Zeszyty Naukowe Politechniki Białostockiej

INFORMATYKA

Numer 8

Oficyna Wydawnicza Politechniki Białostockiej

Białystok 2011

ISSN 1644-0331

ADRES DO KORESPONDENCJI/THE ADRESS FOR THE CORRESPONDENCE:

„*Zeszyty Naukowe Politechniki Białostockiej. Informatyka*”
Wydział Informatyki /Faculty of Computer Science
Politechnika Białostocka /Bialystok University of Technology
ul. Wiejska 45a, 15-351 Białystok
tel. 85 746 90 50, fax 85 746 97 22
e-mail: znpb@irys.wi.pb.edu.pl
http://irys.wi.pb.edu.pl/znpb

Druk: Oficyna Wydawnicza Politechniki Białostockiej
Nakład: 100 egzemplarzy

CONTENTS

ROUGH SET METHODS AND HARDWARE IMPLEMENTATIONS

Maciej Kopczyński, Jarosław Stepaniuk

Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: This paper describes current achievements about hardware realisation of rough sets algorithms in FPGA (**F**ield **P**rogrammable **G**ate **A**rray) logic devices. At the moment only few ideas and hardware implementations have been created. Most of the existing rough set methods implementations are software type. Software solution provides flexibility in terms of data processing and executed algorithms, but is relatively slow. Hardware implementation limits this versatility, but gives a significant increase in calculation speed.
The paper also includes brief description of current authors research on the creation of this type of implementation. The testing environment uses FPGA from *Altera* called *Cyclone II*. This is a high-capacity device providing the ability to create soft-processor core, along with modules allowing to support peripherals of the development board.

Keywords: rough sets, FPGA, programmable logic devices

1. Introduction

Professor Zdzisław Pawlak introduced rough sets assuming that objects are perceived by values of some attributes (for review see e.g., [9,10,13]). Existing implementations of the rough set methods are implemented using a high-level programming languages. This type of implementation provides the ability to comply with any of the algorithms, but the biggest issue is relatively low speed of operation.

The computer processor is a versatile system that executes an arbitrary list of compiler-generated instructions dependant on the source code created by the programmer. For this reason, processors are not optimized to perform specific actions, such as simultaneous rapid execution of elementary logical operations on large amounts of data in a set of objects.

Creating hardware implementation allow us a huge acceleration of the calculation related to the chosen topic, but the disadvantage of this approach is the limit

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 8, pp. 5-17, 2011.

of the applicability of such system only to given issue. A good example of such solutions are GPU's (**G**raphics **P**rocessing **U**nit), which are optimized for parallel execution of calculations related to computer graphics. Most of the mass-produced systems are ASIC-type systems (**A**pplication **S**pecific **I**ntegrated **C**ircuit) that do not allow changes defined in their logical function. Prototype implementations of specialized processors may be implemented in programmable logic devices (e.g., CPLDs (**C**omplex **P**rogrammable **L**ogic **D**evice) and FPGAs) as sequential and combination systems.

Other possibility is a combination of both implementation techniques of the algorithms, which can take advantages of versatility known from software implementations and high-speed calculation of hardware implementations.

Rough sets theory concepts implementation in hardware device can significantly accelerate the execution time of algorithms compared to the software implementation. Logic devices can execute the whole algorithm or just the most time-consuming parts of it.

The paper is organized as follows. Section 2. contains the introduction to rough sets. In Section 3.1, Pawlak's idea of rough set processor is discussed. In Section 3.2, application of cellular networks in rough set methods is shortly recalled. In Section 3.3, some investigations of Kanasugi are presented. Section 4. includes brief description of current authors research on the creation of this type of implementation. In Conclusions, we summarize the results of the paper and we present some directions for further research.

2. Basic notions of rough set theory

Rough set theory due to Zdzisław Pawlak (1926–2006) (see e.g. [9]), is a mathematical approach to imperfect knowledge. The problem of imperfect knowledge has been tackled for a long time by philosophers, logicians and mathematicians. Recently it has become a crucial issue for computer scientists as well, particularly in the area of intelligent systems. There are many approaches to the problem of how to understand and manipulate imperfect knowledge: statistics and probability methods, fuzzy sets, rough sets (see e.g. [4], [11], [15]). One of the most successful is, no doubt, the fuzzy set theory proposed by Lotfi A. Zadeh. Statistics and probability theory can be used to build models and frameworks for particular problems. Rough set theory presents still another attempt to solve this problem. It is based on an assumption that objects are perceived by partial information about them. Due to this some objects can be indiscernible. Indiscernible objects form elementary granules. From this fact it follows that some sets can not be exactly described by available information about objects.

They are rough not crisp. Any rough set is characterized by its (lower and upper) approximations. In this section, we recall some basic definitions of rough set theory.

Let U denote a finite non-empty set of objects, to be called the universe. Further, let A denote a finite non-empty set of attributes. Every attribute $a \in A$ is a function

$$a : U \to V_a,$$

where V_a is the set of all possible values of a, to be called the domain of a. In the sequel, $a(x)$, $a \in A$ and $x \in U$, denotes the value of attribute a for object x.

Definition 1. *A pair $IS = (U,A)$ is an information system.*

Usually, the specification of an information system can be presented in tabular form.

Each subset of attributes $B \subseteq A$ determines a binary $B-indiscernibility$ relation $IND(B)$ consisting of pairs of objects indiscernible with respect to attributes from B. Thus, $IND(B) = \{(x,y) \in U \times U : \forall_{a \in B} a(x) = a(y)\}$. The relation $IND(B)$ is an equivalence relation and determines a partition of U, which is denoted by $U/IND(B)$. The set of objects indiscernible with an object $x \in U$ with respect to B in IS is denoted by $I_B(x)$ and is called $B-indiscernibility$ class. Thus, $I_B(x) = \{y \in U : (x,y) \in IND(B)\}$ and $U/IND(B) = \{I_B(x) : x \in U\}$.

Definition 2. *A pair $AS_B = (U, IND(B))$ is a standard approximation space for the information system $IS = (U,A)$, where $B \subseteq A$.*

The lower and the upper approximations of subsets of U are defined as follows.

Definition 3. *For any approximation space $AS_B = (U, IND(B))$ and any subset $X \subseteq U$, the lower and upper approximations are defined by*

$$LOW\left(AS_B, X\right) = \left\{x \in U : I_B\left(x\right) \subseteq X\right\},$$

$$UPP\left(AS_B, X\right) = \left\{x \in U : I_B\left(x\right) \cap X \neq \emptyset\right\}.$$

The lower approximation of a set X with respect to the approximation space AS_B is the set of all objects, which can be classified with certainty as objects of X with respect to AS_B. The upper approximation of a set X with respect to the approximation space AS_B is the set of all objects which can be possibly classified as objects of X with respect to AS_B.

The difference between the upper and lower approximation of a given set is called its boundary region:

$$BN\left(AS_B, X\right) = UPP\left(AS_B, X\right) - LOW\left(AS_B, X\right).$$

Rough set theory expresses vagueness by employing a boundary region of a set. If the boundary region of a set is empty it means that the set is crisp, otherwise the set is rough (inexact). A nonempty boundary region of a set indicates that our knowledge about the set is not sufficient to define the set precisely. One can recognize that rough set theory is, in a sense, a formalization of the idea presented by a German mathematician Gotlob Frege (1848–1925).

It is possible to express numerically the roughness $R(AS_B, X)$ of a set X with respect to B by assigning

$$R(AS_B, X) = 1 - \frac{card(LOW(AS_B, X))}{card(UPP(AS_B, X))}.$$

In this way, the value of the roughness of the set X being equal 0 means that X is crisp with respect to B, and conversely if $R(AS_B, X) > 0$ then X is rough (i.e., X is vague with respect to B). Detailed information on rough set theory is provided in [9] and [13].

3. Solutions architecture

Designed and implemented rough sets hardware devices use the PC as an external data source and an element of executing all or part of main control program. Hardware systems are used as mechanisms for performing complex calculations, so it is possible to significantly accelerate the calculation time of algorithms. This type of devices can be regarded as a kind of coprocessors. Block diagram of such solutions is shown in Fig. 1.

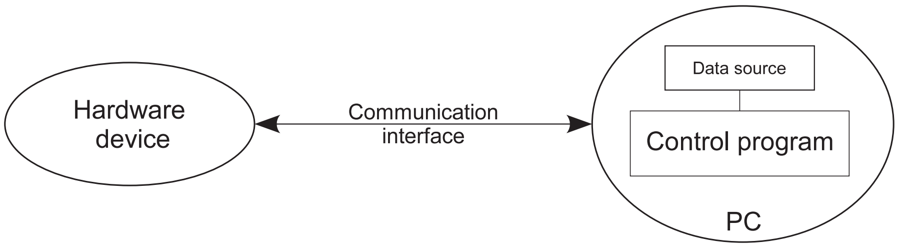


Fig. 1. Block diagram of the rough sets hardware implementation

More powerful FPGA development boards, in addition to the logic device, also contain RAM, flash memory, external flash memory connectors, communication interfaces (USB, Ethernet, etc...) and extensive microcontrollers, e.g. on the ARM cores. Moreover, large FPGAs allow us the implementation of the soft-core processors [16] [19]. Development boards equipped within this type of devices can be used for the implementation of control software without the need for an external, large PC. Currently available versions of operating systems (e.g. Linux [18]) support most common of soft-core processors, what makes it possible to install it on this type of devices. Computing power of such processors is satisfactory for supporting these tasks. This leads to minimization of the amount of space occupied by the resulting device, which finally becomes an independent unit.

3.1 Pawlak's idea of Rough Set Processor

In [8] Pawlak presented an outline of an exemplary RSP (**R**ough **S**et **P**rocessor) structure. The organization of a simple processor is based on elementary rough set granules and dependencies between them. A simplified *RSP* is shown on Fig. 2.

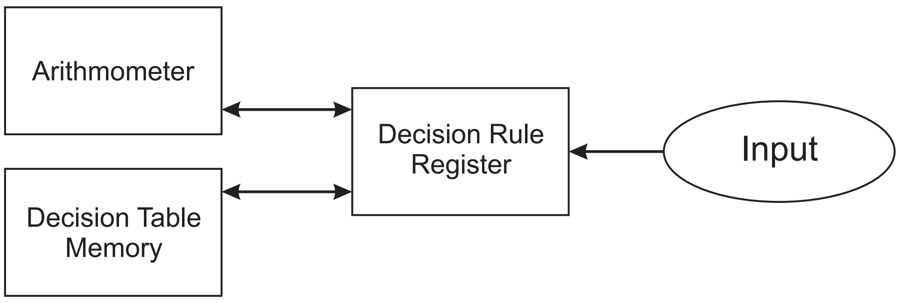


Fig. 2. Block diagram of *Rough Set Processor* [8]

RSP consists of the following units:

- *Decision Table Memory* – this unit keeps the data from the decision table. An ideal situation would be if the memory is large and fast enough to store the whole decision table,
- *Decision Rule Register* – main purpose of this unit is to generate final set of decision rules. This module cooperates with the arithmometer because of need to perform some calculations,
- *Arithmometer* – unit used to perform arithmetic operations for the rest of the modules.

The idea of *RSP* design is as follows. At the beginning, only conditions, decisions, and support of each decision rule are given. Condition and decision are parts of the decision rule. Support is the number of objects from the original decision table matching a given decision rule.

Next operation step is calculation of strength, certainty and coverage factors of each decision rule. These values will be used to find the most important decision rules.

Idea presented by Pawlak was not realized in programmable logic device.

3.2 Cellular networks

Rybinski and Muraszkiewicz in [6] created the concept of describing rough sets methods with usage of cellular networks. On this basis, the idea of the implementation of a device called *PRSComp* (**P**arallel **R**ough **S**ets **Comp**uter) was presented. This is device for parallel processing of basic rough set operations. The description of cellular networks is contained in [7]. Cellular network consists of a matrix of interconnected elements of the same type (cells, that can be treated as a simple, single processors) and a set of control registers. Block diagram of a cellular network with a set of registers is shown in Fig. 3.

The use of cellular network with rough sets is based on the transformation of the input data set to the matrix and definition of the basic operations associated with rough sets using matrix notation. In [6] the following notions are presented along with their pseudocode:

- indiscernibility relation,
- upper approximation,
- lower approximation,
- reducts calculation,
- core calculation.

Given pseudocode allows the implementation of presented matrix notation in programmable logic devices.

Paper [6] provides a basis for the development and expanding *PRSComp* device for another, more complex operations associated with rough sets and matrix notation.

Lewis, Perkowski and Jozwiak [5] described the idea of self-learning rough sets model representation in hardware device. Model is based on cellular networks by Rybinski and Muraszkiewicz. They suggested the possibility of implementing the

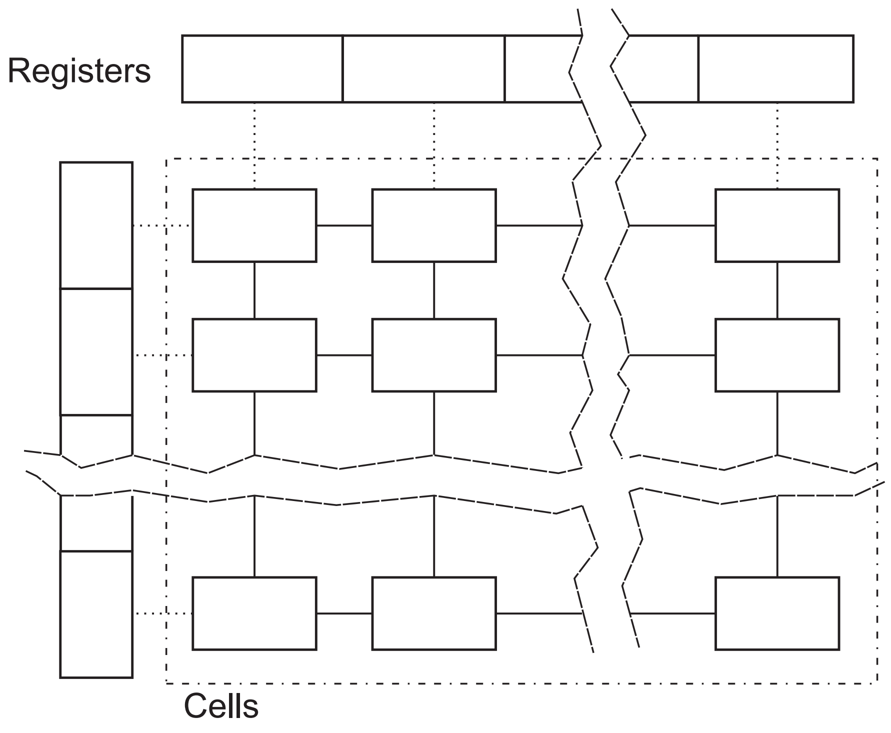


Fig. 3. Block diagram of sample cellular network

solution on *DEC-PERLE-1* board, which is the matrix consisting of 23 type 3090 FPGAs from *Xilinx*. The general principle of device operation is to perform the learning process at the higher level (software), while the later results of this process are transferred to a lower level (hardware). Description of the system working process is as follows:

1. Creation of cellular network logical structure based on an optimized decision matrix (the set of examples) and the requirements for the network construction.
2. Cellular network structure developed on the basis of data from the decision matrix is mapped to the FPGA unit, where each device performs the functions described in the *PRSComp* system. Mapping is created using standard EDA software (e.g., from *Xilinx*).
3. Device's knowledge is stored in the memory of the *DEC-PERLE-1* board as the patterns representing created cellular networks structures. Previously created cellular network patterns are multiplexed in order to choose the best one when working with different data sets. Switching scheme is supervised by an external computer with appropriate control software.
4. During the network training phase, when solving new problems, taken decisions are stored in the memory. Basing on this data the network structure can be reorganized or built completely from scratch in order to avoid the impact of the previously created pattern.

3.3 Direct solutions

Kanasugi and Yokoyama [1] developed a concept of logic device capable of minimizing the large logic functions created on the basis of discernibility matrix. System output are small logical functions representing important decision rules.

The presented system is not independent. It requires an external data source and the mechanisms for creating large logical functions from the database for correct operation. This system can be treated as a coprocessor supporting the central unit.

Block diagram of the logic device is shown in Fig. 4. The project consists of the following functional elements:

- *Core Selector* – main task of this unit is to select rows of binary decision matrix, which include the shortest logical formulas (which have the smallest number of variables with a true value).
- *Covering Unit* – goal of this unit is to check each row of a binary decision matrix and denote them as candidates to remove. Selecting the rows is based on the data prepared by the *Core Selector*.

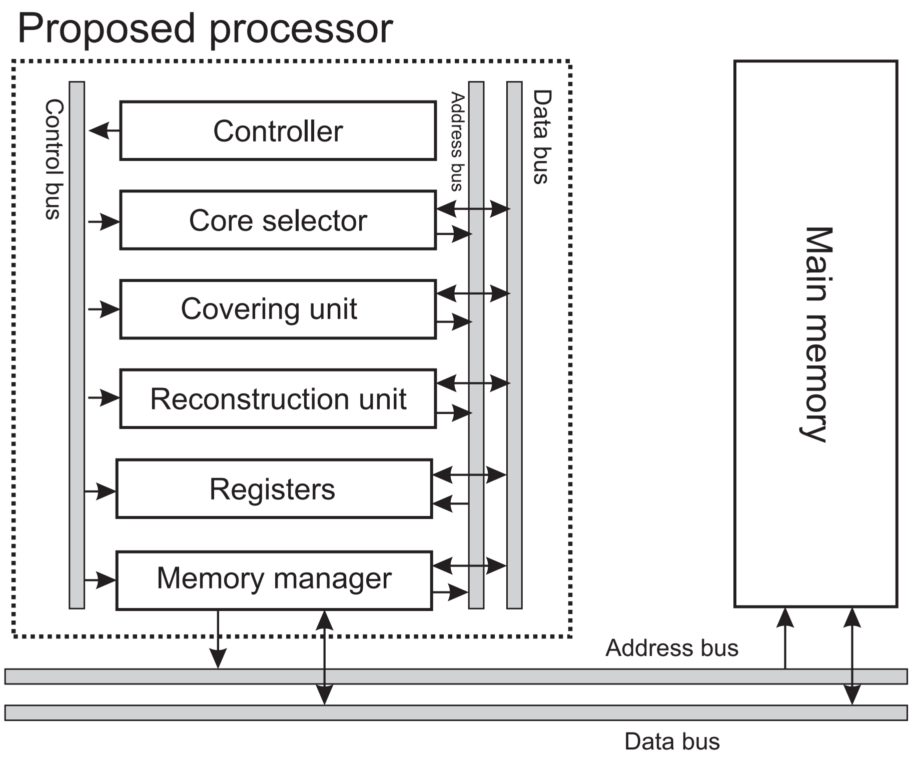


Fig. 4. Block diagram of the logic device [3]

- *Reconstruction Unit* – the role of this unit is to discover dominant variables in binary decision matrix, what helps to find most significant decision rules.

System functionality can be divided into two parts: pre-process (data preparation) and main-process (working with prepared data).
In the data pre-process two units are used: *Core Selector* and *Covering Unit*. The purpose of two mentioned units is to find the rows which contain the least amount of boolean variables (search for cores). The modules also select redundant binary decision matrix rows which can be deleted. The purpose of main-process is to review the whole pre-processed binary decision matrix and select the most important rules (terms). The idea of the algorithm implemented in *Reconstruction Unit* is to find dominant variables in pre-processed binary decision matrix and then create a new decision matrix. This matrix will contain some amount of important decision rules dependent on the algorithm parameter. It should be noted that the implemented algorithm uses approximation technique. Kanasugi has decided on this solution because of time of calculations reduction and the size of the entire system in the FPGA structure.

The solution proposed by Kanasugi and Matsumoto in [3] allowed the nearly 700 times increase in the speed of calculations in comparison to the PC. Binary decision matrix containing 128 rows of data and 2032 attributes was used during the tests. Table 1 shows the results of the experiment.

Table 1. Comparision of calculation time between hardware and software solution [3]

Device type	Speed $[MHz]$	Time $[\mu s]$
Hardware solution	50	7.18
PC	3400	72.54

4. Authors solution

The authors are working on creating a rough set hardware system, which has to be universal for any type of data. The goal of the system is to process the data in accordance with a set of rules, the rules generation from complex sets of data, fast reducts and approximations calculations and so on. Important part of the project is to create a low-level input data transformation algorithms which convert data from decision matrix or database to low-level form (boolean or binary). Inverse operation is also required — the conversion of results returned by the system to similar form found in software implementations. These operations can be performed by software executing on the main unit. At the moment, authors are using development board *DE2-70* made by *Terasic* with FPGA *Cyclone II* (EP2C70F896) by *Altera*. Development board includes a lot of peripheral devices [17]:

- *USB Blaster* for programming (JTAG and Active Serial (AS) programming modes are supported),
- memories: 2 MB SSRAM, two 32 MB SDRAM, 8 MB Flash memory,
- SD Card socket,
- 24-bit audio CODEC with line-in, line-out, and microphone-in jacks
- VGA DAC (10-bit high-speed triple DACs) with VGA-out connector,
- 10/100 Ethernet controller,
- USB Host/Slave controller.

FPGA from *Altera* provides a support to soft-core processors. *Altera* provides the *NIOS II* soft-core processor through IP (**I**ntellectual **P**roperty) cores. Main features of the mentioned processor is [16]:

- separate instruction and data caches (512 B to 64 kB),
- optional MMU (**M**emory **M**anagement **U**nit) or MPU (**M**emory **P**rotection **U**nit),
- access to 2 GB of external address space,
- six-stage pipeline,
- single-cycle hardware multiply,
- hardware divide option,
- JTAG debug module.

Possibility of adding the MMU unit to the processor allows to run the full-featured Linux kernel, what extends the possibilities of creating stand-alone rough set device [18]. Software implemented in soft-core processor will perform control operations (e.g. data conversions, retrieving data from external sources, parallel execution synchronization), while hardware rough set units will do the calculations.

Current work emphases on implementing IO interface between external data source and the created device, creating hardware units performing basic rough sets notions such as indiscernibility relation, generating upper and lower approximations, calculating reducts and core. As soon as the device is finished, the results with comparison to the software implementation will be presented.

5. Conclusions and future research

The hardware implementation is the main direction of using scalable rough sets methods in real time solutions. Software implementations are universal, but rather slow.

Hardware realizations are deprived of this universality, however, allow us performing specific calculations in substantially shorter time.

The system with hardware implementation of rough sets methods can be used in embedded systems such as industrial controllers or as an alternative and very fast method of process control and data classification. The field of potential usage of the system can be very wide due to its versatility.

References

[1] A. Kanasugi, A. Yokoyama, A basic design for rough set processor, In The 15th Annual Conference of Japanese Society for Artificial Intelligence, 2001.

[2] A. Kanasugi, A design of architecture for rough set processor, JSAI 2001 Workshops, LNAI 2253, Springer-Verlag, 2001, pp. 406-412.

[3] A. Kanasugi, M. Matsumoto, Design and implementation of rough rules generation from logical rules on FPGA board, RSEISP 2007, LNAI 4585, Springer-Verlag, 2007, pp. 594-602.

[4] J. Koronacki, J. Cwik, Statystyczne systemy uczace sie, wydanie drugie, Exit, Warsaw, 2008, pp. 327.

[5] T. Lewis, M. Perkowski, L. Jozwiak, Learning in Hardware: Architecture and Implementation of an FPGA-Based Rough Set Machine, euromicro, vol. 1, 25th Euromicro Conference (EUROMICRO '99)-Volume 1, 1999, pp. 1326.

[6] M. Muraszkiewicz, H. Rybinski, Towards a Parallel Rough Sets Computer In: Rough Sets, Fuzzy Sets and Knowledge Discovery, Springer-Verlag, 1994, pp. 434-443.

[7] M. Muraszkiewicz, Sieci komorkowe do przetwarzania danych nienumerycznych, Prace IINTE, no. 52, 1984.

[8] Z. Pawlak, Elementary rough set granules: Toward a rough set processor. In: S. K. Pal, L. Polkowski, and A. Skowron, editors, Rough-Neurocomputing: Techniques for Computing with Words, Cognitive Technologies. Springer-Verlag, Berlin, Germany, 2004, pp. 5-14.

[9] Z. Pawlak, A. Skowron, Rudiments of rough sets. Information Sciences, 177(1) 2007, pp. 3-27.

[10] W. Pedrycz, A. Skowron, V. Kreinovich (Eds.), Handbook of Granular Computing, John Wiley & Sons, New York 2008.

[11] L. Rutkowski, Computational Intelligence, Methods and Techniques, Springer, 2008.

[12] A. Skowron, J. Stepaniuk, Tolerance Approximation Spaces, Fundamenta Informaticae, vol. 27, no. 2–3, 1996, pp. 245-253.

[13] J. Stepaniuk, Rough–Granular Computing in Knowledge Discovery and Data Mining, Springer, 2008.

[14] T. Strakowski, H. Rybinski, A Distributed Decision Rules Calculation Using Apriori Algorithm, T. Rough Sets 11, 2010, pp. 161-176.

[15] L. A. Zadeh, The role of fuzzy logic in the management of uncertainty in expert systems, Fuzzy Sets and Systems 11, 1993, pp. 199-227.

[16] Altera Corporation, `www.altera.com` , cited 30.11.2011.

[17] Terasic Corporation, `www.terasic.com.tw` , cited 30.11.2011.

[18] The Linux Kernel Archives, `www.kernel.org` , cited 30.11.2011.

[19] Xilinx Corporation, `www.xilinx.com` , cited 30.11.2011.

METODY ZBIORÓW PRZYBLIŻONYCH I IMPLEMENTACJE SPRZĘTOWE

Streszczenie Zbiory przybliżone (ang. rough sets) zostały wprowadzone przez Prof. Zdzisława Pawlaka jako narzędzie wnioskowania o pojęciach nieostrych (ang. *vague concepts*). Zarówno podstawy teoretyczne jak i zastosowania zbiorów przybliżonych zostały istotnie rozwinięte. Metody bazujące na zbiorach przybliżonych cieszą się bardzo dużym zainteresowaniem wielu środowisk na świecie.

Praca opisuje bieżące dokonania na polu implementacji sprzętowych w strukturach programowalnych FPGA (ang. **F**ield **P**rogrammable **G**ate **A**rray) metod zbiorów przybliżonych. Do tej pory stworzonych zostało zaledwie kilka takich rozwiązań. Większość istniejących implementacji metod zbiorów przybliżonych jest realizowanych programowo. Rozwiązanie programowe zapewnia uniwersalność działania pod względem przetwarzanych danych oraz wykonywanych algorytmów zapewniając jednocześnie prostotę ich modyfikacji, jednak jest relatywnie powolne. Implementacja sprzętowa ogranicza tą uniwersalność, dając jednak w zamian znaczny przyrost szybkości działania.

W pracy zawarto również krótki opis bieżących badań prowadzonych przez autorów nad stworzeniem tego typu implementacji. Do badań wykorzystywany jest układ FPGA firmy *Altera* o nazwie *Cyclone II*. Jest to układ o dużej pojemności zapewniający możliwość tworzenia procesorów typu soft-core wraz z modułami pozwalającymi na obsługę peryferiów płyty rozwojowej.

Słowa kluczowe: zbiory przybliżone, FPGA, programowalne struktury logiczne

Artykuł zrealizowano w ramach pracy badawczej S/WI/5/08.

COMPETING RISK ANALYSIS - GRAPHICAL REPRESENTATION OF VARIABLE INFLUENCE

Małgorzata Krętowska

Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: In the paper the possibilities of assessing the variable influence on the failure occurrence is shown. Ensemble of dipolar survival trees is used as a prediction tool. The technique is able to cope with censored data (data with incomplete observations) as well as with competing risks data. The results are presented on the base of two real datasets for which the influence of discrete and continuous variables is examined. To this purpose, the cumulative incidence functions and the quartiles of CIF functions are applied.

Keywords: competing risks, survival analysis, ensemble of survival trees, dipolar criterion

1. Introduction

Survival analysis often aims at discovering risk factors - variables that have great impact on failure occurrence. Failure, according to research field, may have different meanings. In medical domain it usually means death or disease relapse. In case of competing risks data there is not only one event under investigation. For each patient we may observe several events, but only the first one is noticed. Each observation (patient) is described by a set of covariates, the time of the first event occurrence and the failure indicator. The value of failure indicator points the type of the first event. The value equal to 0 means that for a given patients there were no events of interest. We only know its follow-up time. Such incomplete observations are called censored cases.

Discovering the risk factors from survival data may be done by using statistical methods, usually non-parametric or semi-parametric ones. Among the non-parametric methods we may distinguish tests for comparing two CIF functions (e.g. Gray's test, logrank test), the well known Cox model [3] belongs to semi-parametric techniques. The main problem with applying the Cox model to the data is a number

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 8, pp. 19-30, 2011.

of assumptions to fulfill. These requirements are often difficult to obey so alternative techniques are proposed.

Classification or regression trees are ones of the methods successfully used for competing risks. [2] and [6] describe similar approaches, where induction of the proposed between-node tree is based on the difference between cumulative incidence function. Additionally Callahan in [2] presents a within-node tree that use event-specific martingale residuals. The method proposed in [7] is available as an R-package. The method based on ensemble of survival tree for competing risk is presented in [9]. Induction of individual tree is based on minimization of, so called, dipolar criterion function created from dipoles. Dipoles are pairs of feature vectors, formed appropriately for a given problem.

In the paper we use the methodology described in [9]. Based on the results received from the ensemble, we try to test the variable influence on failure occurrence. The examination is done by use of CIF functions as well as the graphical representation of quartiles of a given event time. The experiments are performed on two real datasets described patients with follicular cell lymphoma and the other dataset - patients with breast cancer.

The paper is organized as follows. Section 2. describes the survival data with competing risks and introduces the idea of cumulative incidence function as well as the Kaplan-Meier survival function. In Section 3. short description of ensemble of dipolar survival tree is done. Experimental results are presented in Section 4.. They were carried out on the base of two real datasets describing the patients with breast cancer data and follicular type lymphoma. Section 5. summarizes the results.

2. Competing risks

In case of survival data with competing risks, at the beginning of the follow-up the patient is at risk of p ($p > 1$) different types of failure (Fig 1). Assuming that the time of occurrence for ith failure is T_i, we are interested only in the failure for which the time is the shortest $T = \min(T_1, T_2, \ldots, T_n)$. The learning sample L for competing risk data is defined as $L = (\mathbf{x}_i, t_i, \delta_i)$, $i = 1, 2, \ldots, n$, where $\mathbf{x}_i$ is N-dimensional covariates vector, t_i is the time to the first event observed and $\delta_i = \{0, 1, \ldots, p\}$ indicates the case of failure. δ_i equals to 0 represents censored observation, which means that for a given patient has not occurred any failure. Variable t_i represents the follow-up time.

The distribution of the random variable T (time), for an event of type i ($i = 1, 2, \ldots, p$) may be represented by several functions. One of the most popular is cumulative incidence function (CIF) defined as the probability that an event of type i

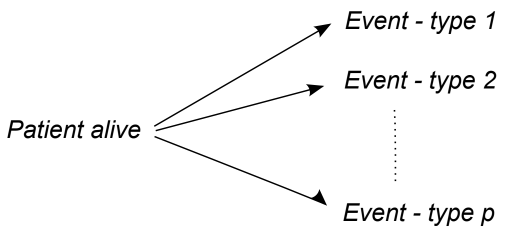


Fig. 1. Competing risks

occurs at or before time t [11]:

$$F_i(t) = P(T \leq t, \delta = i) \tag{1}$$

survival function

$$S_i(t) = P(T > t, \delta = i) \tag{2}$$

or hazard function

$$\lambda_i(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t, \delta = i | T \geq t)}{\Delta t} \tag{3}$$

The estimator of the CIF function is calculated as

$$\hat{F}_i(t) = \sum_{j|t_j \leq t} \frac{d_{ij}}{n_j} \hat{S}(t_{j-1}) \tag{4}$$

where $t_{(1)} < t_{(2)} < \ldots < t_{(D)}$ are distinct, ordered uncensored time points from the learning sample L, d_{ij} is the number of events of type i at time $t_{(j)}$, n_j is the number of patients at risk at $t_{(j)}$ (i.e., the number of patients who are alive at $t_{(j)}$ or experience the event of interest at $t_{(j)}$) and $\hat{S}(t)$ is the Kaplan-Meier estimator of the probability of being free of any event by time t. It is calculated as:

$$\hat{S}(t) = \prod_{j|t_{(j)} \leq t} \left(\frac{n_j - d_j}{n_j} \right) \tag{5}$$

where d_j is the number of events at time $t_{(j)}$. Examples of CIF function as well as Kaplan-Meier estimator are given in figure 2.

The "patients specific" cumulative incidence function for the event of type i is given by $\hat{F}_i(t|\mathbf{x}) = P(T \leq t, \delta = i | \mathbf{X} = \mathbf{x})$. The conditional CIF for the new patient with covariate vector $\mathbf{x}_{new}$ is denoted by $\hat{F}_i(t|\mathbf{x}_{new})$.

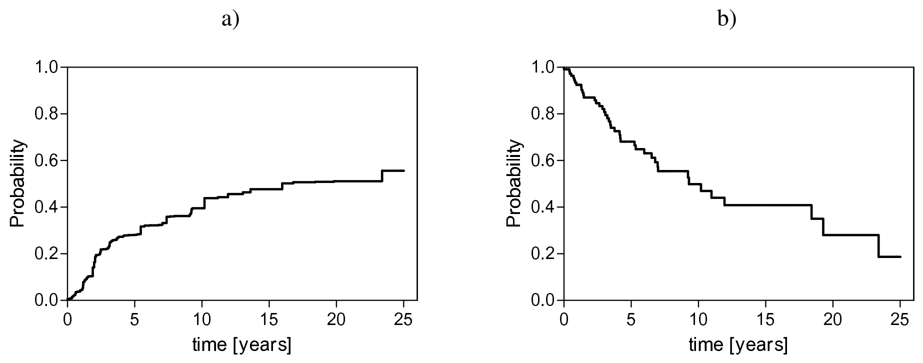


Fig. 2. Examples of a) CIF function; b) Kaplan-Meier estimator.

3. Prediction tool

The analysis of competing risks is performed by the use of ensemble of dipolar survival trees. Detailed description of induction of the ensemble is given in [9]. Below, one can find only the general information about the methodology used.

The general algorithm for generating ensemble of dipolar trees is as follows:

1. Draw k bootstrap samples $(L_1, L_2, \dots, L_k)$ of size n with replacement from L
2. Induction of dipolar survival tree $T(L_i)$ based on each bootstrap sample L_i
3. For each tree $T(L_i)$, distinguish the set of observations $L_i(\mathbf{x}_n)$ which belongs to the same terminal node as $\mathbf{x}_n$
4. Build aggregated sample $L_A(\mathbf{x}_n) = [L_1(\mathbf{x}_n), L_2(\mathbf{x}_n), \dots, L_k(\mathbf{x}_n)]$
5. Compute the Kaplan-Meier aggregated survival function for a new observation $\mathbf{x}_n$ as $\hat{S}^A(t|\mathbf{x}_n)$.
6. Compute the aggregated CIF functions for the ith type of failure for a new observation $\mathbf{x}_n$ as $\hat{F}_i^A(t|\mathbf{x}_n)$.

As one can see the main point in presented above algorithm is induction of dipolar survival tree for each generated bootstrap sample L_i. Each internal node contains a split, which tests the value of an expression of the covariates. In the proposed approach the split is equivalent to the hyper-plane $H(\mathbf{w}, \theta) = \{(\mathbf{w}, \mathbf{x}) :< \mathbf{w}, \mathbf{x} >= \theta\}$. The hyper-planes in the internal nodes of a tree are calculated by minimization of dipolar criterion function (detailed description may be found in [8]). This is equivalent to division of possibly high number of mixed dipoles and possibly low number of pure ones constructed for a given dataset.

The dipole [1] is a pair of different covariate vectors $(\mathbf{x}_i, \mathbf{x}_j)$ from the learning set. Mixed and pure dipoles are distinguished. Assuming that the analysis aims at dividing the feature space into such areas, which would include the patients with the same case of failure and similar survival times, pure dipoles are created between pairs of feature vectors with the same failure type, for which the difference of failure times is small, mixed dipoles - between pairs with distant failure times. Taking into account censored cases the following rules of dipole construction can be formulated:

1. a pair of feature vectors $(\mathbf{x}_i, \mathbf{x}_j)$ forms the pure dipole, if
   - $\delta_i \neq 0$ and $\delta_i = \delta_j = z$ and $|t_i - t_j| < \eta_z$, $z = 1, 2, \ldots, p$.
2. a pair of feature vectors $(\mathbf{x}_i, \mathbf{x}_j)$ forms the mixed dipole, if
   - $\delta_i \neq 0$ and $\delta_i = \delta_j = z$ and $|t_i - t_j| > \zeta_z$, $z = 1, 2, \ldots, p$
   - $(\delta_i = 0, \delta_j = z$ and $t_i - t_j > \zeta_z)$ or $(\delta_i = z, \delta_j = 0$ and $t_j - t_i > \zeta_z)$, $z = 1, 2, \ldots, p$

Parameters η_z and ζ_z are equal to quartiles of absolute values of differences between uncensored survival times for zth type of failure, $z = 1, 2, \ldots, p$. Basing on the earlier experiments, the parameter η_z is fixed as 0.3 quantile and ζ_z - 0.6.

The straightforward graphical representation of the results is the CIF function calculated for all the analyzed types of failure, for a new patient described by $\mathbf{x}_n$. Studying the influence of single variable for failure occurrence or the interaction of two variables, the tool may results the median value, lower quartile or any other centile of event occurrence for any type of failure. It enables drawing surfaces of a given statistics for different values of examined variables.

4. Experimental results

The experiments were performed on the base of two real datasets: breast cancer data and follicular type lymphoma data. The first analyzed dataset was used to show how to examine the influence of discrete variables for the failure (of any type) occurrence. Here, the cumulative incidence functions were used to model the failure prediction. In case of the other data, we presented the influence of continuous variables for the quartiles of CIF functions. We use here the lower quartile and the median values.

All the experiments were performed using the ensemble of 100 survival trees.

4.1 Breast cancer data

Breast cancer data [4] contain information about 641 women (50 years old or older) who had undergone breast-conserving surgery for an invasive adenocarcinoma 5 cm or less in diameter. They were randomly assigned to receive breast irradiation plus tamoxifen (321 women) or tamoxifen alone (320 women). The data were collected between 1992 and 2000. The last follow-up was conducted in summer 2002. Table 1 contains description of the variables [5].

Table 1. Description of variables in breast cancer data

Variable name	Description
tx	Randomized treatment: 1=tamoxifen, 2=radiation + tamoxifen
Variables assessed at the time of randomization	
pathsize	Size of tumor (cm)
hist	Histology: 1=ductal, 2=lobular, 3=medullary, 4=mixed, 5=other
hrlevel	Hormone receptor level: 0=negative, 1=positive
hgb	Haemoglobin (g/l)
nodedis	Whether axillary node dissection was done: 0=no, 1=yes
age	Age (years)
Outcome variables	
time	Time from randomization to event or last follow up (years)
d	Status at last follow up: 0=censored, 1=death, 2=relapse, 3=malignancy,

In figure 3 we can observe the differences between CIF functions calculated separately for tamoxifen alone and tamoxifen plus radiation for two event types: relapse and malignancy. The other variables were set up for theirs median values: pathsize=1.5; hist=1; hrlevel=1; hgb=135; nodediss=1; age=67. As we could observe the probability of relapse is greater in the group of patients treated with tamoxifen alone. Probability of malignancy is less for the group of patients treated with tamoxifen during the first 6 years of observations, later the probability in this group is greater then for patients treated with tamoxifen and radiation.

Figure 4 shows the influence of histology for the probability of relapse and malignancy. Other variable were set up for their median values (see description of figure 3). Two histological types were examined: $hist = 1$ (ductal) and $hist = 4$ (mixed). In figure 4a) we can observe significant differences between two CIF functions calculated for two types of histology. The patients with ductal histology treated only with tamoxifen have greater probability of relapse than patients with mixed histology. Such differences are not visible on figures representing the probability of malignancy (both for $tx = 1$ and $tx = 2$) and for the probability of relapse in group of patients treated with tamoxifen plus radiation.

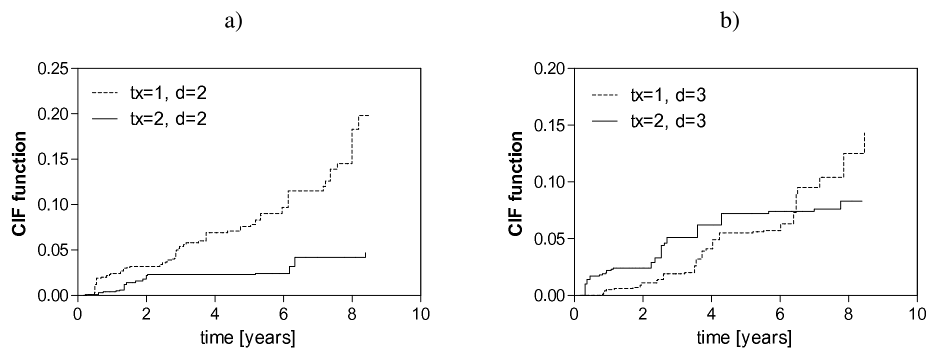


Fig. 3. CIF functions calculated for a) relapse ($d = 2$); b) malignancy ($d = 3$).

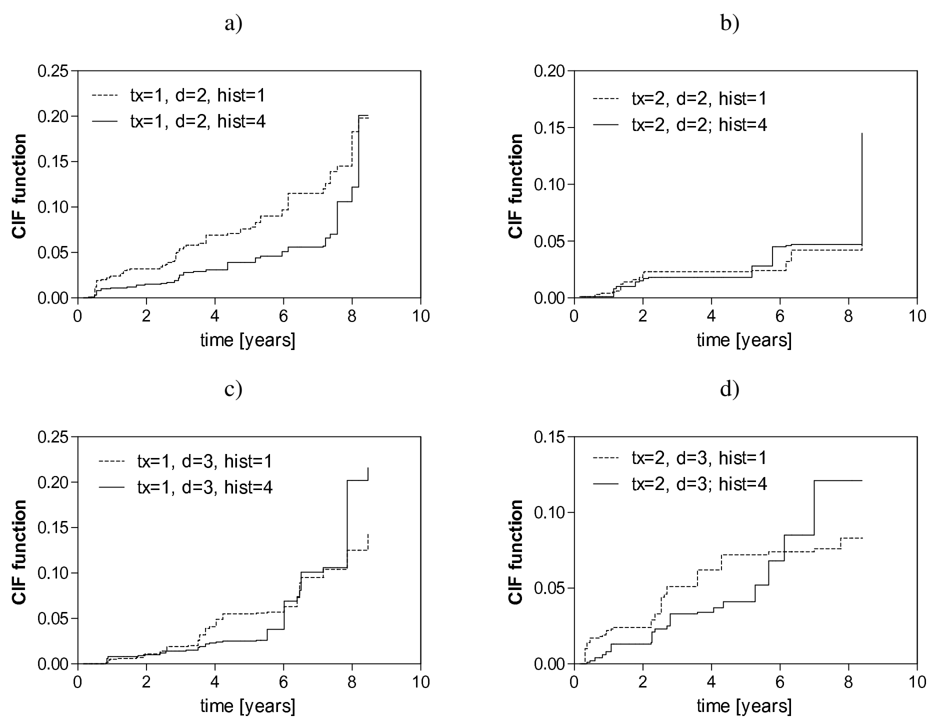


Fig. 4. CIF functions calculated for two types of histology: ductual and mixed for a) relapse ($d = 2$) and $tx = 1$; b) relapse and $tx = 2$; c) malignancy ($d = 3$) and $tx = 1$; d) malignancy and $tx = 3$.

4.2 Follicular cell lymphoma data

Lymphoma patient dataset was created at Princess Margaret Hospital, Toronto [10]. In the experiments we use the subset of 541 patients having follicular type lymphoma, registered for treatment at the hospital between 1967 and 1996, with early stage disease (I or II) and treated with radiation alone or with radiation and chemotherapy. Each patient is described by four variables, described in table 2.

Table 2. Description of variables in follicular type lymphoma data

Variable name	Description
Variables assessed at the time of diagnosis	
age	Age (years)
hgb	Haemoglobin (g/l)
clinstg	Clinical stage: 1=stage I, 2=stage II
ch	chemotherapy: 0=no, 1=yes
Outcome variables	
time	Time from diagnosis to event or last follow up (years)
d	Status at last follow up: 0=censored, 1=no response to treatment or relapse, 2=death

The event of interest is failure from the disease: no response to treatment or relapse. Competing risk type of event is death without failure. There are 272 event of interest and 76 observations with death without relapse.

On the base of lymphoma data the possibility of examination of the impact of continuous variables for probability of event occurrence is shown. For this purpose, described above algorithm of ensemble of dipolar trees generation should return the median value or the value of any other centile of the CIF function calculated for a new observation.

In figures 5 - 8 we can observe the quartiles of CIF function calculated for the first event (no response to treatment or relapse) for patients treated with radiation alone. In figures 5 and 6 the influence of age and hemoglobin for lower quartiles calculated for CIF functions for patients with clinical stage I and II is presented. The probability of relapse at a given value of the lower quartile is equal to 0.25. So the higher values of this statistics are connected with better prognosis for the patient.

For people with clinical stage I (Fig. 5) the lowest values of the first quartile are for haemoglobin at range 100-120. Here the influence of age is not visible. The failure prediction is better for young patients with haemoglobin in range 145-160 and for older patients (age: 50-65) and haemoglobin equal to 130-140. Here we can see the interaction of age and haemoglobin. For patients with clinical stage II (Fig. 6) we

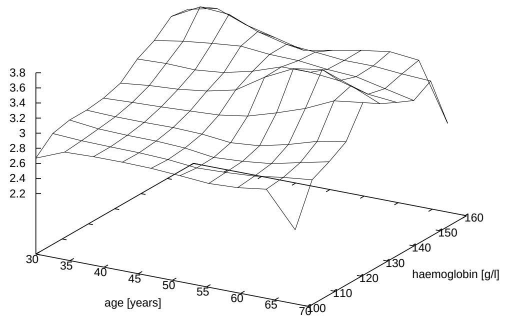


Fig. 5. The influence of age and hemoglobin for lower quartiles calculated for CIF functions ($d = 1$) for patients with clinical stage I

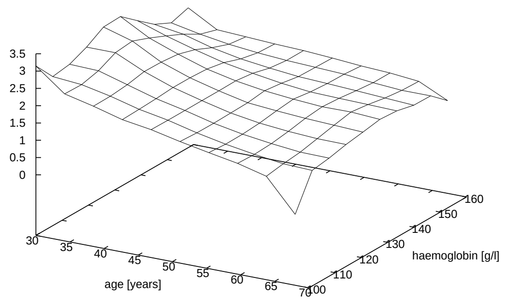


Fig. 6. The influence of age and hemoglobin for lower quartiles calculated for CIF functions ($d = 1$) for patients with clinical stage II

can observe the influence of age (younger people have better prognosis) and there is no influence of haemoglobin.

In figures 7 and 8 the influence of age and haemoglobin for the median values of CIF functions are presented. We can see that on average the medians are greater for people with clinical stage I (Fig. 7) than for people with clinical stage II (Fig. 8). In case of clinical stage I the best prediction is for younger people with greater value

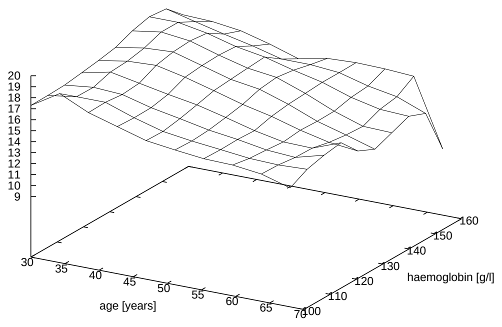


Fig. 7. The influence of age and hemoglobin for median values calculated for CIF functions for patients with clinical stage I

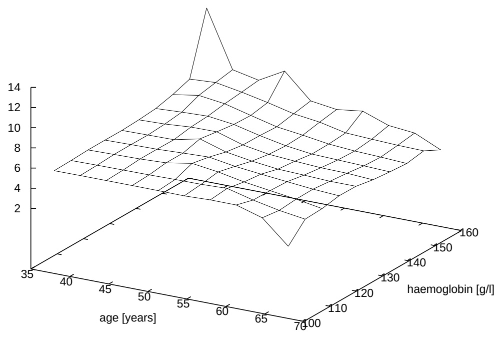


Fig. 8. The influence of age and hemoglobin for median values calculated for CIF functions for patients with clinical stage II

of haemoglobin. The impact of two examined continuous variables is not significant for patient with clinical stage II.

5. Conclusions

In the paper the possibilities of assessing the variables influence for the event occurrence is presented. The methodology based on the ensemble of dipolar survival trees is applied for this purpose. The experiments were performed on two real datasets. The first one - breast cancer data - is served as an example of discrete variables assessing. For this purpose the cumulative incidence functions were drawn for different values of discrete variables. In this case, two types of treatment and histology were used. For the other dataset, follicular type lymphoma data, the influence of two continuous variables for relapse occurrence were assessed. The surfaces of the lower quartile and median values calculated for the CIF functions for different values of age and haemoglobin were analyzed.

As one could see, presented graphs may suggest the influence of a given variable for failure occurrence exists and also may help to establish if the assumptions of statistical methods are fulfilled for examined data.

References

[1] L. Bobrowski, M. Krętowska, M. Krętowski, Design of neural classifying networks by using dipolar criterions, Proc. of the Third Conference on Neural Networks and Their Applications, Kule, Poland, 1997, pp. 689-694.

[2] F.M. Callaghan, Classification trees for survival data with competing risks, Univeristy of Pittsburgh, PhD. thesis, 2008.

[3] D.R. Cox, Regression models and life tables (with discussion), Journal of the Royal Statistical Society B **34**, 1972, pp. 187-220.

[4] A. W. Fyles, D. R. McCready, L. A Manchul., M. E. Trudeau, P. Merante, M. Pintilie, L. M. Weir, and I. A. Olivotto, Tamoxifen with or without breast irradiation in women 50 years of age or older with early breast cancer, New England Journal of Medicine 351, 2004, pp. 963-970.

[5] N.A. Ibrahim, A. Kudus, I. Daud, M.R. Abu Bakar, Decision tree for competing risks survival probability in breast cancer study, World Academy of Science, Engineering and Technology, 38, 2008, pp. 15-19.

[6] N.A. Ibrahim, A. Kudus, Decision tree for prognostic classification of multivariate survival data and competing risks, in: Strangio M. A. (Eds.), Recent Advances in Technologies, 2009.

[7] H. Ishwaran, U.B. Kogalur, R.D. Moore, S.J. Gange, B.M. Lau, Random survival forests for competing risks, 2010.

[8] M. Krętowska, Random forest of dipolar trees for survival prediction, L.Rutkowski et al. (Eds.), ICAISC 2006, LNAI 4029, 2006, pp. 909-918.

[9] M. Krętowska, Competing risks and survival tree ensemble, (submitted).

[10] M. Pintilie, Competing Risks: A Practical Perspective, John Willey & Sons, 2006.

[11] H. Putter, M. Fiocco, R.B. Geskus, Tutorial in biostatistics: Competing risks and multi-stage models, Statistics in Medicine **26**, 2007, pp. 2389-2430.

Małgorzata Krętowska

DANE Z KONKURENCYJNYM RYZYKIEM - GRAFICZNA REPREZENTACJA WPŁYWU CZYNNIKÓW RYZYKA

Streszczenie W pracy przedstawione zostały możliwości graficznej weryfikacji hipotez dotyczących wpływu poszczególnych cech na czas wystąpienia porażki. Jako narzędzie prognostyczne zostały wykorzystane predyktory złożone, w których dipolowe drzewa przeżycia służą jako pojedyncze predyktory. Algorytm tworzenia predyktorów złożonych wykorzystuje informację pochodzącą z obserwacji cenzorowanych, jak również jest przystosowany do danych z konkurencyjnym ryzykiem.
Eksperymenty zostały wykonane przy użyciu dwóch zbiorów danych: zbiór opisujący pacjentki z rakiem piersi i drugi - opisujący pacjentów z chłoniakiem grudkowym. Pierwszy z analizowanych zbiorów posłużył jako przykład do badania wpływu zmiennych dyskretnych. W tym celu wyznaczone zostały dystrybuanty (ang. cumulative incidence function) dla wyróżnionych dwóch zdarzeń konkurencyjnych i dwóch cech: rodzaju leczenia oraz typu histologicznego raka. W przypadku zbioru z chłoniakiem grudkowym badane były cechy ciągłe: wiek oraz wartość hemoglobiny. Analiza tych danych opierała się na wyznaczeniu wartości kwartyla pierwszego oraz mediany z funkcji dystrybuanty, wyznaczonej dla czasu nawrotu choroby.

Słowa kluczowe: dane z konkurencyjnym ryzykiem, analiza przeżyć, predyktory złożone, kryterium dipolowe

Artykuł zrealizowano w ramach pracy badawczej S/WI/2/08.

FEATURE SELECTION METHODS BASED ON MINIMIZATION OF CPL CRITERION FUNCTIONS

Tomasz Łukaszuk

Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: The feature selection is a method of data analysis commonly used as a preliminary step in the techniques of classification and pattern recognition. It is particularly important in situations when data are represented in high-dimensional feature space. Examples of these are collections of bioinformatics data, particularly data obtained from DNA microarrays. The paper presented two methods of feature selection based on minimizing the CPL criterion function: basic SEKWEM/GENET method, in which the selection of features is done in conjunction with the construction of a linear classifier separating objects from different decision classes, and the RLS method extending the primary method by linear separability relaxation stage in order to obtain a subset of features with better generalization ability. The results of the SEKWEM/GENET and RLS methods were confronted with the results obtained from other common feature selection methods in application to the state of the art microarray data sets.

Keywords: feature selection, CPL criterion function, SEKWEM/GENET algorithm, RLS method

1. Introduction

Nowadays, a lot of companies, administrative and scientific institutions has, and still collects data on various aspects of their bussinesses. Based on the collected data it is possible to carry out the necessary studies and obtain useful information and new knowledge. But often it happens that in the stage of data collection, test objects or phenomena are recorded with as large as possible number of parameters. Also, some types of data, research facilities, by their nature are described in a very large number of attributes. Examples of such data are digitized text and bioinformatics data.

The feature selection is a technique commonly used in data mining. Its aim is the selection from all available features the subset of features relevant to the considered

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 8, pp. 31-43, 2011.

problem [10]. Best subset should contain the minimum number of features which most affect the quality of the model relating to this problem.

Feature selection is also known as task that consists in removing irrelevant and redundant features from the initial data (features) set [14]. Irrelevant and redundant features means features with no or minimal effect on later decisions.

There are two ways of selecting features set. One consists in making a ranking of features according to some criterion and selecting certain number of the best features. The other is to select a minimum subset of features without learning performance deterioration [14]. In the second way the quality of the whole subset is evaluated.

Important aspects connected with feature selection are models and search strategies. Typical models are filter, wrapper, and embedded. Filter methods use some own internal properties of the data to select features. Examples of the properties are feature dependence, entropy of distances between data points, redundancy. In the wrapper methods the feature selection is connected with the other data analysis technique, such as classification, clustering algorithm, regression. The accompanying technique helps with evaluation of the quality of selected features set. An embedded model of feature selection integrates the selection in model building. An example of such method is the decision tree induction algorithm. At each node a feature has to be selected. Basic search strategies applied in feature selection are forward, backward, floating, branch-and-bound and randomized strategies [14]. Besides there are a lot of modifications and improvements of them.

This paper is engaged in the feature selection by minimization of a special convex and piece-wise linear (CPL) criterion function. The minimization process allows to calculate the parameters of hyperplane separating the learning sets and to find the best set of features ensured the linear separability of them at once.

The remainder of the paper is structured as follows: Section 2 provides a brief description of exploratory analysis techniques based on minimization of CPL criterion function, Sections 3 and 4 contain a more detailed introduction to developed by author feature selection methods SEKWEM/GENET and RLS. Section 5 presents the course and results of experiments involving the comparison of the SEKWEM/GENET and RLS methods with other feature selection methods. Finally, the work is summarized in Section 6.

2. The exploratory analysis techniques based on minimization of CPL criterion function

Let us consider that the test objects O_j $(j = 1, ..., m)$ are represented by the feature vectors $\mathbf{x}_j[n] = [x_{j1}, ..., x_{jn}]^T$ of the same dimensionality n or by points in the

n-dimensional feature space $F[n]$. Feature (attribute) x_i describes a specific numerical value of the i-th parameter, or the result of a specific i-th measurement made on each object O_j. Features can take discrete ($x_i \in \{0,1,...,p\}$) or continuous ($x_i \in \mathbf{R}^1$) values.

Let us take into consideration two disjointed sets C^+ and C^- composed of m feature vectors $\mathbf{x}_j$:

$$C^+ \cap C^- = \emptyset \; . \tag{1}$$

For example vectors from the first set represent patients suffered from certain disease and vectors from the second one represent patients without the disease. The *positive set* C^+ contains m^+ vectors $\mathbf{x}_j$ and the *negative set* C^- contains m^- vectors ($m = m^+ + m^-$).

We are considering the separation of the sets C^+ and C^- by the hyperplane $H(\mathbf{w},\theta)$ in the feature space $F[n]$.

$$H(\mathbf{w},\theta) = \{\mathbf{x} : \langle \mathbf{w},\mathbf{x} \rangle = \theta\} \tag{2}$$

where $\mathbf{w} = [w_1,...,w_n]^T \in \mathbf{R}^n$ is the weight vector, $\theta \in \mathbf{R}^1$ is the threshold, and $\langle \mathbf{w},\mathbf{x} \rangle$ is the inner product.

One way of finding the hyperplane $H(\mathbf{w},\theta)$ (2) is to minimize a properly defined criterion function $\Phi_\lambda(\mathbf{w},\theta)$ [3].

$$\Phi_\lambda(\mathbf{w},\theta) = \sum_{\mathbf{x}_j \in C^+} \alpha_j \varphi_j^+(\mathbf{w},\theta) + \sum_{\mathbf{x}_j \in C^-} \alpha_j \varphi_j^-(\mathbf{w},\theta) + \lambda \sum_{i \in I} \gamma_i \phi_i(\mathbf{w},\theta) \tag{3}$$

where $\alpha_j \geq 0, \lambda \geq 0, \gamma_i > 0, I = \{1,...,n\}$.
The nonnegative parameters α_j determine relative importance (*price*) of particular feature vectors $\mathbf{x}_j$. The parameters γ_i. represent the *costs* of particular features x_i.

The function $\Phi_\lambda(\mathbf{w},\theta)$ is the sum of the penalty functions $\varphi_j^+(\mathbf{w},\theta)$ or $\varphi_j^-(\mathbf{w},\theta)$ and $\phi_i(\mathbf{w},\theta)$. The functions $\varphi_j^+(\mathbf{w},\theta)$ are defined on the feature vectors $\mathbf{x}_j$ from the set C^+. Similarly $\varphi_j^-(\mathbf{w},\theta)$ are based on the elements $\mathbf{x}_j$ of the set C^-.

$$(\forall \mathbf{x}_j \in C^+) \quad \varphi_j^+(\mathbf{w},\theta) = \begin{cases} 1 + \theta - \langle \mathbf{w},\mathbf{x}_j \rangle & if \quad \langle \mathbf{w},\mathbf{x}_j \rangle < 1 + \theta \\ 0 & if \quad \langle \mathbf{w},\mathbf{x}_j \rangle \geq 1 + \theta \end{cases} \tag{4}$$

and

$$(\forall \mathbf{x}_j \in C^-) \quad \varphi_j^-(\mathbf{w},\theta) = \begin{cases} 1 + \theta + \langle \mathbf{w},\mathbf{x}_j \rangle & if \quad \langle \mathbf{w},\mathbf{x}_j \rangle > -1 + \theta \\ 0 & if \quad \langle \mathbf{w},\mathbf{x}_j \rangle \leq -1 + \theta \end{cases} \tag{5}$$

The penalty functions $\phi_i(\mathbf{w},\theta)$ are related to particular features x_i.

$$\phi_i(\mathbf{w},\theta) = \begin{cases} |w_i| & if \quad 1 \leq i \leq n \\ |\theta| & if \quad i = n+1 \end{cases} \tag{6}$$

The criterion function $\Phi_\lambda(\mathbf{w},\theta)$ (3) is the convex and piecewise linear (*CPL*) function as the sum of the *CPL* penalty functions $\varphi_j^+(\mathbf{w},\theta)$ (4), $\varphi_j^-(\mathbf{w},\theta)$ (5) and $\phi_i(\mathbf{w},\theta)$ (6). The basis exchange algorithm allows to find the minimum efficiently, even in the case of large multidimensional data sets C^+ and C^- [2].

$$\Phi_\lambda^* = \Phi_\lambda(\mathbf{w}^*,\theta^*) = min\,\Phi_\lambda(\mathbf{w},\theta) \geq 0 \tag{7}$$

The parameters $\mathbf{w}^*$ and θ^* define the hyperplane $H(\mathbf{w}^*,\theta^*)$ (2), which in an optimal way in terms of linear separability criterion measured by the value of function $\Phi_\lambda(\mathbf{w},\theta)$ (3) separates the data sets C^+ and C^-.

3. SEKWEM/GENET feature selection method

SEKWEM/GENET is the basic algorithm of feature selection based on the minimization of CPL criterion function $\Phi_\lambda(\mathbf{w},\theta)$ (3). Feature selection is done together with the search for the optimal hyperplane $H(\mathbf{w}^*,\theta^*)$ (2) separating the data sets C^+ and C^-. The resulting vector of parameters $\mathbf{w}^*$ may contain a number of factors w_i equal to or close to zero. This condition occurs especially in the case of multidimensional, so called, "long" data. Features x_i corresponding to the coefficients w_i equal or close to zero are rejected, while the features x_i corresponding to other coefficients form a set of selected features.

In order to simplify further considerations let us assume the following augmented form of the feature vectors $\mathbf{y}_j$ and the vector of parameters $\mathbf{v}$:

$$\mathbf{y}_j = [\mathbf{x}_j^T, 1]^T \tag{8}$$

$$\mathbf{v} = [\mathbf{w}^T, -\theta]^T \tag{9}$$

The equation of the separating hyperplane $H(\mathbf{w},\theta)$ (2) will take the form:

$$H(\mathbf{v}) = \{\mathbf{y} : \langle \mathbf{v},\mathbf{y} \rangle = 0\} \tag{10}$$

Determination of parameter $\mathbf{v}^*$ of the optimal hyperplane $H(\mathbf{v}^*)$ (10) is based on the basis exchange algorithm [2]. The algorithm searches in an oriented way vertices $\mathbf{v}^\mathbf{k}$ resulting from intersections of hyperplanes h_j^+, h_j^- and h_i (12), respectively associated with the features vectors $\mathbf{y}_j$ belonging to the sets C^+, C^- and the unit vectors $\mathbf{e}_i$ (11).

$$\begin{aligned} &\mathbf{e}_i = [e_{i1}, e_{i2}, ..., e_{ik}, ..., e_{in+1}]^T \\ &(\forall i \in \{1,...,n+1\})(\forall k \in \{1,...,n+1\}) \begin{cases} e_{ik} = 1 \; when \; i = k \\ e_{ik} = 0 \; when \; i \neq k \end{cases} \end{aligned} \tag{11}$$

$$\begin{array}{l}(\forall \mathbf{y}_j \in C^{+})\ h_j^{+} = \{\mathbf{v} : \langle \mathbf{y}_j, \mathbf{v}\rangle = 1\} \\ (\forall \mathbf{y}_j \in C^{-})\ h_j^{-} = \{\mathbf{v} : \langle \mathbf{y}_j, \mathbf{v}\rangle = -1\} \\ (\forall i\ \in (1, ..., n+1))\ h_i = \{\mathbf{v} : \langle \mathbf{e}_i, \mathbf{v}\rangle = 0\}\end{array} \tag{12}$$

Each vertex $\mathbf{v}^k$ is the intersection of at least $(n+1)$ hyperplanes h_j^{+}, h_j^{-}, h_i [4]. In the vertex $\mathbf{v}^k$ the following equations are fulfilled:

$$\begin{array}{l}(\forall j \in J_k^{+})\ (\mathbf{y}_j)^T \mathbf{v}^k = 1 \\ (\forall j \in J_k^{-})\ (\mathbf{y}_j)^T \mathbf{v}^k = -1 \\ (\forall i\ \in I_k)\ (\mathbf{e}_i)^T \mathbf{v}^k = 0\end{array} \tag{13}$$

where J_k^{+}, J_k^{-} are the sets of indexes of vectors $\mathbf{y}_j$ belonging respectively to the sets C^{+} and C^{-}, which is compliance with the equation (13), and I_k is the set of indexes of unit vectors $\mathbf{e}_i$ satisfying the last of the equations (13).

Equations (13) can be written in the form of a matrix [4]:

$$\mathbf{B}^k \mathbf{v}^k = \delta \tag{14}$$

$\mathbf{B}^k$ is called the base. The rows of $\mathbf{B}^k$ are formed by features vectors $\mathbf{y}_j$ $(j \in J_k^{+} \cup J_k^{-})$ or unit vectors $\mathbf{e}_i$ $(i \in I_k)$. δ margins is a vector with components equal to 1, -1 or 0 according to (13).

The coefficients v_i^k of the vector (vertex) $\mathbf{v}^k$ associated with the unit vectors $\mathbf{e}_i\,(i \in I_k)$ in base $\mathbf{B}^k$ are equal to zero ($v_i^k = 0$). This follows from the equality (13). Features x_i corresponding to coefficients v_i^k equal to zero may not be taken into account in the vertex $\mathbf{v}^k$. They do not affect the form of separating hyperplane $H(\mathbf{v}^k)$ (10). Vertex $\mathbf{v}^k$ is completely characterized by the subset of the features F^k:

$$F^k = \{x_i : i\ \in I_k'\} \tag{15}$$

where $I_k' = \{1, ..., n\} \setminus I_k$.

Minimizing of the criterion function $\Phi_\lambda(\mathbf{v})$ (3) according to the basis exchange algorithm comes down to appropriate movement between the vertices $\mathbf{v}^k$ until the optimal vertex $\mathbf{v}^*$ is found. Each transition from vertex $\mathbf{v}^k$ to the vertex $\mathbf{v}^{k+1}$ is associated with the replacement of one vector in the base $\mathbf{B}^k$. If a unit vector $\mathbf{e}_l$ exits from the base $\mathbf{B}^k$, it means changing the consideration from the subset of features F^k to the extended subset of features $F^{k+1} = F^k \cup \{x_l\}$. If a unit vector $\mathbf{e}_r$ enters to the base $\mathbf{B}^k$, it means changing the consideration from the subset of features F^k to the reduced subset of features $F^{k+1} = F^k \setminus \{x_r\}$.

Considering the above facts, the process of minimizing the criterion function $\Phi_\lambda(\mathbf{v})$ (3) according to the basis exchange algorithm is connected with a browsing of subsets of features F^k (15) characterized by the vertices $\mathbf{v}^k$. The optimal vector $\mathbf{v}^*$ corresponds to the optimal (in the sense of linear separability criterion of the sets C^+ and C^- (1)) subset of features F^*.

$$F^1 \rightarrow F^2 \rightarrow ... \rightarrow F^k \rightarrow F^{k+1} \rightarrow ... \rightarrow F^* \tag{16}$$

4. RLS feature selection method

A fact that a model behaves very well in relation to objects from training set does not guarantee that equally well handle with objects inactive in the learning process. It is due to the danger of overfitting. It is worth, by reduction the model quality in relation to training data, to obtain better performance in conjunction with test data [17]. This idea underlies the extended feature selection methods, the relaxed linear separability (RLS) method [4].

The RLS method consists of two calculations stages. In the first stage, in accordance with the previously described basic feature selection scheme, there are determined the optimal subset of features F^* (16) and the optimal parameter vector $\mathbf{v}^*$ (7). In the second stage a linear separability relaxation is performed. The linear separability relaxation consists in the controlled removal from the subset F^* (16) the consecutive least significant features and evaluation so obtained subsets of features [4].

Selecting a feature to remove from the subset F^* (16) (and the next resulting subsets of features) is done through the appropriate increasing the value of cost parameter λ occurring in the expression of the criterion function $\Phi_\lambda(\mathbf{v})$ (3). After increasing the value of parameter λ, optimization of the criterion function $\Phi_\lambda(\mathbf{v})$ is performed. The optimization starts from the previously specified vertex $\mathbf{v}^*$ (7). If the value of λ was increased enough, it will lead to an increase the number of unit vectors $\mathbf{e}_i$ in base $\mathbf{B}^{*1}$ associated with the new optimal vertex $\mathbf{v}^{*1}$, and thus to reduce the number of features of optimal subset of features F^{*1} [4]. Further enhancing the value of parameter λ allows to obtain the next subsets of features F^{*k} with a reduced number of features. It is possible to control the value of λ to obtain a sequence of subsets of features $F^{*1}, F^{*2}, ..., F^{*p}$, where each subset $F^{*(k+1)}$ is equal to the subset F^{*k} minus one least significant feature. The last subset F^{*p} has only one feature.

The measure of quality of feature subsets F^{*k} is the classifier error $e_{LOOCV}(F^{*k})$ estimated by leave-one-out cross validation [7] on the set consisting of all objects from subsets C^+ and C^- (1) with reduced features not belonging to subset of features of F^{*k}. The error $e_{LOOCV}(F^{*k})$ is equal to the fraction of incorrectly classified objects with one-element test sets created in the validation process.

$$e_{LOOCV}(F^{*k}) = m_{LOOCV}(F^{*k})/m \tag{17}$$

where $m_{LOOCV}(F^{*k})$ is the number of misclassified objects, and m is the total number of objects in sets C^{+} and C^{-} (1).

The RLS method as the best resulting subset of features considers the subset with the smallest error $e_{LOOCV}(F^{*k})$. If there is more than one subset of features with the smallest error $e_{LOOCV}(F^{*k})$, RLS selects the least numerous subset.

5. Empirical studies

5.1 Experimental setup

Three benchmarking feature selection algorithms were selected for an experimental comparison with the SEKWEM/GENET and RLS methods. One of the selected algorithms, ReliefF, is based on feature ranking procedure proposed by Kononenko [13] as an extension of the Relief algorithm [12]. The ReliefF searches for the nearest objects from different classes and weights features according to how well they differentiate these objects. The second one is a subset search algorithm denoted as CFS-SF (Correlation-based Feature Subset Selection - Sequential Forward) [11]. The CFS-SF algorithm is based on a correlation measure which evaluates the goodness of a given feature subset by assessing the predictive ability of each feature in the subset and a low degree of correlation between features in the subset. The third method, Consistency Subset Evaluation - Selection Forward (CSE-SF) also belongs to the subset search selection methods. It searches the space of solutions using the forward selection procedure, and evaluates the found subsets of features using inconsistency measure proposed by Liu and Setiono at work [15] and then developed in the work [5].

The applied algorithms require the determination of certain parameters controlling their work and having an impact on the results returned. The author used, in most cases, the standard parameters recommended by the creators of algorithms.

Studied feature selection methods were compared on the basis of the returned feature space quality. The quality of the feature space was evaluated based on its discriminative power. Four frequently used classification methods and the CPL method were applied to assess the discriminative power of selected feature spaces:

- k Nearest Neighbours (kNN) [6] with $k = 5$ (arbitrary choice)
- Support Vector Machines (SVM) [18] with linear kernel function

- Naive Bayes Classifier (NBC) [6]
- C4.5 Decision Tree Algorithm (C4.5) [16]
- Convex and Piecewise-Linear criterion functions (CPL) [3] with linear relaxation [4]

These five classifiers were designed in the full feature spaces and in the reduced feature subspaces obtained by the feature selection methods. The result characterizing the effectiveness of a classifier for a given data set is the fraction of misclassified objects from the testing set in the process of leave-one-out cross-validation [7]. The effectiveness of a classifier is an assessment of the quality of the feature space and, consequently, a part of assessment of the feature selection method.

The four first classifiers were designed by using Weka's implementation [20]. The Weka's implementation of ReliefF, CSE-SF and CFS-SF was used also for the feature selection and cross validation evaluation of designed classifiers. The CPL classifiers (the fifth type) based on the search for optimal separating hyperplane through minimization of the CPL criterion functions was applied using author's own implementation. Autor's implementation was also used for the SEKWEM/GENET and RLS methods of feature selection. Calculations were performed on a computer with Intel Core2 T5500 processor and 1GB of RAM.

5.2 Data sets

The experiments were carried out on publicly available data sets concerning classification problems related to four different diseases: colon cancer, leukemia, lung cancer and breast cancer.

The Colon cancer [1] contains expression levels of 2000 genes taken in 62 different samples. For each sample it is indicated whether it came from a tumor biopsy or not.

The Leukemia [8] data set contains expression levels of 7129 genes taken over 72 samples. Labels of objects indicate which of two variants of leukemia is present in the sample: acute myeloid (AML, 25 samples), or acute lymphoblastic leukemias (ALL, 47 samples).

The Lung cancer [9] is made of 181 patients with 12533 markers. The samples belong to two lung cancer classes, malignant pleural mesothelioma (MPM, 31 samples) and adenocarcinoma (ADCA, 150 samples).

The Breast cancer [19] data set describes the patients tested for the presence of breast cancer. The data contains 97 patient samples, 46 of which are from patients who had developed distance metastases within 5 years (labelled as "relapse"), the rest 51 samples are from patients who remained healthy from the disease after their initial diagnosis for interval of at least 5 years (labelled as "non-relapse"). The number of genes is 24481.

Original data sets come with training and test samples that were drawn from different conditions. Here we combine them together for the purpose of cross validation. Data have also been standardized before experiment.

Table 1. The data sets used in testing the feature selection methods

Name	#objects	#features	class sizes	
Colon cancer	62	2000	tumor	normal
			40	22
Leukemia	72	7129	ALL	AML
			47	25
Lung cancer	181	12533	MPM	ADCA
			31	150
Breast cancer	97	24481	relapse	non-relapse
			46	51

5.3 Results

Table 2 summarizes the results of examined feature selection methods obtained in the previously described experiment.

System resources are unfortunately insufficient to apply the CFS-SF method to the lung cancer and breast cancer data sets. Available RAM is insufficient compared to the memory complexity of the algorithm.

Comparing the classification errors obtained on the full data sets ("No selection" group of rows), and classification errors on the data sets composed of features selected by each method (next groups of rows) it should be noted that each of the feature selection methods returns a subset of features improving the properties of classifier. It is in line with expectations and the idea of feature selection. However, the improvement of the quality of classifier is different in relation to particular methods. The methods developed by the author (SEKWEM/GENET and RLS) proved to be significantly better compared to other studied methods. This is clearly evident when compared the values of mean errors obtained for all used classification algorithms, listed in Table 2 in the rows "average".

The second criterion of evaluation the feature selection methods is the number of features returned by the procedure. In this aspect, by far the best method is

Table 2. Comparison of feature selection algorithms in terms of number of selected features and classification errors estimated by leave-one-out cross-validation method

		Colon cancer	Leukemia	Lung cancer	Breast cancer
No selection	#features	2000	7129	12533	24481
	kNN	20,97%	15,28%	6,08%	39,18%
	SVM	16,13%	1,39%	1,11%	31,96%
	NBC	16,13%	0,00%	2,21%	47,42%
	C4.5	20,97%	26,39%	3,87%	42,27%
	CPL	9,68%	2,78%	1,11%	25,77%
	average	16,78%	9,17%	2,88%	37,32%
SEKWEM/GENET	#features	39	43	64	78
	kNN	8,06%	0,00%	0,55%	1,03%
	SVM	0,00%	0,00%	0,00%	0,00%
	NBC	6,45%	0,00%	0,55%	8,23%
	C4.5	33,97%	16,67%	2,76%	29,90%
	CPL	0,00%	0,00%	0,00%	0,00%
	average	9,68%	3,33%	0,77%	7,83%
RLS	#features	14	7	3	19
	kNN	8,06%	0,00%	0,00%	5,15%
	SVM	0,00%	0,00%	0,00%	0,00%
	NBC	4,84%	0,00%	2,76%	9,28%
	C4.5	19,35%	12,50%	3,87%	27,84%
	CPL	12,50%	6,67%	3,72%	21,21%
	average	8,95%	3,83%	2,07%	12,70%
ReliefF	#features	15	32	393	43
	kNN	12,90%	4,17%	3,31%	15,46%
	SVM	11,29%	2,78%	0,55%	23,71%
	NBC	9,68%	4,17%	0,55%	20,62%
	C4.5	20,97%	16,68%	3,31%	35,05%
	CPL	12,90%	5,56%	1,11%	19,59%
	average	13,55%	6,67%	1,77%	22,89%
CSE-SF	#features	4	3	2	6
	kNN	17,74%	4,17%	2,21%	27,83%
	SVM	12,90%	5,56%	2,21%	29,90%
	NBC	11,29%	5,56%	2,76%	46,39%
	C4.5	8,06%	5,56%	1,11%	29,90%
	CPL	12,90%	8,33%	2,21%	31,96%
	average	12,58%	5,84%	2,10%	33,20%
CFS-SF	#features	58	81	n/a	n/a
	kNN	8,06%	1,39%	n/a	n/a
	SVM	12,90%	1,39%	n/a	n/a
	NBC	8,06%	0,00%	n/a	n/a
	C4.5	12,90%	20,83%	n/a	n/a
	CPL	12,90%	4,17%	n/a	n/a
	average	10,96%	5,56%	n/a	n/a

the CSE-SF. Among the author's methods much better on this criterion falls RLS algorithm. The SEKWEM/GENET method is characterized by selecting relatively numerous subsets of features.

On the basis of the results it can be quite definitely say that SEKWEM/GENET and RLS methods very well fit for the purpose of feature selection in relation to high dimensional data sets. They choose subsets of features with high quality, using which makes it possible to build much better classification and decision-making rules than on the basis of the starting sets of features.

6. Concluding remarks

The paper presents basic assumptions of the SEKWEM/GENET and RLS methods of feature selection. The basis of both methods is the minimization of the special CPL criterion function. The work also contains the results obtained from applying of described methods with the state of the art high dimensional microarray data. In comparison with other commonly used feature selection methods the author's methods proved to be better considering the quality of the returned sets of features. The measure of the quality of a subset of features is the classification error obtained in the process of leave-one-out cross-validation.

The experiment described in the article is a repetition of the experiment made by the author for his doctoral dissertation. Some of the results obtained this time is a bit different than the results shown in the dissertation. The reason for this is ongoing work on the development and improvement of the SEKWEM/GENET and RLS methods. The results presented in this article have been obtained on the following slightly modified versions of the implementations of the SEKWEM/GENET and RLS algorithms. Recently completed and potential future development activities are aimed at improving the quality of the results, but also take into account emergency situations that occur sometimes after applying the algorithm to a new custom data set.

References

[1] U. Alon, et al., Broad patterns of gene expressions revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays, PNAS, 96:6745–6750, 1999.

[2] L. Bobrowski, Design of Piecewise Linear Classifiers from Formal Neurons by Some Basis Exchange Technique, pp. 863–870 in: Pattern Recognition, 24(9), 1991.

[3] L. Bobrowski, Data mining based on convex and piecewise linear (CPL) criterion functions (in Polish), Wyd. Politechniki Białostockiej, Białystok, 2005.

[4] L. Bobrowski L., T. Łukaszuk, Feature Selection Based on Relaxed Linear Separability, pp. 43–59 in: Biocybernetics and Biomedical Engineering, 29(2), 2009.

[5] M. Dash, H. Liu, Consistency-based search in feature selection, Artificial Intelligence, 151:155–176, 2003.

[6] O.R. Duda, P.E. Heart, D.G. Stork, Pattern Classification, Second edition, John Wiley & Sons, 2001.

[7] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, San Diego, 1990.

[8] T.R. Golub, et al., Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, Sciences, 286, pp. 531-537, 1999.

[9] J.G. Gordon, R.V. Jensen, L. Hsiao, S.R. Gullans, Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesotheliomar. Cancer Res, 62:4963–4967, 2002.

[10] I. Guyon I., A. Elisseeff, An introduction to variable and feature selection, Journal of Machine Learning Research, 3:1157–1182, 2003.

[11] M.A. Hall, Correlation-based Feature Selection for Machine Learning, PhD thesis, University of Waikato, Dept. of Computer Science, 1998.

[12] K. Kira K., L.A. Rendell, A Practical Approach to Feature Selection, Ninth International Workshop on Machine Learning, 1992, 249-256.

[13] I. Kononenko, Estimating Attributes: Analysis and Extensions of RELIEF, European Conference on Machine Learning, 1994, 171-182.

[14] H. Liu, H. Motoda, Computational methods of feature selection, Chapman & Hall/CRC data mining and knowledge discovery series, Chapman & Hall/CRC, 2008.

[15] H. Liu, R. Setiono, A Probabilistic Approach to Feature Selection - A Filter Solution, 13th International Conference on Machine Learning, pp. 319–327, Morgan Kaufmann, 1996.

[16] J.R. Quinlan, C4.5 - Programs for Machine Learning, Morgan Kaufmann, 1993.

[17] I.V. Tetko, D.J. Livingstone, A.I. Luik, Neural network studies, 1. Comparison of overfitting and overtraining, Journal of Chemical Information and Computer Sciences, 35(5):826–833, 1995.

[18] V.N. Vapnik, Statistical Learning Theory, J. Wiley, New York, 1998.

[19] L.J. van't Veer, et al., Gene expression profiling predicts clinical outcome of breast cancer, Nature, 415(6871), pp. 530-536, 2002.

[20] I.H. Witten, E. Frank, Data Mining - Pracitcal Machine Learning Tools and Techniques with JAVA Implementations, Morgan Kaufmann Publishers, 2000.

METODY SELEKCJI CECH BAZUJĄCE NA MINIMALIZACJI FUNKCJI KRYTERIALNYCH TYPU CPL

Streszczenie Selekcja cech jest metodą analizy danych powszechnie stosowaną jako wstępny krok w technikach klasyfikacji czy rozpoznawania wzorców. Ma ona szczególne znaczenie w sytuacji gdy dane reprezentowane są w wysoko wymiarowej przestrzeni cech. Przykładem takich danych są zbiory bioinformatyczne, a w szczególności dane uzyskane na podstawie mikromacierzy DNA. W pracy przedstawione zostały dwie metody selekcji cech bazujące na minimalizacji funkcji kryterialnych typu CPL: podstawowa metoda SEKWEM/GENET, w której selekcja cech dokonywana jest w połączeniu z budową liniowego klasyfikatora separującego obiekty z różnych klas decyzyjnych, oraz metoda RLS rozszerzająca podstawową metodę o etap relaksacji liniowej separowalności w celu uzyskania podzbioru cech o lepszych zdolnościach generalizacji. Wyniki metod SEKWEM/GENET i RLS zostały także skonfrontowane z wynikami uzyskanymi z innych popularnych metod selekcji cech w zastosowaniu do „benchmarkowych" zbiorów danych mikromacierzowych.

Słowa kluczowe: selekcja cech, funkcja kryterialna typu CPL, algorytm SEKWEM/GENET, metoda RLS

Artykuł zrealizowano w ramach pracy badawczej S/WI/2/2008.

THE CONCEPT OF NEURAL NETWORK APPLICATIONS TO THE ANALYSIS OF WEATHER PARAMETERS FOR RISK PREDICTION

Andrzej Mitas[1], Marcin Bernaś[2], Marcin Bugdol[1], Artur Ryguła[3]

[1] Faculty of Biomedical Engineering, Silesian University of Technology, Gliwice, Poland

[2] Faculty of Transport, Silesian University of Technology, Katowice, Poland

[3] APM, Bielsko-Biała, Poland

Abstract: The weather, especially its fluctuations, has a significant impact on the road driving conditions - leads to a number of collisions and as a consequence to hundreds of deaths, thousands injured and millions of economic losses. The article briefly specifies the methods of particular weather elements description which can be found in literature and their impact on the road conditions. Additionally, weather parameters analysis was performed. On its basis adaptation of a neural network to predict temperature changes and the possibility of icing was proposed. The tests and results obtained by using the designed neural network could be used in fuzzy expert systems.

Keywords: weather prediction, neural networks, transport safety

1. Introduction

Twenty-first century is a period of dynamic climate changes. The weather, especially its fluctuations, has a significant impact on the road driving conditions. In conjunction with increasing traffic in Poland, it leads to a number of collisions and as a consequence to hundreds of deaths, thousands injured and millions of economic losses. To determine the dynamically changing road conditions, the structure of the road, the local topology, traffic and weather conditions need to be considered [1]. The mentioned parameters are affected by, among others, meteorology and geography factors as well as current road works.

The article briefly specifies the methods of particular weather elements description which can be found in literature and their impact on the road conditions. The presented methods were used to experimentally analyse the obtained data. On the basis

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 8, pp. 45-59, 2011.

of the results an adaptation of a neural network to predict temperature changes and the possibility of icing has been introduced.

2. Algorithm

Variability of the weather is a cyclical phenomenon with a period of 24 hours and so it should be analysed. In addition, it varies depending on weather fronts, the detection of which requires a regional / global system. The research literature [3] indicates a close relationship between the specified factors and the deterioration of the weather road conditions.

The study [5] shows correlation between the changes in driving style and the presence of rainfall. Precipitations and their intensity affect the number of overtaking manoeuvres while maintaining larger acceptable gaps between vehicles. The vehicles speed reduction degree is often disproportionate to the loss of adhesion, caused by an extra layer of water on the tire-road contact surface [4]. The imposition result of many weather negative effects (rain or snow and reduced visibility - less than 100m) often leads to a collision several in which several vehicles take part. Rainfall can be measured directly - through the rain gauge.

The phenomenon of the visibility reduction is especially dangerous when it does not occur gradually but suddenly (so-called wall of fog). This leads to frequent collisions caused by drivers different reactions: heavy braking, lack of response or speed reduction. The last reaction is often accompanied by a dangerous reduction of vehicles gaps.

The visibility limitation is mostly caused by fog (air saturated with water molecules), smoke from fires or industrial centres as well as sunrise and sunset (hour interval). Detecting this phenomenon is often complicated. It requires the use of humidity sensors (calibrated for values close to 100%) and directional light sensor or calendar data.

The study [14] indicates a large influence of winds, especially the side gusts, on the vehicles drive ability. Strong wind gusts may cause changes in vehicles handling characteristics (especially motorcycles, buses and trucks) which in consequence may lead to lane change or sliding off the road. The vehicles in open areas: highways without shielding and viaducts are exposed to the wind gusts. The wind above 10 m/s can even be a threat when it is accompanied by wet (slippery) surface or fog. For detection of wind gusts anemometers or barometers recording radical change in pressure, corresponding to passing atmospheric fronts, are used.

Temperature fluctuations around zero, combined with high humidity or rain, can cause the following weather phenomena: the ice, black ice or rime formation. Additionally, extremely high temperatures can affect the driver reaction rate, which increases with the length of travelled route [3].

Detection of icing is possible using sophisticated road embedded sensors or ultrasound, however for their prediction temperature, humidity, dew point, precipitation strength and concentration of the brine are required.

Summarizing, hazardous weather events which have direct impact on the safety are: black ice, hard rime, precipitation, limited visibility, heavy gusts of wind as well as, especially dangerous for traffic, local weather anomalies interpreted as abrupt weather conditions changes. On the basis of studies [4,6,7,17,18] the weather parameters affecting the road conditions (Table 1) have been determined:

Table 1. Weather parameters affecting the road conditions

Weather event	Road condition	Influence on the road transport
rain, snow, hail, flood	reduced visibility, reduced friction surface, covering the road surface marking, damage to infrastructure	reduced road capacity and travel speed, variable travel speed, the possibility of roads, bridges and viaducts closure
strong wind	reduced visibility, infrastructure damage, reduction of vehicle efficiency	increased latency, reduced speed, the possibility of roads, bridges and viaducts closure
fog, smoke	reduced visibility	reduced speed, increased delays, variable travel speed , the possibility of roads, bridges and viaducts closure
lightning, a radical change in temperature	infrastructure damage	ITS system failures, power failures, communication problems

The RWIS system [1] uses predictions based on numerical models to determine the parameters of the weather over a given time horizon. Generated forecast includes assumed surface temperature and road conditions. Using the history of measurements, the system tests the sensitivity of the parameters (their influence on the weather) in different weather conditions. At this stage invalid data are eliminated - incorrect predictions for local road conditions.

Prediction errors occur most frequently by high-sensitivity parameters such as: rain, cloud type and wind speed. Cloudiness is the main cause of errors in numerical models. Wind, at night, impacts the surface temperature through faster cooling. Dew point has little effect on surface temperature change, however is a key parameter to be taken into account in the detection of icing. Additionally, based on study [8] it can be assumed that the creation of black ice or other hazardous conditions (such as mud slip) is a result of simultaneous occurrence of the following factors:

- air temperature in the range from -6°C to +1°C,
- relative humidity greater than 85%,
- variable substrate temperature from positive to negative,
- concentration of the brine on the road.

The RWIS model [1] defines the following classes of slippery surfaces (Table 2):

Table 2. Weather parameters affecting the road conditions

Circumstances	**Type (danger level)**
$T_{air} < T_{freezing\ point}$ and precipitation > 0 and precipitation type = rain	Ice on road surface (high)
$T_{air} < T_{freezing\ point}$ and precipitation > 0 and precipitation type = snow	Lingering snow (average)
$T_{air} < T_{dew\ point} < T_{freezing\ point}$ and precipitation>0 and precipitation type = snow and humidity >80%	Linger snow + hoar-frost (high)
$T_{air} < T_{dew\ point} < T_{freezing\ point}$	Possible hoar-frost (low)
$T_{air} < T_{dew\ point} < T_{freezing\ point}$ and humidity >80%	Black ice (average-high)
$T_{air} < T_{dew\ point} < T_{freezing\ point}$ Wind strength > V_c	Strong hoar-frost (average)
$T_{air} < T_{dew\ point} < T_{freezing\ point}$ Wind strength < V_c	Possible hoar-frost (low)
$T_{air} < T_{freezing\ point}$ and $T_{air-history} > T_{freezing\ point}$ and precipitation _ history>0	Mud or ice on road surface (high)

V_c is determined taking into account the climatic conditions of analysed area (road section)

$T_{freezing\ point}$ – freezing temperature of liquid lingering on the road surface (for water - 0°C)

The parameter V_c in the temperate climate zone is usually assumed to 10 m/s. The $T_{freezing\ point}$ parameter value usually has a value between 0 to 1°C, corresponding to the temperature of water freezing and taking into account the measurement error.

The described analysis indicates a coincidence of factors such as: relative humidity, rainfall, dew point and temperature fluctuations. The order of occurrence of weather conditions is important as well. The most dangerous conditions occur when the favourable conditions change to unfavourable. For example, the temperature changes of only 1°C may result in multiple adhesion reduction, which significantly increases braking distances.

With the use of anti-freezing substance a constant value of 0°C is replaced by the value from the curve showed on figure 1. The presented curve corresponds to the solution concentration of the freezing point. For different types of concentration and coagulation the curve shape can have various forms. The road section considered in the researches was brined in a substance, which lowered the freezing point to -21°C for the 23% solution.

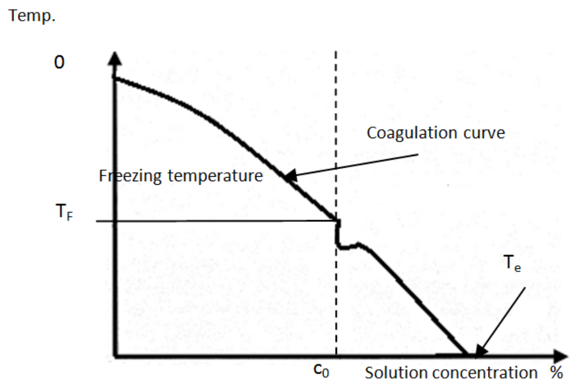


Fig. 1. Solution concentration dependence on the temperature changes [17]

In order to verify the assumptions of the described models a statistical module has been developed containing 9-month historical data from the city of Chorzow (Poland - Upper Silesia region). The data contain the collision information from the police database (SEWIK) and the meteorological parameters from a local weather station. The research has been enriched by the data obtained from the ICM model prediction [4,11]. The first results show the relationships between the parameters: humidity and precipitation (fig. 2).

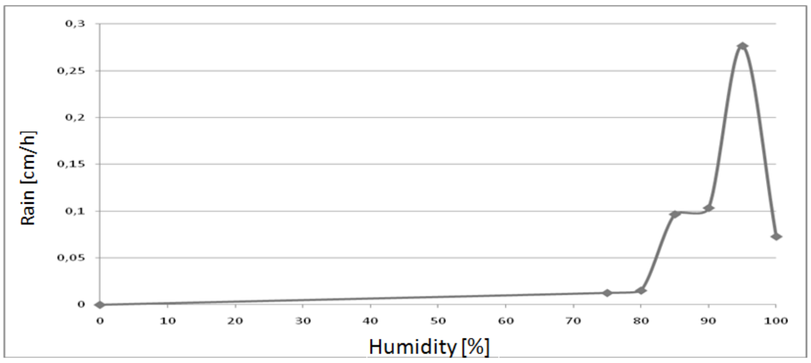


Fig. 2. Dependencies of rain occurrence at a specific humidity [2]

Measurements and the study have been conducted according to the weather parameters for the various weather phenomena: black ice and strong winds. As a statistical measure the arithmetic and geometric means, standard deviation, maximum and minimum, skewness and kurtosis have been used.

At the beginning data from days on which black ice occurred have been analysed. Additionally, for comparative purposes, the parameters for the days on which there was no icing (there was only wet surface - water film thickness greater than 0) have been set.

The parameter distinguishing between the two mention groups is the temperature of the road after taking into account the brine concentration and the dew point.

To determine the complete characteristics of the factors the days on which the water film was 0 and the wind speed did not exceed 5m/s have been analysed. The resulting measurements coincide with the occurrence of icing classes listed in table 2, after taking into account the freezing point of brine.

Next, the daily predicted air temperature changes, derived from the regional model, have been compared with the actual results obtained from the local weather station. Maximum deviation of the test results for 15 days amounted to 4°C.

Additionally, air and road temperature analysis has been performed. The road temperature in summer strongly depends on sunlight and rainfall. Maximum temperature differences between road and air were above 30°C (fig. 3). In winter (December, January, February) the differences do not exceed 12°C (fig. 4). The obtained results and the fact that even changes at the level of 1 - 2 degrees can cause the black ice occurrence underline the need of road surface heat balance model implementation.

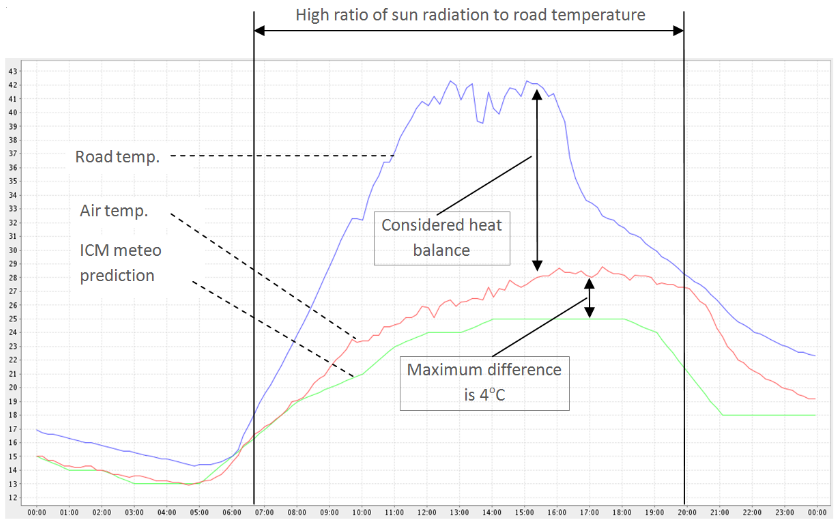


Fig. 3. Dependencies of air and road surface temperature in summer

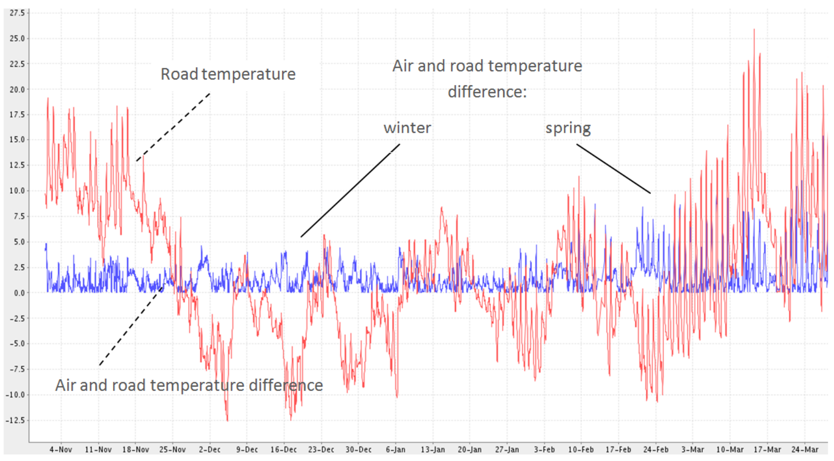


Fig. 4. Dependencies of air and road surface temperature in winter

The obtained results confirm the literature conclusion - more incidents were reported in relatively stable, favourable weather conditions (fig. 5). Studies have also shown the accumulation of accidents in short time periods, with weather condition changes leading to freezing rain, black ice and wind gusts of more than 7 m/s (fig. 6).

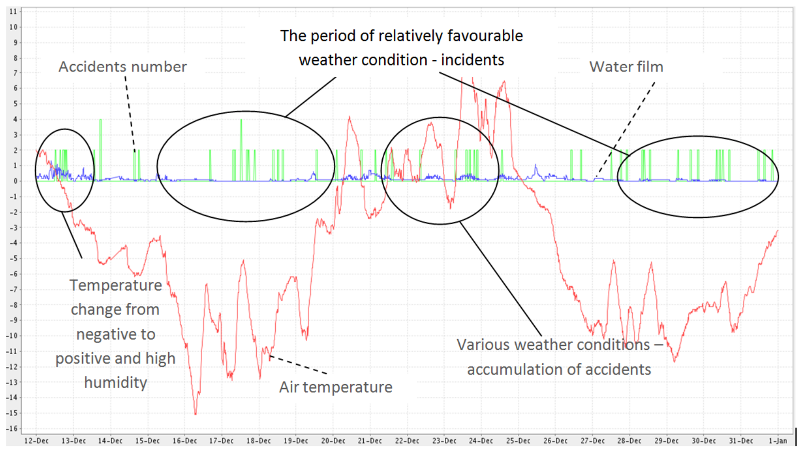


Fig. 5. Dependencies of collisions and accidents on the state of road surface

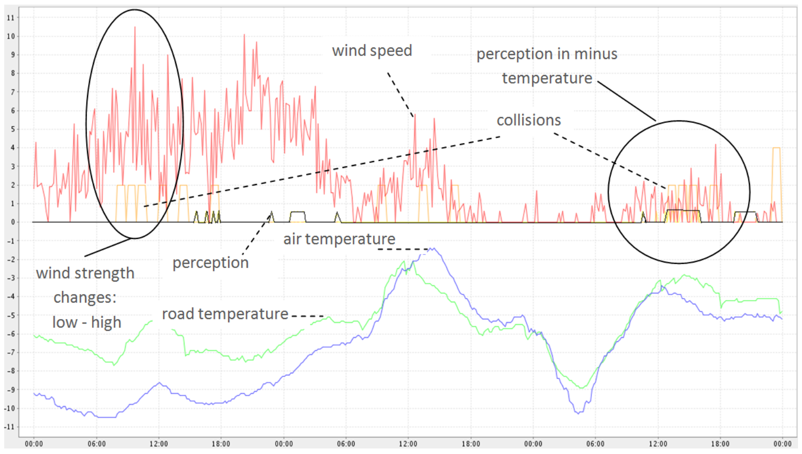


Fig. 6. Effect of wind and rain at negative temperatures on collisions

To confirm empirical evaluation presented on fig. 3, 4, 5 and 6, analysis of correlation and analysis of variance were performed. For further analysis, from pairs of parameters with high correlation rate, one was excluded as well as parameters with low variance.

3. Neural network

In literature there is number of studies describing the use of neural networks for prediction of weather phenomena [17,18,9]. For the needs of the project a neural network with the sigmoid activation function (with β =1) has been used:

$$f(x) = \frac{1}{1+e^{-\beta x}} \tag{1}$$

In the network learning process the gradient drop method has been used [16]. For the learning method the following parameters have been assumed:

- the maximum value of weights = 0.5,
- absence of mixing patterns,
- ratio epsilon = 0.01,
- learning rate = 0.9,
- torque ratio = 0.7,
- teaching tolerance = 0.1.

The prepared data base included samples from each 1 minute of a 6 months period. The database enables the learning process for the following parameters: air temperature, road temperature, wind, precipitation, atmospheric pressure, water film and the level of salinity.

The first experiment has been designed to predict changes in road surface temperature after 15 minutes. Based on a correlation analysis and series of introductory experiments (based on four random parameters) the parameters air, road temperatures, wind speed and rainfall have been used. Characteristics of daily weather changes have been included as the fifth network input. The dynamics of weather changes in the period before prediction have been taken into account by using the time window [13]. The window width, after preliminary tests, has been set to 15, with the sampling performed every 5 minutes.

Based on the Kołmogorow theory [10] $2n+1$ neurons in first hidden layer have been chosen. Additionally, the influence of the second hidden layer on prediction results has been analysed. In the research the network structure shown in figure 7 has been used. The symmetrical connection between hidden layers was implemented.

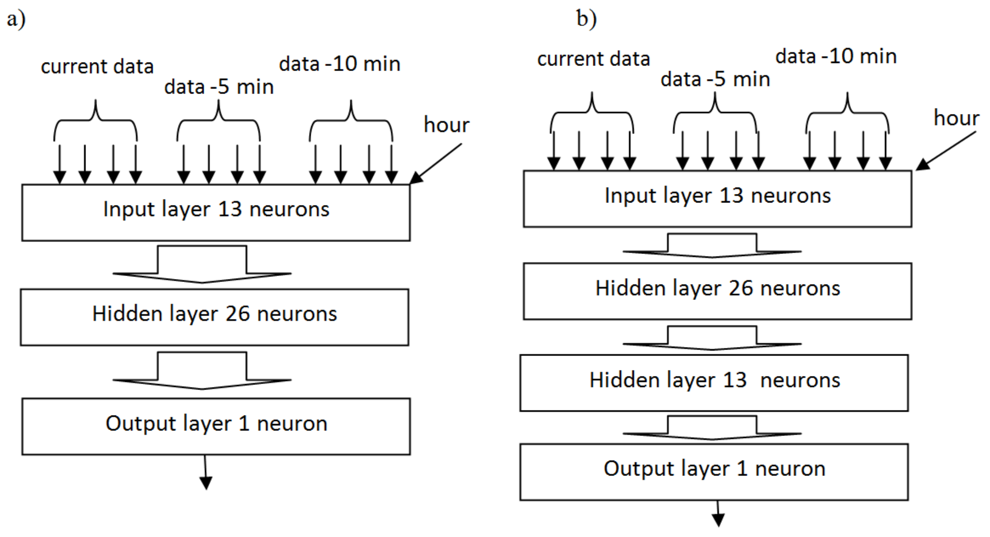


Fig. 7. Schematic diagram of the network for the temperature prediction

One hundred 13-element randomly selected learning vectors have been connected to the input of the network. The procedure has been repeated 20 times. For the forecast a network, yielding the smallest average error for the selected test vectors has been chosen. The network has been built based on two hidden layers enabling better adaptation to rapid weather changes. The obtained results are shown in figure 8.

The experiment demonstrated the effectiveness of neural network for predicting temperature variations in Polish climatic zone. Road surface temperature prediction for the next 15 minutes proved to be correct in 98% of cases with a tolerance of $\pm 2°$C. Decrease of tolerance to $\pm 1°$C strongly decrease prediction rate to 83%. Increasing numbers of hidden layers, above two, did not influence prediction process. However, spread the temperature is too large for the detection of black ice. Large fluctuations in significant part resulted from not including the level of insolation (cloud cover).

Another experiment consisted of defining the level of threat of black ice after 15 minutes. Level of icing threat was defined based on Table 2 and icing sensor with water film (0:none, 1:low, 2:medium, 3:high). The study used a temperature prediction from the first experiment, the wind strength, the level of salinity and water film – parameters connected with ice creating phenomena. In prediction process the network structure shown on fig. 9 was used.

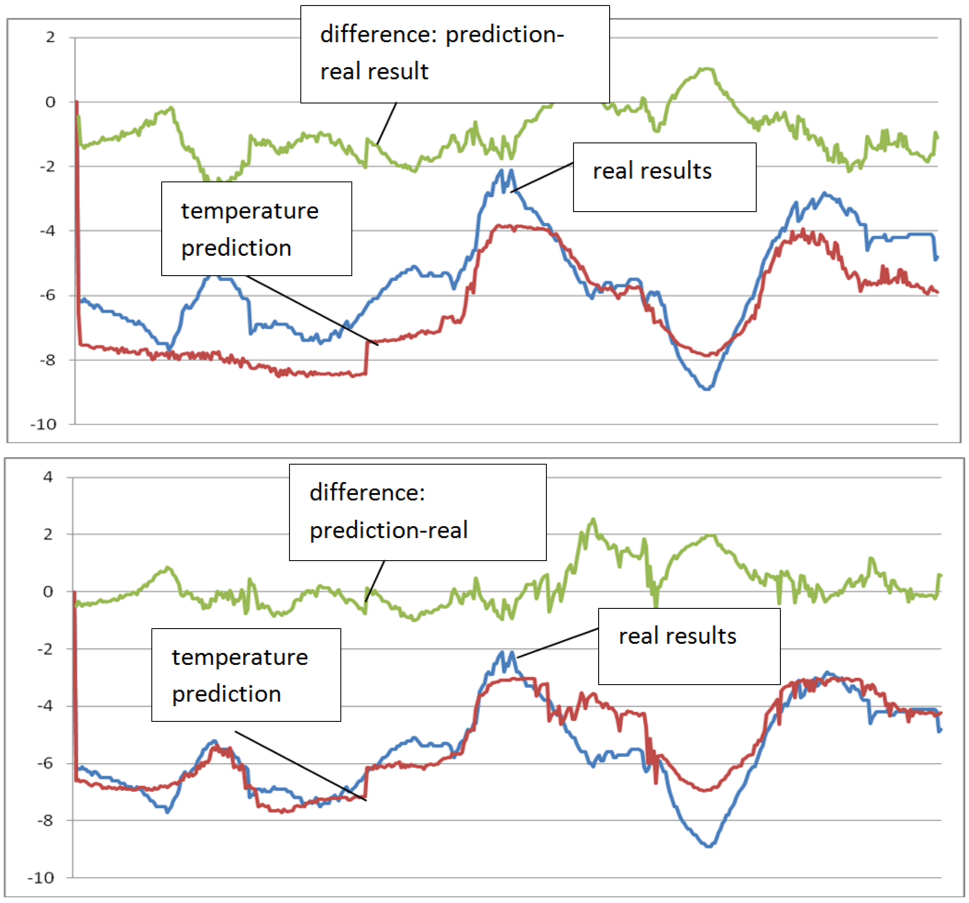


Fig. 8. Results of the temperature prediction

For 80 13-element test vectors, the network has detected black ice in 85% and 90% of cases with the first (fig. 9a) and second (fig. 9b) structure implemented, respectively. Preliminary studies using neural networks confirm the validity of network applications in the process of weather forecasting. According to the literature analysis the proper network combination can reduce the measurement error to 2.5% [**?**] in case of a combining more advanced structures.

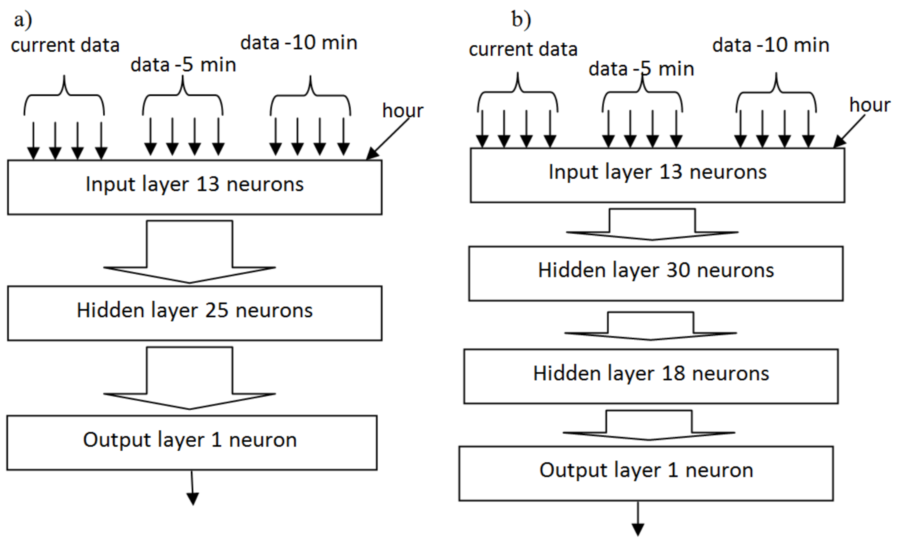


Fig. 9. Schematic diagram of the network for the icing prediction

4. Conclusion

The conducted analysis for the road weather station confirms the data obtained in the present literature [12]. The most common causes of road accidents according to these studies are:

- changing of weather conditions:
  - a temperature change from minus to plus at high humidity,
  - strong side wind gusts,
- precipitation on frozen road surface,
- limited visibility – fog.

In this context, searching for the effective solutions of weather prediction numerical analysis seems to be fully justified. The presented weather prediction system, based on a neural network, provides the opportunity to anticipate significant changes in atmospheric conditions with relatively good accuracy. The results obtained for the data from local meteorological station and forecasts for a 15-minute time horizon are described in Table 3.

Additional studies show the possibility of using neural networks as an icing detector, taking into account the level of sunlight. By using the proposed solution there is a potential possibility for detecting the rapid changes in weather conditions from favourable to unfavourable, which is particularly important since the surveys indicate increased accidents risk in these situations.

The tests and results obtained by using the designed neural network are the basis for expert systems based on fuzzy inference which are the main point of the currently on-going research works.

Table 3. The results of the prediction system based on neural network

Detection parameters	**Road surface temperature prediction**				**Icing prediction**	
	Network 1		**Network 2**		**Network 1**	**Network 2**
Prediction time	15 min		15 min		15 min	15 min
Tolerance / acceptance threshold	±2°C	±1°C	±2°C	±1°C	±0.4	±0.4
Correct prediction	**95%**	**73%**	**98%**	**83%**	**85%**	**90%**
Absolute value of maximal difference	**2.8°C**		**2.2°C**		**2°C**	**1.3°C**
Number of learning vectors	100		100		80	80
Number of test vectors	100		100		100	100

References

[1] M. Agarwal, T. Maze, R. Souleyrette, Impact of Weather on Urban Freeway Traffic Flow Characteristics and Facility Capacity, Integration of Road Weather Information with Traffic Data, 2005, p. 8.

[2] M. Bernaś, M. Dąbkowski, Rozwiązania programowo-sprzętowe dla poprawy bezpieczeństwa transportu drogowego, Zeszyty Naukowe Transport z. 70, 2011, s. 11.

[3] F. Bijleveld, T. Churchill, The Influence of Weather Conditions on Road Safety, SWOV Institute for Road Safety Research, 2009, p. 11.

[4] D. Eisenburg, The Mixed Effects of Precipitation on Traffic Crashed, Accident Analysis & Prevention, Vol. 36, 2004, p. 637.

[5] J. Hogema, Effects of Rain on Daily Traffic Volume and on Driving Behavior, TNO Human Factors Research Institute TM-96-B019 report, 1996, p.113.

[6] R. Hranac, Empirical Studies on Traffic Flow in Inclement Weather, Cambridge Systematic - FHWA, 2006, pp. 2-4.

[7] K. Keay, I. Simmonds, The Association of Rainfall and Other Weather Variables with Road Traffic Volume in Melbourne, Accident Analysis & Prevention, Vol. 37, 2005, p. 109.

[8] R. Krystek, Zintegrowany system bezpieczeństwa transportu, Tom 3, Koncepcja zintegrowanego systemu bezpieczeństwa transportu w Polsce, WKŁ, 2010.

[9] R. Kuligowski, A. Barros, Localized Precipitation Forecasts from a Numerical Weather Prediction Model Using Artificial Neural Networks, Weather Forecast, Vol. 13, 1998, p. 1194.

[10] A. Kwiatkowska, Systemy wspomagania decyzji, Wydawnictwo naukowe PWN, Warszawa, 2007.

[11] Numeric Weather IMC model: http://new.meteo.pl/.

[12] A.W. Mitas, M. Bernaś, M. Bugdol, A. Ryguła, Technologie informacyjne w predykcji pogodowych zagrożeń w ruchu drogowym, Elektronika–konstrukcje, technologie, zastosowania, Warszawa 2011, pp.86-89.

[13] T. Pamuła, Road Traffic Parameters Prediction in Urban Traffic Management Systems Using Neural Networks, Transport Problems , Vol. 6, Iss. 3, 2011, p. 123.

[14] H. Peng, P. Xiaodong, Embankment Wind Velocity Field and Traffic Safety Under the Influence of Sudden Cross-Wind, Computer science and Information technology, 2007, p. 606.

[15] D. Sandeep, S. Sharma S, Impact of Cold and Snow on Temporal and Spatial Variations of Highway Traffic Volumes, Journal of Transport Geography, Vol. 16, 2008, p. 358.

[16] R. Tadeusiewicz, Sieci neuronowe, Akademicka Oficyna Wydawnicza RM, Warszawa, 1993.

[17] K. Zabczyk, Meteorologia drogowa a bezpieczeństwo ruchu, SIGNALCO ltd., Kraków 2008, p. 4.

[18] K. Zabczyk, K. Pierzchała, Mapy termiczne sieci drogowej, SIGNALCO ltd., Kraków 2010, p. 2.

KONCEPCJA ZASTOSOWANIA SIECI NEURONOWYCH DO ANALIZY PARAMETRÓW POGODOWYCH DLA PREDYKCJI ZAGROŻEŃ

Streszczenie W artykule poruszono zagadnienia związane z predykcją warunków meteorologicznych, w ujęciu bezpieczeństwa ruchu drogowego. W pierwszej części pracy podsumowano najważniejsze wnioski oraz rezultaty przeprowadzonych studiów literaturowych oraz, na ich podstawie, dokonano obserwacji określonych artefaktów w zdefiniowanym zbiorze danych statystycznych. Dane obejmowały parametry meteorologiczne ze stacji pogodowej, informacje o lokalnych kolizjach drogowych oraz dane modelu predykcji zjawisk atmosferycznych. W finalnej części pracy przestawiono przykładowe implementacje sieci neuronowych wykorzystane celu predykcji omawianych zagrożeń pogodowych.

Słowa kluczowe: predykcja pogody, sieci neuronowe, bezpieczeństwo transportu

THE COMPARISON OF GENETIC ALGORITHMS WHICH SOLVE ORIENTEERING PROBLEM USING COMPLETE AND INCOMPLETE GRAPH

Krzysztof Ostrowski, Jolanta Koszelew

Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: The purpose of this work was to compare two forms of genetic algorithm (complete and incomplete graph version) which solves Orienteering Problem (OP). While in most papers concerning OP graph is complete and satisfies triangle inequality, in our versions such assumptions may not be satisfied. It could be more practical as transport networks are graphs which do not have to satisfy those conditions. In such cases, graphs are usually complemented with fictional edges before they can be used by classic OP solving algorithms which operate on complete graphs. This paper answers the question: Is it better (in terms of results quality and time consumption) to transform graphs to classic OP form before running algorithm (complete graph version) or to solve OP on graphs without any assumptions and changes (incomplete graph version)? The computer experiment was conducted on the real transport network in Poland and its results suggest that it is worth checking both versions of the algorithm on concrete networks.

Keywords: orienteering problem, OP, transport network, genetic algorithm, GA, incomplete graph, complete graph

1. Introduction

The orienteering problem (OP) is still one of the most challenging optimization problems. It is related to the travelling salesman problem (TSP). The main difference is that not all cities have to be visited and each of them has some profit. The goal is to maximize total profit within a given time frame. The OP has a lot of practical applications (i.e. logistics, planning and tourism [1] [2]). For example, it could be very helpful in trip planning and such systems for tourists are developed [3][12]. Several exact solutions of the OP were proposed, including linear and dynamic programming [4][5]. However, the OP (like TSP) is an NP-hard problem [13][14]

Zeszyty Naukowe Politechniki Bialostockiej. Informatyka, vol. 8, pp. 61-77, 2011.

and exact solutions are impractical in terms of time consumption for bigger sized problems. Thus, algorithms with various heuristic strategies are implemented to solve the problem more efficiently [15]. One of the first heuristics (including Monte Carlo method) were applied by Tsiligirides [11]. Others proposed methods include i.e. 2-opt and 3-opt procedures and centre of gravity usage [6][13][7]. One of the most effective heuristic solving the OP is a guided local search heuristic [10][19]. Artificial neural networks and genetic algorithms were also used to solve the OP [8][9]. The OP itself has also several extensions and variants i.e. team orienteering problem (TOP) and orienteering problem with time windows (OPTW) [16][17].

In the paper two versions of genetic algorithm (GA) with mutation are presented to solve the OP. One of them (IG) operates on an incomplete graph and the other (CG) performs edge completion before running the main genetic algorithm on a complete graph. The paper is organised as follows. Section 2 presents the definition of the OP with a network example. Section 3 gives a detailed specification of both GA versions with several examples. In section 4 the experimental results of these two algorithms (with a couple of different heuristics) were compared on a real transport network. Conclusions are presented in section 5.

2. Problem definition

Given a set of n vertices (each vertex has some nonnegative profit), travel time between every pair of vertices and the starting point (s) and the end point (e), the purpose of the OP is to find the path (limited by travel time t_{max}) between vertices s and e that maximizes the total profit (computed as the sum of profits of visited vertices). Each point can be visited at most once.

The OP can be also defined using an undirected, complete graph $G = (V,E)$, when V ($|V| = n$) is the vertex set and E is the edge set. Each vertex i is associated with some nonnegative profit p_i and each edge connecting vertices i and j is associated with some travel time t_{ij}. The goal is to find a Hamiltonian path of a subgraph of G (between start (s) and end (e) vertices) which maximizes the total collected profit and is limited by constraint t_{max}.

The problem introduced in this article is OP with one substantial modification: each vertex can be visited more than once during the travel, but the total profit is increased only when a given vertex is visited for the first time. In addition, graph edges do not have to satisfy triangle inequality. The paper presents two versions of algorithms solving OP. Both of them are genetic algorithms (GA) with local search procedure in the form of mutation. In the first algorithm version presented in the article there are direct connections (travel times) only between some pairs of vertices (incomplete and undirected graph) and no edge completion is performed before running the OP solving algorithm. In the second version it is assumed that there is a direct connection (travel time) between every pair of points (complete and undirected graph). If this assumption is not satisfied, a graph is complemented with virtual edges using Dijkstra algorithm. There is an important note about the score calculation: if a given path includes a virtual edge between vertices i and j then all vertices on the shortest path from i to j are considered when computing the total profit.

At the problem input there are graph G (with matrix of travel times t and vector of profits p) and maximum travel time t_{max}. At the problem output route r in graph G is obtained. It starts and ends in the vertex number 1, its travel time is not greater than t_{max} and its total profit is maximized. In our experiment graph G is a real transport network with cities (and profits) and connections between them.

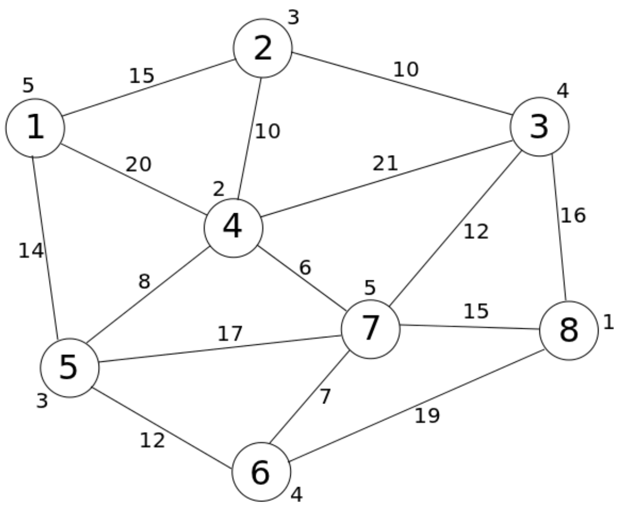


Fig. 1. A graph representing an exemplary transport network

In the picture (fig. 1) there is an exemplary network of 8 cities. Travel time (t_{ij}) values are marked on edges and profit (p_i) values are marked near vertices (cities). This network is used in all examples from the paper. Let $t_{max} = 80$. A cycle r=1, 5, 4, 7, 6, 7, 3, 2, 1 can be a solution. Its travel time is equal to 79 (14+8+6+7+7+12+12+15) and its total profit is 26 (5+3+2+5+4+4+3 - only first visits to cities 1 and 6 are counted).

3. Algorithm specification

Both presented versions are genetic algorithms with mutation. Main steps in both versions are the same but genetic operations are different in some important details. Tours are encoded into a chromosome as a sequence of vertices (cities). It is the most natural way of adopting GA for OP. For example, the cycle from previous section can be represented as an individual (1, 5, 4, 7, 6, 7, 3, 2, 1) and its fitness is 26 (equal to the total profit).

3.1 Incomplete graph version

This algorithm is an improved (in terms of time complexity) version of [18]. First, an initial population of P_{size} solutions is generated. At the beginning a random vertex v adjacent to vertex 1 (the start point) is chosen and t_{1v} is added to the current travel time. If the current travel time does not exceed $0.5 \cdot t_{max}$, the tour generation is continued. Now we start at the vertex v and choose random vertex u adjacent to v. At every step we exclude the visited vertex from the set of possibilities - it prevents from visiting a given vertex repeatedly. If the current tour length is greater than $0.5 \cdot t_{max}$, the last vertex is rejected and we return to vertex 1 the same way in reverse order. This way of generating the individuals of the initial population means that they are symmetrical in respect of the middle vertex in the tour. However, these symmetries are removed by the algorithm. An example of an initial population is shown in table 1.

After generating initial population, the GA starts to improve the current population through repetitive application of selection, crossover and mutation. The algorithm stops after N_g generations and the resulting tour is the best individual from the final generation. First, tournament selection is applied - we select t_{size} random, different individuals from the current population and the best one from the group is copied to the next population. The whole tournament group is returned to the old population. After P_{size} repetitions of this step a new population is created (a selection example in table 2).

The example presented in tab. 2 shows how selection improves average population fitness. However, if t_{size} is too high (relatively to P_{size}), the population converges very fast.

The crossover is performed as follows: first, two random parental individuals are selected. Afterwards we randomly choose a common gene (crossing point) in both parents (first and last genes are not considered). If there are no common genes, crossover cannot be done and no changes in chosen chromosomes are applied.

Table 1. An initial population example ($P_{size} = 5$, $t_{max} = 80$)

No	Individual	Fitness	Travel time
1	(1, 2, 3, 7, 3, 2, 1)	17	74
2	(1, 5, 6, 7, 6, 5, 1)	17	66
3	(1, 4, 7, 6, 7, 4, 1)	16	66
4	(1, 5, 7, 5, 1)	13	62
5	(1, 5, 4, 7, 6, 7, 4, 5, 1)	19	77

Table 2. An example of a tournament selection performed on the population from table 1

No	Numbers of individuals selected into the tournament group	Number of the best individual	The best individual (next population member)	Fitness of the best individual
1	1, 3, 4	1	(1, 2, 3, 7, 3, 2, 1)	17
2	2, 4, 5	5	(1, 5, 4, 7, 6, 7, 4, 5, 1)	19
3	2, 3, 4	2	(1, 5, 6, 7, 6, 5, 1)	17
4	1, 2, 5	5	(1, 5, 4, 7, 6, 7, 4, 5, 1)	19
5	1, 2, 3	1	(1, 2, 3, 7, 3, 2, 1)	17

After that, two new individuals are created as a result of exchanging chromosome fragments (from the crossing point to the end of the chromosome) in both parents. If one of the children does not preserve t_{max} constraint, it is replaced by the fitter parent in the new population. If both children do not preserve this constraint, the parents replace them in the new population (no changes applied).

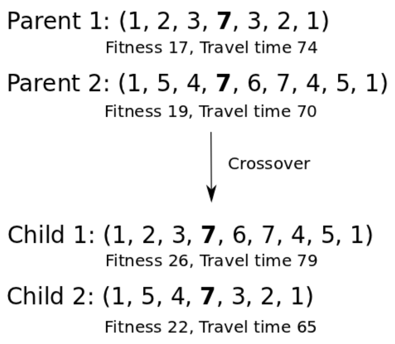

Parent 1: (1, 2, 3, **7**, 3, 2, 1)
Fitness 17, Travel time 74
Parent 2: (1, 5, 4, **7**, 6, 7, 4, 5, 1)
Fitness 19, Travel time 70
Crossover
Child 1: (1, 2, 3, **7**, 6, 7, 4, 5, 1)
Fitness 26, Travel time 79
Child 2: (1, 5, 4, **7**, 3, 2, 1)
Fitness 22, Travel time 65

Fig. 2. An example of crossover ($t_{max} = 80$), crossing point in bold

In the example presented in fig. 2 two children with improved fitness and low enough travel time are created. Both parents are symmetrical but the symmetry in the offspring individuals is removed by crossover. After selection and crossover the population undergo mutation. First we select a random individual to be mutated. There are two possible kinds of mutation: inserting a new gene and removing an existing gene. Several mutation versions (with different heuristics) have been implemented in the presented algorithm. In some of them only inserting mutation can be performed, but in others both kinds of mutation are possible with the probability of 0.5. During the inserting mutation all the possibilities of inserting a new gene that is not present in the chromosome are considered (without exceeding t_{max}) and the best is chosen. Depending on the heuristic used it can be the one with lowest travel time increase, highest fitness gain or best $fitness/travelTime$ ratio.

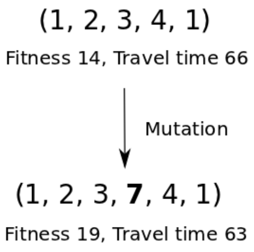


Fig. 3. An example of mutation ($t_{max} = 80$, highest fitness gain heuristic), inserted gene in bold

In the example presented in fig. 3 there are two possibilities of inserting a new gene into chromosome (1, 2, 3, 4, 1) without exceeding t_{max}. One of them is inserting gene 7 between genes 3 and 4 (fitness gain is 5) and the other is inserting gene 5 between genes 4 and 1 (fitness gain is equal to 3). The first of them (with higher fitness gain) is chosen. There are also several variants of removing mutation. In all of them we consider only genes which can be removed without perturbing path continuity - between their neighbouring genes in the chromosome there should be an edge in the graph. After obtaining the set of possibilities, an appropriate heuristic is performed. In the first heuristic only genes that appear in the chromosome more than once are considered (except first and last genes). If there are no such candidates, the removing mutation is not performed. Otherwise we choose the gene in order to shorten the travel time as much as possible. In the second heuristic we concentrate on the lowest fitness loss. In the best-case scenario the fitness loss is zero (we remove a gene that has duplicates in the chromosome), but it is possible that fitness loss is greater than zero (no duplicates in the chromosome).

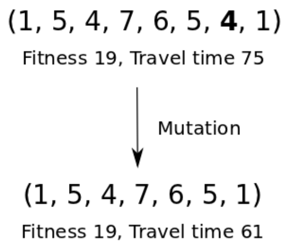


Fig. 4. An example of mutation ($t_{max} = 80$, highest fitness gain heuristic), removed gene in bold

In the example presented in fig. 4 we have three candidates to remove: gene 5 (between genes 1 and 4), gene 4 (between genes 5 and 7) and gene 4 (between genes 5 and 1). These genes meet two conditions: they have duplicates and their chromosome neighbours are connected in the graph. If we remove gene 5, the path is shorten by 2 (14+8-20), but if we remove gene 4 (between genes 5 and 7) travel time is even greater. Thus, the best candidate is gene 4 (between genes 5 and 1) - we shorten travel time by 14 (20+8-14).

3.2 Complete graph version

In the second algorithm version there are some necessary steps to do before performing the GA. The graph is complemented with virtual edges. First, Dijkstra algorithm is run from every vertex (altogether n times). If there is no edge between vertices i and j we add virtual edge (i, j) computed during the Dijkstra algorithm. It contains more information than a real edge - besides the travel time, vertices on the shortest path between i and j are also remembered. Virtual edge (j, i) is constructed independently during a different run of Dijkstra algorithm. Its travel time in undirected graphs is the same as in edge (i, j), but the path itself could be totally different (a lot of different shortest paths possible). This partially reduces the chromosome symmetry obtained during generating the initial population - although the chromosome is still symmetrical, its inner paths (virtual edges) could differ significantly. In our exemplary graph (presented in fig. 1) both virtual edges between vertices 1 and 8 have travel time of 41, but the associated shortest paths are different (1-4-7-8 and 8-3-2-1).

As a result of the edge completion described above, calculating the individual fitness is different from the one applied the first algorithm version. While encoding tour vertices into a chromosome is the same, vertices "hidden" in virtual edges (shortest paths) are also considered when calculating fitness. It means that profits of all vertices (hidden or not) are summed up only during the first visit of a given vertex. In this section term ’main path’ means the sequence of chromosome genes (vertices) whereas term ’full path’ refers to the sequence of all vertices in the route (including those hidden in virtual edges).

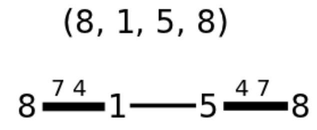


Fig. 5. An example of a chromosome. Virtual edges (shortest paths) in bold and hidden vertices above

In the example presented in fig. 5 there is a chromosome representing tour 8-1-5-8 (main path) which actually is 8-(7-4)-1-5-(4-7)-8 (full path). The shortest path from vertex 8 to vertex 1 is 8-7-4-1 and the shortest path from vertex 5 to vertex 8 is 5-4-7-8 - these are virtual edges. The edge from vertex 1 to vertex 5 is real. To calculate overall profit (fitness) we take into account vertices 8, 7, 4, 1, and 5 - the fitness is 16. The total travel time is 84 - the sum of 41 (shortest path from vertex 8 to vertex 1), 14 and 29 (shortest path from vertex 5 to vertex 8).

The first step of the GA is to generate an initial population. It is similar to the procedure performed in the incomplete graph version. The main difference is that the graph is full and from a given vertex we can construct the path to any other (except those already included in the main path). In the process of creating individuals both real and virtual paths could be added.

After generating the initial population, the GA starts its standard procedure of repeated selection, crossover and mutation. The tournament selection and crossover are performed in the same way as in the incomplete graph version of the algorithm. During the crossover only vertices from the main path are considered when selecting crossing point (virtual and real edges treated in the same way).

Mutation is the only step which is strongly different in both versions of the GA. We select a random individual and then inserting or removing mutation is performed, both with the probability of 0.5 (in some variants inserting mutation is the only option). During inserting mutation all the possibilities are checked - between every pair of neigbouring genes in the chromosome (main path) we can insert any gene different than these two (the graph is full and gene duplicates in the chromosome are allowed) and t_{max} is the only limit. From this set of possibilities the best one is chosen - it depends on the heuristic used. The first heuristic selects insertion which results in the greatest fitness gain. The second one is based on finding the greatest $fitness/travelTime$ ratio in the newly mutated individual. Normally during inserting mutation one edge is removed and two edges are added. In this version of the algorithm any edge could be a path (virtual edge). Thus, the algorithm operates on whole paths and heuristics are more complex - when calculating fitness and travel time, every hidden vertex in virtual edges has to be considered.

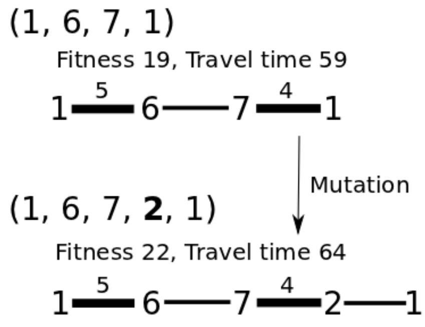


Fig. 6. An example of mutation (t_{max}=80, max $fitness/travelTime$ heuristic), inserted gene in bold

In the example presented in fig. 6 we have chromosome (1, 6, 7, 1). Edges 1-6 and 7-1 are virtual and the full path is 1-(5)-6-7-(4)-1. There are a few possibilities of inserting a new gene without exceeding t_{max}. The best is presented in the example - inserting gene 2 between genes 7 and 1 results in maximum $fitness/travelTime$ ratio of 0.344 (fitness 22, travel time 64). One of other options is inserting gene 3 between genes 7 and 1. The mutated chromosome would be (1, 6, 7, 3, 1) with the full path 1-(5)-6-7-3-(2)-1. Its fitness would be 24, which is better than in the chosen possibility, however with travel time of 70 its $fitness/travelTime$ ratio is slightly worse (0,343).

Removing mutation also checks all possibilities - any gene (except first and last) could be removed. There are several heuristics presented - two of them are the same as those in the inserting mutation (greatest fitness loss (could be less than 0) and greatest $fitness/travelTime$ ratio in a mutated individual). The third heuristic chooses the option with the greatest $fitness^2/travelTime$ ratio. Evaluating new fitness and travel time is performed in a similar way to the inserting mutation - if some removed or inserted edge is virtual, the algorithm takes into account all vertices in the shortest path. Thus, calculation is more complex but it is compensated by shorter chromosomes than those in the first algorithm version (less genes in the main path).

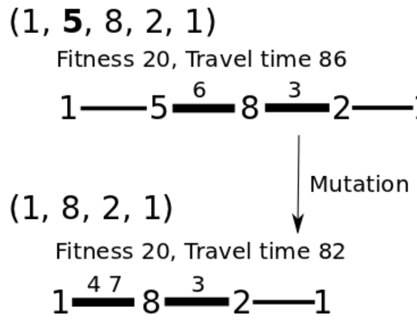


Fig. 7. An example of mutation ($t_{max} = 100$, max $fitness^2/travelTime$ heuristic), removed gene in bold

In the chromosome presented in fig. 7 there are three potential candidates to remove - genes 5, 8 and 2. If gene 5 is extracted (shown in the example) fitness (20) is unchanged and travel time is shorter (82) - the resulted $fitness^2/travelTime$ is 4.88, which is the best of all possibilities. During this operation we remove edge 1-5 and virtual edge 5-(6)-8. Instead, shortest path 1-(4-7)-8 from vertex 1 to vertex 8 is added. The second option is eliminating gene 8 and obtaining the chromosome (1, 5, 2, 1) - the real path would be much shorter: 1-5-(4)-2-1 (travel time 47, fitness 13, $fitness^2/travelTime3,60$). The third option (removing gene 2) results in the chromosome (1, 5, 8, 1) (real path 1-5-(6)-8-(7-4)-1). With fitness (20) and travel time (86) unchanged, its $fitness^2/travelTime$ ratio would be 4.65.

4. Experimental results

Experiments were conducted on the real road network of 306 cities in Poland. The tested data of the network can be found on the website [20] in two text files: cities.txt and distances.txt. The network file (distances.txt) was created from a real map - it includes main segments of roads from the whole Poland. The capital of Poland, Warsaw, was established as the central depot (vertex 1). Profits associated with a given city (written in the file cities.txt) were determined according to the number of inhabitants in the city. The more inhabitants the higher profit associated with a given city but its maximum value is 5. The profit was calculated as follows:

$$profit = \frac{inhabitants}{10000} \tag{1}$$

Both versions of the algorithm (for complete and incomplete graph) with several different heuristic variants were tested on the network presented above and its results were compared. The chosen heuristics were among the best of all heuristics tested on the transport network used in our experiment. Algorithm parameters were the same in almost all runs ($P_{size} = 300$, $t_{size} = 3$, $N_g = 100$). Only number of generations (N_g) is increased to 150 in the last two presented comparisons for IG. Increasing P_{size} brings minor changes in both versions and increasing t_{max} in IG has similar but generally less pronounced effects than changing N_g. CG converges more quickly and in this case increasing algorithm parameters results in marginal differences. Experiments were conducted on six t_{max} values: 500, 1000, 1500, 2000, 2500, 3000. The result of a given algorithm run is the best chromosome (highest fitness) from the final population. Statistics were obtained from 30 runs of each algorithm variant. Analysed parameters were: average result (mean), 95% confidence interval (CI) for mean and the best result (max) from 30 algorithm runs.

First, algorithms without removing mutation were compared. Two different inserting heuristics (highest fitness gain, highest $fitness/travelTime$ ratio after mutation) were chosen.

Table 3. Results compared (inserting mutation: highest fitness gain, no removing mutation)

	Complete graph version (CG)			Incomplete graph version (IG)		
t_{max}	Mean	CI for mean	Max	Mean	CI for mean	Max
500	59.6	±2.3	79	44.2	±3.2	65
1000	102.3	±3.6	118	86.7	±5.7	142
1500	127.1	±5.3	160	140.3	±10.3	193
2000	161.3	±6.5	215	188.6	±8.4	227
2500	179.2	±8.8	227	221.4	±8.1	264
3000	200.1	±6.6	244	241.9	±9.8	285

When using highest fitness gain heuristic (tab. 3), CG performs better for the lowest t_{max} values. However, IG is significantly better for higher t_{max} (>1000) in terms of both average and best results.

Table 4. Results compared (inserting: highest $fitness/travelTime$ after mutation, no removing mutation)

	CG			IG		
t_{max}	Mean	CI for mean	Max	Mean	Ci for mean	Max
500	61.8	±3.4	79	46.9	±2.9	63
1000	112.6	±5.0	149	99.6	±7.6	157
1500	140.7	±7.9	209	160.9	±11.6	206
2000	169.9	±8.1	209	202.8	±8.3	237
2500	193.3	±9.5	253	228.3	±10.2	267
3000	220.5	±10.7	295	254.7	±7.4	287

Results of both algorithms improve when using heuristic of highest $fitness/travelTime$ ratio after mutation. The tendency is the same as in the previous heuristic - while CG is on average better for lower t_{max}, IG excels for higher t_{max}. However, the best results (max) from 30 runs are more similar than in the previous heuristic, with CG better even for some higher t_{max} values (i.e. 3000).

After first comparisons removing mutation was added to both versions of GA. Lowest fitness loss heuristic (when removing) combined with highest $fitness/travelTime$ ratio after mutation (when inserting) was the best combination for complete graph version of GA and one of the best variants (in terms of mean results) for incomplete graph algorithm version.

Table 5. Results compared (inserting: highest $fitness/travelTime$ after mutation, removing: lowest fitness loss)

	CG			IG		
t_{max}	Mean	CI for mean	Max	Mean	Ci for mean	Max
500	61.9	±3.5	92	52.3	±3.2	71
1000	109.5	±5.4	144	92.7	±6.2	123
1500	146.7	±8.8	204	151.5	±9.7	192
2000	190.6	±9.7	248	198.2	±8.0	241
2500	219.1	±10.7	281	228.6	±6.9	264
3000	256.9	±11.4	320	244.7	±8.8	279

It can be seen that removing mutation significantly improved results of CG for higher t_{max} values. At the same time performance of IG slightly dropped. As a result, CG is still better for lower t_{max} values but its mean values are also closer for higher t_{max}. What is more, for longest routes (t_{max} 2500-3000) CG achieves highest maximum values.

There is one more combination of heuristics regarding incomplete graph algorithm version which should be mentioned. When inserting a gene we choose the option with highest $fitnessGain^2/travelTimeIncrease$ ratio. When removing we consider only genes which have some duplicate in the chromosome. If any of them are found, we choose the option with highest travel time loss. In terms of mean results this combination is similar to the version from the previous table but it is better for $t_{max} = 1000$. More importantly, it is one of algorithm variants which improves further for higher number of GA generations (N_g=150). It is shown in the tab. 6.

Table 6. Comparison 1 between results for N_g=100 and N_g=150 - incomplete graph version (inserting: highest $fitnessGain^2/travelTimeIncrease$ ratio, removing: highest travel time loss)

	$N_g = 100$			$N_g = 150$		
t_{max}	Mean	CI for mean	Max	Mean	Ci for mean	Max
500	52.4	±3.0	69	54.3	±3.0	69
1000	101.4	±7.1	149	105.6	±6.9	157
1500	162.3	±7.4	194	178.8	±9.1	215
2000	195.3	±7.4	230	213.5	±9.1	269
2500	225.4	±7.9	263	243.6	±9.5	288
3000	241.8	±7.1	270	261.6	±8.3	295

For mediocre and high t_{max} values there is an improvement of about 10% in results between 100th and 150th generation. Mean values for higher t_{max} are the best of all results obtained during the experiment. In tab. 7 there is one more comparison - inserting heuristic is the same as in tab. 6 but removing heuristic is not performed.

Table 7. Comparison 2 between results for N_g=100 and N_g=150 - incomplete graph version (inserting: highest $fitnessGain^2/travelTimeIncrease$ ratio, no removing mutation)

	$N_g = 100$			$N_g = 150$		
t_{max}	Mean	CI for mean	Max	Mean	Ci for mean	Max
500	46.7	±3.1	66	47.5	±3.3	66
1000	88.2	±6.0	132	88.3	±6.0	132
1500	153.2	±11.3	213	157.4	±13.5	225
2000	206.9	±10.3	249	216.6	±12.8	283
2500	222.2	±12.2	276	227.2	±13.9	292
3000	247.2	±12.3	295	257.3	±15.2	331

While mean results differ slightly, one can see a significant improvement of best runs for higher t_{max} - these are best routes obtained during the experiment.

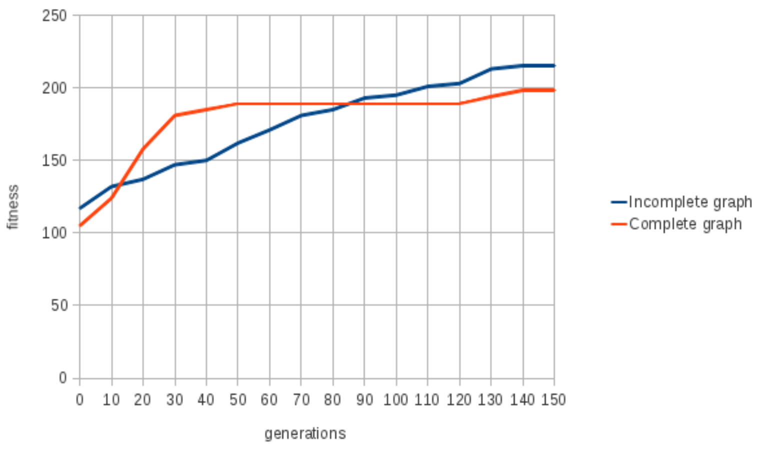


Fig. 8. Average fitness of the best individual from the n-th generation for both algorithms ($t_{max} = 2000$)
IG heuristic: inserting: highest $fitnessGain^2/travelTimeIncrease$, removing: highest travel time loss
CG heuristic: inserting: highest $fitness/travelTime$ after mutation, removing: lowest fitness loss

CG converges more quickly (as seen in fig. 8) and improvements between 100th and 150th generation are generally very small. While increasing number of generations leads to the best mean and maximum algorithm output in IG, it also results in more time consumption.

Table 8. Average execution time (in miliseconds) of both algorithm versions and various t_{max} values

t_{max}	500	1000	1500	2000	2500	3000
CG	60	90	120	140	170	220
IG ($N_g = 100$)	40	60	80	90	100	110
IG ($N_g = 150$)	60	90	110	120	140	160

One can see (table 8) that IG is faster than CG. It results from the fact that inserting mutation is more time consuming when the graph is complete - there are more different possibilities to check. Increasing N_g to 150 in IG lengthen execution time by 40-50% and then it is similar to CG time. Edge completion time in CG was not included. Algorithms were implemented in C++ and run on Intel Pentium M740 1.7 GHz CPU.

5. Conclusions

Comparison results show that the performance of both algorithm versions depends strongly on t_{max} value. For lower t_{max} (500-1000) complete graph version (CG) gives better mean results. It also gives routes with higher maximum profits. For higher t_{max} values incomplete graph version (IG) excels in terms of average results. This effect is even more pronounced when increasing N_g. In this case maximum results are also higher for IG. It can be also seen that the choice of heuristic influences the algorithm results i.e. adding removing mutation improves CG significantly contrary to IG.

It should be noted that a lot depends on a network itself. Results obtained using the network from our experiment can be different when using another network. Thus, it is recommended to check both algorithm version with various heuristics on a given network and we plan to test algorithms on even bigger network of 500 tourist facilities. It is also recommended to compare CG with GLS [19] on benchmark tests, as the latter is considered the most effective heuristic operating on complete graphs.

References

[1] J. Koszelew, Logistics for globetrotters - innovative software component for e-tourism, Logistics, vol. 6, 54-56, 2010.

[2] W. Souffriau, P. Vansteenwegen, Tourist Trip Planning Functionalities: State-of-the-Art and Future, Springer, vol. 6385/2010, 474-485, 2010.

[3] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, D. Van Oudheusden, The City Trip Planner: An expert system for tourists, Elsevier, vol. 38, 2011.

[4] G. Laporte, S. Martello, The Selective Traveling Salesman Problem, Discrete Applied Mathematics, vol. 26, pp. 193-207, 1990.

[5] M. Hayes, J. M. Norman, Dynamic Programming in Orienteering: Route Choice and the Siting of Controls, Journal of the Operational Research Society, vol. 35, no. 9, pp. 791-796, 1984.

[6] R. Ramesh, K. M. Brown, An Efficient Four-Phase Heuristic for the Generalized Orienteering Problem, Computers and Operations Research, vol. 18, no. 2, pp. 151-165, 1991.

[7] B. Golden, Q. Wang, L. Liu, A Multifaceted Heuristic for The Orienteering Problem, Naval Research Logistics, vol. 35, pp. 359-366, 1988.

[8] Q. Wang, X. Sun, B. L. Golden, J. Jia, Using Artificial Neural Networks to Solve the Orienteering Problem, Annals of Operations Research, vol. 61, pp. 111-120, 1995.

[9] M. F. Tasgetiren, A. E. Smith, A Genetic Algorithm for the Orienteering Problem, Proceedings of the 2000 Congress on Evolutionary Computation, San Diego, CA, pp. 1190-1195, 2000.

[10] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, D. Van Oudheusden, A guided local search metaheuristic for the team orienteering problem, European Journal of Operational Research in Press, vol. 196, 2008.

[11] T. Tsiligirides, Heuristic Methods Applied to Orienteering, Journal of Operational Research Society, vol. 35, no. 9, pp. 797-809, 1984.

[12] A. Garcia, M.T. Linaza, O. Arbelaitz, P. Vansteenwegen, Intelligent Routing System for a Personalised Electronic Tourist Guide, Springer, 4, 185-197, 2009.

[13] B. Golden, L. Levy, R. Vohra, The orienteering problem, Naval Research Logistics 34, 1987.

[14] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, 1979.

[15] P. Vansteenwegen, W. Souffriau, D. Van Oudheusden, The orienteering problem: A survey, Elsevier, vol. 209, 2010.

[16] I. Chao, B. Golden, E. Wasil, Theory and methodology - the team orienteering problem, European Journal of Operational Research 88, 464-474, 1996.

[17] F. Tricoire, M. Romauch, K. Doerner, R. Hartl, Heuristics for the multi-period orienteering problem with multiple time windows, Computers and Operations Research 37 (2), 351-367, 2010.

[18] A. Piwonska, J. Koszelew, A Memetic Algorithm for a Tour Planning in the Selective Travelling Salesman Problem on a Road Network Springer, vol. 6804, 684-694, 2011.

[19] W. Souffriau, P. Vansteenwegen, J. Vertommen, G. Vanden Berghe, D. Van Oudheusden, A personalized tourist trip design algorithm for mobile tourist guides, Applied Artificial Intelligence, 22:10, 964-985, 2008.

[20] http://piwonska.pl/research

PORÓWNANIE ALGORYTMÓW GENETYCZNYCH ROZWIĄZUJĄCYCH ORIENTEERING PROBLEM PRZY POMOCY GRAFU PEŁNEGO I NIEPEŁNEGO

Streszczenie Celem pracy było porównanie dwóch odmian algorytmu (wersja dla grafu pełnego i niepełnego) rozwiązujących Orienteering Problem (OP). W większości artykułów dotyczących OP graf jest pełny, a jego krawędzie spełniają nierówność trójkąta, natomiast w naszej wersji takie założenia mogą nie być spełnione. Może to być bardziej praktyczne ponieważ sieci transportowe są grafami, ktore nie muszą spełniać tych warunków. W takich przypadkach grafy sa zazwyczaj uzupełniane fikcyjnymi krawędziami, a następnie działają na nich algorytmy rozwiązujące klasyczną wersje OP, które operują na grafie pełnym. Artykuł odpowiada na pytanie: czy pod względem jakości wyników i czasu obliczeń lepiej jest przekształcać graf do klasycznej formy OP przed uruchomieniem algorytmu w wersji dla grafu pełnego czy rozwiązywać OP na grafie niezmienionym i nie spełniającym dodatkowych założen (wersja dla grafu niepełnego)? Eksperyment został przeprowadzony na prawdziwej sieci transportowej w Polsce, a jego wyniki sugerują, że warto sprawdzać obie wersje algorytmu na konkretnych sieciach.

Słowa kluczowe: orienteering problem, OP, sieć transportowa, algorytm genetyczny, GA, grap pełny, graf niepełny

Artykul zrealizowano w ramach pracy badawczej S/WI/01/10.

MULTICLASS CLASSIFICATION STRATEGY BASED ON DIPOLES

Magdalena Topczewska

Faculty of Computer Science, Bialystok University of Technology

Abstract: The problem of multiclass classification is considered and resolved through the approach based on dipoles. The found hyperplane separates objects from different classes cutting between them and not through their middle. The crux is to define a suitable functional, which is small on lines with good separation power and little damage, easy to calculate and to minimize. The numerical tests were performed and the criterion modified in a way that preserves the intention of finding cuts between classes, which separate as many data points as possible. The approach was tested on some synthetic data sets using a recursive implementation.

Keywords: classification, multiclass problem, dipole

1. Introduction

The problem of classification is encountered in various areas, such as medicine to identify a disease of a patient, or industry to decide whether a defect has appeared or not. The aim of supervised classification methods is to construct a learning model from a labeled training data set to be able to classify new objects with unknown labels.

Assume that a training data set is given of the form $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathbb{R}^n$ is a vector of attributes of the ith object and y_i is the ith class label of C_l where $l \in \{1,\dots,k\}$. We aim at finding a learning model H such that $H(\mathbf{x}_i) = y_i$ for new unlabeled objects. The problem is simply formulated in the two class case, where the labels y_i are just +1 or -1 for the two classes involved. In such a case many different approaches have been proposed and developed over last few decades, for instance *LDA* (*Fisher's Linear Discriminant Analysis*) [11], *optimal Bayes rule*, *Support Vector Machines* (*SVM*) [22,5], classifiers using convex and piecewise linear (*CPL*) criterion functions [3,20], et al.

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 8, pp. 77-90, 2011.

However, the case of multicategory classification is more complicated. Four groups of methods to solve such problems can be described. The first group includes methods, which can be naturally extended from binary problems. The second group is constituted by methods using decomposition into binary classification tasks. The third group consists of methods converted from binary to multiclass approaches by changing the criterion functions and the forth group is described by hierarchical classification methods.

1.1 Extensible methods

This group of methods gathers algorithms which may be applied regardless of the number of classes in a data set. This includes *Naive Bayes algorithm*, *k Nearest Neighbour* method, *Classification and Regression Trees* (*CART*) or *neural networks* (*NN*).

The *Naive Bayes* algorithm is a classification algorithm based on the Bayes rule. It assumes the attributes in a data set are all conditionally independent of one another. Regardless of whether the number of classes equals two or more, the method computes the posterior probability of that sample belonging to each class. The new object is classified according to the largest posterior probability using the maximum a posteriori decision rule [16].

The main idea of *k Nearest Neighbour* (*kNN*) method is to find the nearest *k* neighbours of a chosen or new object and then use a majority decision rule to classify the new sample [7,10]. All objects are treated as points in *n*-dimensional space and we imply that there is a distance or dissimilarity measure that can be computed between samples based on the independent variables, for instance Euclidean distance calculated typically. The voting majority rule of the *kNN* algorithm is not affected by the number of classes.

Next approach concerns building of decision trees. A tree tries to infer a split of the data based on the values of the attributes to produce a good generalization. The split at each node is based on the attribute that gives the maximum information gain and the leaf nodes correspond to class labels. A new unlabeled object is classified according to a path from the root node to the leaf node. Among considerable number of *Classification and Regression Trees* (*CART*) algorithms C4.5 and ID3 are widely known [17,18].

Neural networks can also be attached to the extensible methods group. *NN*s are feed-forward neural networks with signals propagated only forward through the layers units and consist of three types of layers: the input layer, which is feed with the data; the hidden layer(s) of units and the output layer of units. The output layer has one unit for each diagnostic category, so-called 1-*of*-*k* encoding [16]. During the training phase the backpropagation learning algorithm adjusts weights of connections between units by propagation of the error among output layer and true classification labels. As the optimization tool the gradient descent method is applied to find the minimum of the error function.

The second type of neural networks in this paragraph are probabilistic neural networks (*PNN*). They belong to the Radial Basis Function (RBF) neural networks [16] and are comprised of three layers: input, hidden and output layers. The hidden layer consists of a pattern layer and a competitive layer of units. The pattern layer contains one unit for each object in the data set and applies a Gaussian density activation function, whereas the competitive layer consists of one unit for each class label which are activated only by pattern units associated only with the class of the trained object. In every unit of the competitive layer the probability of the object's membership to specific class is calculated. The output unit is activated according to the maximum probability value and the new object is classified to the activated unit's corresponding class.

1.2 Decomposition into binary classification problems

The most popular as well as basic and conceptually the simplest approach used in multiclass classification is to decompose the problem into multiple two-class classification problems and then solve them using efficient binary classifiers. There are a number of different approaches to decompose a k-class classification problem into two-class problems.

The first approach is called *one-versus-rest* (*OVR*) and the ith constructed binary classifier separates the ith class versus all other $k-1$ classes. The combined *OVR* decision function chooses the class for a new object that usually corresponds to the minimum value of the Hamming distance or the maximum value of a posteriori probability of the object's membership to each class. Even if described method is very simple, it might not be effective.

Another approach is *one-versus-one* (*OVO*) method. This method constructs one binary classifier for every pair of distinct classes and so, all together $\binom{k}{2} = k*(k-1)/2$ binary classifiers are constructed and using max-wins voting to decide to which class the new object should be placed.

In the *p-versus-q* (*PVQ*) approach p of the k classes are separated from the other q of the k classes. The process is repeated several times, each time a mix of p different classes against q different classes are chosen.

More complicated decomposition scheme is the Error-Correcting Output Codes (ECOC) method. The task is to convert k class classification problem into a large number l of binary problems and to use a unique codeword to a class instead of assigning each class label. An error correcting code is a l bit long, having unique codewords with a Hamming distance. Several methods for generating error correcting codewords and determining the l number were presented such as BCH codes [12,4], exhaustive codes [9], random codes [14] or scheme by Allwein et al. [1].

1.3 Reformulations of the objective function

This group of methods uses the conversion of criterion functions from binary to multi-class classification problems and includes among others multiclass Support Vector Machines (*MC-SVM*) and approaches by Weston and Watkins [24] and Crammer and Singer [8].

Method by Weston and Watkins is viewed as a natural extension of the binary SVM classification task. In the k-class case a single quadratic optimization problem of size $(k-1)*n$ is solved. This is identical to binary SVM when the number of classes is equal 2. The method reduces the number of support vectors needed to describe the decision functions [24].

Method proposed by Crammer and Singer [8] is similar to above-mentioned and the same size of $(k-1)*n$ quadratic problem is solved. The difference lies in using smaller number of slack variables in the constraints of the optimization problem. For both approaches the use of decompositions can provide a significant speed-up in the solution of the optimization problem [13].

1.4 Hierarchical classification

Another group of methods to solve the multiclass classification problem consists of approaches dividing the input space in hierarchical manner. Starting from the root node classes are divided into clusters and such a process is continued for each child node until the leaf nodes contain all objects from all classes of the data set. Finally, all k classes are arranged as a tree and classification of a new unlabeled object goes according to the path from the root to the leaf node. Detailed description of such methods can be found in [19]. Below only few methods are mentioned among many other .

Hierarchical Support Vector Machines (*HSVM*) solve a series of max-cut problems. Hierarchical and recursive partition of the set of classes into two-subsets, until the pure leaf nodes that have only one class label, are obtained [6]. The edge weights of the undirected graph represent the Kulback-Leiber distance between the classes and that is used to find subclusters mostly distant from each other. Then at each internal node the SVM is applied to construct the discriminant function for a binary classifier.

Binary Hierarchical Classifier [15] is another approach. The algorithm builds a binary tree with k leaf nodes, each corresponding to one class, using $k-1$ binary classifiers. At root the algorithm finds the best feature projection that distinguishes the two groups and then the binary split into two clusters of classes is done. The process is repeated for subsequent nodes. This approach was performed comparable to ECOC, with the added advantage of using fewer classifiers [2]

Next described approach is called *Divide-By-2* [23] and as previous one uses only $k-1$ binary classifiers to form a binary tree. Instead of Fisher Discriminant, either k-means algorithm is applied for clustering the class means into two groups or the classes grand mean is used as a threshold. The method puts classes with means smaller to the grand mean in one cluster and those with larger mean to the other [2]. Binary classification is performed by binary SVM.

2. Approach based on dipols

The proposed approach for multiclass classification belongs to the hierarchical methods family [21]. In the first step the algorithm finds the *best* split of the data. We try to divide all the objects into two clusters to obtain the smallest number of wrongly classified objects. Objects belonging to different classes should not be situated on different sides of the cutting hyperplane. Next steps of the algorithm are repeated in two subspaces separately finding best splits of clusters of remained data. Finally, the hierarchy in a form of a tree is obtained. The path from the root to the leaf nodes determines the membership of the object to the appropriate class.

Linear classification for the two class case assumes that a suitable functional Φ: $\mathbb{R}^n \to \mathbb{R}$, $\Phi(\mathrm{x}) = \mathrm{w}^T\mathrm{x} - b$, can be constructed, such that for some indices l $\Phi(\mathrm{x}) < 0$ for all $\mathrm{x} \in C_l$, while for the remaining indices the corresponding value will be positive. In the case of k=2, one such functional suffices.

In the multiclass case the essential is to formulate the criterion function which is unaffected by scattered data and outliers that can occur in real data sets. The proposed criterion function is based on the dipoles that denote ordered pairs of data vectors. Concerning the two class classification case two types of dipoles are available - clean and mixed ones. The dipole is clear if both objects belong to the same class, otherwise it is mixed. Having k classes, there are k sorts of clean dipoles and $k*(k-1)$ mixed dipole types. A mixed dipole is good to be reflected by a functional featuring different signs on the dipole's elements. On the other hand, mixed signs on a clean dipole are considered bad. One might simply count good and bad situations and make a balance. But of course, such an assessment is difficult to optimize, since it leads to an integer, hence discontinuous, objective.

Moreover, we prefer our assessment also to reflect how bad a bad cut is, and how safe a clean cut is. For example, a clean dipole (x,y) with $\Phi(x) = 0.001$ and $\Phi(y) = 1000$ is a potential risk, that a small perturbation of Φ may render it mixed, for x may go through the line. Analogously, a mixed dipole separated by Φ is not so safe if one of its parts has a small value $|\Phi(x)|$.

If the whole data set is represented by an $m \times (n+1)$ matrix D, where m denotes number of objects in a data set and n number of comprising values of attributes and knowing class indices, the proposition is to minimize a sum of the form

$$F(\Phi;\mathbf{D}) = \sum_i \sum_j \sum_{\mathbf{x}\in C_i} \sum_{\mathbf{y}\in C_j} F_{ij}(\Phi,\mathbf{x},\mathbf{y};\mathbf{D}) \tag{1}$$

where F is a function of the linear affine functional Φ. The space of all such functionals may be parametrized by a vector $\mathbf{p} = (\mathbf{w},b) \in \mathbb{R}^{m+1}$. The value of F depends obviously on the data, i.e. on the set of all feature vectors and their assignment to the k different classes.

The assessment function F is a double sum over class indices of contributions being double sums of terms of the form $F_{ij}(\mathbf{x},\mathbf{y})$. Each such term is a function of two real values that the functional Φ takes on the dipole with elements $\mathbf{x}$ and $\mathbf{y}$. For $i = j$ a dipole is clean. If $\Phi(\mathbf{x})$ and $\Phi(\mathbf{y})$ differ in sign, F_{ii} should be positive correspondingly. The bigger term in absolute value sets the sign of the dipole. The error is measured by the distance of the other one from the selected sign. The convex and piecewise linear functions studied in [3,21] may be applied to such a problem. We make F_{ii} the bigger the farther the smaller of the terms lays on the wrong side, starting from a given threshold on the good side.

The way of presenting the contributions F_{ij} opens another option. Basing on the whole set of data, we may decide that class C_i is positive (negative), and then calculate errors as in before. This would avoid making an individual decision for each dipole with both objects belonging to C_i. In fact, it is feasible to calculate the total error contribution for the clean dipoles from this class for both orientations and then take the smaller one.

In the case of $i \neq j$, different signs of $\Phi(\mathbf{x})$ and $\Phi(\mathbf{y})$ are in principle desired. To avoid problems with the mixed orientation, from i,j the index of the class with the highest value of Φ is chosen and class with this index is declared as the positive one. Then the piecewise linear error contributions for data from the positive class with too small Φ value are added, and likewise for data from the negative class which are too large. It proved sensible to consider values below a positive ε already as too small for the positive class, and likewise above a negative ε as too big for the negative class. Obviously, for each pair (i,j) with $i \neq j$ there is no need to consider (j,i), hence the second sum may be restricted to $j \leqslant i$.

The main drawback of the method is evidently the high number of terms in the quadruple sum. Besides, in the classical setting, the sum of error terms is convex and piecewise linear, which makes it comparatively amiable for minimization. The present task is inherently nonconvex.

3. Results

With the aim of presenting the performance of proposed approach, a few examples are shown below.

3.1 Example 1

As a first example a synthetic data set containing four classes was generated. In every class there are 250 objects.

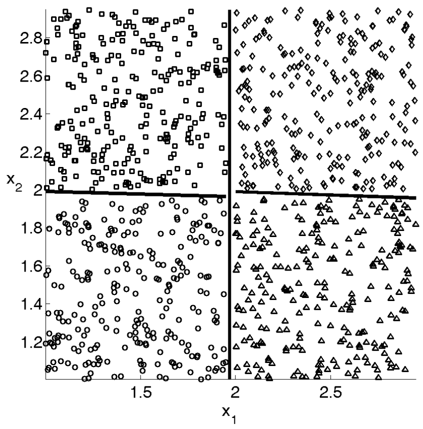
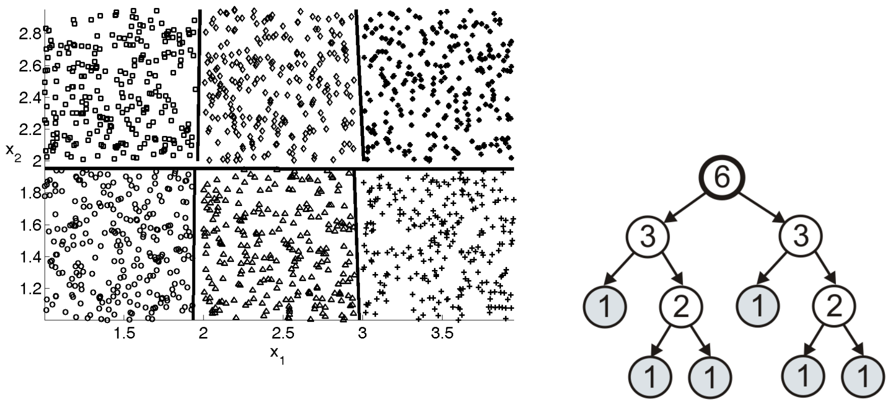


Fig. 1: Hierarchical classification of four classes

In the first step of the algorithm two classes are separated from the other two. Next, each halfplane is divided separately into two quarters with the final classification accuracy of 100%, see Fig. 1. Every quarter is associated with only one class.

3.2 Example 2

The second presented example describes classification problem of synthetic data sets and consisting of objects belonging to six and twelve different classes respectively. In each class there are 250 objects for the six classes data set and 100 objects for the remain data set.

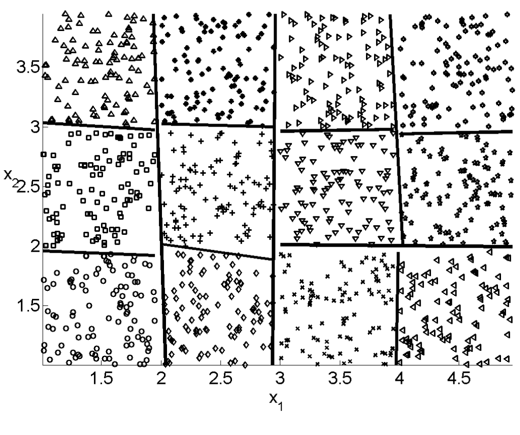


(a) data and successive splits (b) number of divided classes

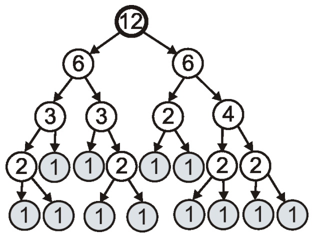
Fig. 2: Hierarchical classification of six classes

In the first step of the algorithm six classes are divided into two clusters, each of three classes. In the next step, separately for two halfplanes the space is divided into two and one class and finally two remained classes are separated by a line. Analogically, three areas each associated with different class are obtained on the other side of a halfplane, see Fig. 2.

In the twelve class case the first split divided all classes into two groups containing 6 classes each. Next,recursively two clusters of 3 classes were achieved, whereas at the other side of the hyperplane the split gave a cluster of 2 and a cluster of 4 classes. Next eight splits divided space into membership areas of every class, see Fig. 3.



(a) data and successive splits



(b) number of divided classes

Fig. 3: Hierarchical classification of twelve classes

3.3 Example 3

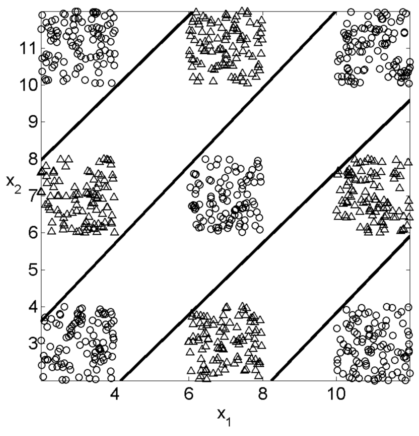
The third presented example describes classification problem of synthetic data sets and consisting of objects belonging to two classed set as a chessboard.

First division selected two clusters - 6 at the positive side of the hyperplane and 3 at the negative one. Next three splits gave areas of membersip of objects to two classes, see Fig. 4.

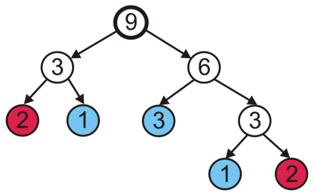
Calculations were performed in the Matlab system. Improvement of implementation by application of meshgrid shortened time about twenty times in relation to primal version of the method.

4. Conclusions

Finding a separating hyperplane for two classes by minimizing an error functional summing contributions for each poorly classified data point is by now common practice. Effective implementations in the framework of SVM or in terms of CPL functions are available and shown to work well for quite large sets of data. An approach based on dipoles is not needed. If, however, the number of classes increases, a dipole based criterion may be helpful to split large data first into smaller subsets, each containing not only less feature vectors, but above all a smaller number of classes. We start from a theoretical formulation of a criterion suitable for this task, modeled on



(a) data and successive splits



(b) number of divided clusters

Fig. 4: Hierarchical classification of two classes

the approach presented in [3]. Next, we perform numerical tests and modify the criterion in a way that preserves the intention of finding cuts between classes and not through their middle, which separate as many data points as possible. However, we do not count in integers, but weigh by taking the distance of the object to the wrong side. This approach was tested on some synthetic data and performed well enough to be considered for implementation into a recursive production code.

The impact of quick and automatic classification of data in any field of computer aided activity human, like design, manufacturing, quality control or medicine, diagnostics, image analysis can hardly be overestimated. The presented problem is hard because of its inherently non-convex character. The solution applicable since it does not require human intervention. It easily and quickly breaks down a multiclass problem by divide and conquer into a sequence of smaller and smaller problems, which in a post-processing step may be handled by classical methods available for the two-class case.

The plan is to perform comparative analysis with the algorithms of the hierarchical classification family.

References

[1] E.L. Allwein, R.E. Schapire, Y. Singer, Reducing multiclass to binary: a unifying approach for margin classifiers, *Journal of Machine Learning Research*,

1:113-141, 2001.

[2] M. Aly, Survey on multiclass classification methods, Technical Report, Caltech, USA, 2005.

[3] L. Bobrowski, *Data exploration based on convex and piecewise criterion functions* (in Polish), Wydawnictwo Politechniki Białostockiej, Białystok, 2005.

[4] R.C. Bose, D.K. Ray-Chaudhuri, On A Class of Error Correcting Binary Group Codes, *Information and Control* 3:68–79, 1960.

[5] C.J. Burges, A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, 2:121-167, 1998.

[6] Y. Chen M.M. Crawford, J. Ghosh, Integrating Support Vector Machines in a hierarchical output space decomposition framework, In Proceedings of 2004 International Geoscience and Remote Sensing Symposium, Anchorage, Alaska, vol. 2, 949–953, 2004.

[7] T.M. Cover, P.E. Heart, Nearest neighbor pattern classification, *IEEE Trans. Inform. Theory*, IT 13:21-27, 1967.

[8] K. Crammer, Y. Singer, On the learnability and design of output codes for multiclass problems. *Proceedings of the Thirteen Annual Conference on Computational Learning Theory* (COLT 2000), Stanford University, Palo Alto, CA, June 28 - July 1, 2000.

[9] T.G. Dietterich, G. Bakiri, Solving multiclass learning problem via error correcting codes, *Journal of Artificial Intelligence Research*, 2:263-386, 1995.

[10] O.R. Duda, P.E. Heart, D.G. Stork, *Pattern Classification*, Second edition, John Wiley & Sons, 2001.

[11] R. A. Fisher, The use of multiple measurements in taxonomic problems, *Annals of Eugenics*, 7:179–188, 1936.

[12] A. Hocquenghem, Codes correcteurs d'erreurs (in French), Chiffres, Paris, 2, 147–156, 1959

[13] C.W. Hsu, C.J. Lin, A comparison of methods for multi-class support vector machines, *IEEE Trans. Neural Netw.*, 13:415-425, 2002.

[14] J.E. Gentle, *Random Number Generation and Monte Carlo Methods*, Springer Verlag, 1998.

[15] S. Kumar, J. Ghosh, M.M. Crawford, Hierarchical fusion of multiple classifiers for hyperspectral data analysis, *Pattern Analysis& Applications*, 5:210-220, 2002.

[16] T.M. Mitchell, *Machine Learning*, McGraw-Hill Science, New York, 1997.

[17] J.R. Quinlan, Induction of decision trees, *Machine Learning* 1(1):81-106, 1986.

[18] J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufman Publishers, Inc., 1993.

[19] C.N. Silla Jr., A.A. Freitas A survey of hierarchical classification across different application domains, Data Mining and Knowledge Discovery, 22:31-72, 2011.

[20] M. Topczewska, K. Frischmuth, Numerical aspects of weight calculation in classification methods, *In PTSK Conference*, Krynica Górska, Poland, Sept. 26-29, 2007.

[21] M. Topczewska, K. Frischmuth, Classification strategies based on dipols (in Polish), *Pomiary, Automatyka, Kontrola*, 56(6):632-635, 2010.

[22] V. N. Vapnik, *Statistical learning theory* Wiley J., 1998.

[23] V. Vural, J.G. Dy, A hierarchical method for multi-class support vector machines. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 105-112, 2004.

[24] J. Weston, C. Watkins, Support vector machines for multi-class pattern recognition, In *Proceedings of the Seventh European Symposium On Artificial Neural Networks* (ESANN 99), Bruges, April 21-23, 1999.

STRATEGIA KLASYFIKACJI WIELOKLASOWEJ OPARTA NA DIPOLACH

Streszczenie W pracy rozpatrywane jest zagadnienie klasyfikacji w przypadku wieloklasowym oraz podejście oparte na dipolach. Poszukiwana hiperpłaszczyzna powinna rozdzielać obiekty należące do różnych klas, ale nie przecinając środka żadnej klasy. Zdefiniowano w tym celu odpowiedni funkcjonał, by przyjmował on małe wartości w przypadku prawidłowej klasyfikacji większości obiektów, był prosty do obliczenia i minimalizacji. Przeprowadzono testy numeryczne oraz dokonano modyfikacji kryterium, by znaleźć takie rozdzielenie klas, by odseparować możliwie dużo obiektów. Podejście było testowane na wybranych syntetycznych zbiorach danych przy wykorzystaniu implementacji w postaci wywołań rekurencyjnych.

Słowa kluczowe: klasyfikacja, problem wieloklasowy, dipol

LISTA RECENZENTÓW / THE LIST OF REVIEWERS (2010-2011)

1. Piotr Augustyniak (Kraków)
2. Alexander Barkalov (Zielona Góra)
3. Michał Bereta (Kraków)
4. Johanne Bezy-Wendling (Rennes)
5. Marcin Blachnik (Katowice)
6. Cezary Bołdak (Białystok)
7. Wojciech Bożejko (Wrocław)
8. Rafał Dreżewski (Kraków)
9. Marcin Dziubiński (Warszawa)
10. Piotr Kisielewski (Wrocław)
11. Adam Klimowicz (Białystok)
12. Mieczysław A. Kłopotek (Warszawa)
13. Marek Kociński (Łódź)
14. Eugeniusz Kornatowski (Szczecin)
15. Romuald Kotowski (Warszawa)
16. Jarosław Kozik (Kraków)
17. Marek Krętowski (Białystok)
18. Wojciech Kwedlo (Białystok)
19. Elżbieta Majewska (Białystok)
20. Robert Milewski (Białystok)
21. Andrzej Mitas (Katowice)
22. Hung Son Nguyen (Warszawa)
23. Andrzej Obuchowicz (Zielona Góra)
24. Joanna Olbryś (Białystok)
25. Tadeusz Pankowski (Poznań)
26. Marek Parfieniuk (Białystok)
27. Marek Piotrów (Wrocław)
28. Agnieszka Rossa (Łódź)
29. Khalid Saeed (Kraków)
30. Władysław Skarbek (Warszawa, Białystok)
31. Marcin Skoczylas (Białystok)
32. Piotr M. Szczypiński (Łódź)
33. Krzysztof Ślot (Łódź)
34. Marek Tabędzki (Białystok)
35. Tomasz Traczyk (Warszawa)
36. Krzysztof Trojanowski (Warszawa)
37. Alicja Wakulicz-Deja (Katowice)
38. Rafał Wcisło (Kraków)
39. Mariusz Wrzesień (Rzeszów)