetric block permutation cipher. The main idea of this cipher is that we divide the input text into blocks, i.e. into symbol strings with the fixed length equal to N, and then symbols in every block are rearranged in accordance with the given permutation

$$P = \begin{pmatrix} 1 & 2 & 3 & \dots & i & \dots & N \\ p_1 & p_2 & p_3 & \cdots & p_i & \cdots & p_N \end{pmatrix},$$

where $p_i \in \{1,2,3,\dots,N\}$. In other words, symbol at the position i is moved to the position p_i. The secret key of that cipher is the permutation P. Deciphering is performed using the inverse permutation P^{-1}.

The genetic algorithm

A permutation will play the role of individual in the genetic algorithm, because the solution of the cryptanalysis problem here is also a permutation. At first we suppose that we know the key's length and we must find only the key, i.e. the permutation with the fixed length *N*.

The main problem we must solve is the role of separate genes of an individual. The first simple solution which seems the most evident is to associate separate genes with elements of the permutation *P*, i.e. associate the i^{th} gene with the number p_i. Obviously in that case genes will depend on each other. If some gene is equal to *j*, then no other gene of this individual can be equal to *j*, because all numbers between 1 and *N* are presented only once in the permutation *P*. The dependence of genes results in important limitations to operators of mutation and crossover. We can't use standard operators in this case, because all of them work with a string of independent bits. Such interconnections between genes don't reside in wildlife. This leads to prejudices in the effectiveness of using genetic algorithms. But this association is subconsciously comprehensible and doesn't need any additional computations to form genes.

The alternative solution is to use an intermediate presentation of an individual in a form of an object which can be easily transformed to a permutation. At the same time such object should be presented as a string of independent bits to make applicable mutation and crossover operators. In this case the problem of choosing a proper intermediate presentation can be very complex. In this article we choose the first solution which is the most comprehensible. It means that in the rest part of the article we will regard separate genes as elements of the permutation *P*.

The next problem is how to calculate fitness value for individuals. We suppose that input text is a literary text in Russian. Fitness function is taken from Jakobsen's works [Jakobsen, 1995, Agranovskiy, 2002]. In 1995 Thomas Jakobsen suggested an automatic method for deciphering a simple substitution cipher. In Jakobsen's method the information about frequency sharing of digrams in literary texts is used. Digram is a pair of two neighboring symbols in text. Jakobsen suggested calculating the fitness value as the sum of modules of differences between predefined etalon digram frequencies and real digram frequencies in the ciphertext. Let T_{ij} be the relative frequencies of digrams (*i*, *j*) in the text *T*. Then the fitness function would look like

$$W(T) = \sum_{ij} \left| T_{ij} - E_{ij} \right|$$

where E_{ij} is the predefined etalon frequencies of digrams (*i*, *j*). Etalon frequencies' matrix *E* is calculated beforehand under big literary text in Russian. Obviously, the more literary is the text, the less is the fitness value and the "closer" is the found key to the true secret key. It means that fitness value is in inverse proportion with individual's fitness. It is important that fitness function can be not equal to zero even if the ciphertext is deciphered with the true secret key.

So, if we have the ciphertext *S*, then we should perform the following steps to calculate fitness value of the individual *P*.

1. Decipher ciphertext *S* with the selected key *P*. As a result we would get the text $T = Decrypt_P(S)$..
2. Calculate digrams frequencies T_{ij} in the text *T*.
3. Calculate the fitness value *W*(*T*).

Let's focus on the properties of the fitness function *W*(*T*). Let P_1 and P_2 be two individuals and let them differ in only two first genes (obviously they can't differ in the only one gene). Suppose for instance

$$P_1 = \begin{pmatrix} 1 & 2 & 3 & \dots & N \\ p_1 & p_2 & p_3 & \cdots & p_N \end{pmatrix},$$

$$P_2 = \begin{pmatrix} 1 & 2 & 3 & \dots & N \\ p_2 & p_1 & p_3 & \cdots & p_N \end{pmatrix}.$$

Suppose, we have ciphertext *S*. Texts $Decrypt_{P_1}(S)$ and $Decrypt_{P_2}(S)$ deciphered with keys P_1 and P_2 differ only in symbols which are at positions p_1, p_2, $p_1 + N$, $p_2 + N$, $p_1 + 2N$, $p_2 + 2N$ and so on. These numbers form two arithmetic progressions with the difference of *N*. Because texts $Decrypt_{P_1}(S)$ and $Decrypt_{P_2}(S)$ can differ only in these positions, each block with a length of *N* has no more than 3 digrams which differ in their frequencies. So, when the key's length *N* is big enough (this is true for real schemes), a small change of an individual would cause a small change of a fitness value.

Also notice that in long literary texts digram frequencies T_{ij} are close to corresponding etalon frequencies E_{ij}. That's why in the result of deciphering the text *T* with the true secret key fitness value would be close to zero. So, if *K* is the true secret key then the value $W(Decrypt_K(S))$ would be the minimal possible value.

As it was mentioned above we can use a standard crossover operator only if an individual is presented as a string of independent bits. In our case individuals are not strings of independent bits, so we can't use standard operators. We should develop a special crossover operator. The main requirement to the crossover operator is that the result of its execution should give a correct permutation. In this article we suggest the next crossover operator:

1. Enumerate all genes with numbers 1, 2, 3, …, *N* and scan them in increasing order.
2. Copy gene with the next number from one of the parents, if it is possible.
3. Assign any possible number to gene if genes from both parents are impossible.

Let $P(i)$ be the number which replaces the number *i* in permutation *P*. So, we have the next crossover algorithm for two parents P_1 and P_2 and their child P^*.

1. Assign a value of Ø to the set *Used*. *Used* is the set of already used gene values. Set the number of current gene *i* to 1.
2. Define whether numbers $P_1(i)$ and $P_2(i)$ belong to *Used* or not.
3. If only one of numbers $P_1(i)$ and $P_2(i)$ doesn't belong to *Used* then assign it to the i^{th} gene of P^*. If both numbers don't belong to *Used* then choose number $P_1(i)$ with probability p_c or choose $P_2(i)$ with probability $(1 - p_c)$. Assign the selected number to the i^{th} gene of P^*. If both numbers belong to *Used*, then assign random number from the set $\{1, 2, 3, \ldots, N\}\backslash Used$ to the i^{th} gene of P^*.
4. Include the number which was assigned to the i^{th} gene of P^* in the set *Used*.
5. Move to the next gene, i.e. increase *i* by 1 and go to step 2.

We can use a standard exchange operator as the mutation operator. The idea of exchange operator is very simple. Some two genes of the individual exchange their positions with some probability. To develop the selection operator we use two main classical ideas: the "wheel principle" and the "elite principle" [Mitchell, 1999]. The population's size remains constant because every time new two children are obtained we delete two individuals with the smallest fitness from the population. Finally, we selected the condition of the end of calculations. Calculations stop when era number reaches a fixed number *M*. Because of this condition we can predict time that is needed to complete the calculations or we can define the parameter *M* so that the algorithm ends after specified time.

The influence of genetic algorithm's parameters

So, we developed a genetic algorithm for solving the problem of cryptanalysis of the block permutation cipher. We supposed that we know the key's length *N*. Let's discuss the optimal values for parameters. The main parameters which were mentioned early in this article and which influence the algorithm's speed and the solution's quality are:

- the population's size *k*;
- the probability of copying a gene from one of the parents p_c;
- the mutation's probability p_m;
- the number of generations *M*.

There are no theoretical recommendations on how to choose these parameters. We can only say that the mutation's probability should be small and p_c must be close to 0.5. The number of generations can be chosen depending on time resources. Generally speaking, if the number of generations is the only condition for the end of calculations it must be big enough. The population's size at the start is also important because the population's size is constant in our genetic algorithm and is always equal to *k*. Notice that the population's size has a considerable influence on the complexity of computations and the time required for them.

Conducted experiments show that small population's size gives bad results because the algorithm quickly finds a local minimum point of the fitness function and all further populations are formed in its vicinity. According to these experiments if key's length equals to 10 then the optimal value for population's size is between 15 and 17. Changing the mutation's probability in a reasonable range doesn't have a significant impact on the results. It can be partially explained that we used a specific crossover operator which already implies some mutation of individuals. Indeed, when the next gene cannot be copied from one of the parents it is simply chosen randomly, i.e. child doesn't "resemble" any of his parents in this gene. The optimal number of generations significantly depends on the key's length *N*. If *N* is small, a big number of generations isn't effective because after some populations algorithm finds a local minimum and all further calculations don't lead to a better result. Possible solutions' space grows with the growth of the key's length. It requires more generations to find the best solution.

At the end of the genetic algorithm's work we obtain a text called decrypted text. That decrypted text doesn't considerably differ from the starting open text. Usually we can restore separate symbols from the context. So, algorithm doesn't completely solve the cryptanalysis problem but greatly helps to decrypt the given text.

Finding the key's length

The genetic algorithm described above works on the assumption that the key's length *N* is known. The result of its work is the permutation which corresponds to the best individual from the final population. Because this permutation has the minimum value of the fitness function *W*(*T*) genetic algorithm also calculates the value *F*(*N*) by the given key's length *N*. *F*(*N*) indicates the minimal found value of the fitness function.

Let's now focus on the key's length finding problem from the given ciphertext. Let *N* be the supposed key's length and *L* be the true key's length. If *N* is equal to *L* then we can expect that the permutation found by the genetic algorithm is close to the searched permutation and the fitness function corresponding to this permutation would have a relatively small value.

Let the true secret key be the permutation

$$K = \begin{pmatrix} 1 & 2 & \cdots & L \\ p_1 & p_2 & \cdots & p_L \end{pmatrix}$$

Consider the key with length 2·*L* of a form

$$K' = \begin{pmatrix} 1 & 2 & \cdots & L & L+1 & \cdots & 2\cdot L \\ p_1 & p_2 & \cdots & p_L & L+p_1 & \cdots & L+p_L \end{pmatrix}$$

In K' first *L* columns are absolutely equal to first *L* columns in permutation *K*. The rest columns are obtained from the first *L* columns by adding *L* to all numbers. We can correctly decipher the ciphertext with the key K' because both keys *K* and K' identically transform any text. It means that when genetic algorithm searches the key with the length 2·*L* it would find a permutation which is close to K'. At the same time a small fitness value would correspond to the found permutation. Therefore the value *F*(2·*L*) would be relatively small. Similar statements are true for all keys have length divisible by *L*. In other words, if we pass the ciphertext and the key's length $N = m\cdot L$ (where *m* is a natural number) to the input of genetic algorithm then we can expect a permutation with a small fitness value at the output. Otherwise, if *N* isn't divisible by *L* then there is no permutation with the length *N* which are close to the key *K*. In this case fitness value and value *F*(*N*) are expected to be relatively large. So, value *F*(*m*·*L*) is expected to be much smaller than *F*(*N*) for all N which are not divisible by *L*.

So, we offer the following algorithm for the finding of the key's length. At first, choose some initial value N_0 for the key's length. The value $F(N_0)$ is calculated using the genetic algorithm. At the next step we increase the supposed key's length by 1 and calculate value $F(N_1)$ (where $N_1 = N_0 + 1$) using the genetic algorithm and so on. At the *i*th step we calculate value $F(N_i)$ where $N_i = N_0 + i$. This algorithm continues while $N_i < N^*$ where N^* is a predefined value (is an upper bound). As a result of this process we have a sequence of numbers

$$F(N_0),\ F(N_0+1),\ F(N_0+2),\ \ldots,\ F(N_0+i),\ \ldots,\ F(N^*).$$

In this sequence we can mark out small values having positions expected to form an arithmetical progression with the step L. The value of L is the searched true key's length.

Notice that we can use a small population's size and a small number of generations while searching the key's length to save time resources. When the length of the key is found we can restart genetic algorithm with big values for parameters to find the secret key.

An example of algorithm's work

As the input text we took a fragment of literary text in Russian:

Было тихое летнее утро. Солнце уже довольно высоко стояло на чистом небе, но поля еще блестели росой, из недавно проснувшихся долин веяло душистой свежестью, и в лесу еще сыром и не шумном, весело распевали ранние птички. На вершине пологого холма, сверху донизу покрытого…

In experiment this text was encrypted using the block permutation cipher with the key's length equal to 10. In the result we had the next ciphertext:

т иолоеыхБенеуеттл оСлц. еонрд оожелув ывско оноьляонтоас отм ис ч,н бепеонщеебя лл оил отессрези е, дйнон рснонвоасхядшиов уевяон иллтсо шисуйдтсь,же еювелс в е уирыю сиемщушмое мнн есл вер о,авл пе сиаеи тнниапрН ави.ек ч еплиношорх омгоаолгрех свд у, упкизрноо о ог т ы…

From this meaningless set of symbols it is difficult to restore the start text manually. Then this ciphertext was passed to the input of the genetic algorithm in order to find the secret key's length. The range of the supposed key's lengths was [4; 34], i.e. N_0=4, N^*=34. For all supposed key's lengths we calculated corresponding values $F(4)$, $F(5)$, …, $F(34)$. Figure 1 shows the reverse values 11/$F(4)$, 1/$F(5)$, …, 1/$F(34)$. The higher point at the diagram the closer its abscissa to the true key's length L or to a value, divisible by L.

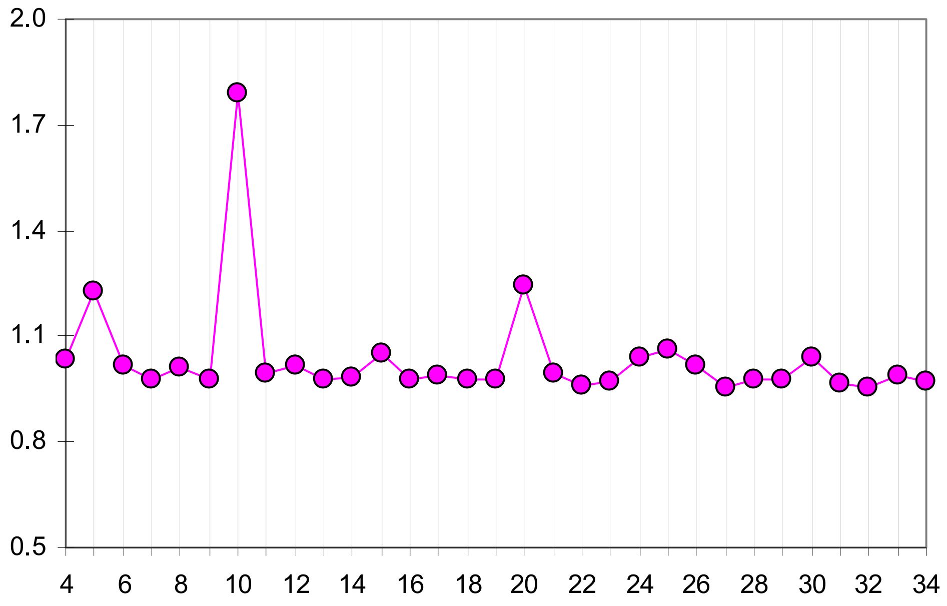


Fig1. Diagram of the dependence 1/F(N) where N is the supposed key's length.

On the figure we can see the maximal “spike” of the fitness function which corresponds to the length 10. Notice that 10 is the true secret key’s length. The next noticeable local maximum is at N=20 which corresponds to the double value $2{\cdot}L$. Further there are no noticeable “spikes” but at N=30 (which equals to the triple value $3{\cdot}L$) we can see a local maximum.

Results of this experiment conform well to the theory. So, in this experiment we can affirm that the secret key’s length L is equal to 10. Then genetic algorithm was restarted to find the secret key. We used the following values for algorithm’s parameters: the population’s size – 15, the number of generations – 30 and the mutation’s probability – 0.2. In the result we obtained the next text:

Былохти ое лет еенутро. нолСце ужевдо ольно оысвко сто лояна чис омт небе он, поля щееблестери лосой, нз иедавноопр снувши сяхдолин леяво душийтос свежеютьс, и в уесл еще смроы и не нумшом, веоелс распеиалв раннпе итички. На вершино пелогоголхо ма, свурхе донизо пукрытог о…

As we can see, algorithm didn’t completely solve the problem because the result text isn’t identical to the input text. But we can see that this text is close to the input text. For example, we can see several words which are equal to the corresponding words in the input text: “было”, “уже”, “поля”, “небе” and so on. It seemed easy to finish the decryption manually using this result text.

Moreover, we can compare the true secret key and the result of the genetic algorithm. The true secret key is the permutation

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 10 & 8 & 5 & 6 & 2 & 1 & 3 & 9 & 4 & 7 \end{pmatrix}.$$

The result of the genetic algorithm is

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 10 & 8 & 5 & 6 & 9 & 1 & 3 & 2 & 4 & 7 \end{pmatrix}.$$

These permutations differ in only one pair of values. It seems easy to look through a small number of variants and to find the true secret key manually.

Certainly the genetic algorithm for finding the secret key described in this article doesn’t completely automate the text decryption but it makes the decryption faster. After using this algorithm we can easily finish the decryption process manually.

Bibliography

[Delman, 2005] Delman B. GeneticAlgorithms in Cryptography // A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering. New York, 2004.

[Лебедев, 2005] Лебедев А. В криптографии мы способны конкурировать / ЛАН Крипто, 2005.

[Jakobsen, 1995] Jakobsen T. A Fast Method for the Cryptanalysis of Substitution Ciphers, 1995.

[Agranovskiy, 2002] Аграновский А. В., Хади Р.А. Практическая криптография: алгоритмы и их программирование. М.:СОЛОН-Пресс, 2002.

[Mitchell, 1999] Mitchell M. An Introduction to Genetic Algorithms. Fifth printing. Cambridge, MA: The MIT Press, 1999.

Authors' Information

Aleksey Gorodilov – University Undergraduate, Computer Science Department, Perm State University, Bukireva street 15, Perm 614990, Russia; e-mail: gora830@yandex.ru

Vladimir Morozenko – Assistant Professor, Computer Science Department, Perm State University, Bukireva street 15, Perm 614990, Russia; e-mail: v.morozenko@mail.ru

TABLE OF CONTENTS OF VOLUME 15, NUMBER 1