Zeszyty Naukowe Politechniki Białostockiej

INFORMATYKA

Zeszyt 2

Wydawnictwo Politechniki Białostockiej
Białystok 2007

Redaktor naukowy:
dr inż. Marek Krętowski

Recenzenci:
dr hab. inż. Ryszard S. Choraś, prof. UTP
dr hab. inż. Krzysztof Goczyła, prof. PG
dr hab. inż. Mieczysław A. Kłopotek, prof. AP
prof. dr hab. inż. Jacek Koronacki
prof. dr hab. inż. Juliusz L. Kulikowski
dr hab. inż. Halina Kwaśnicka, prof. PWr
dr hab. Grażyna Mirkowska-Salwicka, prof. PJWSTK
prof. dr hab. inż. Stanisław Osowski
dr hab. Christoph Schwarzweller
prof. dr hab. inż. Kazimierz Subieta
prof. dr hab. inż. Sławomir T. Wierzchoń

Opracowanie redakcyjne:
Jadwiga Żukowska

Sekretarz redakcji:
mgr inż. Tomasz Łukaszuk

ISSN 1644-0331

Druk:
Dział Wydawnictw i Poligrafii Politechniki Białostockiej

Nakład:100 egz.

SPIS TREŚCI

CONTENTS

1. Grzegorz **Bancerek** 5
EXPLORING MIZAR LIBRARY WITH MML QUERY
EKSPLORACJA BIBLIOTEKI MIARA Z UŻYCIEM MML QUERY

2. Leon **Bobrowski** 19
SEPARABLE DATA AGGREGATION BY LAYERS OF ELEMENTARY CLASSIFIERS
SEPAROWALNA AGREGACJA DANYCH W WARSTWACH KLASYFIKATORÓW ELEMENTARNYCH

3. Andrzej **Chmielewski**, Sławomir T. **Wierzchom** 39
ON THE DISTANCE NORMS FOR DETECTING ANOMALIES IN MULTIDIMENSIONAL DATASETS
O METRYKACH ODLEGŁOŚCI DLA WIELOWYMIAROWYCH ZBIORÓW DANYCH WYKORZYSTYWANYCH W ALGORYTMIE SELEKCJI NEGATYWNEJ O WARTOŚCIACH RZECZYWISTYCH

4. Dorota **Duda**, Marek **Krętowski**, Johanne **Bézy-Wendling** 51
EKSTRAKCJA CECH TEKSTURALNYCH W KLASYFIKACJI OBRAZÓW TOMOGRAFICZNYCH WĄTROBY
TEXTURE FEATURE EXTRACTION IN LIVER CT IMAGE ANALYSIS

5. Małgorzata **Krętowska** 67
LASY LOSOWE – OCENA JAKOŚCI PROGNOSTYCZNEJ CECH
RANDOM FORESTS – EVALUATION OF PREDICTIVE ACCURACY

6. Tomasz **Łukaszuk**, Leon **Bobrowski** 79
TEMPORALNOŚĆ W MODELACH RANGOWYCH
TEMPORALITY IN RANDEK MODELS

7. Joanna **Olbryś** 93
SIEĆ BAYESOWSKA JAKO NARZĘDZIE POZYSKIWANIA WIEDZY Z EKONOMICZNEJ BAZY DANYCH

BAYESIAN NETWORK AS A TOOL OF EXTRACTING KNOWLEDGE FROM AN ECONOMIC DATABASE

8. Agnieszka **Oniśko** 109
KNOWLEDGE ACQUISITION FROM HUMAN EXPERTS FOR BUILDING BAYESIAN NETWORK MODELS
POZYSKIWANIE WIEDZY OD EKSPERTÓW W BUDOWANIU MODELI SIECI BAYESOWSKICH

9. Tomasz **Rybak** 121
PROBLEMS WITH STORING TEMPORAL DATA
PROBLEMY ZE SKŁADOWANIEM DANYCH TEMPORALNYCH

10. Marta K. **Smolińska**, Zenon A. **Sosnowski** 141
PODSUMOWANIA LINGWISTYCZNE Z GRUPOWANIEM ROZMYTYM
LINGUISTIC SUMMARIES WITH FUZZY CLUSTERING

11. Paweł **Tadejko** 155
ADDITIONAL DATA PREPROCESSING AND FEATURE EXTRACTION IN AUTOMATIC CLASSIFICATION OF HEARTBEATS
DODATKOWE PRZETWARZANIE WSTĘPNE I EKSTRAKCJA CECH W PROCESIE AUTOMATYCZNEJ KLASYFIKACJI RYTMU SERCA

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2007
Informatyka – Zeszyt 2

Grzegorz Bancerek[1]

EXPLORING MIZAR LIBRARY WITH MML Query

Abstract: MIZAR, a proof-checking system, is used to build the MIZAR Mathematical Library (MML). MML Query is a semantics-based tool for managing the mathematical knowledge in MIZAR including searching, browsing and presentation of the evolving MML content. The tool is becoming widely used as an aid for MIZAR authors and plays an essential role in the ongoing reorganization of MML.
In the paper, we briefly present MIZAR system including language, tools for logical verification and publishing, foundations of MML, its content and maintenance, and the problems raising when using the MML (information retrieval and rendering). We also present the possibilities offered by MML Query to solve these problems.

Keywords: MIZAR, sematic searching, repositories of formalized mathematics

1. Overview of the Mizar project

The MIZAR language is a language used for such a formalization of mathematics that is close to the vernacular used in mathematical publications. An implemented MIZAR verifier is available for checking correctness of MIZAR texts according to Jaśkowski natural deduction. The perpetual development of the MIZAR system (see [17]) has resulted in the MIZAR Mathematical Library (MML)—a centrally maintained library of formalized mathematics based on Tarski–Grothendieck set theory. Contributions to MML have been the main activity of the MIZAR project since the late 1980s. MML is organized as an interrelated collection of MIZAR articles. At this moment—December 2006—there are 959 articles in MML, occupying 70 MB, containing 43149 theorems and 8185 definitions. The most important facts included in MML are

- Jordan Curve Theorem (JCT), [16],
- Gödel Completeness Theorem (GCT), [12],
- Fundamental Theorem of Algebra (FTA), [18]
- Reflection Theorem, [6]

[1] Faculty of Computer Science, Białystok Technical University, Białystok

JCT is a substantial achievement of the project and is the result of a long-lasting cooperation between Shinshu University and the University of Białystok which was initiated by Yatsuka Nakamura in 1992 and has involved 16 people. About 70 articles[2] from the MML are devoted, directly or indirectly, to the JCT project.

Another large project within MML, called the CCL project [3,9], is aimed at formalization of the theory of continuous lattices as presented in *A Compendium of Continuous Lattices*, [14]. 58 MIZAR articles written by 16 authors cover about 65% of the main course of the book at the moment.

MML is commonly considered the biggest library of computer proof-checked mathematics. The scale of the development of MML could be measured by the number formalized theorems from the list The Hundred Greatest Theorems[3]. Currently, MML contains 35 of 100 theorems. It place the MIZAR system in fourth position in the ranking[4] maintained by Freek Wiedijk.

Notwithstanding the above, MML's coverage of mathematical knowledge is still minuscule. Even so, information retrieval in MML became a burning issue a long time ago. The lack of searching tools, which would be more advanced than some `grep`-based utilities, had delayed work in the CCL project when several authors formalized interrelated parts of the theory using various and barely compatible formalizations from the MML. This prompted in 2000 efforts aiming at development of a semantics-based searching tool for MML [3] and it was the origin of the MML Query system [10].

2. MML Query semantic searcher

The first release of MML Query was completed in 2001 and included basic queries enabling semantic searching of library items only. The second release completed in 2002 is described in [10]. It introduced a number of searchable resources and a variety of queries enabling more advanced searching. Additionally, beginnings of semantic presentation of MML were available in this release. The third release developed in 2004-2005 was inspired by the works aimed at presentation of the content of MML for different purposes:

– an application of MML Query in the Trial-Solution project [13] to generate semantically linked slicing of MIZAR articles,

[2] These articles are not devoted only to JCT but also to Brouwer Theorem, Urysohn Theorem, Tietze Theorem, etc.

[3] `http://personal.stevens.edu/~nkahl/Top100Theorems.html`

[4] Formalizing 100 Theorems, `http://www.cs.ru.nl/~freek/100/`

– translation of MML into the OMDOC format [15],
– semantic browsing in Emacs [11].

These investigations as well as continual development of the web interface to the MML Query system resulted in a text transformation processor MMLQT[5] which is able to interpret the MML Query language. The language in the third release was improved to satisfy requirements of MMLQT (ordered queries, version queries, and metadata queries) and to make searching with MML Query somewhat easier (non-expert searching, rough queries).

Currently, MML Query provides the following functionalities: semantic searching, semantic browsing, semantic presentation, collection of MML statistics,[6] and assistance for authoring MIZAR articles with Josef Urban's Mizar mode for Emacs [21,11]. In consequence, the tool facilitates individual authoring as well as collaborative work in larger projects by enabling adequate searching and uniform and unambiguous presentation. In particular, MML Query provides the possibility to create monographs–the uniform ordered semantic presentation of a specified piece of a theory which may be spread over the MML. These features of MML Query are also used in the ongoing reorganization of MML into the Encyclopedia of Mathematics in MIZAR.

3. MML Query language and its processing

The description of the syntax and semantics of the MML Query language is available on the web[7]. It was also presented in [10] and [4].

The parser of the MML Query language realize the idea of *non-expert queries* (see [4]). It means that first it tries to parse the query according to the syntax. If it is impossible, the parser tries to recognize the input as a part of a MIZAR formula. This is done by recognition of MIZAR symbols and their contexts. For each symbol and for each context the parser generates appropriate queries and composes them in a correct way. Finally, two versions are presented: a strict query and a rough query. Both queries are completed by ordering part which orders the result according to the number of concepts used. The strict query is executed first and if the result is empty, then the rough query is also executed. Such realization helps less experienced users to get better querying results. It also can save time for experts and gives a good base for editing more precise queries.

For example, parsing for the input `theorem 0 = 1` which is not a query according to the syntax of MML Query language generates the following strict query Q_1:

[5] MML Query Templates or MML Query Transformation
[6] Accessible on WWW at `http://mmlquery.mizar.org/`
[7] `http://mmlquery.mizar.org/`

```
1   number 0 occur and
2   (symbol '=' [[[notation | constructor] or vocabulary] | occur]) and
3   number 1 occur
4   | filter th
5   ordered by number of ref
```

and the following rough query Q_2:

```
1   rough max-max(
2     number 0 occur,
3     (symbol '=' [[[notation | constructor] or vocabulary] | occur]),
4     number 1 occur
5   )
6   | filter th ordered by number of ref
```

The strict query Q_1 results in 1544 elements and then the rough query Q_2 is not executed (if it was executed the result would be the same). The elements are ordered by the number of constructors referred to (ordering part in line 5 in Q_1 above). It means that at the beginning we can find the elements using the smallest number of concepts. All elements are theorems (filtering query in line 4) which include number 0 (line 1), number 1 (line 3) and the equality (line 2). Actually, they might not include the equality but any other concept denotated with the symbol '='. But because all 23 notations from MML[8] using the symbol '=' concern the original equality[9] there is no such possibility. The first 4 theorems returned as a result of the query Q_1 are the following

```
CARD_1:87     1 = {0};

COMPTRIG:57   Arg 1 = 0;

NEWTON:18     0! = 1;

NEWTON:27     0 choose 0 = 1;
```

and concern, respectively, the von Neumann form of number 1, the angle of complex number 1, the factorial of number 0, and the Newton binomial coefficient $\binom{0}{0}$. The first column gives MML Query names of theorems which are also MML names used in MIZAR articles for referring (justifying reasoning steps).

[8] the query: `symbol '=' notation` returns 23 elements

[9] the query `symbol '=' notation | constructor | origin` results in one element `HIDDEN:pred 1` which is MML Query name for the original equality.

Execution of the rough query Q_2 starts from executing each partial query (lines 2-4). Each element from obtained results gets the number of partial queries it comes from. E.g., theorems `CARD_1:87` is the answer of each partial query and then it gets the number 3. The biggest number is set as `max` (for Q_2 it is 3) and all elements with numbers between `max` and `max` are returned as the result (all elements satisfying the conjunction of three partial queries). Rough query allows also to provide the limiting numbers directly, e.g., `rough 1-3 (...)`, and to use the word `count` which stands for the number of partial queries. The query `rough count (q1, ..., qn)` is an abbreviation of `rough count-count (q1, ..., qn)` and is equivalent to the conjunction `q1 and ... and qn`

Queries Q_1 and Q_2 above include *proper* parts and *ordering* parts. The *proper* part is obligatory in correct queries and *ordering* parts are optional. A correct query may also include *selection* or *presentation* parts. The *ordering* and *selection* parts may be nested inside the *proper* part if necessary but the *presentation* part may occur only at the end of whole query. For example, in the correct query

```
1   at least minus 3 * (
2      {TOPGEN_3:17,TOPGEN_3:30,CARD_2:44} | ref
3      ordered by number of occur reversed
4      select 0-6
5   )
6   ordered by number of ref
7   presented with version 4.66.942
```

an *ordering* part occurs twice (lines 3 and 6). The first *ordering* part and a *selection* part (line 4) are included in the main *proper* part (lines 1-5). A *presentation* part appears at the end (line 7). The *proper* part of the subquery from lines 2-4 is included in line 2. The theorems in line 2 are theorems from the list of 100 Greatest Theorems:

- The Denumerability of the Rational Numbers, [5]
- The Non-Denumerability of the Continuum, [5]
- The Number of Subsets of a Set (PS[10]), [2]

The sub-subquery in line 2 selects all constructors (concepts) referred to by these theorems. Next, the ordering rule `number of occur reversed` in line 3 orders them according to descending number of all occurrences in whole MML. So, the most popular constructors are at the beginning. The selection in line 4 takes first 7 of them. The rough variant of group query `at least minus 3 * ...` finds all elements from MML which refer to at least $4 = 7 - 3$ chosen constructors. Finally, the result is ordered

[10] Power Set

by the number of constructors referred to and presented according to the version 4.66.942 of MML.

Examples of less complicated (basic) queries are given below.

```
CARD_2:44 ref                                                    (1)
{TOPGEN_3:17,TOPGEN_3:30,CARD_2:44}                              (2)
list of th from TOPGEN_3                                         (3)
(CARD_2:44 ref) butnot (list of constr from CARD_2)              (4)
(list of th from TOPGEN_3) | ref                                 (5)
(TOPGEN_3:def 4 ref) & occur                                     (6)
list of th where [ref | filter struct]                           (7)
list of constr where notation > 1                                (8)
list of th where positive ref <= negative ref                    (9)
```

The queries listed above may be read as follows: (1) all constructors appearing in PS, (2) list of three theorems mentioned above, (3) all theorems from article TOPGEN_3, (4) all constructors from PS defined in articles other than CARD_2, (5) all constructors appearing in any theorem from article TOPGEN_3, (6) all items which refer to all constructors appearing in the definitional theorem of $\mathfrak{c}$ (continuum), (7) all theorems concerning structures, (8) all constructors with more than one notation, (9) all theorems that refer positively to a bigger number of constructors than they refer negatively. Other examples of queries can be found in [10] and [4].

4. MML Query Transformation

The presentation of the results of querying as well as the presentation of the content of MML for other purposes prompted the development of MML Query Transformation (MMLQT). At the moment, MMLQT is a text processor used for rendering MML content when browsing MML Query results, making MML statistics, or generating semantically linked abstracts [11] and OMDoc repository [15].

The transformation processor is based on a bunch of templates which include directives to the processor. Each purpose above has its own bunch of templates. MMLQT templates are text files which could be written in different styles, e.g., XML or LaTeX. The key role in XML style is played by the XML element <mmlq> which holds the directives. The transformation processor parses only the occurrences of <mmlq> and does not count any other XML elements—the text outside <mmlq> elements is simply rewritten. The directives are specified with the XML attribute type of <mmlq>

element. Namely, the value of the attribute determines the task to be performed by the processor. The performance depends on changing environment of the MMLQT processor which includes inter alia the current *argument* and the *version* of MML. Directives may change the environment directly (e.g., directive `change` for *argument* and `changever` for *version*) or indirectly (directive `foreach` for *argument*).

The base directive is `explain`. The role of this directive is presenting the meaning of an appropriate MML Query element according to the current presentation style which may differ in different purposes. The directive may have, for example, the following forms

```
<mmlq type="explain"/>
<mmlq type="explain" argument="CARD_2:44"/>
<mmlq type="explain" operation="ref"/>
<mmlq type="explain" query="list of notat from TOPGEN_3"/>
```

The first directive explains simply the meaning of *argument* and the second explains the meaning of theorem PS (`CARD_2:44`). The third explains the first element from the list of elements referred to by *argument*. The last explains the first notation from article `TOPGEN_3`. To explain all notations from `TOPGEN_3` we must use a little bit more complicated directive

```
<mmlq type="foreach" query="list of notat from TOPGEN_3">
  <mmlq type="explain"/>
</mmlq>
```

MMLQT templates could be created with the web interface Template Maker available at

`http://mmlquery.mizar.org/template-maker.php`

It has some support for editing directives in XML style and allows to render templates with the presentation style prepared for MML semantic browsing and statistics. It means that all symbols in rendered MIZAR formulae are linked to their definitions with suggestions for further browsing.

We may use Template Maker to render the meaning of 3 theorems mentioned in the previous section. It can be done by writing the following template

```
<mmlq type="foreach" query="{TOPGEN_3:17,TOPGEN_3:30,CARD_2:44}">
  <center><mmlq type="value"/></center>
  <mmlq type="explain"/>
  <hr>
</mmlq>
```

The template includes query (2) and the directive `value` which renders the MML Query name of *argument*. Moreover, two HTML elements, `<center>` and `<hr>`, are added to beautify the rendering:

```
                          CARD_2:44
theorem
for b1 being set holds
   exp(2,Card b1) = Card bool b1;
```

```
                          TOPGEN_3:17
theorem
Card RAT = alef 0;
```

```
                          TOPGEN_3:30
theorem
alef 0 in continuum;
```

The body of `<mmlq>` element with `foreach` directive is a subtemplate which is processed by MMLQT processor for each MML Query item retrieved with the query from XML attribute `query`.

MML Query statistics are generated by processing MMLQT templates. The main role in these templates is played by the directive `count` which renders the quantity of the result of appropriate query. For example, the main statistic page shows quantities of different resources in MML.

Version 4.76.959:

- 189 authors,
- 960 articles,
- 43149 theorems,
- 8185 definitions,
- 739 schemes,
- 9791 constructors: 106 aggregates, 1846 attributes, 6330 functors, 410 modes, 851 predicates, 142 selectors, and 106 structures

...

It is rendered from the following template:

```
Version <mmlq type="version" part="MML"/>:
<ul>
<li><mmlq type="link" template="authors.mqt">
       <mmlq type="count" query="list of article|author"/> authors
    </mmlq>,
<li><mmlq type="count" query="list of article"/> articles,
<li><mmlq type="count" query="list of th"/> theorems,
<li><mmlq type="count" query="list of def"/> definitions,
<li><mmlq type="count" query="list of sch"/> schemes,
<li><mmlq type="count" query="list of constr"/> constructors:
    <mmlq type="count" query="list of aggr"/> aggregates,
    <mmlq type="count" query="list of attr"/> attributes,
    <mmlq type="count" query="list of func"/> functors,
    <mmlq type="count" query="list of mode"/> modes,
    <mmlq type="count" query="list of pred"/> predicates,
    <mmlq type="count" query="list of sel"/> selectors, and
    <mmlq type="count" query="list of struct"/> structures
...
</ul>
```

The other important role in statistics plays an *ordering* part together with a *selection* part of queries. They allow to choose top n MML Query items according to an appropriate criterion and to render them in an appropriate order. For example, the statistic page "The most popular theorems" uses the following query

```
list of thdef ordered by number of in by ref reversed select 0-29
```

It shows top 30 (select 0-29) theorems and definitional theorems (thdef) and the choosing criterion is the number of items which refer to the theorem in their proofs. The relation `in by ref` is the inverse of the basic relation `by ref` which connect a MML Query item i with a theorem t if in the proof of i there is a reference to t. The most popular theorem is `TARSKI:def 1` which is definitional theorem of singleton set. It is referred in the proof by 3134 items.

Some statistics require a complicated ordering. For example, the page "The Longest Type Widening" requires the calculation of iteration degree for the following compound relation W:

```
[not = HIDDEN:mode 1
  | [ mothertype ref or prefix ref ]
    | [ mode or struct ]
]
```

The widening of MIZAR types is described in [7]. It is a transitive-reflexive closure of mother type and prefix relations. MML Query relations `mothertype ref` and `prefix ref` include mother type and prefix relations, respectively, but they concern also other constructors occurring in widening. Therefore, the relation W filters type constructors by `[ mode or struct ]` filter. The type `HIDDEN:mode 1` (*set*) is the widest type. Therefore, the relation W uses `not = HIDDEN:mode 1` relation as the type *set* is a mother type of itself. Summarizing, two types T_1 and T_2 are in the relation W if T_1 is different from the type *set* and T_2 occurs in the mother type or prefix of T_1 and T_2 is type constructor. The query resulting in top 20 types with longest widening is the following:

```
list of mode ordered by iteration of W reversed select 0-19
```

One of the longest widening starts from type `Leaf of T` where `T` is a `Tree`. It widens to `Element of Leaves T` which is `Element of T`. In further widening there occur a finite sequence of natural numbers, a partial function from the set $\mathbb{N}$ of natural numbers into $\mathbb{N}$, a subset of Cartesian product $\mathbb{N} \times \mathbb{N}$, and, finally, a set.

5. MML Query interfaces

The basic interface for MML Query is the program `mmlquery` written in the Perl language. It is a base for every other MML Query interface. It can be downloaded from MML Query home page together with zipped database files[11]. The full installation (see README file) requires 5.2GB of free disk space and is ready to use after unpacking and customizing a few variables. The program allows to call queries from the command line and to input them with or without support. The support requires Perl module `TermReadKey` which is a part of some Linux distributions[12] (e.g., Fedora Core 5) and is also available from CPAN Perl library[13].

To call a query from command line one has, for example, to type:

```
mmlquery --single --present=decode --query="list of th from CARD_2"
```

or simply

```
mmlq decode list of th from CARD_2
```

The first argument of `mmlq` determines a presentation style and others form a query. The program called in a terminal window for inputting queries with the command

[11] `http://mmlquery.mizar.org/mmlquery/downloads/`
[12] It may be installed with command: `yum install perl-TermReadKey`
[13] `http://www.perl.com/CPAN/`

```
mmlquery --input --present
```

shows the following message:

```
MML Query, version 1.4.01
Copyright (c) 2001-2007 Association of Mizar Users
Report bugs to Grzegorz Bancerek, bancerek@mizar.org
For help type \h or point your web browser to http://mmlquery.mizar.org/
Different presentation - see option \op present
Current presentation style = all_fields
Presentation styles: standard, all_fields, paths, decoded_exp, decoded,
                     mxa, html, emacs
mmlquery> _
```

The option `--input` turns on the input support of the interface. The input queries are interpreted and executed by the program and the result is presented with one of presentation styles. The simplest style `standard` gives only MML Query names. A more advanced style, `decoded`, gives reconstructed MIZAR formulae. The style `html` is used by the web interface[14] [10,4] and equips MML Query names with links for explanation and further browsing. The explanations and further browsing is rendered with MMLQT transformation. The style `emacs` is used by MIZAR mode for Emacs [21] when browsing GABs (Generated MIZAR abstracts, see [11]). GABs are produced with MMLQT transformation and offer integrated semantic browsing and demonstrate the information hidden in MIZAR articles. They offer also possibilities for querying with adequate constructors in provided interface.

MML Query interface is called in one of three modes: `--plain`, `--command`, and `--admin`. The plain mode is used only for querying. The command mode allows additionally

- to make abbreviations for MML Query items (e.g., `set Set = HIDDEN:mode 1`),
- to make abbreviations for MML Query relations and results,
- to set resource relations which are used in *non-expert queries* as default relations,
- to set explanation functions which are used in the result window of the web interface.

The command mode is set as default for the web interface. The interface in MIZAR mode for Emacs maintained by Josef Urban keeps also a running process of the `mmlquery` program in command mode and with `emacs` presentation style. The MIZAR mode takes over the results of queries and presents them integrated with other features.

[14] `http://mmlquery.mizar.org/mmlquery/three.html`

The admin mode is used to make indexing of all resources. It provides the means to change the MML Query database and to process additional MIZAR resources such as bibliographic files and translation patterns of the journal *Formalized Mathematics*. Translation patterns include English phrases or $\mathcal{AMS}$ TEX symbols for each MIZAR symbol which can be used in searching. E.g., the MIZAR symbol `VectSp` which corresponds to MML Query item `VECTSP_1:modenot 6` is translated in *Formalized Mathematics* as *vector space*. The last form is probably the best for beginners. (More about automatic translation in *Formalized Mathematics* could be found in [1])

6. Further work

Further work concerns the development of *non-expert queries* and the improvement of the functionality of MML Query interfaces. These tasks require better integration of MML Query with the MIZAR system. To strengthen the integration, MML Query tools must use the internal XML format of MIZAR articles which is newer than MML Queryś own format. Successful porting to MIZAR XML format requires an extension of this format used for the first stage of MIZAR text processing, since some information important for searching is lost with current implementation.

Another direction in further work concerns investigations on data mining and theorem prover techniques to extract from MML new basic relations which could improve searching.

References

[1] Bancerek, G.: Automatic translation in Formalized Mathematics, Mechanized Mathematics and Its Applications, 5(2): 19-31, 2006, MIZAR Japan.

[2] Bancerek, G.: Cardinal Arithmetics, *Formalized Mathematics*, 1(3):543–547, 1990.

[3] Bancerek, G.: Development of the theory of continuous lattices in MIZAR, in M. Kerber and M. Kohlhase (eds), *Symbolic Computation and Automated Reasoning*, 65-80, A. K. Peters, 2001.

[4] Bancerek, G.: Information retrieval and rendering with MML Query, in J.M. Borwein, W.M. Farmer (eds), Proceedings of MKM 2006, Wokingham, UK, *LNAI*, 4108: 266-279, 2006.

[5] Bancerek, G.: On Constructing Topological Spaces and Sorgenfrey Line, *Formalized Mathematics*, 13(1):171–179, 2005.

[6] Bancerek, G.: The Reflection Theorem, *Formalized Mathematics*, 1(5):963-972, 1990.

[7] Bancerek, G.: On the structure of Mizar types. Electronic Notes in Theoretical Computer Science, Vol. 85 (7), Elsevier, 2003.

[8] Bancerek, G., Endou, N., Shidama, Y.: Lim-inf Convergence and its Compactness. *Mechanized Mathematics and Its Applications*, 2(1): 29-35, 2002.

[9] Bancerek, G., Rudnicki, P.: A Compendium of Continuous Lattices in MIZAR, *Journal of Automated Reasoning*, 29(3-4): 189-224, 2002.

[10] Bancerek, G., Rudnicki, P.: Information retrieval in MML, in A. Asperti, B. Buchberger, J. H. Davenport (eds), Proceedings of MKM 2003, Bertinoro, *LNCS*, 2594: 119-131, 2003.

[11] Bancerek, G., Urban, J.: Integrated Semantic Browsing of the Mizar Mathematical Library for Authoring Mizar Articles, in Asperti, A., Bancerek, G., Trybulec, A. (eds), Proceedings of MKM 2004, Białowieża, *LNCS*, 3119: 44-57, 2004.

[12] Braselmann, P., Koepke, P.: Gödel's Completeness Theorem, *Formalized Mathematics*, 13(1):49-53, 2005.

[13] Dahn, I.: Management of Informal Mathematics Knowledge–Lessons Learned from the Trial-Solution Project, in Bai, F., Wegner, B., Electronic Information and Communication in Mathematics, LNCS 2730:29-43, 2003.

[14] Gierz, G., K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, D. S. Scott. *A Compendium of Continuous Lattices*, Springer-Verlag, Berlin, 1980.

[15] M. Kohlhase (joint work with Bancerek, G.). Towards MIZAR Mathematical Library in OMDOC format. Electronic Proceedings of the Workshop *Mathematics on the Semantic Web*, Eindhoven, 2003, `http://www.win.tue.nl/dw/monet/proceedings/kohlhase-mizarinomdoc.ps`

[16] Korniłowicz, A.: Jordan Curve Theorem, *Formalized Mathematics*, 13(4):481-491, 2005.

[17] R. Matuszewski, Rudnicki, P.: MIZAR: the first 30 years. *Mechanized Mathematics and Its Applications*, **4**(1):3–24, 2005.

[18] R. Milewski. Fundamental Theorem of Algebra, *Formalized Mathematics*, 9(3):461-470, 2001.

[19] Rudnicki, P., Ch. Schwarzweller, A. Trybulec. Commutative Algebra in the Mizar System. *Journal of Symbolic Computation*, **32**:143–169, 2001.

[20] Rudnicki, P., A. Trybulec. On equivalents of well-foundedness. *Journal of Automated Reasoning*, 23(3-4):197–234, 1999.

[21] J. Urban. MizarMode - an integrated proof assistance tool for the Mizar way of formalizing mathematics. Journal of Applied Logic, 2005, `doi:10.1016/j.jal.2005.10.004`

[22] J. Urban. MoMM - fast interreduction and retrieval in large libraries of formalized mathematics. International Journal on Artificial Intelligence Tools, 15(1): 109–130, 2006.
[23] J. Urban. XML-izing Mizar: making semantic processing and presentation of MML easy. in M. Kohlhase (ed), Proceedings of MKM 2005, Bremen, *LNCS*, 3863: 346-360, 2006.
[24] J. Urban, Bancerek, G.: Presenting and Explaining Mizar. in S. Autexier and C. Benzmüller (eds), Proceedings of UITP 2006, IJCAR'06 Workshop, Seattle, 97-108, 2006.
[25] Wiedijk, F.: *Mizar: An Impression*. http://www.cs.kun.nl/~freek/notes.

Partially supported by Białystok Technical University grant W/WI/1/06

EKSPLORACJA BIBLIOTEKI MIZARA Z UŻYCIEM MML QUERY

Streszczenie: MIZAR(komputerowy system weryfikacji dowodów) jest używany do budowania MIZAR Mathematical Library (MML). MML Query jest narzędziem opartym o semantykę tekstu mizarowego służącym do zarządzania wiedzą w systemie MIZAR włączając w to wyszukiwanie, przeglądanie i prezentację ewolującej zawartości MML. Narzędzie to stało się szeroko używane jako pomoc dla autorów mizarowych oraz odgrywa istotną rolę w nieustających reorganizacjach MML.
W artykule zaprezentowane są elementy systemu MIZAR i MML Query jak język, narzędzia logicznej weryfikacji i automatycznej publikacji, podstawy biblioteki MML, jej zawartość i konserwacja oraz problemy pojawiające się przy używaniu MML (odzyskiwanie informacji oraz jej przedstawianie). Ponadto, są przedstawione możliwości MML Query w rozwiązywaniu tych problemów.

Słowa kluczowe: MIZAR, wyszukiwanie semantyczne, repozytoria sformalizowanej matematyki

Artykuł zrealizowano w ramach pracy badawczej W/WI/1/06

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2007
Informatyka – Zeszyt 2

Leon Bobrowski[1,2]

SEPARABLE DATA AGGREGATION BY LAYERS OF ELEMENTARY CLASSIFIERS

Abstract: Data exploration or data mining goals can be reached by using variety of methods such as the fuzzy set theory or the rough sets theory. An interesting group of data exploration methods is based on minimization of convex and piecewise linear (*CPL*) criterion functions. This method originated from the theory of neural networks (multilayer *Perceptrons*). Powerful methods of data mining based on the support vector machines (*SVM*) can be also linked to this concept.

Hierarchical networks of formal neurons or multivariate decision trees can be induced from learning sets through minimization *CPL* criterion functions specified for classification problem. Another type of the *CPL* criterion functions can be used for designing visualizing data transformations. Separability of the transformed learning sets is a fundamental concept in the *CPL* approach to designing data mining tools.

Keywords: data transformations, data aggregation, separable data sets, elementary classifiers, convex and piecewise linear (*CPL*) criterion function

1. Introduction

Data exploration is aimed at discovering regularities (*patterns*) in data sets and at designing such data models which take into account these patterns. Many data exploration goals can be reached through the process of pattern recognition [1], [2], [3]. In this approach each object or event is represented as a feature vector or as a point in a multidimensional feature space. The pattern recognition process includes three basic stages: feature selection, feature extraction and classification. The primary goal of classification is proper allocation of each object or event into one of the classes (categories). This goal is often achieved by using variety of tools originating, among others, from case based reasoning techniques, neural networks models, decision trees approach. Feature selection stage is aimed at reducing dimensionality of the feature space through neglecting such features which are not

[1] Faculty of Computer Science, Bialystok Technical University
[2] Institute of Biocybernetics and Biomedical Engineering, PAS, Warsaw, Poland

important in the classification process. The dimensionality reduction can be achieved also during feature extraction stage in result of feature space transformations.

Many concepts in the theory and applications of artificial neural networks and pattern recognition has his beginning in the model of the Perceptron [4]. Hierarchical layers of formal neurons (multilayer perceptrons) still belong to the most fundamental models of neural networks [1]. Designing a neural network relates to the choice of a neural network structure (e.g. the number of layers and the number of elements in particular layers) and the weights of connections between elements of successive layers. The above designing tasks can be performed through minimization of the convex and piecewise linear (*CPL*) criterion functions deriving from the Perceptron model [4]. Linear separability of learning sets in a selected feature space is a big issue of the perceptron theory and plays a central role in applications the perceptron *CPL* criterion function. Similar *CPL* criterion functions can be used, for example, in designing decision trees, designing data transformation for feature extraction, feature selection, or data visualization. These topics are discussed more closely in the presented paper. Particular attention is paid to problems of designing separable layers of elementary classifiers.

2. Separable learning sets

Let us assume that each of the m analysed objects O_j ($j = 1,...,m$) is represented as the feature vector $\mathbf{x}_j = [x_{j1},...,x_{jn}]^T$ or as a point in the n -dimensional *feature space* $F[n]$ ($\mathbf{x}_j \in F[n]$). The components (*features*) x_{ij} of the vector $\mathbf{x}_j$ are supposed to be numerical results of a variety of examinations of the given object O_j . The feature vectors $\mathbf{x}_j$ can be of mixed, qualitative-quantitative type with n binary or real components x_{ij} ($x_{ij} \in \{0,1\}$ or $x_{ij} \in R$).

We assume that the database contains descriptions $\mathbf{x}_j(k)$ of m objects $O_j(k)$ ($j = 1,...,m$) labelled according to their *category* (*class*) ω_k ($k = 1,...,K$). The learning sets C_k can be created on this basis. One learning set C_k contains m_k feature vectors $\mathbf{x}_j(k)$ assigned to the k -th category ω_k

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

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

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

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

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

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

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

Definition 2: The learning sets (1) are *linearly separable* in the *n*-dimensional feature space $F[n]$ if each of these sets C_k can be fully separated by some hyperplane $H(\mathbf{w}_k, \theta_k)$ (3) from the sum $\cup C_i$ ($i \neq k$) of the remaining sets C_i:

$$\begin{aligned} (\exists k \in \{1, ..., K\})(\exists \mathbf{w}_k, \theta_k) \quad (\forall \mathbf{x}_j(k) \in C_k) \quad \mathbf{w}_k{}^T \mathbf{x}_j(k) > \theta_k \\ and \quad (\forall \mathbf{x}_{j'}(k') \in C_{k'}, k' \neq k) \quad \mathbf{w}_k{}^T \mathbf{x}_{j'}(k') < \theta_k \end{aligned} \tag{4}$$

In accordance with relation (4), all the vectors $\mathbf{x}_j(k)$ belonging to the learning set C_k are situated on the positive side ($\mathbf{w}_k{}^T \mathbf{x}_j(k) > \theta_k$) of the hyperplane $H(\mathbf{w}_k, \theta_k)$ (3) and all the feature vectors $\mathbf{x}_{j'}(k')$ from the remaining sets C_i are situated on the negative side ($\mathbf{w}_k{}^T \mathbf{x}_{j'}(k') < \theta_k$) of this hyperplane.

The separation of data sets C_k by the hyperplanes $H(\mathbf{w}_k, \theta_k)$ (3) can by linked to data transformation by a layer of K formal neurons $FN(\mathbf{w}_k, \theta_k)$. The formal neuron $FN(\mathbf{w}_k, \theta_k)$ is defined by the threshold decision rule $q(\mathbf{w}_k, \theta_k; \mathbf{x})$:

$$q = q(\mathbf{w}_k, \theta_k; \mathbf{x}) = \begin{array}{lll} 1 & if & \mathbf{w}_k{}^T \mathbf{x} \geq \theta_k \\ 0 & if & \mathbf{w}_k{}^T \mathbf{x} < \theta_k \end{array} \quad (5)$$

where q is the output, $\mathbf{w}_k = [w_{k1},...,w_{kn}]^T \in R^n$ is the weight vector, $\theta_k \in R^1$ is the threshold and $\mathbf{x} = [x_1,...,x_n]^T$ is the input feature vector.

The feature vector $\mathbf{x}$ activates ($r = 1$) the formal neuron $FN(\mathbf{w}_k, \theta_k)$ if and only if $\mathbf{x}$ is situated on the positive side of the hyperplane $H(\mathbf{w}_k, \theta_k)$ ($\mathbf{w}_k{}^T \mathbf{x} \geq \theta_k$).

Layer of K formal neurons $FN(\mathbf{w}_i, \theta_i)$ transforms the feature vectors $\mathbf{x}$ into the binary vectors $\mathbf{q} = \mathbf{q}(\mathbf{x})$, where $\mathbf{q} = [q_1,...,q_K]$, $q_i = q(\mathbf{w}_i, \theta_i; \mathbf{x})$ (5). Such a layer can be used as the classifier with the allocation rule given below

$$if \quad (q(\mathbf{w}_k, \theta_k; \mathbf{x}) = 1) \quad and \quad (\forall i \neq k) q(\mathbf{w}_i, \theta_i; \mathbf{x}) \quad then \quad (\mathbf{x} \in \omega_k) \quad (6)$$

A vector $\mathbf{x}$ is allocated to the class ω_k if only one neuron $FN(\mathbf{w}_k, \theta_k)$ in this layer is activated. We can remark that if the learning sets C_k (1) are linearly separable (4), then the layer of K formal neurons $FN(\mathbf{w}_i, \theta_i)$ with the rule (6) can allocate properly all the feature vectors $\mathbf{x}_j(k)$ (1).

3. Elementary classifiers

Let us take into account a layer of L *elementary classifiers* $Q_i = Q_i(\mathbf{v}_i)$ ($i = 1,...,L$) with the binary outputs q_i ($q_i \in \{0,1\}$). Each classifier Q_i is defined on the feature vectors $\mathbf{x}$ by an individual decision rule $q_i = q_i(\mathbf{v}_i; \mathbf{x})$:

$$q_i = q_i(\mathbf{v}_i; \mathbf{x}) \quad (i = 1,...,L) \quad (7)$$

where $\mathbf{v_i} = [v_{i1},...,v_{in'}]^T$ is n' - dimensional vector of parameters.

The classifier Q_i is activated by the feature vectors $\mathbf{x}$ if and only if $q_i(\mathbf{v}_i; \mathbf{x}) = 1$. Formal neurons $FN(\mathbf{w}_k, \theta_k)$ (5) can be used as the elementary classifiers Q_i.

Definition 3: The *activation field* S_i of the elementary classifier $Q_i = Q_i(\mathbf{v}_i)$ is defined as the set of such feature vectors $\mathbf{x}$, which activates ($q_i(\mathbf{v}_i;\mathbf{x}) = 1$) this classifier.

$$S_i = \{\mathbf{x} : q_i(\mathbf{v}_i;\mathbf{x}) = 1\} \tag{8}$$

The layer of L elementary classifiers Q_i transforms each feature vector $\mathbf{x}_j(k)$ from the sets C_k (1) into the vector $\mathbf{q}_j(k)$ with L binary components $q_i = q_i(\mathbf{v}_i;\mathbf{x}_j(k))$.

$$\mathbf{q}_j(k) = [q_1(\mathbf{v}_1;\mathbf{x}_j(k)),...,q_L(\mathbf{v}_L;\mathbf{x}_j(k))]^T \tag{9}$$

where $\mathbf{v_i} = [v_{i1},...,v_{in'}]^T$ is a vector of parameters.

Vectors $\mathbf{q}_j(k)$ form new data representation which can be useful in designing valuable decision rules for classification purpose [2]. The decision rules could be designed more efficiently on the basis of the transformed vectors $\mathbf{q}_j(k)$ than on the basis of the feature vectors $\mathbf{x}_j(k)$. An example of the decision rule is given by (6). The transformed vectors $\mathbf{q}_j(k)$ form the sets D_k:

$$D_k = \{\mathbf{q}_j(k)\} \quad (j \in I_K) \tag{10}$$

One of the fundamental goals in designing layers of elementary classifiers Q_i could be the separability (2) or the linear separability (4) of the transformed sets D_k. Additionally, we could demand the *separable aggregation* of the learning sets C_k (1).

Definition 4: The transformation (9) results in the *separable aggregation* of the learning sets C_k (1) if and only if the transformed sets D_k (10) are separable (2), each feature vector $\mathbf{x}_j(k)$ (1) activates at least one elementary classifiers Q_i (7) of the layer, and the number m' of different vectors $\mathbf{q}_j(k)$ (9) in the sets D_k is less than m.

Classification postulate I: The transformation (9) defined by the layer of L' elementary classifiers Q_i should result in the separable (2) sets D_k (10) with a low number m' of different vectors $\mathbf{q}_j(k)$ (9) and a low dimensionality L' of these vectors.

Few examples of the elementary classifiers Q_i are given below:

Example 1: Formal neurons $FN(\mathbf{w}_i,\theta_i)$ (5) can be treated as the elementary classifiers Q_i (8). In this case, the decision rule $q_i = q_i(\mathbf{v}_i;\mathbf{x}_j(k))$ (7) is based on the vector of parameters $\mathbf{v}_i$ given below:

$$\mathbf{v}_i = [\mathbf{w}_i^T, \theta_i]^T \tag{11}$$

The activation field S_i (8) of the formal neuron $FN(\mathbf{w}_i,\theta_i)$ (5) is the positive half-space defined by the hyperplane $H(\mathbf{w}_i,\theta_i)$ (3).

Example 2: The number n of inputs $\mathbf{x}_j$ to formal neuron $FN(\mathbf{w}_i,\theta_i)$ (5) can be reduced to one. Such reduced neuron will be called as *logical element* $LE(w_i,\theta_i)$. The elementary classifiers Q_i are determined in this case by the vector of parameters $\mathbf{v}_i = [w_i,\theta_i]^T$ (11) with only two components w_i and θ_i. The decision rule $q_i = q_i(\mathbf{v}_i;\mathbf{x}_j(k))$ (8) can be reduced to the below form:

$$\textit{if} \quad (w_i x_{ji}(k) \geq \theta_i) \quad \textit{then} \quad (q_i(\mathbf{v}_i;\mathbf{x}_j(k)) = 1) \quad \textit{else} \quad (q_i(\mathbf{v}_i;\mathbf{x}_j(k)) = 0) \tag{12}$$

The activation field S_i (8) of the logical element $LE(w_i,\theta_i)$ is the positive half-space defined by such hyperplanes $H(\mathbf{w}_k,\theta_k)H(\mathrm{w_k},\theta_\mathrm{k})$ (3), which are parallel to all but one axis of the feature space $F[n]$.

Example 3: The elementary classifier Q_i (7) can be based on the Euclidean ball $K_E(\mathbf{w}_i,\rho_i)$ centered in the point $\mathbf{w}_i$ and with the radius ρ_i in the feature space $F[n]$:

$$K_E(\mathbf{w}_i,\rho_i) = \{\mathbf{x} : (\mathbf{x}-\mathbf{w}_i)^T(\mathbf{x}-\mathbf{w}_i) \leq \rho_i\} \tag{13}$$

The vector of parameters $\mathbf{v}_i = [\mathbf{w}_i^T, \rho_i]^T$ defines the decision rule $q_i = q_i(\mathbf{v}_i;\mathbf{x}_j(k))$ (7) in the below manner

$$\begin{aligned} &\textit{if} \quad ((\mathbf{x}_j(k)-\mathbf{w}_i)^T(\mathbf{x}_j(k)-\mathbf{w}_i) \leq \rho_i) \quad \textit{then} \quad (q_i(\mathrm{v}_i;\mathbf{x}_j(k)) = 1) \\ &\qquad\qquad \textit{else} \quad (q_i(\mathrm{v}_i;\mathbf{x}_j(k)) = 0) \end{aligned} \tag{14}$$

The activation field S_i (8) of such elementary classifier Q_i (7) is the ball $K_E(\mathbf{w}_i, \rho_i)$ (13).

Example 4: The $L1$ ball $K_{L1}(\mathbf{w}_i, \rho_i)$ centered in the point $\mathbf{w}_i$ and with the radius ρ_i in the feature space $F[n]$ also can serve as an elementary classifier Q_i (8).

$$K_{L1}(\mathbf{w}_i, \rho_i) = \{\mathbf{x} : |x_1 - w_1| + ... + |x_n - w_n| \le \rho_i\} \tag{15}$$

The decision rule $q_i = q_i(\mathbf{v}_i; \mathbf{x}_j(k))$ (7) has now the following form:

$$\begin{aligned} &if \quad (|x_1 - w_1| + ... + |x_n - w_n| \le \rho_i) \quad then \quad (q_i(\mathrm{v}_i; \mathbf{x}_j(k)) = 1) \\ &\qquad\qquad else \quad (q_i(\mathrm{v}_i; \mathbf{x}_j(k)) = 0) \end{aligned} \tag{16}$$

The activation field S_i (8) of this elementary classifier Q_i (7) is the ball $K_{L1}(\mathbf{w}_i, \rho_i)$ (15).

Example 5: The $L1$ ball $K_{L1}(\mathbf{w}_i, \rho_i)$ (15) can be generalized to the ball $K_P(\mathbf{w}_i, \rho_i)$

$$K_{L1}(\mathbf{w}_i, \alpha_{i,}\rho_i) = \{\mathbf{x} : \alpha_{i1}|x_1 - w_1| + ... + \alpha_{in}|x_n - w_n| \le \rho_i\} \tag{17}$$

where $\alpha_i = [\alpha_{i1}, ..., \alpha_{in'}]^T$ is the vector of *features costs* α_{ik}.

The decision rule $q_i = q_i(\mathbf{v}_i; \mathbf{x}_j(k))$ (16) is generalised with the parameters $\mathrm{v}_i = [\mathbf{w}_i^T, \alpha_i^T, \rho_i]^T$ to:

$$\begin{aligned} &if \quad (\alpha_{i1}|x_{j1}(k) - w_1| + ... + \alpha_{in}|x_{jn}(k) - w_n| \le \rho_i) \\ &\quad then \quad (q_i(\mathbf{v}_i; \mathbf{x}_j(k)) = 1) \quad else \quad (q_i(\mathbf{v}_i; \mathbf{x}_j(k)) = 0) \end{aligned} \tag{18}$$

Let us remark that the number of parameters in the above rule has been increased to $(2n+1)$ in comparison to $(n+1)$ parameters used in the rule (16).

4. Dipolar strategy of separable layers designing

Lets us take into consideration the problem of designing separable layers of elementary classifiers Q_i $(i = 1, ..., L)$ (7). "Separable layer"is such a layer of L

elementary classifiers Q_i (7) which results in the separability (2) of the transformed sets D_k (10). The dipolar and the ranked strategies of designing separable layers of formal neurons were proposed earlier [6], [7]. Now, we will generalize these strategies to the layers of elementary classifiers Q_i (7). We will start with the description of the dipolar strategy. This strategy is based on the concept of clear and mixed dipoles [5].

Definition 5: A pair of different feature vectors $(\mathbf{x}_j(k), \mathbf{x}_{j'}(k'))$ $(\mathbf{x}_j(k) \neq \mathbf{x}_{j'}(k'))$ constitutes a *mixed dipole* if and only if these vectors belong to different classes ω_k $(k \neq k')$. Similarly, a pair of different feature vectors from the same class ω_k constitutes the *clear dipole* $(\mathbf{x}_j(k), \mathbf{x}_{j'}(k'))$.

Definition 6: The elementary classifier Q_i (7) *separates* (*divides*) the dipole $(\mathbf{x}_j(k), \mathbf{x}_{j'}(k'))$ if only one feature vector $\mathbf{x}_j(k)$ or $\mathbf{x}_{j'}(k')$ from this pair activates this element $(q_i(\mathbf{v}_i; \mathbf{x}_j(k)) = 1$ and $q_i(\mathbf{v}_i; \mathbf{x}_{j'}(k')) = 0$ or $q_i(\mathbf{v}_i; \mathbf{x}_j(k)) = 0$ and $q_i(\mathbf{v}_i; \mathbf{x}_{j'}(k')) = 1)$.

Lemma 1: The necessary and sufficient condition for the separability (*Def. 1*) of the sets D_k (10) transformed by the layer (9) is the separation of each mixed dipole $(\mathbf{x}_j(k), \mathbf{x}_{j'}(k'))$ by at least one elementary classifier Q_i (7) of the layer.

The proof of similar result for layer of formal neurons $FN(\mathbf{w}, \theta)$ (4) has been given in [5], [8]. In accordance with the *Lemma 1*, a layer which divides all mixed dipoles transforms separable sets C_k (1) into separable sets D_k (10). In order to preserve the chance for correct classification of all feature vectors $\mathbf{x}_j(k)$ (1), an additional postulate is introduced:

Classification postulate II: Each feature vector $\mathbf{x}_j(k)$ (1) should activate $(q_i(\mathbf{v}_i; \mathbf{x}_j(k)) = 1)$ at least one elementary classifier Q_i (7) of a given layer (9).

5. Ranked strategy of separable layers designing

Let us take into consideration the ranked strategy of designing separable layers of elementary classifiers Q_i $(i = 1, ..., L)$ (7). This strategy uses a fixed order between

elementary classifiers Q_i of the layer which is based on the indexing of these classifiers. The relation "*prior to*" is defined between any two elementary classifiers Q_l and Q_i of the layer on the base of the indices l and i in the below manner.

Definition 7: The classifier Q_i (7) is *prior to* the classifier Q_l if and only if $i < l$.

Definition 8: The l-th *ranked field* $R_l\,(l = 1,...,L)$ of the layer of L elementary Q_i classifiers is a set of such feature vectors $\mathbf{x}_j(k)$ (1) which activate the l-th classifier Q_l and do not activate any of the prior classifiers Q_i.

$$R_l = \{\mathbf{x}_j(k) : q_l(\mathrm{v}_i; \mathbf{x}_j(k)) = 1 \quad and \quad (\forall i < l) q_i(\mathrm{v}_i; \mathbf{x}_j(k)) = 0\} \tag{19}$$

Definition 9: The *r*anked field R_i (19) is *deterministically admissible* if and only if it contains feature vectors $\mathbf{x}_j(k)$ from only one learning set C_k (1).

Definition 10: The *r*anked field R_i (19) is *statistically admissible* at the level α ($0 < \alpha < 0.5$) if and only if it contains feature vectors $\mathbf{x}_j(k)$ not only from the dominant set C_k but also from other sets C_i (1) in a fraction f_i less than α ($f_i < \alpha$).

The fraction f_i of elements $\mathbf{x}_j(l)$ from non-dominant sets C_l is defined by the expression below:

$$f_i = \frac{m_i'(k)}{m_i(k) + m_i'(k)} \tag{20}$$

where $m_i(k)$ is the number of elements $\mathbf{x}_j(k)$ from the dominant set C_k in the ranked field R_i (19) and $m_i'(k)$ is the number of elements $\mathbf{x}_j(l)$ in this field from all non-dominant sets C_l (1) ($m_i(k) > m_i'(k)$).

The layer of L elementary classifiers Q_i (7) with admissible ranked fields R_i (19) will be called an admissible one (deterministically or statistically admissible). It can be seen that the number L of classifiers R_i in an admissible layer fulfills below condition.

$$K \le L \le m \tag{21}$$

where K is the number of the learning sets C_k (1), and m is the number of feature vectors $\mathbf{x}_j(k)$ in these sets.

The lowest possible number $L = K$ appears when the ranked fields R_i (19) are extremely large and contains whole learning sets C_k ($\forall k \in \{1,...,K\}$ $R_k = C_k$ (1). The highest possible number $l = m$ appears when each ranked field R_i (19) contains only one feature vector $\mathbf{x}_j(l)$. It can be expected that layer of classifiers Q_i with large ranked fields R_i (19) should have greater generalizing power than the layer with small active fields.

Definition 11: The layer of elementary classifiers Q_i (7) with deterministically admissible (*Def. 7*) ranked fields R_i (19) forms the *ranked layer* if and only if each feature vector $\mathbf{x}_j(k)$ from the sets C_k (1) belongs to one of this fields.

The ranked layer of L elementary classifiers Q_i transforms each feature vector $\mathbf{x}_j(k)$ into the vector $\mathbf{q}_j(k)$ (9) with L binary components $q_i = q_i(\mathbf{v}_i; \mathbf{x}_j(k))$. The separability (2) of the sets C_k (1) is preserved during the transformation by the ranked layer as it is proven below.

Lemma 2: If the sets C_k (1) are separable (2), then the sets D_k (10) at the output of the ranked layers are also separable.

Proof: The sufficient condition for the sets D_k (10) separability has the form (2).

$$(k \neq k') \Rightarrow (\forall j \in I_k) \quad and \quad (\forall j' \in I_{k'}) \mathbf{q}_j(k) \neq \mathbf{q}_{j'}(k') \tag{22}$$

The above condition results directly from the definition of the ranked fields R_i (19). Two vectors $\mathbf{q}_j(k)$ and $\mathbf{q}_{j'}(k')$ related to the ranked fields R_j and $R_{j'}$ are linked to different classes ω_k and $\omega_{k'}$. So, these vectors cannot be equal ($\mathbf{q}_j(k) \neq \mathbf{q}_{j'}(k')$).

Theorem 1: The ranked layer (*Def. 9*) of L elementary classifiers Q_i with the decision rules $q_i(\mathbf{v}_i; \mathbf{x})$ (7) transforms the separable sets C_k (1) into linearly separable (4) sets D_k (10).

Proof: Let us assign the following parameter α_i to each ranked field R_i (19).

$$(\forall i \in \{1,...,L\}) \quad \alpha_i = \frac{1}{2^i} \tag{23}$$

The hyperplane $H(\mathbf{z}_k, \theta_k)$ (3) used for separation of the set D_k (10) from the sum $\cup D_i$ of the remaining sets D_i ($i \neq k$) can be defined by the weight vector $\mathbf{z}_k = [z_{k1},..., z_{kL}]^T$ with the following components z_{ki}

$$\begin{aligned}(\forall i \in \{1,...,L\}) \quad & \textit{if} \quad R_i \in C_k \quad \textit{then} \quad z_{ki} = \alpha_i \\ & \textit{and} \quad \textit{if} \quad R_i \notin C_k \quad \textit{then} \quad z_{ki} = -\alpha_i\end{aligned} \tag{24}$$

By direct computations we can verify the inequalities below.

$$\begin{aligned}(\exists k \in \{1,...,K\})(\forall \mathbf{q}_j(k) \in D_k) \quad & \mathbf{z}_k{}^T \mathbf{q}_j(k) > 0 \\ \textit{and} \quad (\forall \mathbf{q}_j(k) \in D_k, i \neq k) \quad & \mathbf{z}_k{}^T \mathbf{q}_j(i) < 0\end{aligned} \tag{25}$$

where $\mathbf{z}_k$ is the weight vector with the components z_{ki} (24). The inequalities (25) mean that the sets D_k (10) are linearly separable (4).

The considerations above are similar to the proof given in the paper [7]. The notions used in *Theorem 1* are illustrated by the below Figure.

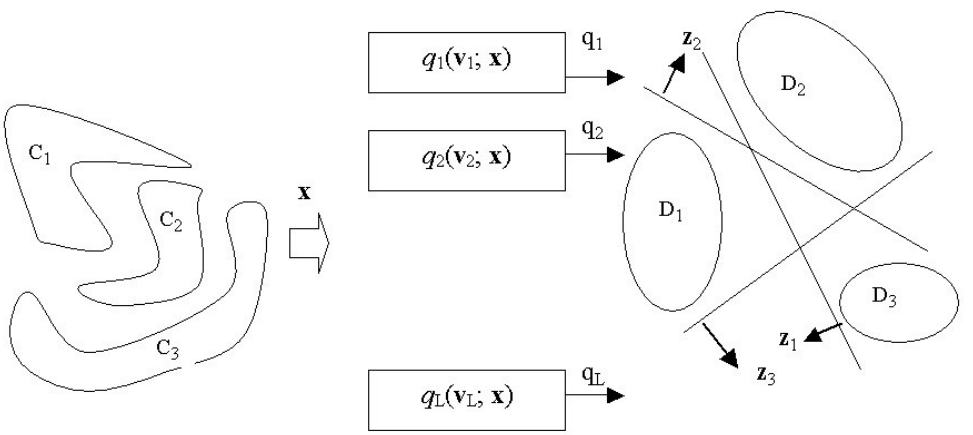


Fig. 1. Linearization (25) of three data sets C_k by the ranked layer of L elementary classifiers Q_i (7)

Linearization (26) of data sets C_k (1) by the ranked layer of L elementary classifiers Q_i has such consequence, that the second layer of K formal neurons $FN(\mathbf{w}_k, \theta_k)$ defined on the vectors $\mathbf{q}_j(k)$ (9) can separate exactly the sets D_k(10). As a consequence, each feature vector $\mathbf{x}_j(k)$ (1) can be correctly classified by the hierarchical network formed by such two layers.

6. Optimization of separable layers

A layer of L elementary classifiers Q_i (7) transforms each feature vector $\mathbf{x}_j(k)$ (1) into the *output vector* $\mathbf{q}_l = [q_1,...,q_L]^T$ (9) with L binary components q_i ($q_i \in \{0,1\}$). Many feature vectors $\mathbf{x}_j(k)$ can be transformed into the same vector $\mathbf{q}_l$ ($(l \neq l') \Rightarrow \mathbf{q}_l(k) \neq \mathbf{q}_{l'}(k')$) in this manner. The set of such feature vectors $\mathbf{x}_j(k)$ is called as the l-th *activation field* S_l of the layer of elementary classifiers.

$$S_l = \{\mathbf{x}_j(k) : [q_1(\mathbf{v}_1, \mathbf{x}_j(k)),..., q_L(\mathbf{v}_L, \mathbf{x}_j(k))]^T = \mathbf{q}_l\} \tag{26}$$

where $(\forall l \neq l')$ $\mathbf{q}_l(k) \neq \mathbf{q}_{l'}(k)$.

Definition 12: The set S_l (26) will be called the *clear activation field* if all feature vectors $\mathbf{x}_j(k)$ (1) from this set ($\mathbf{x}_j(k) \in S_l$) belong to the same class ω_k. Similarly, the set S_l is the *mixed activation field* if it contains feature vectors $\mathbf{x}_j(k)$ (1) from different classes ω_k.

The field $S_l(k)$ and the output vector $\mathbf{q}_l(k)$ will be linked to the k-th class ω_k if and only if the most of the labeled feature vectors $\mathbf{x}_j(k)$ from the set S_l (26) is labeled to the class ω_k.

All feature vectors $\mathbf{x}_j(k)$ from the l-th activation field S_l are *aggregated* by the layer of elementary classifiers into one vector $\mathbf{q}_l$. In other words, the vector $\mathbf{q}_l$ *generalizes* all feature vectors $\mathbf{x}_j(k)$ from the field S_l (26). It can be expected that the layer of elementary classifiers with large and clear activation fields S_l (26)

will have a great *generalization power*. Such layer could be used also as a classifier with the following decision rule

$$if \quad \mathbf{x}_0 \in S_l(k) \quad then \quad \mathbf{x}_0 \in \omega_k \tag{27}$$

where $S_l(k)$ is such activation field (26) that most of the labeled feature vectors (1) from this field belong to the class ω_k.

A quality of the decision rule can be evaluated by the *error rate er* [9]. The classification error rate *er* is often evaluated as

$$\hat{er} = \frac{m_e}{m} \tag{28}$$

where m_e is the number of such feature vector $\mathbf{x}_j(k)$ from the sets C_k (1) which are wrongly allocated by the decision rule (27). The error rate evaluation (28) is positively biased (*optimistic bias*). The unbiased error rate *er* evaluations are based on such technique as cross-validation or on using testing sets [1].

Optimization problem I: To design such a layer of L of elementary classifiers Q_i (7) which will produce the decision rule (27) with the minimal *error rate er*.

Definition 13: A layer L of elementary classifiers Q_i (7) will be called *separable* if each feature vector $\mathbf{x}_j(k)$ from the sets C_k (1) belongs to some clear activation field S_l (26).

Optimization problem II: To design a separable layer of L of elementary classifiers Q_i (7) with minimal number L' of activation fields $S_l(k)$ or the output vectors $\mathbf{q}_l$ (26).

The minimal number L' of the activation fields S_l (26) can not be less than the number K of the classes ω_k ($L' \geq K$).

One can see, that a separable layer of elementary classifiers Q_i (7) with the decision rule (27) allocates correctly all feature vectors $\mathbf{x}_j(k)$ from the learning sets C_k (1). In this case, the estimator (28) of the error rate *er* is equal to zero. The classifiers which have error rate evaluation (28) on the sets C_k (1) equal to zero are often overfittning to these sets. As a consequence, such classifier can have a low generalisation power and the classification of new objects $\mathbf{x}$ might often be

wrong. So, the classification (27) based only on the clear activation fields S_l (26) could be far from optimal. In order to improve the classification rule (26), the clear activation fields S_l (26) should be replaced by such fields S_l (26) which can be "slightly" mixed.

Definition 14: A layer of L elementary classifiers Q_i (7) will be called the ε - s*eparable,* if and only if the ratio m_e / m (28) of the wrongly classified feature vectors $\mathbf{x}_j(k)$ by the rule is no greater than ε ($m_e / m \le \varepsilon$), where ε is a positive parameter ($\varepsilon > 0$).

Optimization problem III: Design a ε-separable layer of L of elementary classifiers Q_i (7) with minimal number L' of activation fields $S_l(k)$ (26).

A separable layer of L elementary classifiers Q_i (*Def. 9*) can serve also in data aggregation. Let us define the *aggregation coefficient* η_a f such layer a in the following manner

$$\eta_a = \frac{m - m'}{m - K} \tag{29}$$

where m is the number of the feature vectors $\mathbf{x}_j(k)$ from the sets C_k (1), m' is the number of different output vectors $\mathbf{q}_l$ (26) from a separable layer, and K is the number of the classes ω_k or the learning sets C_k (1).

The minimal number m' of the output vectors $\mathbf{q}_l$ (26) from a separable layer is equal to K ($m' = K$). The aggregation coefficient η_a (29) takes the maximal value equal to one ($\eta_a = 1$) in this ideal situation. The aggregation coefficient η_a (29) of a layer of formal neurons $FN(\mathbf{w}_i, \theta_i)$ (5) can take the maximal value $\eta_a = 1$ if and only if the learning sets C_k (1) are linearly separable. The maximal value of the number m' is equal to m. There is no aggregation in this case and the aggregation coefficient η_a (29) takes the minimal value equal to 0 ($\eta_a = 0$). As a result.

$$0 \le \eta_a \le 1 \tag{30}$$

It can be noted that a solution of the *Optimization problem II* leads to the maximisation of the aggregation coefficient η_a (29).

In some cases, the above optimization problems can be solved through minimisation of the convex and piecewise linear (*CPL*) criterion functions [7]. We will pay particular attention to the perceptron criterion function (*CPL*). This function is linked to the beginning of the theory of neural networks.

7. Convex and piecewise linear criterion function (CPL)

Let us consider designing a separable layer of the formal neurons $FN(\mathbf{w}_i,\theta_i)$ (5) or the logical elements $LE(\mathbf{w}_i,\theta_i)$ (12). In this case, the designing procedure can be based on a sequence of minimisation of the convex and piecewise linear (*CPL*) criterion functions $\Psi_l(\mathbf{w},\theta)$ ([3], [4]). The perceptron criterion function $\Psi_l(\mathbf{w},\theta)$ belongs to the *CPL* family. It is easy to define the functions $\Psi_l(\mathbf{w},\theta)$ by using the positive G_l^+ and the negative G_l^- sets of the feature vectors $\mathbf{x}_j=[x_{j1},...,x_{jn}]^T$ (1).

$$G_l^+=\{\mathbf{x}_j\} \quad j\in J_l^+ \quad and \quad G_l^-=\{\mathbf{x}_j\} \quad j\in J_l^- \tag{31}$$

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

$$\varphi_j^+(\mathbf{w},\theta)=\begin{matrix}1-\mathbf{w}^T\mathbf{x}_j+\theta & if \quad \mathbf{w}^T\mathbf{x}_j-\theta\le 1\\ 0 & if \quad \mathbf{w}^T\mathbf{x}_j-\theta>1\end{matrix} \tag{32}$$

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

$$\varphi_j^-(\mathbf{w},\theta)=\begin{matrix}1+\mathbf{w}^T\mathbf{x}_j-\theta & if \quad \mathbf{w}^T\mathbf{x}_j-\theta\ge -1\\ 0 & if \quad \mathbf{w}^T\mathbf{x}_j-\theta<-1\end{matrix} \tag{33}$$

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

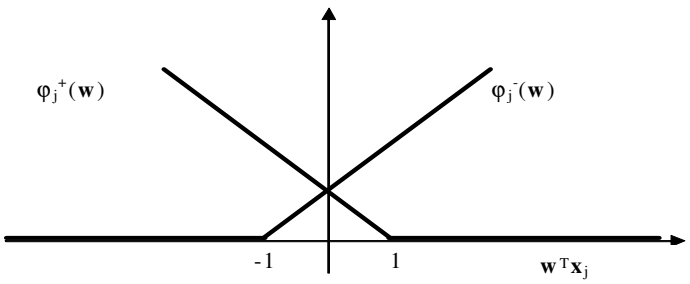


Fig. 2. The penalty functions $\varphi_j^+(w)$ (32) and $\varphi_j^-(w)$ (33).

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

$$\Psi_l(\mathbf{w},\theta) = \sum_{j\in J_l^+} \alpha_j^+ \varphi_j^+(\mathbf{w},\theta) + \sum_{j\in J_l^-} \alpha_j^- \varphi_j^-(\mathbf{w},\theta) \tag{34}$$

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

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

$$\Psi_l^* = \Psi_l(\mathbf{w}_l^*,\theta_l^*) = \min \Psi_l(\mathbf{w},\theta) > 0 \tag{35}$$

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

It has been proved that the minimal value Ψ_l^* of the peceptron criterion function $\Psi_l(\mathbf{w},\theta)$ (32) is equal to zero ($\Psi_l^* = 0$) if and only if the positive G_l^+ and the negative G_l^- sets (29) are linearly separable (4). In this case, all elements $\mathbf{x}_j$ of the set G_l^+ (29) are located on the positive side of the hyperplane $H(\mathbf{w}_l^*,\theta_l^*)$ (3) and all elements $\mathbf{x}_j$ of the set G_l^- are located on the negative side:

$$\begin{aligned} &(\forall \mathbf{x}_j \in G_l^+) \quad (\mathbf{w}_l^*)^T \mathbf{x}_j > \theta_l^* \\ \textit{and} \quad &(\forall \mathbf{x}_{j'} \in G_l^-) \quad (\mathbf{w}_l^*)^T \mathbf{x}_{j'} < \theta_l^* \end{aligned} \tag{36}$$

If the sets G_l^+ and G_l^- (22) are not linearly separable (4), then $\Psi_l^* > 0$ and the inequalities (34) are fulfilled only partly, not by all, but by the majority of the elements $\mathbf{x}_j$ of the sets (22).

Minimization of the function $\Psi_l(\mathbf{w},\theta)$ (32) allows one to find optimal parameters $(\mathbf{w}_l^*,\theta_l^*)$ which define such hyperplane $H(\mathbf{w}_l^*,\theta_l^*)$ (3), which separates relatively well two sets G_l^+ and G_l^- (22). The parameters $(\mathbf{w}_l^*,\theta_l^*)$ can be also used in defining the l-th element $FN(\mathbf{w}_l^*,\theta_l^*)$ (5) of a neural layer.

The perceptron criterion function $\Psi_l(\mathbf{w},\theta)$ (32) can be used in designing separable layers of formal neurons $FN(\mathbf{w}_l,\theta_l)$ (5) both in accordance with the dipolar strategy described in Paragraph 4 as well as in accordance with the ranked strategy described in Paragraph 5. Specification of the criterion function $\Psi_l(\mathbf{w},\theta)$ (32) to particular strategy is achieved through an adequate choice of the sets G_l^+ and G_l^- (22) and the prices α_j^+ or α_j^- of the feature vectors $\mathbf{x}_j(k)$.

Designing separable layers of the formal neurons $FN(\mathbf{w}_l,\theta_l)$ (5) or the logical elements $LE(\mathbf{w}_i,\theta_i)$ (12) can be done in a sequential manner. During the l-th stage the l-th element $FN(\mathbf{w}_l^*,\theta_l^*)$ (5) or $LE(\mathbf{w}_i,\theta_i)$ (12) of the layer is designed through minimization of the criterion function $\Psi_l(\mathbf{w},\theta)$ (32).

Both the dipolar and the ranked strategy of separable layer designing can be optimised in accordance with the postulates described in Paragraph 6. In order to obtain a layer with large activation fields S_l (26) (*Optimization problem II*) the following postulate has been formulated in the framework of the sequential dipolar strategy:

"… First neuron should be designed in such a manner that its hyperplane divides the greatest number possible of mixed dipoles and a possibly low number of the clear dipoles. Second neuron should divide the greatest number possible of mixed dipoles undivided by the first neuron, and so on. The procedure is stopped after all mixed dipoles are divided…." [5]

Similar goal in the framework of the ranked strategy is realized through the postulate of large *r*anked fields R_i (19). These postulates are aimed at achieving a separable layer with a large generalization power. Such layer should allow for considerable data aggregation (29) or for classification rules (27) with a low error rate.

8. Concluding remarks

Designing separable layers from different types of elementary classifiers Q_i (7) was discussed in this paper. The dipolar and the ranked strategy of separable layers designing was described. The dipolar strategy allows for preserving separability (2) of the learning sets C_k (1) by the design layer of elementary classifiers Q_i Ranked layers have a fundamental property of linearization of learning sets. This means that the separable data sets C_k (1) are transformed by the ranked layer into linearly separable (4) sets D_k (12). A simplified representation of a classification problem can be reached as a result of such transformation. Linearization of data sets by the ranked layers could find important applications also in the methods originating from Support Vector Machines (*SVM*) [3]. Both the dipolar, as well as the ranked layers, can be used as a tool for separable data aggregation.

The deterministic version of the dipolar and the ranked strategies was discussed in this paper. The deterministic approach has a constraint in the form of data overfitting. It can be expected that the statistical approach towards designing ranked layers (e.g. *Def.* 8) combined with feature selection techniques will increase the chance of obtaining accurate classifiers with a large discriminative power.

The dipolar and the ranked strategies of designing separable layers of the formal neurons $FN(\mathbf{w}_l,\theta_l)$ (5) or the logical elements $LE(\mathbf{w}_l,\theta_l)$ (12) can be done in a sequential manner. The optimisation of the parameters $(\mathbf{w}_l,\theta_l)$ (5) or $(\mathbf{w}_l,\theta_l)$ (12) during the l-th stage of designing can be done through minimisation of the convex and piecewise linear (*CPL*) criterion functions $\Psi_l(\mathbf{w},\theta)$ (32). The basis exchange algorithms which are similar to linear programming allow to find the minimum of the criterion functions $\Psi_l(\mathbf{w},\theta)$ [5]. Designing separable layers from such elementary classifiers Q_i (7) which are based on the Euclidean balls $K_E(\mathbf{w}_i,\rho_i)$ (13) demand other types of algorithms. For example, the genetic algorithms can be used in the designing process.

References

[1] Duda, O.R., Heart, P.E., Stork D.G.: *Pattern Classification*, Wydanie drugie, zmienione, John Wiley & Sons, 2001.

[2] Fukunaga, K.: *Introduction to Statistical Pattern Recognition*, Academic Press 1990.

[3] Vapnik, V.N.: *Statistical Learning Theory*, J. Wiley, New York 1998.
[4] Rosenblatt, F.: *Principles of neurodynamics*, Spartan Books, Washington 1962.
[5] Bobrowski, L.: *Piecewise-Linear Classifiers, Formal Neurons and separability of the Learning Sets*, Proceedings of ICPR'96, pp. 224-228, (13th International Conference on Pattern Recognition", August 25-29, 1996, Wienna, Austria).
[6] Bobrowski, L,: *Design of piecewise linear classifiers from formal neurons by some basis exchange technique*, Pattern Recognition, 24(9), pp. 863-870, 1991.
[7] Bobrowski, L.: *The ranked neuronal networks*, Biocybernetics and Biomedical Engineering, Vol. 12, No. 1-4, pp. 61-75, 1992.
[8] Bobrowisk, L.: *Eksploracja danych oparta na wypukłych i odcinkowo-liniowych funkcjach kryterialnych*, Politechnik Białostocka, 2005.

SEPAROWALNA AGREGACJA DANYCH W WARSTWACH KLASYFIKATORÓW ELEMENTARNYCH

Streszczenie: Cele eksploracji danych mogą być osiągnięte przy użyciu różnorodnych metod, takich jak teoria zbiorów rozmytych lub teoria zbiorów przybliżonych. Interesująca grupa metod eksploracji danych bazuje na minimalizacji wypukłych i odcinkowo-liniowych (CPL) funkcji kryterialnych. Metody te wywodzą się z teorii sieci neuropodobnych (wielowarstwowy perceptron). Do tej grupy mogą być także zaliczone silne obliczeniowo metody eksploracji danych bazujące na maszynach wektorów podpierających (SVM).
Hierarchiczne sieci neuronów formalnych lub wielowymiarowe drzewa decyzyjne mogą być zbudowane na podstawie zbiorów uczących poprzez minimalizację funkcji kryterialnych typu CPL dostosowanych do problemu klasyfikacji. Inny typ funkcji kryterialnych CPL może być użyty do projektowania wizualizacyjnych transformacji danych. Podstawą w omawianym podejściu CPL do projektowania narzędzi eksploracji danych jest separowalność transformowanych zbiorów uczących.

Słowa kluczowe: transformacje danych, agregacja danych, separowalne zbiory danych, klasyfikatory elementarne, wypukła i odcinkowo-liniowa (CPL) funkcja kryterialna

Artykuł zrealizowano w ramach projektu badawczego „Temporalna reprezentacja wiedzy i jej implementacja w informatycznych systemach wspomagania postępowania medycznego", nr 3 T11F 011 30.

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2007
Informatyka – Zeszyt 2

Andrzej Chmielewski[1], Sławomir T. Wierzchoń[2]

ON THE DISTANCE NORMS FOR DETECTING ANOMALIES IN MULTIDIMENSIONAL DATASETS

Abstract: One of the key parameters of algorithms for anomaly detection is the metric (norm) applied to calculate the distance between every two samples which reflect its proximity. It is especially important when we operate on real-valued high dimensional datasets, i.e. when we deal with the problem of intruders detection in computer networks. As observed, the most popular Euclidean norm becomes meaningless in higher than 15-dimensional space. This means that other norms should be investigated to improve the effectiveness of real-valued negative selection algorithms. In this paper we present results for the following norms: Minkowski, fractional distance and cosine.

Keywords: negative selection, anomaly detection, Minkowski norm, fractional distance metric

1. Introduction

In recent years, the problem of calculating distance between two vectors (samples) in highly dimensional space was studied in great detail, for example in [14]. Nearest neighbor search, clustering and indexing are the examples of applications (issues) where behavior of some norms in high dimensions is the major obstacle to implement effective algorithms, and choice of the distance metric is not obvious. As it was shown in [1], some metrics, like i.e. Euclidean norm, loose its meaningfulness of proximity with increasing dimensionality. Thus, they can not be applied to highly dimensional datasets.

This problem was also observed, e.g. in [3] and [11], during the process of anomaly detection in datasets containing descriptions of network connections (e.g. the well-known KDD Cup 1999 dataset with 41 attributes [8]). Even after reduction of dimension for selected subsets to about 20, the average efficiency

[1] Faculty of Computer Science, Technical University of Bialystok, Poland
[2] Institute of Computer Science, Polish Academy of Science, Warsaw, Poland

of applied negative selection V-Detector algorithm never exceed 70%. We showed in [3]-[5] that after some modifications, this algorithm can produce quite good results, comparable with results generated by Support Vector Machine - a very strong classification tool. However, we are convincing that the choice of appropriate metric, especially for high dimensional datasets, is crucial. Therefore, in this paper, we present some experiments with selected norms which, should make possible to gain better results, in comparison to Euclidean distance.

2. Selected distance norms and its behavior in high dimensional spaces

In this section we present metrics (norms) used in the experiments described in Section 5. Below, there are good-known expressions to calculate the distance between two points: $\mathrm{x}=[x_1, x_2, \ldots, x_n]$ and $\mathrm{y}=[y_1, y_2, \ldots, y_n]$ in the space $\Re^n$. We also discuss their behavior in high dimensional space.

2.1. Minkowski norm and fractional distance metric

Minkowski norm of order m (L_m-norm distance) in space $\Re^n$ is defined as:

$$L_m(\mathrm{x},\mathrm{y})=\left(\sum_{i=1}^{n}\left|x_i-y_i\right|^m\right)^{\frac{1}{m}} \tag{1}$$

This is the generalized metric distance for $m \geq 1$. When $m=1$, it becomes Manhattan distance and when $m=2$, it becomes Euclidean distance.

Based on Minkowski norm, Aggarwal et al. [1] introduced *fractional distance metric* with $m<1$ which is more appropriate in high dimensional spaces. They proved that for the uniform distribution of k points

$$\left(\frac{C}{(m+1)^{1/m}}\right)\cdot\sqrt{\frac{1}{2\cdot m+1}} \leq \lim_{n\to\infty} E\left[\frac{Dmax_n^m - Dmim_n^m}{n^{1/m-1/2}}\right] \leq \left(\frac{C\cdot(k-1)}{(m+1)^{1/m}}\right)\cdot\sqrt{\frac{1}{2\cdot m+1}} \tag{2}$$

where $E[X]$ is expected value of the random variable X, $Dmax_n^m$ and $Dmin_n^m$ are farthest and nearest distance to the origin (measured by the distance metric L_m), and C is some constant. Eqn. (2) means that in a high-dimensional space

the absolute difference between $Dmax_n^m$ and $Dmin_n^m$ increases at the rate of $n^{1/m-1/2}$ when m decrease, independently of data distribution. Thus, *fractional distance metric* should provide better contrast between farthest and nearest neighbor than Manhattan and Euclidean norms. Moreover, the Euclidean norm should not be used for $n>5$ and it completely loose its meaningfulness of proximity distance for $n>15$ (distance is always very close to 0) – see e.g. [2] for a deeper discussion.

The similar proof, but only for Euclidean distance, was presented by Stibor in [10] to explain the very poor results obtained for the V-Detector algorithm (see Section 4) for the already mentioned KDD Cup 1999 dataset.

2.2. Cosine norm

Cosine norm for non-zero vectors x and y is defined as

$$D_{COS}(\mathrm{x},\mathrm{y}) = 1 - \frac{\langle \mathrm{x},\mathrm{y} \rangle}{\|\mathrm{x}\|\|\mathrm{y}\|} \tag{3}$$

where $\langle \mathrm{x},\mathrm{y} \rangle$ is the inner product of vectors x and y.

This norm is a popular distance measure for comparing (classifying) documents in the information retrieval applications. Thus, it seems to be good metric even for high dimensional spaces – see e.g. [2].

3. Negative selection

One of the major algorithms developed within emerging field of artificial immune systems (AIS) is Negative Selection Algorithm, proposed by Forrest et al. in [6]. It is based on the principles of self/nonself discrimination in the immune system. More formally, let U stands for the problem space, e.g. a set of all possible bit strings of fixed length, and S stands for the set of strings representing typical behavior. Then the set of strings characterizing anomalous behavior, N can be viewed as the set-theoretical complement of S:

$$N = U \setminus S \tag{4}$$

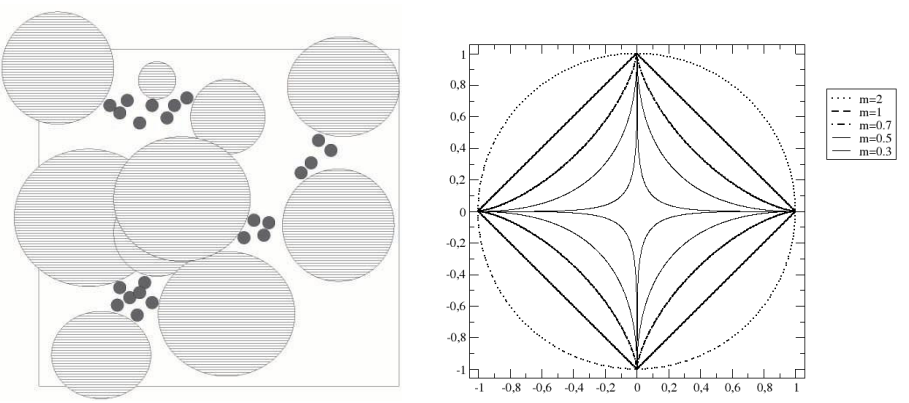
The elements of S are called self, and those of N are termed as non-self.

To apply the negative selection algorithm it is necessary to generate a set $D \subset N$ of detectors, such that each $d \in D$ recognizes at least one element $n \in N$,

and does not recognize any self element. Thus, we must designate a rule, $match(d,u)$, specifying when d recognizes an element u, consult [12] for details. This approach, although intuitive and simple, admits at least two serious drawbacks. First, it is hard to specify the full set S; typically we observe only a subset $S' \subset S$. Second, majority of detection rules induce so-called holes, i.e. regions of N which are not covered by any detector.

Instead of the binary representation of the space U we can use real-valued representation, originally proposed in [7]. This paper is focused only on real-valued detectors, described in Section 4.

4. V-Detector algorithm



(a) (b)

Fig. 1. (a) Examples of performance V-Detector algorithm in 2D. Self samples and detectors are represented as circles. Grey circles denotes self samples, dashed circles denotes v-detectors. (b) Unit spheres for selected L_m norms in 2D.

The V-Detector algorithm was formally proposed by Ji and Dasgupta [9]. It operates on (normalized) vectors of real-values attributes being points in the *n*-dimensional unit hypercube, $U = [0,1]^n$. Each self sample, $s_i \in S$, is represented as a hypersphere with the center at $c_i \in U$ and constant radius r_s, i.e. $s_i = (c_i, r_s)$, $i = 1,\ldots,|S|$, where $|S|$ is the number of self samples. Every point $u \in U$ belonging to any hypersphere is considered as a self element. Also, detectors d_j are represented as hyperspheres: $d_j = (c_j, r_j)$, $i = 1,\ldots,|D|$ where $|D|$ is a number

of detectors. In contrast to self elements, the radius r_j is not fixed but is computed as the Euclidean distance from a randomly chosen center c_j to the nearest self element (this distance must be greater than r_s, otherwise detector is nor created). Formally, we define r_j as

$$r_j = \max\left\{0, \min_{1 \le i \le l} dist(c_j, c_i) - r_s\right\} \tag{5}$$

```
Input:    S = set of self samples
          r_s = self radius
          T_MAX = max. number of V-detectors
          co = estimated coverage
Output:   D = set of generated V-detectors

begin
   D ← ∅
   repeat
      find ← false
      t ← 0
         repeat
            x ← random point from [0,1]^n
            foreach d_i∈D do
                  //calculate the distance between c_di (center of detector d_i) and x

                  //if distance is lesser than r_di (radius of detector d_i)
                  if dist(c_di, x) ≤ r_di then
                     // point x is covered by detector
                     t ← t+1

                     //check, if the estimated coverage (co) was achieved
                     if t ≥ 1/(1-co) then
                           return D;
                  else
                     //point x is not covered by detector
                     find ← true;
                     break;
                  endif
            endfor
         until find = true

         //now, x is the candidate for detector

         //calculate the distance to the nearest self sample (r_d)
         r_d ← ∞
         foreach s_i∈S do
            l ← dist(c_si, x)
            if l - r_s < r_d then
                  r_d = l - r_s
            endif
         endfor

         //radius of detector (r_d) should be equal or grater then r_s
         if r_d ≥ r_s then
            //add new detector d to set D
            D ← D ∪ {d=(x,r_d)}
         endif

   until |D|=T_MAX
end
```

Fig.2. Pseudocode of V-Detector algorithm.

The algorithm terminates if predefined number T_{max} of detectors is generated or the space $U \setminus S$ is sufficiently well covered by these detectors (parameter co). The pseudocode of V-Detector algorithm is presented in Fig. 2.

As mentioned above, in original version, V-Detector utilizes Euclidean distance to calculate similarity between samples, thus both: v-detectors and self samples are represented as hyperspheres (circles in 2D, see Fig. 1(a)). However, this shape will change, when we choose another metric. Fig. 1 (b) presents unit spheres for selected values of m for L_m norm in 2D.

5. Experiments

To evaluate the performance of the original V-Detector algorithm two indices were used: *Detection Rate* (*DR*) and *False Alarm Rate* (*FAR*), computed as follow:

$$DR = \frac{TP}{TP + FN}$$
$$FAR = \frac{FP}{FP + TN} \tag{6}$$

where *TP* (*true positive*) is the number of correctly classified anomalous (nonself) samples, *FN* (*false negative*) is the number of self samples recognized as nonself, *FP* (*false positive*) is the number of nonself samples recognized as self and *TN* (*true negative*) is the number of correctly classified self samples.

It is worth to notice that for V-Detector algorithm *FAR* is always equal 0 when the same dataset is used at learning and classification process.

Experiments were performed for the following metrics: $L_{0.3}, L_{0.4}, L_{0.5}, L_{0.7}$, L_1 (Manhattan), L_2 (Euclidean) and cosine with following values of radius of self samples (r_s): 0.01, 0.001, 0001. As a test dataset we take some parts of KDD Cup 1999, namely, K_{ICMP} and K_{UDP} containing description of ICMP and UDP protocols, respectively. Moreover, these sets were divided into several subsets containing connections specific for particular services (consult [3] for details and reasons of such a decomposition):

$$K_{ICMP} = K_{ICMP\,eco\_i} \cup K_{ICMP\,ecr\_i} \cup K_{ICMP\,red\_i} \cup K_{ICMP\,tim\_i} \cup$$
$$\cup K_{ICMP\,urh\_i} \cup K_{ICMP\,urp\_i}$$
$$K_{UDP} = K_{UDP\,domain\_u} \cup K_{UDP\,ntp\_u} \cup K_{UDP\,other} \cup K_{UDP\,private} \cup K_{UDP\,tftp\_u}.$$

All experiments were repeated 20 times.

6. Results

Results presented in Fig. 3 and Fig. 4 shows that the highest values of *DR* were obtained for $L_{m=0.5}$ norm (with some exceptions). It is worth to notice than for $m < 0.5$, the V-Detetcor algorithm generated much worse results than for $m = 0.5$, in despite of tendered theoretical proofs mentioned in Section 2. And for $m \leq 0.2$ none of nonself samples was detected (*DR*=0) in despite of maximal number of detectors was generated ($T_{\max} = 100000$) – this suggest the existence of optimal value of *m* in the interval [0, 1].

One of the disadvantages of the *fractional distance metric* is that the duration of learning and classification stages rapidly increase with lower values of m (see Fig. 5). For example, duration of learning and classification for $m = 1$ is 2-3 times shorter than for $m = 0.5$. It is related with higher number of generated detectors (see Fig. 6) and complexity of calculation of the distances.

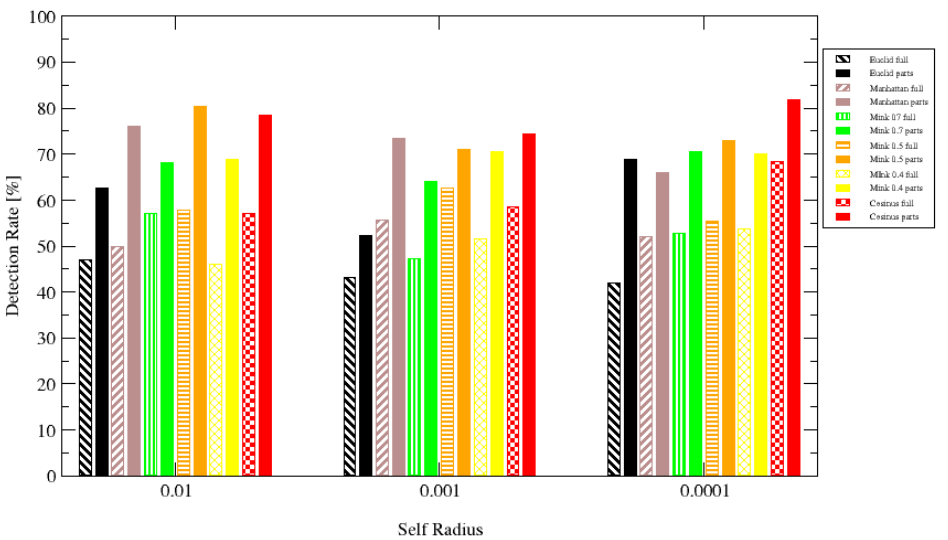


Fig. 3. *Detection Rate* values for ICMP protocol for different norms. "Parts" denotes that overall rate for protocol was calculated from rates obtained for its all subsets.

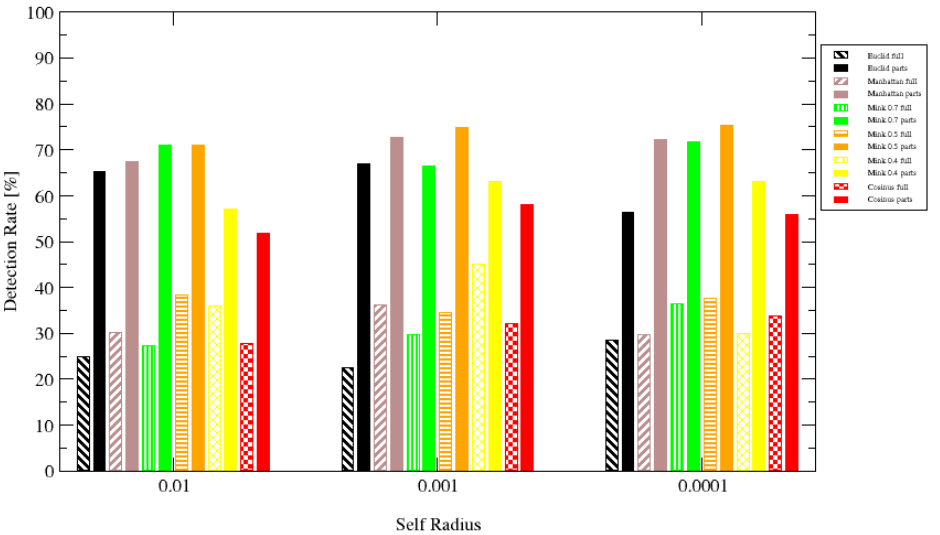


Fig. 4. *Detection Rate* values for UDP protocol for different norms. "Parts" denotes that overall rate for protocol was calculated from rates obtained for its all subsets.

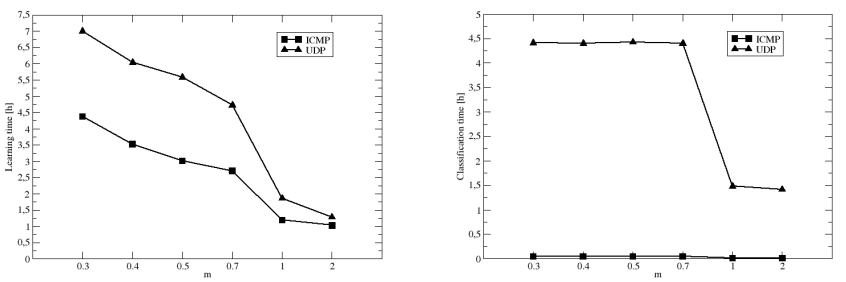


Fig. 5. Duration of learning and classification processes for different values of *m* for L_m -norm.

Cosine metric results in two extreme different values: good results for K_{ICMP} dataset (*DR* over 80%), and poor for K_{UDP}. It seems that this metric can be used only for special type of datasets, after experiment verifying its usability. As cosine metric is usually used to compare documents, it could be very effective in finding anomalies among their.

It worth to notice that, independently on metric, in all cases the *DR* was much grater when testing dataset was divided on subsets (about 2 times).

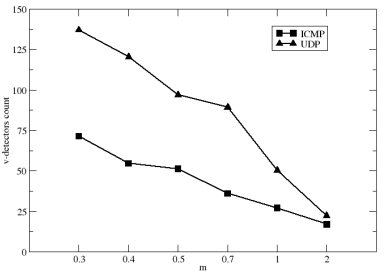


Fig. 6. Number of v-detectors generated for different values of m for of *m* for L_m-norm.

7. Conclusions

The results of experiments affirm that non-Euclidean metrics are more appropriate to calculate distance (proximity) between samples in highly dimensional spaces. Decreasing the value m for L_m-norms causes improvement of the *DR* ratio. However, for low values of m ($m < 0.5$) the efficiency of the algorithm decreases, what implies that the optimal value of m locates somewhere in the interval [0.5, 1.0]; hence, for all the datasets, this value should be properly tuned.

Performed experiments showed also a trade-off between efficiency and time complexity for $L_{m>0.5}$-norms. This is very important information in the case of intrusion detection in computer networks, where the efficiency understood as the time of reaction (to a potential intruder) is of primary interest.

Acknowledgement

Experiments were performed on the computer cluster at Faculty of Computer Science, Bialystok Technical University.

This work was partly supported by Bialystok Technical University grant S/WI/5/03.

References:

[1] Aggarwal C., Hinneburg A., Keim D.A., *On the surprising behavior of distance metrics in high dimensional space*, In Proc. of the 8th International Conference on Database Theory, ICDT 2001, London, pp. 420-434.

[2] Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U. *When is „nearest neighbor" meaningful*, In Proc. of the 7th ICDT, LNCS 1540, Springer 1999, pp. 217-235.

[3] Chmielewski A., Wierzchoń S. T., *Badanie przydatności algorytmu generującego V-detektory do klasyfikacji wybranych zbiorów*, Inżynieria wiedzy i systemy ekspertowe. T.1, Wrocław (Poland), 2006, pp. 13-22.

[4] Chmielewski A., Wierzchoń S. T., *Comparing real-valued negative selection algorithms for intrusion detection applications*, In Proc. of the 13th International Multi-Conference, Advanced Computer Systems (ACS'2006), Vol. 1, Międzyzdroje (Poland), 2006, pp. 387-395.

[5] Chmielewski A., Wierzchoń S.T., *V-Detector algorithm with tree-based structures*, In Proc. of the International Multiconference on Computer Science and Information Technology, Wisła (Poland), 2006, pp. 9-14.

[6] Forrest S., Perelson A., Allen L., Cherukuri R., *Self-nonself discrimination in a computer*. In Proceedings IEEE Symposium on Research in Security and Privacy, Los Alamitos, CA, 1994. IEEE Computer Soc. Press, pp. 202-212.

[7] Gonzalez F., Dasgupta D., Nino L.F., *A randomized real-valued negative selection algorithm*. In: J. Timmis, P.J. Bentley, E. Hart, eds., Proceedings of the 2nd International Conference on Artificial Immune Systems (ICARIS-2003), LNCS 2787, Springer-Verlag, 2003, pp. 261-272.

[8] Hettich S., Bay S. D., KDD Cup 1999 Data (1999), http://kdd.ics.uci.edu.

[9] Ji Z., Dasgupta D., *Real-valued negative selection algorithm with variable-sized detectors*, In Genetic and Evolutionary Computation GECCO-2004, Seattle (USA), Part I. LNCS 3102, Springer-Verlag, 2004, pp. 287-298.

[10] Stibor, T., J. Timmis und C. Eckert: *On the use of hyperspheres in artificial immune systems as antibody recognition regions*, In Proc. of the 5th International Conference on Artificial Immune Systems (ICARIS-2006), LNCS, Springer-Verlag, Oeiras (Portugal), 2006, pp. 215-228.

[11] Stibor, Thomas, Jonathan Timmis und Claudia Eckert: *A comparative study of real-valued negative selection to statistical anomaly detection techniques*, In Proc. of the 4th International Conference on Artificial Immune Systems(ICARIS-2005), Banff (Canada), LNCS, Springer, Berlin/Heidelberg, 2005, pp. 262-275.

[12] Wierzchoń S.T., *Deriving concise description of non-self patterns in an artificial immune system*, In L. C. Jain, J. Kacprzyk, eds., New Learning Paradigms in Soft Computing, Physica-Verlag 2001, pp. 438-458.
[13] Wierzchoń S.T. *Sztuczne systemy immunologiczne*. *Teoria i zastosowania.* Akademicka Oficyna Wydawnicza ELIT, Warszawa 2001.
[14] Weber R., Schek H.-J., Blott S,, *A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces*, In Proc. of VLDB Conference, 1998.

O METRYKACH ODLEGŁOŚCI DLA WIELOWYMIAROWYCH ZBIORÓW DANYCH WYKORZYSTYWANYCH W ALGORYTMIE SELEKCJI NEGATYWNEJ O WARTOŚCIACH RZECZYWISTYCH

Streszczenie: Jednym z kluczowych parametrów algorytmów wykrywania anomalii jest metryka (norma) służąca do obliczania odległości pomiędzy dwiema próbkami, która odzwierciedla ich podobieństwo. Jest ona szczególnie istotna w przypadkach operowania na zbiorach o wielu wymiarach takich, z jakimi mamy do czynienia w przypadku wykrywania intruzów w sieciach komputerowych. Zaobserwowano, że najczęściej stosowana norma euklidesowa staje się bezużyteczna w przestrzeniach o wymiarach większych niż 15. Oznacza to konieczność stosowania innych norm, które pozwoliłyby na zwiększenie skuteczności algorytmu selekcji negatywnej o wartościach rzeczywistych. W artykule prezentujemy wyniki uzyskane dla normy Minkowskiego, L_m, przy zmianach parametru m w zakresie (0, 2] oraz dla odległości kosinusowej.

Słowa kluczowe: selekcja negatywna, wykrywanie anomalii, norma Minkowskiego z ułamkowym wykładnikiem, odległość kosinusowa

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2007
Informatyka – Zeszyt 2

Dorota Duda[1,2], Marek Krętowski[1], Johanne Bézy-Wendling[2]

EKSTRAKCJA CECH TEKSTURALNYCH W KLASYFIKACJI OBRAZÓW TOMOGRAFICZNYCH WĄTROBY

Streszczenie: W pracy przedstawiono nową metodę opisu tekstur, przystosowaną do analizy grupy obrazów, przedstawiających na różne sposoby ten sam fragment organu. Charakteryzując obszary zainteresowania, uwzględniono nie tylko cechy teksturalne wyliczone na ich podstawie, ale również ich zależność od warunków pozyskiwania obrazów. Zaproponowano kilka sposobów konstrukcji przestrzeni parametrów odzwierciedlających zmianę tekstury, która zachodzi pod wpływem zmian warunków akwizycji. Proponowaną metodę zweryfikowano doświadczalnie w klasyfikacji obrazów tomograficznych wątroby. Rozpoznawano cztery typy tkanki, dla każdego przypadku rozważono trzy momenty akwizycji, związane z obecnością i propagacją środka kontrastującego. Wyniki uzyskane przy użyciu różnych zestawów cech teksturalnych i klasyfikatora w postaci dipolowych drzew decyzyjnych pokazują, że uwzględnienie zmian tekstury pod wpływem propagacji środka kontrastującego znacznie poprawia diagnozę.

Słowa kluczowe: klasyfikacja obrazów medycznych, analiza tekstur, tomografie komputerowe wątroby

1. Wprowadzenie

Tomografia komputerowa jest narzędziem szeroko stosowanym w diagnostyce chorób narządów brzusznych. Przy rozpoznawaniu patologii wątroby, standardem stało się wykonywanie trzech serii obrazów: pierwszej – przed podaniem pacjentowi środka kontrastującego oraz kolejnych dwóch – po podaniu kontrastu, w wyróżnionych przez lekarzy radiologów fazach jego propagacji (tętniczej i żylnej). Wizualna analiza różnic w wyglądzie określonego fragmentu wątroby na trzech obrazach odpowiadających trzem kolejnym fazom akwizycji pozwala lekarzowi zlokalizować i wstępnie rozpoznać typ patologii. W wielu przypadkach nie jest

[1] Wydział Informatyki, Politechnika Białostocka, ul. Wiejska 45A, 15-351 Białystok
[2] Laboratoire Traitement du Signal et de l'Image, INSERM, Université de Rennes 1, Campus de Beaulieu, Bâtiment 22, 35042 Rennes CEDEX, France

jednak wystarczająca do postawienia definitywnej diagnozy i musi być uzupełniona badaniem histopatologicznym (biopsją cienkoigłową). Zastosowanie komputerowych metod analizy obrazów może znacznie podwyższyć jakość diagnozy, ograniczając tym samym konieczność stosowania szkodliwych dla organizmu człowieka metod inwazyjnych [5].

Kluczowym zagadnieniem w procesie komputerowego przetwarzania obrazów jest jednoznaczny i obiektywny opis obszarów na nich występujących – tak zwanych obszarów zainteresowania (ang. *Regions of Interests*, *ROIs*). Cennym źródłem informacji o analizowanych obszarach obrazów może być ich tekstura [16]. Analiza tekstur ma na celu znalezienie zbioru parametrów (nazywanych cechami teksturalnymi), z których każdy jest liczbową miarą określonej właściwości tekstury. Rozpatrywaną właściwością może być, na przykład, ziarnistość, kierunek ułożenia wzoru, jednorodność, lokalny kontrast, czy też średni poziom jasności pikseli danego obszaru obrazu. Do tej pory zaproponowano bardzo wiele sposobów ekstrakcji cech teksturalnych, stale pojawiają się nowe propozycje. Spośród nich największą popularność zdobyły metody statystyczne, polegające na analizie histogramu poziomów szarości, macierzy gradientów, macierzy współwystępowania (ang. *co-occurrence matrices*, *COM*) [15] lub macierzy jednorodnych ciągów pikseli (ang. *run-length matrices*, *RLM*) [12]. Powszechnie stosowane są też metody oparte na przekształceniach obrazu (transformacje Fouriera, Gabora [8] lub falkowe [23]), wykorzystujące modele (pola fraktalne [6] i Markowa [9]), lub operacje matematycznej morfologii [17]. Wymienione metody (w połączeniu z odpowiednimi algorytmami klasyfikacji) znalazły szerokie zastosowanie w diagnostyce chorób narządów wewnętrznych zobrazowanych różnymi metodami. Przykładem takiego zastosowania może być rozpoznawanie łagodnych i złośliwych mikrozwapnień na obrazach mammograficznych piersi (promienie X) [30], klasyfikacja blaszek miażdżycowych w tętnicach wieńcowych (ultrasonografia wewnątrznaczyniowa) [26], identyfikacja odmian złośliwego nowotworu mózgu (rezonans magnetyczny) [4], wykrywanie zmian ogniskowych w wątrobie (tomografia komputerowa) [25], ocena struktury kostnej w diagnostyce osteoporozy (mikrotomografia komputerowa) [1], czy też wykrywanie niedokrwienia mięśnia sercowego na obrazach echokardiografii kontrastowej (ultradźwięki) [2].

Pierwsze badania nad możliwościami wykorzystania analizy tekstur w procesie rozpoznawania patologii wątroby przedstawionej na obrazach tomograficznych opisano w pracy [24]. Jej autorzy charakteryzowali dwa typy tkanki wątrobowej (chorą i zdrową) przy pomocy cech teksturalnych policzonych na podstawie macierzy współwystępowania, macierzy jednorodnych ciągów pikseli oraz macierzy różnic poziomów szarości (ang. *gray level difference matrices*, *GLDF*) [31]. Użycie zbioru wybranych parametrów, w skład którego wchodziły: entropia (ang. *entropy*) i lokalna jednorodność (ang. *local homogeneity*), policzone na pod-

stawie macierzy współwystępowania, oraz rozkład poziomów szarości (ang. *grey level distribution*), policzony na bazie macierzy jednorodnych ciągów pikseli, pozwoliło trafnie rozpoznać niemal 99% analizowanych przypadków. W kolejnej pracy [7] zaproponowano system komputerowego wspomagania diagnostyki, który był zdolny do samodzielnego znajdowania konturów wątroby oraz rozpoznawania jej dwóch najczęściej występujących nowotworów. W systemie tym klasyfikacja tekstur, opisanych wymiarem fraktalnym oraz cechami policzonymi na bazie macierzy współwystępowania, odbywała się przy użyciu probabilistycznych sieci neuronowych. Podobną aplikację (wykorzystującą cechy policzone na podstawie macierzy współwystępowania oraz sieci neuronowe z propagacją wsteczną), przystosowaną do rozróżniania zdrowej i patologicznie zmienionej wątroby, opisano w pracy [19]. Sariyanni *et al*. [27] próbowali rozpoznawać tkankę odpowiadającą wątrobie zdrowej oraz rakowi wątrobowokomórkowemu (*carcinoma hepatocellulare*). W tym przypadku analizowane tekstury charakteryzował wymiar fraktalny policzony z zastosowaniem czterech różnych metod. W pracy [13] zaprezentowano system, który, wykorzystując parametry teksturalne policzone na podstawie macierzy współwystępowania oraz trzy sekwencyjnie ułożone jednokierunkowe sieci neuronowe, był zdolny określić występowanie schorzeń wątroby. Kolejny system przedstawiono w pracy [29]. Jej autorzy badali przydatność systemu do wykrywania zmian ogniskowych w wątrobie. Do opisu właściwości czterech typów tkanki (cysty, naczyniak krwionośny (*haemangioma*), rak wątrobowokomórkowy, wątroba zdrowa), wykorzystali macierze współwystępowania, macierze różnic poziomów szarości pikseli, filtrację obrazu z zastosowaniem masek Lawsa [21] oraz wymiar fraktalny. Klasyfikacji natomiast dokonywano stosując sieci neuronowe oraz klasyfikatory statystyczne. Bardo podobny system opisano w pracy [28]. Mala *et al*. [22] zaproponowali system, który pozwalał rozróżnić stłuszczenie wątroby (*steatosis*) od marskości. Opisując tekstury, autorzy wykorzystali ortogonalną transformację falkową oraz cechy statystyczne drugiego rzędu; w procesie klasyfikacji użyto dwuwarstwowej probabilistycznej sieci neuronowej.

Spośród systemów zaprezentowanych w ostatnim roku, na uwagę zasługuje *DIAGNOSIS* [25] – zintegrowany system umożliwiający zarządzanie bazą obrazów medycznych, zdalny dostęp do bazy, półautomatyczną segmentację obrazów, ekstrakcję cech teksturalnych oraz klasyfikację. W aplikacji zastosowano wiele rozwiązań przedstawionych w pracach [29], [28]. Kolejny przykład systemu można znaleźć w pracy [18]. Charakteryzowanie obszarów zainteresowania odbywa się tu przy użyciu znormalizowanych współczynników autokowariancji (ang. *normalized autocovariance coefficients*) [14] poziomów szarości pikseli. W procesie poszukiwania klasyfikatora wykorzystano maszyny wektorów wspierających (ang. *Support Vector Machines*, *SVM*). System testowano na danych odpowiadających złośliwym i łagodnym zmianom nowotworowym w wątrobie.

Cechą wspólną opisanych aplikacji jest analiza tomografii komputerowych wątroby, wykonanych bez wykorzystania środka kontrastującego. W prowadzonych przez nas badaniach [20], do rozpoznawania przerzutów nowotworowych w wątrobie, posłużono się także obrazami tomograficznymi, uzyskanymi w dwóch fazach (tętniczej i żylnej) po podaniu pacjentom kontrastu. Każdą z trzech serii obrazów, odpowiadających trzem fazom akwizycji, rozpatrywano oddzielnie. Wyniki przeprowadzonych wówczas doświadczeń okazały się lepsze w przypadku analizy obrazów wykonanych po podaniu pacjentom środka kontrastującego, przy czym najwyższej jakości diagnozę uzyskano rozpatrując grupę obrazów wykonanych w fazie tętniczej. W kolejnej pracy [11] po raz pierwszy zaproponowaliśmy jednoczesną analizę trójek obrazów przedstawiających ten sam fragment wątroby w trzech kolejnych momentach akwizycji. Wektory parametrów opisujące trójkę obszarów zainteresowania składały się z trzech takich samych zestawów cech teksturalnych, policzonych kolejno dla fazy bez kontrastu, fazy tętniczej i fazy żylnej. W ten sposób uwzględniono zmiany tekstury pod wpływem propagacji środka kontrastującego. Wyniki uzyskane w pracach [11] oraz [10] pokazują, że jednoczesna analiza obrazów w trzech następujących po sobie fazach akwizycji pozwala lepiej rozpoznawać patologie. Nierozwiązanym problemem pozostaje jednak wybór przestrzeni parametrów, odzwierciedlających ewolucję tekstury w czasie.

Dalsza część pracy zorganizowana jest w następujący sposób. W kolejnym paragrafie opisano proces projektowania klasyfikatorów na podstawie danych obrazowych oraz zaproponowano kilka metod konstrukcji wektora cech teksturalnych na podstawie trzech obrazów tego samego fragmentu wątroby (w kolejnych trzech fazach akwizycji). W punkcie 3 opisano przeprowadzone eksperymenty z wykorzystaniem proponowanych zestawów parametrów oraz porównano i przedyskutowano wyniki. Ostatni – czwarty paragraf jest podsumowaniem pracy oraz przedstawia planowane na przyszłość kierunki prac badawczych.

2. Klasyfikacja tekstur

W praktyce klinicznej diagnoza jest często stawiana na podstawie jednoczesnej analizy grupy obrazów, które w różny sposób przedstawiają ten sam fragment organu. Liczbę wykonywanych serii obrazów dyktuje rodzaj badanego organu, sposób propagacji środka kontrastującego (jeśli go zaaplikowano), technika obrazowania, czy też potrzeba porównania obrazów w różnych profilach. Wykonując tomografie komputerowe wątroby rozpatruje się zwykle trzy serie obrazów, związane z obecnością i propagacją środka kontrastującego. Pojawienie się nowotworu lub innych patologii pociąga za sobą zmiany ukrwienia chorych tkanek oraz wpły-

wa na sposób dostarczania krwi do wątroby. Oglądając obrazy wykonane w kolejnych fazach propagacji kontrastu można zlokalizować miejsca o nadmiernym lub zbyt ubogim ukrwieniu, świadczące o obecności choroby. W zależności od jej rodzaju, tekstura rozpatrywanych fragmentów obrazów ulega charakterystycznym zmianom, które można zaobserwować porównując obrazy wykonane przed podaniem pacjentowi płynu kontrastującego oraz po jego podaniu. Na podstawie analizy wizualnej zmian tekstury na obrazach przedstawiających ten sam fragment organu, lecz w innych fazach akwizycji, lekarz próbuje postawić diagnozę.

Projektując system komputerowego wspomagania diagnostyki postąpiono w sposób analogiczny do stosowanego przez lekarzy. Przedmiotem badań było nie tylko znalezienie zestawu cech teksturalnych, odzwierciedlających właściwości analizowanych obszarów zainteresowania, ale również opis ich zmian pod wpływem zmiany warunków pozyskiwania obrazów (w przypadku wątroby – pod wpływem propagacji środka kontrastującego). Analizie poddawano zatem nie pojedyncze obrazy, lecz uporządkowane trójki obrazów, spośród których każdy pochodził z innej serii.

2.1. System komputerowego wspomagania diagnostyki

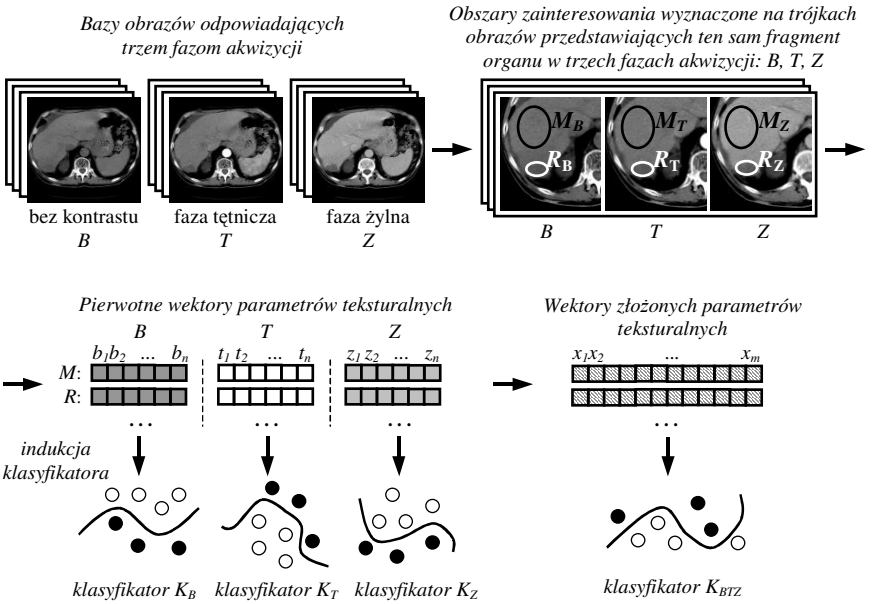
Wstępnym etapem pracy z komputerowym systemem wspomagania diagnostyki jest przygotowanie tego systemu do rozpoznawania ustalonej grupy patologii. W przypadku systemów bazujących na użyciu klasyfikatorów, przygotowanie to polega na stworzeniu zbioru uczącego, w skład którego wchodzą obserwacje o znanej etykiecie (przynależności do klas), a następnie wykorzystaniu go do wygenerowania klasyfikatora. Po nauczeniu klasyfikatora system można wykorzystać w procesie rozpoznawania nowych, niezdiagnozowanych jeszcze przypadków.

Proces budowy klasyfikatorów na podstawie opisanego przez lekarza specjalistę zbioru tomografii komputerowych wątroby przedstawiono na rysunku 1. Pierwszym etapem procesu jest zgromadzenie możliwie jak największej bazy danych składającej się z trzech grup obrazów. Każda z wyróżnionych grup odpowiada innej fazie pozyskiwania obrazów: pierwsza (B) – fazie przed podaniem pacjentowi środka kontrastującego, druga (T) – fazie tętniczej propagacji środka, ostatnia (Z) – fazie żylnej propagacji. Obrazy przedstawiające ten sam fragment organu w trzech kolejnych fazach akwizycji są grupowane i powstałe trójki obrazów są analizowane jednocześnie. Na każdym z obrazów tworzących trójkę wyznacza się taki sam obszar zainteresowania (położenie i rozmiary obszaru są jednakowe dla wszystkich obrazów) i nadaje się mu etykietę. Proces ten przebiega przy udziale lekarza, który bazując na analizie wizualnej i histopatologicznej jest w stanie trafnie określić typ schorzenia. Kolejnym etapem jest policzenie zestawu cech teksturalnych, oddzielnie dla każdego z trzech rozpatrywanych obszarów. Powstają

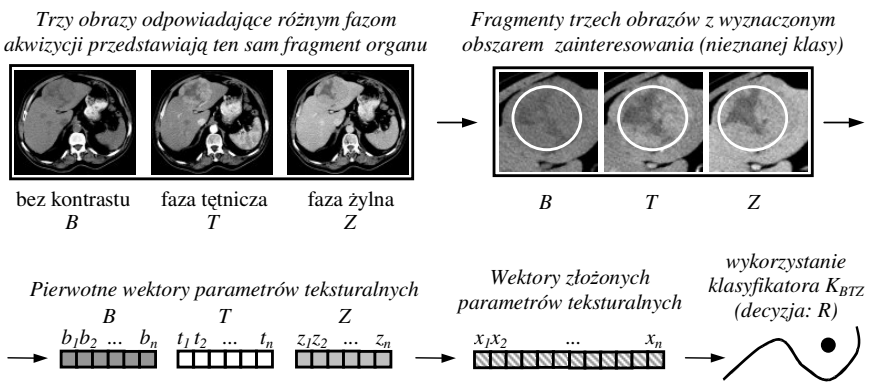
w ten sposób trzy wektory cech (które będziemy nazywać pierwotnymi wektorami cech), $w_B = [b_1, b_2, ..., b_n]$, $w_T = [t_1, t_2, ..., t_n]$, $w_Z = [z_1, z_2, ..., z_n]$, przy czym każde trzy cechy b_i, t_i, z_i $(i = 1,...n)$ liczone są taką samą metodą. Grupy wektorów cech pierwotnych, odpowiadających każdej z trzech faz mogą być analizowane oddzielnie [20] – można poszukiwać klasyfikatora na podstawie wektorów cech odpowiadających jedynie fazie bez kontrastu (klasyfikator K_B), jedynie fazie tętniczej (klasyfikator K_T) lub jedynie żylnej (klasyfikator K_Z). Trójka wektorów (w_B, w_T, w_Z) może posłużyć do konstrukcji nowego zestawu parametrów (nazywanego zestawem parametrów złożonych) $x = [x_1, x_2, ..., x_m]$, które wyrażają zmiany właściwości tekstury na trzech obszarach zainteresowania. Na podstawie zbioru obserwacji opisanych zestawem parametrów złożonych powstaje klasyfikator K_{BTZ}.

Po utworzeniu klasyfikatora (lub grupy klasyfikatorów, wykorzystujących różne algorytmy klasyfikacji), system jest gotowy do wspomagania diagnostyki znanych mu chorób (takich, jakie stwierdzono u pacjentów, których obrazy wykorzystano przy konstrukcji zbioru uczącego).

Rysunek 2 przedstawia schematyczny proces wykorzystania klasyfikatora w rozpoznawaniu typu patologii wątroby. Spośród obrazów wchodzących w skład trzech serii należy wybrać takie (trzy), które przedstawiają ten sam fragment wątroby (każdy obraz ma odpowiadać innej serii). Kolejnym etapem jest wyznaczenie trzech obszarów zainteresowania (po jednym na każdym obrazie) o takich samych wymiarach i współrzędnych. Obszary zainteresowania zakreślone są w obrębie tego fragmentu tkanki, dla którego chcemy postawić diagnozę. Następnie liczy się cechy teksturalne, osobno dla każdego z trzech obszarów zainteresowania. Zestawy cech pierwotnych wyznacza się przy zastosowaniu metod identycznych jakich użyto w procesie indukcji klasyfikatora. Na podstawie trzech zestawów cech jest konstruowany wektor cech złożonych, opisujący trójkę obszarów zainteresowania. Ostatnim etapem jest wykorzystanie zbudowanego klasyfikatora, który przyporządkuje obserwację do jednej z klas. Na tej podstawie można postawić diagnozę.



Rys. 1. Tworzenie klasyfikatora na podstawie zbioru uczącego, zbudowanego z pierwotnych wektorów cech (klasyfikator K_B, klasyfikator K_T, klasyfikator K_Z), oraz na podstawie zbioru wektorów cech złożonych (klasyfikator K_{BTZ}). Na każdym z obrazów wyznaczono dwa obszary zainteresowania, odpowiadające marskości wątroby (M) oraz rakowi wątrobowokomórkowemu (R). Dolne indeksy w oznaczeniach obszarów (B, T, Z) odpowiadają kolejnym fazom akwizycji



Rys. 2. Wykorzystanie klasyfikatora K_{BTZ} w rozpoznawaniu patologii wątroby na podstawie trzech obrazów tego samego fragmentu wątroby, wykonanych w różnych fazach propagacji środka kontrastującego. Decyzja R odpowiada przyporządkowaniu badanego obszaru zainteresowania do klasy „rak wątrobowokomórkowy"

3. Przeprowadzone eksperymenty

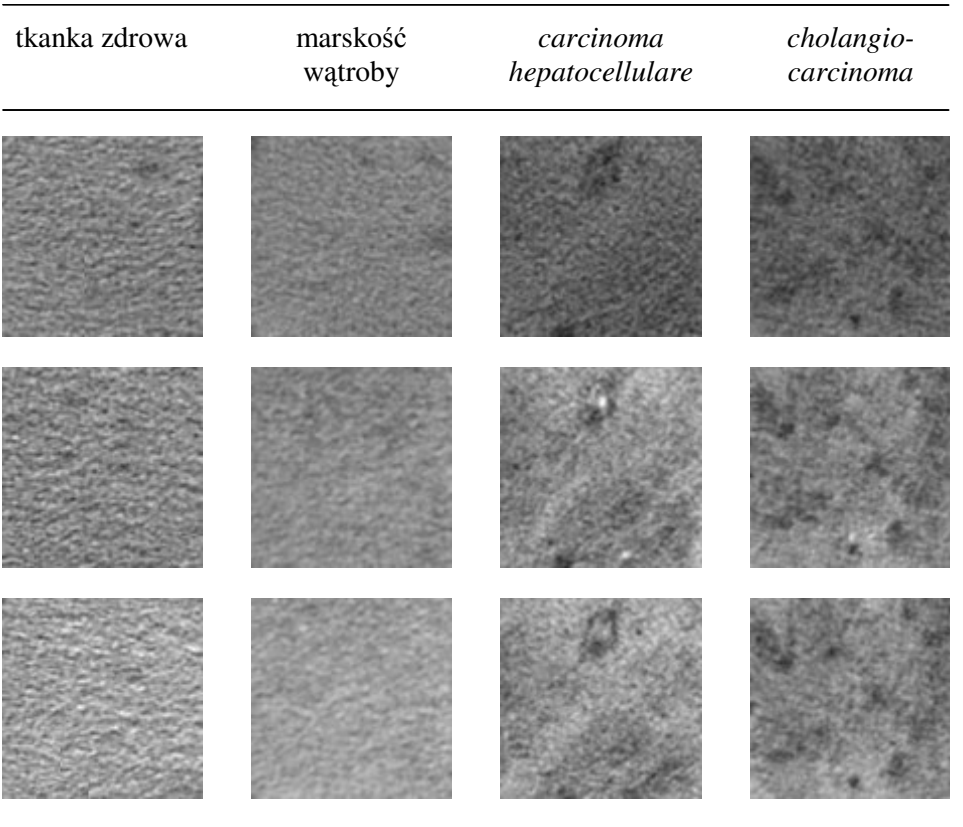
Zaproponowana metoda analizy tekstur była wykorzystana w procesie klasyfikacji obrazów tomograficznych wątroby, wykonanych w trzech fazach związanych z propagacją środka kontrastującego. Rozpoznawano cztery typy tkanki wątrobowej. Dwa pierwsze odpowiadały najczęściej występującym nowotworom pierwotnym wątroby: rakowi wątrobowokomórkowemu (*carcinoma hepatocellulare*) oraz rakowi przewodów żółciowych wewnątrzwątrobowych (*cholangiocarcinoma*). Kolejny typ stanowiła tkanka dotknięta marskością. Ostatnia, czwarta klasa odpowiadała tkance wątroby zdrowej.

3.1. Baza danych obrazowych

Analizowane obrazy pochodziły z dwóch ośrodków w Rennes (Francja): Centrum Leczenia Nowotworów Eugène Marquis (tomograf HiSpeed CT firmy GE Medical Systems) oraz Szpitala Uniwersyteckiego Pontchaillou (tomograf LightSpeed16 firmy GE Medical Systems). W obu szpitalach wykorzystywane były zbliżone protokoły akwizycji: stosowany był tryb skanowania spiralnego, grubość warstwy wynosiła 5 lub 7 mm. Płyn kontrastujący (z reguły 100 ml) podawany był pacjentom przez żyłę przedłokciową, z szybkością 4 ml/s. Pozyskiwanie obrazów odpowiadających fazie tętniczej propagacji środka kontrastującego rozpoczynało się około 20 sekund po rozpoczęciu jego wstrzykiwania. Obrazy odpowiadające fazie żylnej wykonano z opóźnieniem 50-60 sekund. Wszystkie obrazy zapisano w formacie DICOM i miały rozmiar 512x512 pikseli.

W doświadczeniach wykorzystano obrazy pochodzące od 76 pacjentów. Utworzona z nich baza danych zawierała 2961 obrazów, po 987 obrazów uzyskanych w każdej z trzech faz akwizycji. Obrazy pogrupowano w trójki, na których był przedstawiony ten sam fragment wątroby w kolejnych fazach. Na obrazach wykreślono w sumie 1511 trójek obszarów zainteresowania: 537 odpowiadających wątrobie zdrowej, 433 – marskości wątroby, 319 – rakowi wątrobowokomórkowemu oraz 222 – rakowi przewodów żółciowych wewnątrzwątrobowych.

Na potrzeby programu do analizy tekstur (*TextureAnalyser*) obrazy przekonwertowano do formatu BMP, nie tracąc przy tym pełnej informacji o zmienności poziomów szarości pikseli w obrębie zakreślonych obszarów zainteresowania. Przykładowe wycinki analizowanych obrazów przedstawia rysunek 3.



Rys. 3. Przykładowe wycinki (100x100) obszarów zainteresowania, reprezentujące 4 klasy analizowanych tkanek wątroby. Obrazy znajdujące się w górnym wierszu uzyskane były bez podawania pacjentowi środka kontrastującego, kolejne dwa wiersze odpowiadają fazie tętniczej (wiersz środkowy) i żylnej (wiersz dolny) propagacji kontrastu

3.2. Ekstrakcja cech teksturalnych

Obszary zainteresowania (wykreślone na każdym z trzech obrazów) scharakteryzowano przy użyciu następujących zestawów (pierwotnych) cech teksturalnych:

- 4 parametry wyznaczone na podstawie histogramu poziomów szarości,
- 11 cech wyznaczonych na podstawie macierzy współwystępowania,
- 8 cech wyznaczonych na podstawie macierzy jednorodnych ciągów pikseli,
- entropie obrazu przefiltrowanego przy użyciu masek Lawsa (14 parametrów),
- wszystkie cztery wyżej wymienione grupy parametrów jednocześnie (37 cech).

Macierze współwystępowania oraz macierze jednorodnych ciągów pikseli utworzono dla czterech standardowych kierunków ułożenia par (ciągów) pikseli: 0º, 45º, 90º, 135º. W przypadku obu metod zredukowano liczbę poziomów szarości (z 256 początkowo używanych do 64). Podczas tworzenia macierzy współwystępowania wzięto pod uwagę odległości par pikseli od 1 do 5. Cechy liczone dla różnych kierunków oraz różnych odległości pikseli uśredniono. W przypadku filtracji obrazów zastosowano 24 maski Lawsa o wymiarach 5x5. Suma elementów każdej macierzy konwolucji wynosiła zero. Po wykonaniu filtracji obrazy odpowiadające poszczególnym maskom oraz ich transpozycjom dodano. Obrazy odpowiadające maskom symetrycznym zostały pomnożone przez dwa.

Każdy z pięciu wymienionych wcześniej zestawów cech był liczony oddzielnie dla każdego z trzech obszarów zainteresowania. Następnie, na podstawie parametrów teksturalnych odpowiadających trzem obszarom zainteresowania, wyznaczono zestawy parametrów złożonych mające postać:

- $w_{od} = \left[t_1 - b_1, t_2 - b_2, ..., t_n - b_n, z_1 - b_1, z_2 - b_2, ..., z_n - b_n\right]$ ($2n$ parametrów),
- $w_{dz} = \left[t_1/b_1, t_2/b_2, ..., t_n/b_n, z_1/b_1, z_2/b_2, ..., z_n/b_n\right]$ ($2n$ parametrów),
- $w_{zm} = \left[w_1, ..., w_n\right]$ (n parametrów),

gdzie w_i ($i = 1,...n$) jest klasycznym współczynnikiem zmienności i-tej cechy, liczonym na podstawie jej trzech wartości (b_i, t_i, z_i), opowiadających rozpatrywanym fazom akwizycji.

Ponadto wykonano doświadczenia, w których analizowano:

- grupy wektorów opisujących obserwacje w każdej z trzech faz pozyskiwania obrazów: w_B, w_T, w_Z (oddzielna grupa dla każdej z faz),
- zbiór obserwacji opisanych wektorami parametrów odpowiadających jednocześnie trzem fazom akwizycji: $w_{BTZ} = \left[b_1, b_2, ..., b_n, t_1, t_2, ..., t_n, z_1, z_2, ..., z_n\right]$ ($3n$ parametrów) [11],
- zbiór obserwacji opisanych przez $2n$ pierwszych (spośród $3n$) składowych głównych (wektory w_{PCA2n}), które policzono na podstawie zbioru, o którym mowa w poprzednim punkcie,
- zbiór obserwacji opisanych przez n pierwszych składowych głównych (wektory w_{PCAn}), o których mowa powyżej.

We wszystkich eksperymentach wykorzystano klasyfikator w postaci Dipolowych Drzew Decyzyjnych (ang. *Dipolar Decision Trees*) [3] z wbudowaną selekcją cech. W każdym doświadczeniu jakość klasyfikacji oceniano z wykorzystaniem 10-krotnie powtórzonej kroswalidacji 10-przedziałowej. Tabela 1 zawiera zestawienie uzyskanych wyników.

Tabela 1

Jakość klasyfikacji (%) uzyskana dla różnych zestawów pierwotnych cech teksturalnych oraz różnych metod konstrukcji wektora parametrów złożonych, opisujących trójkę obszarów zainteresowania. Kolejne wiersze tabeli odpowiadają różnym metodom ekstrakcji cech, zastosowanym do policzenia parametrów pierwotnych (oddzielnie dla każdej z trzech faz). Przy każdej metodzie podano liczbę wyznaczonych parametrów (n). Kolumny tabeli odpowiadają zaś różnym metodom liczenia złożonych wektorów cech. W nawiasie podano liczby otrzymywanych cech (wykorzystując w każdym przypadku liczbę cech liczonych oddzielnie dla każdej z faz)

Cechy pierwotne		Wektory cech złożonych								
Zestaw cech	Liczba cech, n	w_{BTZ} ($3n$)	w_{PCA2n} ($2n$)	w_{od} ($2n$)	w_{dz} ($2n$)	w_{PCAn} (n)	w_{zm} (n)	w_B (n)	w_T (n)	w_Z (n)
FO	n =4	78,2	78,0	71,5	64,7	69,1	64,0	66,2	52,2	62,7
RLM	n =8	77,4	79,7	75,7	74,9	78,6	72,6	70,4	69,5	71,4
COM	n =11	76,4	76,5	74,0	72,0	78,5	70,6	69,9	66,1	69,9
Laws	n =14	69,1	70,9	56,4	55,9	74,0	70,1	64,3	66,9	66,0
wszystkie	n =37	78,3	77,4	72,8	68,4	77,8	74,5	71,8	70,6	72,5

3.3. Dyskusja

Na podstawie wyników przeprowadzonych doświadczeń można stwierdzić, że analiza zmian charakterystyk teksturalnych, towarzyszących propagacji środka kontrastującego, może prowadzić do poprawy jakości diagnozy. Niemal we wszystkich przypadkach wyniki uzyskane przy zastosowaniu zestawów złożonych cech teksturalnych okazały się lepsze od rezultatów otrzymanych przy oddzielnej analizie trzech grup wektorów, odpowiadających trzem fazom pozyskiwania obrazu (w_B , w_T , lub w_Z). Największe ilości poprawnie rozpoznawanych tkanek wątroby (do 80% analizowanych) odnotowywano przy opisie obserwacji z jednoczesnym zastosowaniem trzech grup nieprzekształconych parametrów (wektory w_{BTZ} o rozmiarze $3n$) lub części składowych głównych ($2n$ lub n pierwszych składowych), otrzymanych na podstawie zbioru obserwacji opisanych $3n$ -wymiarowymi wektorami w_{BTZ} . W przypadku zastosowania zestawów pierwotnych cech teksturalnych, policzonych na podstawie macierzy współwystępowania, macierzy jednorodnych ciągów pikseli lub wszystkich czterech metod, podobne wyniki uzyskano przy różnych liczbach składowych głównych (n lub $2n$) oraz przy wszystkich nieprzekształconych cechach pierwotnych (w liczbie $3n$). W przypadku zestawów cech pierwotnych, otrzymanych z użyciem masek Lawsa, użycie n pierwszych składowych głównych prowadziło do najlepszego rozpoznawania tkanek. Można zatem wnioskować, że wykorzystanie większej liczby składowych głównych (w naszym przypadku powyżej n) nie prowadzi do podwyższenia jakości klasyfikacji, jedynie może wydłużyć czas indukcji klasyfikatora. Co prawda w przypadku parametrów pierwotnych, liczonych na podstawie histo-

gramu poziomów szarości, stosowanie jedynie n składowych głównych prowadziło do znacznego zmniejszenia liczby trafnie rozpoznawanych obiektów (9,1% w stosunku do wyników uzyskanych z zastosowaniem $3n$ nieprzetworzonych parametrów). Jednak w tym przypadku liczba cech pierwotnych wynosiła zaledwie 4, co mogło nie wystarczyć do pełnego opisu zobrazowanych tkanek.

Wyniki uzyskane przy zastosowaniu pozostałych wektorów o $2n$ parametrach (w_{od}, w_{dz}) pokazują, że lepszym rozwiązaniem jest rozpatrywanie różnic wartości parametrów pierwotnych, odpowiadających fazie tętniczej i bez kontrastu oraz żylnej i bez kontrastu (wektory w_{od}). Takie rozwiązanie prowadzi jednak do gorszego rozpoznawania obserwacji (od 2,5% dla zestawu parametrów pierwotnych policzonych na podstawie macierzy współwystępowania do 14,5% dla parametrów pierwotnych policzonych z zastosowaniem masek Lawsa), niż w przy stosowaniu wektorów w_{PCA2n}, o których mowa w poprzednim akapicie. W przypadku rozpatrywania ilorazów wartości parametrów (wektory w_{dz}) jakość klasyfikacji okazywała się gorsza nawet od uzyskanej w sytuacji, gdy analizowano cechy odpowiadające jedynie fazie bez kontrastu (w_B), jedynie fazie tętniczej (w_T) lub jedynie fazie żylnej (w_Z). Tak było przy zastosowaniu zestawu parametrów pierwotnych, policzonych na podstawie histogramu poziomów szarości (1,5% różnicy), obrazów przefiltrowanych z zastosowaniem masek Lawsa (do 11,0% różnicy) oraz przy zastosowaniu kombinacji wszystkich parametrów (do 4,1% różnicy).

Najgorszym rozwiązaniem wydaje się opisywanie zobrazowanych tkanek przy użyciu n współczynników zmienności parametrów pierwotnych. Dla trzech metod ekstrakcji parametrów pierwotnych (wykorzystujących histogram poziomów szarości, macierze jednorodnych ciągów pikseli lub macierze współwystępowania) wspomniana metoda konstrukcji złożonych wektorów cech okazała się najgorsza spośród wszystkich proponowanych. Względnie dobre wyniki uzyskano dla zestawu współczynników zmienności parametrów policzonych na podstawie obrazów przefiltrowanych z zastosowaniem masek Lawsa oraz dla kombinacji współczynników zmienności parametrów policzonych z wykorzystaniem wszystkich czterech metod ekstrakcji cech pierwotnych. Rozpoznawalność tkanek wątroby określono: w pierwszym przypadku ponad 70%, (porównywalny wynik otrzymano wykorzystując wektory zawierające $3n$ nieprzekształconych parametrów, w_{BTZ}), w drugim – 74,5%.

Na koniec warto zauważyć, że, na otrzymywane wyniki, wpływ ma także specyfika użytego klasyfikatora. Dzięki selekcji cech, dokonującej się w trakcie budowy drzewa decyzyjnego, w każdym węźle drzewa, wybierane są cechy, które uznano za najistotniejsze w rozróżnianiu obiektów w węźle. Tym samym może się okazać, że spośród $3n$ parametrów, początkowo opisujących obserwacje w zbiorze

uczącym, tylko część jest wykorzystywana do podejmowania decyzji o przynależności obiektów do klas

4. Podsumowanie

W pracy skupiono się na poszukiwaniu metod możliwie jak najlepszego opisu obszarów zainteresowania na dynamicznych tomografiach komputerowych wątroby. Analizowane obrazy były wykonane w trzech fazach akwizycji, wyróżnionych ze względu na obecność środka kontrastującego: pierwszej – przed podaniem pacjentowi kontrastu, kolejnych dwóch (tętniczej i żylnej) – po jego podaniu. Trójki obrazów (odpowiadających trzem następującym po sobie fazom akwizycji), na których przedstawiony był ten sam fragment wątroby, były analizowane jednocześnie. Uwzględniono w ten sposób nie tylko właściwości tekstury rozpatrywanych obszarów zainteresowania, ale również ich zmiany w czasie, dokonujące się pod wpływem propagacji płynu kontrastującego. Zaproponowano kilka metod opisu ewolucji tekstury w czasie, opierających się na wykorzystaniu różnych funkcji trzech argumentów, którymi były wartości tego samego parametru, odpowiadające trzem fazom pozyskiwania obrazów. Wykorzystano również składowe główne policzone na podstawie zbioru obserwacji opisanych jednocześnie parametrami odpowiadającymi trzem wymienionym fazom. Doświadczenia przeprowadzone na zbiorze danych opisujących cztery typy tkanki wątrobowej pokazały, że znacznie lepszą diagnozę można uzyskać, gdy się uwzględnia informacje o zmianach tekstury w czasie pod wpływem propagacji kontrastu. Najlepszym z proponowanych rozwiązań okazało się zastosowanie zestawów cech zawierających wszystkie nieprzekształcone parametry (odpowiadające fazie bez kontrastu, fazie tętniczej i fazie żylnej). Należy jednak podkreślić, że wybór przestrzeni parametrów opisujących trójki obszarów zainteresowania nie jest definitywny i pozostanie nadal przedmiotem badań.

W przyszłości kontynuowane będą prace nad rozpoznawaniem typów tkanki wątrobowej na dynamicznych tomografiach komputerowych. Baza danych obrazowych będzie rozszerzona o przypadki nowych pacjentów, pojawi się również możliwość klasyfikacji innych, niż dotychczas analizowane, patologii wątroby (na przykład nowotworów wtórnych). Planowane są dalsze poszukiwania przestrzeni parametrów złożonych, charakteryzujących zmiany właściwości tekstury podczas propagacji środka kontrastującego. Przeprowadzone będą również badania nad możliwością zastosowania w systemie klasyfikatorów złożonych. W dalszej perspektywie autorzy rozważają wykorzystanie analogicznych rozwiązań w rozpoznawaniu obrazów innych organów, które wykonano różnymi technikami (na przykład w klasyfikacji glejaków mózgu na obrazach rezonansu magnetycznego).

Literatura:

[1] Apostol L., Peyrin F., Yot S., Basset O., Odet C., Tabary J., Dinten J. M., Boller E., Boudousq V., Kotzki P. O.: *A procedure for the evaluation of 2D radiographic texture analysis to assess 3D bone micro-architecture*, Proc. of SPIE, Vol. 5370 Medical Imaging, 2004, pp. 195-206.

[2] Bae R. Y., Belohlavek M., Greenleaf J. F., Seward J. B.: *Myocardial contrast echocardiography: texture analysis for identification of nonperfused versus perfused myocardium*, Echocardiography, 18(8), 2001, pp. 665-672.

[3] Bobrowski L., Krętowski M.: *Induction of multivariate decision trees by using dipolar criteria*, LNCS Vol. 1910, Springer-Verlag, 2000, pp. 331-336.

[4] Brown R. A., Zlatescu M. C., Cairncross J. G., Mitchell J. R.: *Texture Analysis for Non-Invasive Identification of Brain Tumor Genotype from MRI*, Proc. of the Fifth IASTED International Conference on Visualization, Imaging, and Image Processing, ACTA Press, 2005, pp. 459-464.

[5] Bruno A., Collorec R., Bézy-Wendling J., Reuzé P., Rolland Y.: *Texture analysis in medical imaging*, W: Roux C., Coatrieux J. L. (edytorzy): Contemporary Perspectives in Three-dimensional Biomedical Imaging, IOS Press, 1997, pp. 133-164.

[6] Chen C., Daponte J. S., Fox M. D.: *Fractal feature analysis and classification in medical imaging*, IEEE Transactions on Medical Imaging, 8, 1989, pp. 133-142.

[7] Chen E. L., Chung P. C., Chen C. L., Tsai H. M., Chang C. I.: *An automatic diagnostic system for CT liver image classification*, IEEE Transactions on Biomedical Engineering, 45(6), 1998, pp. 783-794.

[8] Clausi D. A., Jernigan M. E.: *Designing Gabor filters for optimal texture separability*, Pattern Recognition, 33, 2000, pp. 1835-1849.

[9] Cross G. R., Jain A. K.: *Markov random fields texture models*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 5(1), 1985, pp. 25-39.

[10] Duda D., Krętowski M., Bézy-Wendling J.: *Texture characterization for Hepatic Tumor Recognition in Multiphase CT*, Biocybernetics and Biomedical Engineering, 26(4), 2006, pp. 15-24.

[11] Duda D., Krętowski M., Bézy-Wendling J.: *Texture-based classification of hepatic primary tumors in multiphase CT*, Proc. of 7th MICCAI, LNCS Vol. 3217, Springer-Verlag, 2004, pp. 1050-1051.

[12] Galloway M. M.: *Texture analysis using gray level run lengths*, Computer Graphics and Image Processing, 4, 1975, pp. 172-179.

[13] Gletsos M., Mougiakakou S. G., Matsopoulos G. K., Nikita K. S., Nikita A. S., Kelekis D.: *A computer-aided diagnostic system to characterize*

CT focal liver lesions: design and optimization of a Neural Network classifier, IEEE Transactions on Information Technology in Biomedicine, 7(3), 2003, pp. 153-162.

[14] Gonzalez R. C., Woods R. E.: *Image Compression*, W: Digital Image Processing, Wydanie drugie, Reading, MA: Addison-Wesley, 2002, pp. 409-518.

[15] Haralick R. M., Shanmugam K., Dinstein I.: *Textural features for image classification*, IEEE Transactions on Systems, Man and Cybernetics, 3, 1973, pp. 610-621.

[16] Haralick R. M.: *Statistical and structural approaches to texture*, Proc. of the IEEE, 67, 1979, pp. 786-804.

[17] Haralick R., Sternberg S. R., Zhuang X.: *Image analysis using mathematical morphology*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 9(4), 1989, pp. 532-550.

[18] Huang Y. L. Chen J. H. Shen W. C.: *Diagnosis of Hepatic Tumors With Texture Analysis in Nonenhanced Computed Tomography Images*, Academic Radiology, 13(6), 2006, pp. 713-720.

[19] Husain S. A., Shigeru E.: *Use of neutral networks for feature based recognition of liver region on CT images*, Proc. of the IEEE Signal Processing Society Workshop, Vol. 2, 2000, pp. 831-840.

[20] Krętowski M., Bezy-Wendling J., Duda D.: *Classification of hepatic metastasis in enhanced CT images by dipolar decision tree*, Proc. of 19th GRETSI, 2003, pp. 327-330.

[21] Laws K. I.: *Texture Energy Measures*, Proc. of the DARPA Image Understanding Workshop, 1979, pp. 47-51.

[22] Mala K., Sadasivam V.: *Automatic Segmentation and Classification of Diffused Liver Diseases using Wavelet Based Texture Analysis and Neural Network*, Annual IEEE INDICON Conference, 2005, pp. 216 - 219.

[23] Mallat S. G.: *A theory for multiresolution signal decomposition: The wavelet representation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(7), 1989, pp. 674-693.

[24] Mir A. H., Hanmandlu M., Tandon S. N.: *Texture analysis of CT-images*, IEEE Engineering in Medicine and Biology, 5, 1995, pp. 781-786.

[25] Mougiakakou S. G., Valavanis I., Mouravliansky N., Nikita A., Nikita K. S.: *DIAGNOSIS: A Telematics Enabled System for Medical Image Archiving, Management and Diagnosis Assistance*, Proc. of the IEEE IST International Workshop, 2006, pp. 43-48.

[26] Nair A., Kuban B. D., Tuzcu E. M., Schoenhagen P., Nissen S. E., Vince D. G.: *Coronary plaque classification with intravascular ultrasound radiofrequency data analysis*, Circulation, 106(17), 2002, pp. 2200-2206.

[27] Sariyanni C. P. A., Asvestas P., Matsopoulos G. K., Nikita K. S., Nikita A. S. Kelekis D.: *A fractal analysis of CT liver images for the discrimination of hepatic lesions: A comparative study*, Proc. of the 23rd Annual EMBS International Conference, 2001, pp. 1557-1560.

[28] Stoitsis J., Valavanis I., Mougiakakou S. G., Golemati S., Nikita A., Nikita K.: *Computer aided diagnosis based on medical image processing and artificial intelligence methods*, Nuclear Instruments and Methods in Physics Research, 2006, pp. 591-595.

[29] Valavanis I., Mougiakakou S. G., Nikita K. S., Nikita A.: *Computer aided diagnosis of CT focal liver lesions by an ensemble of neural network and statistical classifiers*, Proc. of the IEEE International Joint Conference on Neural Networks, Vol. 3, 2004, pp. 1929-1934.

[30] Wei L., Yang Y., Nishikawa R. M. Jiang Y.: *A study on several Machine-learning methods for classification of Malignant and benign clustered microcalcifications*, IEEE Trans. Med. Imaging, 24(3), 2005, pp. 371-380.

[31] Weszka J. S. Dyer C. R., Rosenfeld A.: *A Comparative Study of Texture Measures for Terrain Classification*, IEEE Trans. Systems, Man, Cybernetics, 6, 1976, pp. 269-285.

TEXTURE FEATURE EXTRACTION IN LIVER CT IMAGE ANALYSIS

Abstract: In the work, a new method of texture characterization from multiple scan series is presented. Images with the same slice position, acquired at different conditions, are analyzed simultaneously. Thereby not only texture characteristics of the considered region of interest are taken into account, but also their variations over the different acquisition moments. A few approaches to description of these variations were proposed. They were applied in recognition of four types of hepatic tissue. Liver CT images were acquired during the three typical phases related to presence and propagation of contrast material. Experiments with various sets of texture parameters and dipolar decision tree as a classifier showed that simultaneous analysis of texture features derived from three subsequent acquisition moments could considerably improve the classification accuracy.

Keywords: computer-aided diagnosis, texture analysis, liver CT images

Artykuł zrealizowano w ramach prac badawczych W/WI/1/05 oraz W/WI/5/05.

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2007
Informatyka – Zeszyt 2

Małgorzata Krętowska[1]

LASY LOSOWE – OCENA JAKOŚCI PROGNOSTYCZNEJ CECH

Streszczenie: W pracy bezwzględny błąd predykcji jest wykorzystywany do oceny jakości prognostycznej poszczególnych cech. Narzędzie prognostyczne – lasy losowe – jest konstruowane w celu uzyskania estymatora funkcji przeżycia. Jest on następnie porównywany z estymatorem funkcji przeżycia Kaplana-Meiera, utworzonym przy założeniu jednorodności populacji. Elementem składowym lasów są dipolowe drzewa przeżycia. Zastosowanie dipolowej funkcji kryterialnej pozwala wykorzystać niepełną informację o czasie zajścia porażki, pochodzącą z obserwacji obciętych.

Słowa kluczowe: lasy losowe, analiza przeżyć, bezwzględny błąd predykcji

1. Wstęp

Ocena stanu zdrowia pacjenta oparta jest z reguły na weryfikacji pewnych cech (np. testów laboratoryjnych), odpowiedzialnych za występowanie określonego schorzenia. Dzisiejszy stan wiedzy o wielu chorobach pozwala dosyć precyzyjnie ocenić obecność czy też nasilenie choroby. Mamy jednak bardzo wiele poważnych schorzeń, których źródła nie znamy. W takich sytuacjach postawienie właściwej diagnozy jest zadaniem bardzo trudnym. Pojawia się wówczas potrzeba weryfikacji posiadanej wiedzy i ocena poszczególnych cech pacjenta pod kątem ich wpływu na analizowaną chorobę.

W analizie przeżyć chcemy zbadać wpływ cech na przeżycie pacjentów. Przez pojęcie przeżycie rozumiemy tu czas do zajścia porażki. Mianem porażki określamy zgon pacjenta, czy też nawrót choroby, przy czym zdarzeniem początkowym, od którego rozpoczynamy odliczanie czasu, jest z reguły rozpoznanie choroby lub operacja. Analizę tego typu danych utrudnia specyfika tych danych. Nie posiadamy dokładnych informacji o czasie wystąpienia porażki u wszystkich pacjentów. Są to tzw. obserwacje obcięte. Zmienna czasowa t występująca w opi-

[1] Wydział Informatyki, Politechnika Białostocka, Białystok

sie pacjenta oznacza tutaj jedynie czas, do którego porażka nie miała miejsca. W przypadku obserwacji pełnej zmienna t określa dokładny czas porażki.

Występowanie obserwacji obciętych uniemożliwia w praktyce stosowanie standardowych modeli regresyjnych, jak też ocenę dopasowania modeli do danych empirycznych poprzez analizę np. współczynnika determinacji. Modelem, który najczęściej jest wykorzystywany do analizy tego typu danych, jest model proporcjonalnych hazardów Cox'a [8]. Dosyć restrykcyjne założenia tego modelu wymuszają rozwój nowych, alternatywnych metod analizy danych przeżycia. Wykorzystywane są tutaj zarówno sieci neuronowe [1, 9], jak też drzewa regresyjne [7] czy modele złożone, takie jak np. lasy losowe [6, 12]. Rozwój tych ostatnich ma na celu stabilizację wyników uzyskanych przy użyciu drzew regresyjnych. Algorytmy indukcji drzew są często algorytmami heurystycznymi, co determinuje możliwość uzyskania różnych drzew dla tych samych danych. Stabilność otrzymanych w ten sposób wyników jest często dyskusyjna, stąd potrzeba stworzenia modeli, które umożliwiłyby uzyskanie rozwiązań powtarzalnych. Pozwoli to również wyodrębnić cechy, które mają rzeczywisty wpływ na przeżycie.

Wybór metody oceny jakości otrzymanego modelu jest zdeterminowany w dużym stopniu sposobem prezentacji wyników: dokładny czas porażki, funkcja przeżycia czy też hazardu. Metodami bezpośrednimi porównuje się wartości empiryczne z teoretycznymi, uzyskanymi jako wynik działania określonego narzędzia prognostycznego. Wśród tych metod możemy wyróżnić współczynniki korelacji rangowej Kendall'a i Somer'a [15], jak też współczynnik zaproponowany przez Schemper'a i Hendersona [19]. W przypadku pośrednim porównuje się nie wartości występujące w zbiorze, ale pewne funkcje, np. funkcję przeżycia, otrzymaną na bazie wartości ze zbioru uczącego, z funkcją przeżycia – rezultatem działania analizowanej metody. Przykładem tego typu rozwiązania jest współczynnik Briera zaproponowany przez Graf i in. [11], czy też miary dokładności predykcji rozwijane przez Schempera [18].

W pracy dokładność predykcji mierzona bezwzględnym błędem predykcji [18] jest wykorzystywana do oceny jakości prognostycznej poszczególnych cech. Dodatkowo weryfikacji podlega stworzone narzędzie – lasy losowe [12], budowane na bazie dipolowych drzew przeżycia [16]. Zastosowanie dipolowej funkcji kryterialnej pozwala wykorzystywać informacje niepełne, pochodzące z danych obciętych. Wynikiem działania lasów losowych jest sumaryczna funkcja przeżycia Kaplana-Meiera, budowana dla nowego pacjenta opisanego pewnym wektorem cech $\boldsymbol{x}$.

Praca składa się jest z sześciu rozdziałów. W rozdziale drugim przedstawiono ogólną charakterystykę danych analizy przeżyć, jak również opisano metodę estymacji funkcji przeżycia. Rozdział trzeci przedstawia schemat budowy lasów losowych wraz z algorytmem indukcji pojedynczego drzewa przeżycia. Opis metod

oceny jakości predykcji zawarty jest w rozdziale czwartym. Rozdział piąty zawiera wyniki eksperymentów, wykonanych na bazie dwóch zbiorów danych: *Primary Biliary Cirrhosis,* zawierający informacje o pacjentach z pierwotną marskością żółciową wątroby, oraz zbiór „*VA lung cancer*" z opisem pacjentów chorych na raka. Rozdział szósty to podsumowanie uzyskanych rezultatów.

2. Dane analizy przeżyć

Niech T oznacza nieujemną zmienną losową reprezentującą czas przeżycia pacjentów, natomiast C - zmienną losową reprezentującą czas obcięcia. Dane analizy przeżyć są reprezentowane przez zmienną $O=(\boldsymbol{X}, T, \Lambda)$, gdzie $\boldsymbol{X}=(X_1, X_2 \ ..., X_N)$ jest zbiorem N zmiennych z pewnej przestrzeni cech, natomiast $\Lambda=\mathrm{I}(T \leq C)$ jest wskaźnikiem przeżycia, gdzie I(*wyrażenie*) przyjmuje wartość 1, jeżeli wyrażenie jest prawdziwe i 0, gdy nie jest prawdziwe. Załóżmy, że mamy dany n elementowy zbiór uczący $L=[\boldsymbol{x}_i, t_i, \delta_i]$, $i=1,2,...,n$, gdzie $\boldsymbol{x}_i$ jest N-wymiarowym wektorem cech, t_i jest czasem przeżycia, natomiast δ_i – oznacza wskaźnik przeżycia, który w przypadku danych obciętych przyjmuje wartość równą 0, a dla danych pełnych jego wartość wynosi 1.

Rozkład zmiennej losowej T może być opisany m. in. przy użyciu funkcji przeżycia, określonej dla danego czasu t jako prawdopodobieństwo tego, że porażka nie miała miejsca wcześniej: $S(t)=P(T>t)$. Najczęściej wykorzystywanym estymatorem funkcji przeżycia jest estymator Kaplana-Meiera [14], który oznaczamy jako $\hat{S}(t)$:

$$\hat{S}(t) = \prod_{j|t_{(j)} \leq t} \left(\frac{m_j - d_j}{m_j} \right) \tag{1}$$

gdzie $t_{(1)}<t_{(2)}<...<t_{(D)}$ są uporządkowanymi rosnąco czasami porażki pacjentów ze zbioru uczącego L, d_j jest liczbą porażek, które miały miejsce w czasie $t_{(j)}$, m_j jest liczbą obserwacji, dla których czas porażki jest nie mniejszy niż $t_{(j)}$. Funkcję przeżycia (warunkową) wyliczoną dla nowego pacjenta opisanego wektorem cech $\boldsymbol{x}$ będziemy oznaczać symbolem $\hat{S}(t \mid x)$.

3. Modele złożone w analizie przeżyć

Wykorzystywane w pracy lasy losowe są zbiorem drzew regresyjnych indukowanych na podstawie danych przeżycia. W odróżnieniu od klasycznych drzew regre-

syjnych [15] poszczególne liście reprezentują krzywe przeżycia Kaplana-Meiera wyznaczone na podstawie obserwacji ze zbioru uczącego, które dany liść osiągnęły.

3.1. Indukcja dipolowego drzewa przeżycia

Większość algorytmów indukcji drzew rozpoczyna swoje działanie od korzenia drzewa. W każdym węźle wyliczany jest test optymalizujący zadane kryterium. Test ten w drzewach wielowymiarowych przyjmuje postać hiperpłaszczyzny $H(\boldsymbol{w}, \theta) = \{\boldsymbol{x}: <\boldsymbol{w}, \boldsymbol{x}>=\theta\}$. Odpowiednio dobrana funkcja decyzyjna dzieli zbiór uczący na dwa rozłączne podzbiory (drzewo binarne). W przypadku danych analizy przeżyć powinny powstać podzbiory różniące się długością czasu przeżycia. Proces podziału jest powtarzany dla każdego nowo utworzonego węzła-potomka do momentu, aż zostanie spełnione kryterium stopu i węzeł zostanie uznany za liść, czyli powstałe podzbiory są jednorodne z punktu widzenia czasu przeżycia.

Celem algorytmu indukcji drzewa jest wyznaczenie odpowiedniej liczby węzłów wewnętrznych drzewa, jak też położenia hiperpłaszczyzn $H(\boldsymbol{w}, \theta)$ w każdym węźle [4]. Proponowana w pracy metoda podziału (budowy testów w węzłach drzewa regresyjnego) bazuje na wykorzystaniu dipolowej funkcji kryterialnej [3], której budowa opiera się na pojęciu dipola. Dipolem nazywany parę wektorów cech $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$. Wyróżniamy dwa rodzaje dipoli - mieszane i czyste. Dipol mieszany tworzymy pomiędzy parą wektorów cech, które powinny zostać rozdzielone, dipol czysty pomiędzy wektorami cech jednorodnymi w punktu widzenia analizowanego kryterium. W przypadku analizy przeżyć dipole czyste tworzymy pomiędzy parami wektorów cech, dla których różnica czasu przeżycia jest odpowiednio mała, dipole mieszane pomiędzy tymi parami, dla których różnica czasu przeżycia jest odpowiednio duża. Uwzględniając zróżnicowanie obserwacji (pełne, obcięte) można sformułować następujące reguły konstrukcji dipoli [17]:

Para wektorów cech $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$ tworzy dipol czysty, gdy

- $\delta_i = \delta_j = 1 \wedge | t_i - t_j | < \eta$

Para wektorów cech $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$ tworzy dipol mieszany, gdy

- $\delta_i = \delta_j = 1 \wedge | t_i - t_j | > \zeta$
- $\delta_i = 0, \delta_j = 1 \wedge t_i - t_j > \zeta$ lub $\delta_i = 1, \delta_j = 0 \wedge t_j - t_i > \zeta$

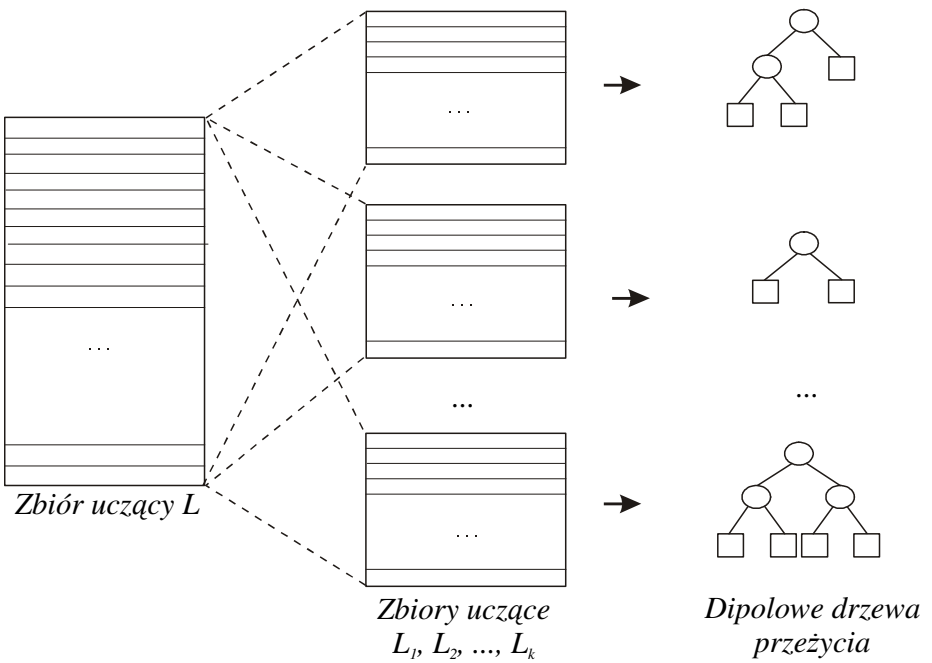
Parametry η oraz ζ przyjmują wartości równe kwantylom wartości bezwzględnych różnic czasu porażki obserwacji pełnych. Parametr η ustalany jest jako kwantyl rzędu 0.1-0.3, ζ jako kwantyl rzędu 0.6-0.9.

Z każdym dipolem związana jest odcinkowo-liniowa funkcja kary. Z dipolami mieszanymi wiążemy funkcje, których minimalizacja pozwoli znaleźć hiperpłaszczyznę $H(\boldsymbol{w}, \theta)$, przecinającej te dipole. Z dipolami czystymi natomiast wiążemy takie funkcje, aby hiperpłaszczyzna $H(\boldsymbol{w}, \theta)$ ich nie przecięła. Suma funkcji kary

po wszystkich dipolach tworzy dipolową funkcję kryterialną. W wyniku jej minimalizacji, przeprowadzanej przy użyciu algorytmów wymiany rozwiązań bazowych [2], otrzymujemy wartości współczynników $\boldsymbol{w}$ i θ hiperpłaszczyzny $H(\boldsymbol{w}, \theta)$ w kolejnych węzłach drzewa.

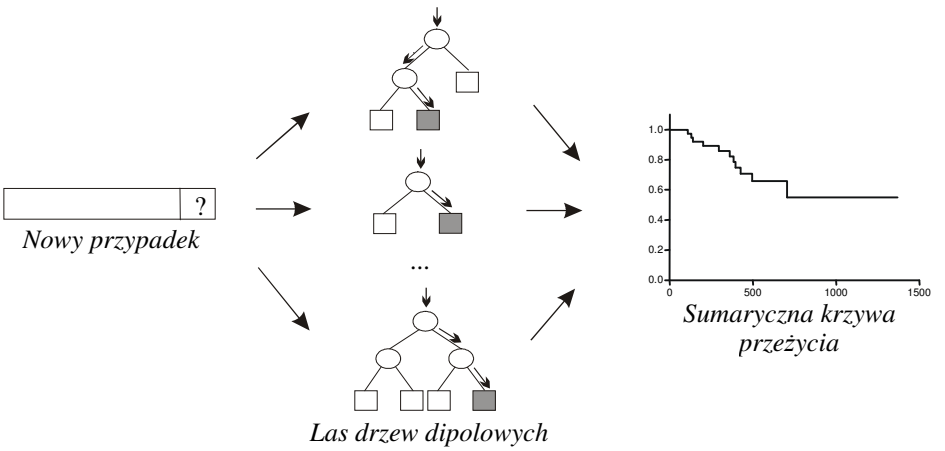
3.2. Algorytm budowy lasów losowych

Metoda lasów losowych [12] pozwala na estymację warunkowej funkcji przeżycia $\hat{S}(t \mid x_{n'})$. Jest ona budowana na bazie k zbiorów uczących ($L_1, L_2, ..., L_k$) losowanych ze zwracaniem ze zbioru uczącego L. Dla każdego zbioru uczącego L_i (i=1,2,.., k) wyznaczany jest zbiór obserwacji $L_i(\boldsymbol{x}_{n'})$ bliskich wektorowi $\boldsymbol{x}_{n'}$.



Rys. **1**. Konstrukcja lasu losowego

Dipolowe drzewo przeżycia jest indukowane dla każdego zbioru L_i, i=1,2,.., k (rys. 1). Wektor cech $\boldsymbol{x}_j$ jest włączany do zbioru $L_i(\boldsymbol{x}_{n'})$, jeżeli należy do tego samego liścia co wektor $\boldsymbol{x}_n$. Mając dane k zbiorów $L_i(\boldsymbol{x}_{n'})$, tworzy się rodzinę zbiorów $L_A(\boldsymbol{x}_{n'})$: $L_A(\boldsymbol{x}_{n'})=[L_1(\boldsymbol{x}_{n'}); L_2(\boldsymbol{x}_{n'}); ... ; L_k(\boldsymbol{x}_{n'})]$. Sumaryczna, warunkowa funkcja przeżycia Kaplana-Meiera, wyznaczana na bazie otrzymanego zbioru $L_A(\boldsymbol{x}_{n'})$ będzie określana mianem $\hat{S}_A(t \mid x_{n'})$.



Rys. **2**. Wykorzystanie lasu losowego

Algorytm budowy lasów losowych możemy zatem przedstawić w następujących punktach:

- Wylosowanie ze zwracaniem k n-elementowych zbiorów (L_1, L_2, ..., L_k) ze zbioru uczącego L
- Indukcja dipolowych drzew przeżycia $T(L_i)$ na bazie kolejnych zbiorów L_i, i=1,2,.., k
- Tworzenie rodziny zbiorów danych $L_A(\boldsymbol{x}_{n'})=[L_1(\boldsymbol{x}_{n'}); L_2(\boldsymbol{x}_{n'}); \ldots ; L_k(\boldsymbol{x}_{n'})]$ (dla nowego pacjenta, opisanego wektorem cech $\boldsymbol{x}_{n'}$).
- Estymacja sumarycznej funkcji przeżycia Kaplana-Meiera $\hat{S}_A(t \mid x_{n'})$ dla nowej obserwacji $\boldsymbol{x}_{n'}$ (rys. 2).

4. Ocena jakości predykcji

Ocena jakości prognostycznej danego narzędzia, a przez to poszczególnych cech tworzących to narzędzie, wykonana jest przy użyciu miary zaproponowanej przez Schempera [18]. Bezwzględny błąd predykcji jest wyliczany na bazie wartości funkcji przeżycia Kaplana-Meiera $\hat{S}(t)$, wyliczanej przy założeniu jednorodności populacji (bez uwzględniania wartości zmiennych) i warunkowej funkcji przeżycia otrzymanej w wyniku działania analizowanego narzędzia $\hat{S}(t \mid x)$.

Estymator bezwzględnego błędu predykcji, wyliczany dla dowolnego czasu $t_{(j)}$, jest zdefiniowany jako:

$$\tilde{M}(t_{(j)}) = 2\hat{S}(t_{(j)})(1-\hat{S}(t_{(j)})) \tag{2}$$

oraz

$$\tilde{M}(t_{(j)} \mid x) = 2n^{-1}\sum_i \hat{S}(t_{(j)} \mid x_i)(1-\hat{S}(t_{(j)} \mid x_i)) \tag{3}$$

Ponieważ z reguły zainteresowani jesteśmy przeżyciem przez określony czas $(0, \tau)$, uogólnione estymatory bezwzględnego błędu predykcji wylicza się jako ważoną średnią estymatorów (2) i (3) po wszystkich czasach porażki ze zbioru uczącego. Wyliczanie wartości wag pozwala na uwzględnienie występowania obserwacji obciętych w zbiorze. Otrzymujemy następujące estymatory:

$$\tilde{D}_s = w^{-1}\sum_j \hat{G}(t_{(j)})^{-1} d_j \tilde{M}(t_{(j)}) \tag{4}$$

$$\tilde{D}_{s,x} = w^{-1}\sum_j \hat{G}(t_{(j)})^{-1} d_j \tilde{M}(t_{(j)} \mid x) \tag{5}$$

gdzie $w = \sum_j \hat{G}(t_{(j)})^{-1} d_j$, d_j jest liczbą porażek, które miały miejsce w czasie $t_{(j)}$ natomiast $\hat{G}$ oznacza estymator Kaplana-Meiera, przy czym jako porażkę traktuje się obcięcie obserwacji.
Wariancja wyjaśniona modelem jest zdefiniowana jako:

$$\tilde{V}_s = \frac{(\tilde{D}_s - \tilde{D}_{s,x})}{\tilde{D}_s} \tag{6}$$

5. Wyniki eksperymentów

Analizę wykonano na bazie dwóch zbiorów danych. Pierwszy zbiór, *Primary Biliary Cirrhosis – PBC,* zawiera dane z kliniki w Mayo [10]. Obserwacji poddani byli pacjenci z pierwotną marskością żółciową wątroby. 312 pacjentów brało udział w randomizowanym badaniu klinicznym, mającym na celu ocenę leczenia z wykorzystaniem D-penicillaminy. Okres obserwacji trwał 10 lat: od 1974 do 1984 roku. Czas przeżycia liczony był od momentu rejestracji do zgonu, transplantacji wątroby lub zakończenia okresu obserwacji. Do analizy wzięto następujące cechy: wiek (w latach) - WIEK, występowanie obrzęku (tak, nie), stężenie albuminy we krwi [g/dl] - ALBUMINA, log(poziom bilirubiny we krwi [mg/dl]) -LOGBIL oraz log(czas protrombinowy [sek]) -LOGPRO. Zbiór zawiera 60% obserwacji obciętych.

Stosując model regresji wielokrotnej Cox'a uzyskano następujące zmienne znaczące: LOGBIL ($p<0.00001$), ALBUMINA ($p<0.0001$), WIEK ($p=0.0001$), LOGPRO ($p=0.001$) [18].

Tabela 1

Bezwzględny błąd predykcji i wariancja wyjaśniona modelem dla zbioru *PBC*

Model	Bezwzględny błąd predykcji (średnia ± odch. std.)	Wariancja wyjaśniona (średnia ± odch. std.)
Estymator KM	0,37	-
Model regresji Cox'a	0,23	0,40
Lasy losowe		
wszystkie zmienne	0,233 ± 0,005	0,37 ± 0,013
LOGBIL	0,244 ± 0,005	0,34 ± 0,014
ALBUMINA	0,289 ± 0,007	0,22 ± 0,02
WIEK	0,335 ± 0,013	0,095 ± 0,035
LOGPRO	0,3 ± 0,003	0,189 ± 0,009

W skład wykorzystywanych w analizie lasów losowych wchodzi 100 drzew dipolowych. Dla każdego zestawu danych analiza była powtórzona 20 razy. Uzyskane wartości średnie bezwzględnego błędu predykcji oraz wariancji wyjaśnionej modelem wraz z odchyleniem standardowym przedstawione są w tabeli 1. Możemy zauważyć, że bezwzględny błąd predykcji jest porównywalny z błędem uzyskanym przy modelu regresji Cox'a (0,23), tym samym procent wariancji wyjaśnionej jest podobny (40% i 37%, odpowiednio dla modelu Cox'a i lasów losowych). Analizowano również wpływ poszczególnych pojedynczych cech. Najlepsze własności predykcji dla pacjentów z pierwotną marskością żółciową wątroby ma poziom bilirubiny we krwi (logarytm). Wprowadzenie tej zmiennej redukuje błąd predykcji średnio o 12,6. Wariancja wyjaśniona modelem wynosi 34%. Zmiennymi mającymi mniejszy wpływ są wiek i czas protrombinowy, dla których bezwzględny błąd predykcji jest równy odpowiednio 0,335 i 0,3.

Drugi zbiór „*VA lung cancer*" (ang. *Veteran's Administration lung cancer*) [13] zawiera informacje o 137 pacjentach (9 obserwacji obciętych), u których wykryto raka płuc. Pacjenci, mężczyźni, zostali poddani terapii standardowej (69 przypadków) oraz chemoterapii (68). Dodatkowo są oni opisani następującymi zmiennymi: indeks KPS (stan pacjenta w trakcie randomizacji: 10-30 – w trakcie pobytu w szpitalu, 40-60 – okresowo hospitalizowany i okresowo pod opieką poradni ambulatoryjnej, 70-90 – pod opieką ambulatoryjną), czas trwania choroby w miesiącach, wiek, wcześniejsza terapia (tak, nie) oraz typ komórek rakowych

(0 – rak płaskonabłonkowy, 1 – rak drobnokomórkowy, 2 – gruczolakorak, 3 – rak wielkokomórkowy).

Tabela 2

Bezwzględny błąd predykcji i wariancja wyjaśniona modelem dla zbioru „*VA lung cancer*"

Model	Bezwzględny błąd predykcji (średnia ± odch. std.)	Wariancja wyjaśniona (średnia ± odch. std.)
Estymator KM	0,335	-
Lasy losowe		
wszystkie zmienne	0,214 ± 0,008	0,357 ± 0,03
WIEK	0,314 ± 0,012	0,061 ± 0,037
CZAS_CHOROBY	0,315 ± 0,008	0,061 ± 0,023
KPS	0,253 ± 0,008	0,245 ± 0,023
TYP_KOMÓREK	0,297 ± 0,006	0,115 ± 0,018

Wyniki uzyskane dla zbioru „*VA lung cancer*" zawiera tabela 2. Bezwzględny błąd predykcji dla estymatora funkcji przeżycia Kaplana-Meiera wynosi 0,335. W przypadku lasów losowych, brano pod uwagę model wykorzystujący wszystkie wymienione wyżej cechy oraz modele budowane na podstawie pojedynczych cech: wiek pacjentów (WIEK), indeks KPS (KPS), czas trwania choroby w miesiącach (CZAS_CHOROBY) oraz typ komórek rakowych (TYP_KOMÓREK). Można zaobserwować, że najlepszą jakość predykcji ma zmienna KPS (średni błąd wynosi 0,253, co daje nam 24,5% wariancji wyjaśnionej modelem), następnie typ komórek rakowych (0,297 i 11,5%, odpowiednio błąd i wariancja wyjaśniona modelem). Wpływ wieku i czasu choroby na przeżycie jest podobny. Dla tych zmiennych błąd predykcji wynosi odpowiednio 0,314 i 0,315, co wyjaśnia tylko 6,1% wariancji.

6. Podsumowanie

W pracy przedstawiono możliwość wykorzystania bezwzględnego błędu predykcji oraz wariancji wyjaśnionej modelem do oceny jakości prognostycznej cech. Miary dokładności predykcji budowane były przez porównanie, uzyskanych na bazie lasów losowych, sumarycznych estymatorów funkcji przeżycia Kaplana-Meiera z funkcją przeżycia, utworzoną przy założeniu jednorodności populacji. Dodatkowo jakość narzędzia prognostycznego – lasów losowych – została zweryfikowana przez porównanie z modelem Cox'a. Uzyskane wyniki potwierdzają przydatność lasów losowych do analizy danych przeżycia, a tym samym do oceny jakości prognostycznej poszczególnych zmiennych.

Literatura

[1] Biganzoli, E., Boracchi, P., Mariani, L., Marubini, E., *Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach*, Statistics in Medicine 17(10), 1998, str. 1169-1186

[2] Bobrowski, L., *Design of piecewise linear classifiers from formal neurons by some basis exchange technique*, Pattern Recognition 24(9), 1991, str. 863-870.

[3] Bobrowski, L., Krętowska, M., Krętowski, M., *Design of neural classifying networks by using dipolar criterions*, Proceedings of the 3rd Conference on Neural Networks and their Applications, Częstochowa, 1997, str. 689-694.

[4] Bobrowski, L., Krętowski, M., *Induction of multivariate decision trees by using dipolar criteria*, Zighed D.A., Komorowski J., Żytkow J. (Eds.): PKDD 2000, LNAI 1910, Springer-Verlag, 2000, str. 331-336

[5] Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J., *Classification and Regression Trees*, Wadsworth, 1984.

[6] Breiman, L., *How to use survival forest.* [http://stat-www.berkeley.edu/users/breiman]

[7] Ciampi, A., Negassa, A., Lou, Z., *Tree-structured prediction for censored survival data and the Cox model,* Journal of Clinical Epidemiology 48(5), 1995, str. 675-689

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

[9] Faraggi, D., Simon, R., *A neural network model for survival data*, Statistics in Medicine 14, 1995, str. 73-82

[10] Fleming, T. R., Harrington, D. P., *Counting Processes and Survival Analysis*, John Wiley & Sons, Inc., 1991

[11] Graf, E., Schmoor, C., Sauerbrei, W., Schumacher, M., *Assessment and comparison of prognostic classification schemes for survival data*, Statistics in Medicine 18, 1999, str. 2529-2545

[12] Hothorn, T., Lausen, B., Benner, A., Radespiel-Troger, M., *Bagging survival trees*, Statistics in medicine 23, 2004, str. 77-91

[13] Kalbfleisch, J.D., Prentice, R.L., *The statistical analysis of failure time data*, Wiley, New York, 1980.

[14] Kaplan, E.L., Meier, P., *Nonparametric estimation from incomplete observations*, Journal of the American Statistical Association 5, 1958, str. 457-481

[15] Korn, E. L., Simon, R., *Measures of explained variation for survival data*, Statistics in medicine 9, 1990, str. 487-503

[16] Krętowska, M.: *Random forest of dipolar trees for survival prediction*, Lecture Notes in Computer Science 4029, Lecture Notes in Artificial Intelligence, 2006, str. 909-918.
[17] Krętowska, M., *Dipolar regression trees in survival analysis,* Biocybernetics and biomedical engineering 24(3), 2004, str. 25-33
[18] Schemper, M., *Predictive accuracy and explained variation*, Statistics in medicine 22, 2003, str. 2299-2308
[19] Schemper, M., Henderson, R., *Predictive accuracy and explained variation in Cox regression*, Biometrics 56(1), 2000, str. 494-255

RANDOM FORESTS – EVALUATION OF PREDICTIVE ACCURACY

Abstract: In the paper, predictive accuracy measured as the absolute predictive error is used to evaluate the quality of covariates. The prognostic tool – random forests – is built to receive the aggregated survival function. The function is compared to Kaplan-Meier estimator of survival function with assumption that the population is homogenous. The induction of individual dipolar survival tree is based on minimization of a piece-wise linear function – dipolar criterion. The algorithm allows using the information from censored observations for which the exact survival time is unknown.

Keywords: random forest, survival analysis, predictive accuracy, explained variation

Artykuł zrealizowano w ramach pracy badawczej W/WI/4/05.

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2007
Informatyka – Zeszyt 2

Tomasz Łukaszuk[1], Leon Bobrowski[1,2]

TEMPORALNOŚĆ W MODELACH RANGOWYCH

Streszczenie: W zbiorze danych określony jest pewien porządek czasowy dla wybranych obiektów. Poprzez model rangowy rozumiemy taką liniową transformację, która zachowuje w najlepszym możliwym stopniu wiedzę a priori o uporządkowaniu obiektów. W artykule przedstawiono koncepcję budowy modelu rangowego opierając się na minimalizacji wypukłej i odcinkowo-liniowej (CPL) funkcji kryterialnej. Zagadnienie zostało sprowadzone do problemu znalezienia optymalnej hiperpłaszczyzny rozdzielającej zbiory zbudowane z elementów powstałych z różnic arytmetycznych wektorów cech tworzących pary, dla których określony jest porządek czasowy.

Słowa kluczowe: model rangowy, wypukła i odcinkowo-liniowa funkcja kryterialna (CPL), liniowa separowalność zbiorów danych

1. Wprowadzenie

W związku z ogromnym wzrostem w ostatnich latach liczby gromadzonych danych, wzrosło także zainteresowanie wydobywaniem ukrytych w tych danych informacji. To wydobywanie informacji polega głównie na klasyfikowaniu, grupowaniu i odnajdywaniu zależności w danych [8], [9]. Jednym z ważniejszych problemów analizy jest postępowanie z danymi zawierającymi zależności czasowe.

Zależności czasowe mogą być określone w postaci pewnego porządku czasowego pomiędzy wybranymi obiektami ze zbioru danych. Na przykład, możemy posiadać informację, że pewne obiekty są starsze (bardziej zaawansowane w rozpatrywanym procesie) niż obiekty z jednego zbioru, natomiast te same obiekty są młodsze (mniej zaawansowane w procesie) niż obiekty z drugiego zbioru. Tego typu wiedza a priori na temat relacji wybranych obiektów może być podstawą do utworzenia modelu rangowego.

Poprzez model rangowy rozumiemy tutaj taką liniową transformację, która zachowuje w najlepszym możliwym stopniu wiedzę *a priori* o uporządkowaniu obiektów. Proces budowy modelu rangowego polega na znalezieniu parametrów

[1] Wydział Informatyki, Politechnika Białostocka, Białystok
[2] Instytut Biocybernetyki i Inżynierii Biomedycznej, PAN, Warszawa

przekształcenia liniowego na podstawie wiedzy *a priori* o porządku istniejącym w danych.

Celem autorów artykułu jest przedstawienie procedury budowy modelu rangowego, bazującej na minimalizacji wypukłej i odcinkowo-liniowej (CPL) funkcji kryterialnej. Funkcja kryterialna tego typu jest sumą dodatnich i ujemnych funkcji kary typu CPL, które są zdefiniowane na podstawie różnic arytmetycznych pomiędzy wektorami cech tworzącymi dipole (pary obiektów, co do których posiadamy wiedzę *a priori* o relacji czasowej pomiędzy nimi) [2]. W ten sposób zadanie budowy modelu rangowego może być sprowadzone do problemu zapewnienia liniowej separowalności dwóch zbiorów danych w zadanej przestrzeni cech.

2. Liniowa transformacja rangowa

Niech badane obiekty (np. pacjenci, samochody, obrazy graficzne) O_j $(j = 1,...,m)$ będą reprezentowane przez n-wymiarowe wektory cech $\mathbf{x}_j = [x_{j1},...,x_{jn}]^T$. Cecha (atrybut) x_i opisuje wartość liczbową określonego $i-$tego parametru lub wynik określonego badania obiektu O_j. Cechy mogą być binarne ($x_i \in \{0,1\}$) lub ciągłe ($x_i \in R^1$).

W zbiorze O_j obiekty są w pewien sposób uporządkowanie. Uporządkowanie to ma (może mieć) charakter jakościowy i wskazuje, że np. pewne obiekty są starsze lub lepsze pod określonym względem niż inne obiekty. Zakładamy, że wiedza *a priori* o uporządkowaniu obiektów dana jest w postaci relacji następstwa „$\prec$" wybranych par wektorów cech.

$$\mathbf{x}_j \prec \mathbf{x}_{j'} \Leftrightarrow \mathbf{x}_{j'} \quad nastepuje\_po \quad \mathbf{x}_j \tag{1}$$

Jeżeli wektory cech $\mathbf{x}_j$ i $\mathbf{x}_{j'}$ pozostają w relacji następstwa (1), oznacza to, że obiekt $O_{j'}$ reprezentowany przez wektor $\mathbf{x}_{j'}$ jest bardziej zaawansowany ze względu na rozważany czynnik niż obiekt O_j, reprezentowany przez wektor $\mathbf{x}_j$.

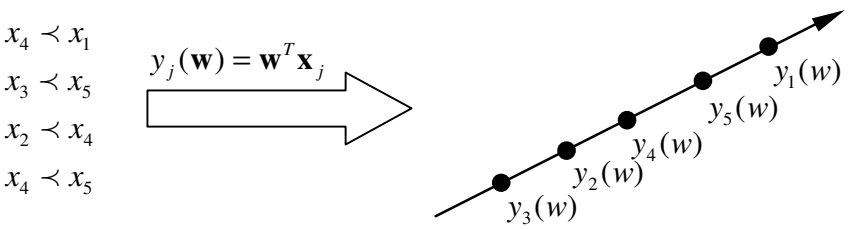
Rozważamy odwzorowanie liniowe postaci

$$y_j = \mathbf{w}^T \mathbf{x}_j \quad (j = 1,...,m) \tag{2}$$

gdzie $\mathbf{w} = [w_1,...,w_N]^T \in R^N$ jest wektorem parametrów. Odwzorowanie (2) przyporządkowuje poszczególnym wektorom cech $\mathbf{x}_j$ punkty y_j na prostej. Punkty y_j mogą być uporządkowane na prostej zgodnie z relacją większościową

$$y_{j(1)} < y_{j(2)} < ... < y_{j(m)} \tag{3}$$

Jesteśmy zainteresowani wyborem takiego wektora parametrów $\mathbf{w}$, który daje największą możliwą zgodność uporządkowania punktów y_j na prostej (2) z relacją następstwa „$\prec$" wektorów cech $\mathbf{x}_j$. Wyznaczenie wektora $\mathbf{w}$ o tej właściwości nazywamy zagadnieniem regresji rangowej.



Rys. 1. Przykład relacji następstwa oraz uporządkowania punktów na prostej zgodnie z tą relacją

3. Dodatnio i ujemnie zorientowane dipole

Relacja następstwa „$\prec$" może być użyta w określaniu orientacji dipoli $\{\mathbf{x}_j, \mathbf{x}_{j'}\}$ ($j < j'$) utworzonych z wektorów cech, dla których relacja jest dana.

Definicja 1: Para $(\mathbf{x}_j, \mathbf{x}_{j'})$ ($j < j'$) wektorów cech $\mathbf{x}_j$ i $\mathbf{x}_{j'}$ tworzy dipol z orientacją dodatnią $\{\mathbf{x}_j, \mathbf{x}_{j'}\}$ ($(j, j') \in I^+$) wtedy i tylko wtedy, gdy $\mathbf{x}_j \prec \mathbf{x}_{j'}$.

$$(\forall (j, j') \in I^+) \quad \mathbf{x}_j \prec \mathbf{x}_{j'} \tag{4}$$

gdzie I^+ jest zbiorem indeksów (j, j') dipoli z orientacją dodatnią $(\mathbf{x}_j, \mathbf{x}_{j'})$ ($j < j'$).

Definicja 2: Para $(\mathbf{x}_j, \mathbf{x}_{j'})$ $(j < j')$ wektorów cech $\mathbf{x}_j$ i $\mathbf{x}_{j'}$ tworzy dipol z orientacją ujemną $\{\mathbf{x}_j, \mathbf{x}_{j'}\}$ $((j, j') \in I^-)$ wtedy i tylko wtedy gdy $\mathbf{x}_{j'} \prec \mathbf{x}_j$.

$$(\forall (j, j') \in I^-) \quad \mathbf{x}_{j'} \prec \mathbf{x}_j \tag{5}$$

gdzie I^- jest zbiorem indeksów (j, j') dipoli z orientacją ujemną $(\mathbf{x}_j, \mathbf{x}_{j'})$ $(j < j')$.

Zgodnie z relacją (4) drugi wektor $\mathbf{x}_{j'}$ w parze $(\mathbf{x}_j, \mathbf{x}_{j'})$ następuje po $\mathbf{x}_j$. W przypadku relacji (5) pierwszy wektor $\mathbf{x}_j$ następuje po $\mathbf{x}_{j'}$.

Definicja 3: Uporządkowanie punktów y_j na prostej (2) jest zgodne z relacją „$\prec$" (1) pomiędzy wektorami cech $\mathbf{x}_j$ wtedy i tylko wtedy gdy spełnione są poniższe relacje.

$$\begin{aligned} &(\forall (j, j') \in I^+) \quad y_j < y_{j'} \\ &(\forall (j, j') \in I^-) \quad y_j > y_{j'} \end{aligned} \tag{6}$$

4. Zbiory C+ i C- i ich liniowa separowalność

Relacje (6) mogą być przedstawione w równoważnej poniższej postaci.

$$\begin{aligned} &(\forall (j, j') \in I^+) \quad \mathbf{w}^T (\mathbf{x}_{j'} - \mathbf{x}_j) > 0 \\ &(\forall (j, j') \in I^-) \quad \mathbf{w}^T (\mathbf{x}_{j'} - \mathbf{x}_j) < 0 \end{aligned} \tag{7}$$

Zdefiniujmy dwa zbiory C^+ i C^- składające się z wektorów $\mathbf{r}_{jj'}$ utworzonych z różnic arytmetycznych wektorów cech $\mathbf{x}_j$ i $\mathbf{x}_{j'}$ tworzących dipole $\{\mathbf{x}_j, \mathbf{x}_{j'}\}$.

$$\begin{aligned} C^+ &= \{\mathbf{r}_{jj'} = (\mathbf{x}_{j'} - \mathbf{x}_j) : (j, j') \in I^+\} \\ C^- &= \{\mathbf{r}_{jj'} = (\mathbf{x}_{j'} - \mathbf{x}_j) : (j, j') \in I^-\} \end{aligned} \tag{8}$$

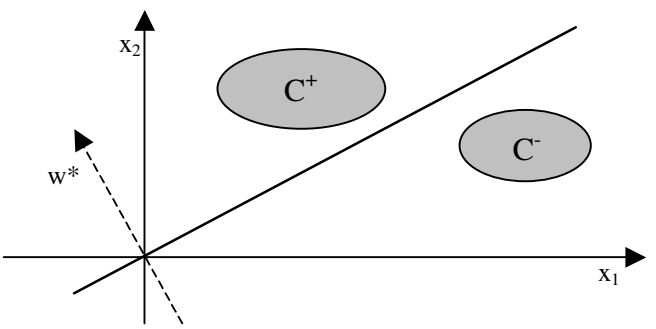
Liniowa separowalność zbiorów C^+ i C^- przez hiperpłaszczyznę $H(\mathbf{w})$ przechodzącą przez początek układu współrzędnych zapewnia spełnienie relacji (7).

$$H(\mathbf{w}) = \{\mathbf{x} : \mathbf{w}^T \mathbf{x} = 0\} \tag{9}$$

Uwzględniając definicje (8) zbiorów C^+ i C^-, relacje (7) można przedstawić w następujący sposób [5]:

$$(\exists \mathbf{w}^*) \quad \begin{array}{ll} (\forall (j, j') \in I^+) & (\mathbf{w}^*)^T \mathbf{r}_{jj'} \geq 1 \\ (\forall (j, j') \in I^-) & (\mathbf{w}^*)^T \mathbf{r}_{jj'} < -1 \end{array} \tag{10}$$

Zagadnienie regresji rangowej sprowadza się do wyznaczenia parametrów $\mathbf{w}^*$ optymalnej hiperpłaszczyzny $H(\mathbf{w}^*)$ separującej zbiory C^+ i C^-. Optymalność hiperpłaszczyzny $H(\mathbf{w}^*)$ rozumieć należy w ten sposób, że jest to hiperpłaszczyzna zapewniająca największy możliwy margines pomiędzy nią samą a elementami zbiorów C^+ i C^- [2]. Tak wyznaczona hiperpłaszczyzna nie powinna nadmiernie dopasowywać się do danych uczących i mieć dużą zdolność generalizacji.



Rys. 2. Zbiory C^+ i C^- w przestrzeni dwuwymiarowej i hiperpłaszczyzna separująca H(w*)

5. Funkcja kryterialna CPL

Hiperpłaszczyznę $H(\mathbf{w}^*)$ optymalnie rozdzielającą zbiory C^+ i C^- można wyznaczyć minimalizując regresyjno-rangową funkcję kryterialną $\Phi_r(\mathbf{w})$ [2].

$$\Phi_r(\mathbf{w}) = \sum_{(j,j') \in I^+} \varphi^+_{jj'}(\mathbf{w}) + \sum_{(j,j') \in I^-} \varphi^-_{jj'}(\mathbf{w}) \tag{11}$$

Funkcja $\Phi_r(\mathbf{w})$ jest sumą dodatnich $\varphi^+_{jj'}(\mathbf{w})$ i ujemnych $\varphi^-_{jj'}(\mathbf{w})$ funkcji kary.

$$(\forall (j, j') \in I^+)$$

$$\varphi^+_{jj'}(\mathbf{w}) = \begin{cases} 1 - \mathbf{w}^T \mathbf{r}_{jj'} & jezeli \quad \mathbf{w}^T \mathbf{r}_{jj'} < 1 \\ 0 & jezeli \quad \mathbf{w}^T \mathbf{r}_{jj'} \geq 1 \end{cases} \tag{12}$$

$$(\forall (j, j') \in I^-)$$

$$\varphi^-_{jj'}(\mathbf{w}) = \begin{cases} 1 - \mathbf{w}^T \mathbf{r}_{jj'} & jezeli \quad \mathbf{w}^T \mathbf{r}_{jj'} > -1 \\ 0 & jezeli \quad \mathbf{w}^T \mathbf{r}_{jj'} \leq -1 \end{cases} \tag{13}$$

Funkcja kryterialna $\Phi_r(\mathbf{w})$ jest funkcją wypukłą i odcinkowo-liniową (CPL). Algorytm wymiany rozwiązań bazowych, technika zbliżona do programowania liniowego, pozwala znaleźć minimum tego typu funkcji w sposób efektywny, nawet przy dużych, wysokowymiarowych zbiorach danych C^+ i C^- [2].

$$\Phi^* = \Phi(\mathbf{w}^*) = \min \Phi(\mathbf{w}) \geq 0 \tag{14}$$

Wektor parametrów $\mathbf{w}^*$ definiuje prostą $y_j = (\mathbf{w}^*)^T \mathbf{x}_j$ (2), będącą najlepszą z punktu widzenia zagadnienia regresji rangowej. Jeżeli wartość funkcji kryterialnej Φ^* jest równa 0, to model prawidłowo zachowuje wszystkie relacje (1) wektorów $\mathbf{x}_j$ i $\mathbf{x}_{j'}$ ze zbioru danych [4]. Gdy wartość funkcji Φ^* jest większa od 0, model uwzględnia największą możliwą liczbę relacji (1). Taka sytuacja oznacza, że niemożliwe jest uwzględnienie wszystkich relacji w przestrzeni cech o zadanym wymiarze.

6. Wyniki eksperymentów

Przy zastosowaniu opisanych wcześniej metod wykonane były eksperymenty na dwóch zbiorach danych. W pierwszym eksperymencie użyto części zbioru Primary Biliary Cirrhosis (PBC) z repozytorium UCI [1], w drugim danych z systemu komputerowego wspierania diagnostyki chorób wątroby „Hepar" [7].

6.1. Eksperyment 1

Zbiór PBC zawiera informacje opisujące pacjentów cierpiących na pierwotną żółciową marskość wątroby. Dane były zbierane w klinice Mayo w USA w latach

1974-1984. Każdy pacjent O_j w tym zbiorze opisany jest przez 17 atrybutów ($x_1, x_2, ..., x_{17}$). Ponadto dla każdego pacjenta przypisano czas przeżycia t_j i wskaźnik niepowodzenia δ_j ($\delta_j = \{0,1\}$). Czas przeżycia jest to liczba dni od momentu stwierdzenia choroby i rozpoczęcia obserwacji do momentu śmierci pacjenta, transplantacji wątroby lub zakończenia badań w lipcu 1986 roku. Wskaźnik niepowodzenia $\delta_j = 1$ oznacza, że obserwacja pacjenta była przerwana z powodu jego śmierci, $\delta_j = 0$ oznacza, że obserwację przerwano przed śmiercią pacjenta ze względu na transplantację lub zakończenie badań. Sytuację drugiego typu nazywamy cenzorowaniem lub ucinaniem [10].

Dane użyte do eksperymentu zawierały obserwacje $(\mathbf{x}_j, t_j, \delta_j)$ 30 pacjentów O_j. Wśród nich 18 obserwacji było uciętych ($\delta_j = 0$). Zbiory C^+ i C^- wektorów różnicowych $\mathbf{r}_{jj'} = (\mathbf{x}_{j'} - \mathbf{x}_j)$ zostały utworzone na podstawie 30 wektorów cech $\mathbf{x}_j$. Bazują one na wszystkich dipolach $\{\mathbf{x}_j, \mathbf{x}_{j'}\}$ ($j < j'$) zorientowanych zgodnie z następującą zasadą

$$\mathbf{x}_j \prec \mathbf{x}_{j'} \Leftrightarrow \delta_j = 1 \quad i \quad t_j < t_{j'} \tag{15}$$

Model rangowy otrzymany w wyniku minimalizacji funkcji $\Phi_r(\mathbf{w})$ (11) ma formę

$$\begin{aligned} y_j = {} & \text{-}0{,}0016x_{j1} - 34{,}5864x_{j6} + 1{,}0555x_{j7} - 0{,}0424x_{j8} - 8{,}5786x_{j9} + \\ & + 0.0915x_{j10} - 0{,}0005x_{j11} - 0{,}1322x_{j12} - 4{,}2516x_{j16} \end{aligned} \tag{16}$$

Wartość funkcji kryterialnej $\Phi(\mathbf{w}^*) = 0$, więc model (16) zachowuje wszystkie informacje o uporządkowaniu obiektów w zbiorze danych.

Otrzymany model ma zastosowanie do prognozowania czasu przeżycia dla obserwacji uciętych. Aby móc bezpośrednio wykorzystywać generowane przez niego wyniki zastosowane zostało dodatkowe przekształcenie skalujące:

$$y_j' = \alpha y_j + \beta \tag{17}$$

gdzie α i β są parametrami skalującymi wyznaczonymi poprzez minimalizację sumy różnic $\left| t_j - \alpha y_j - \beta \right|$ dla wszystkich nieuciętych obserwacji. W wyniku tej operacji otrzymano przekształcenie o następujących parametrach:

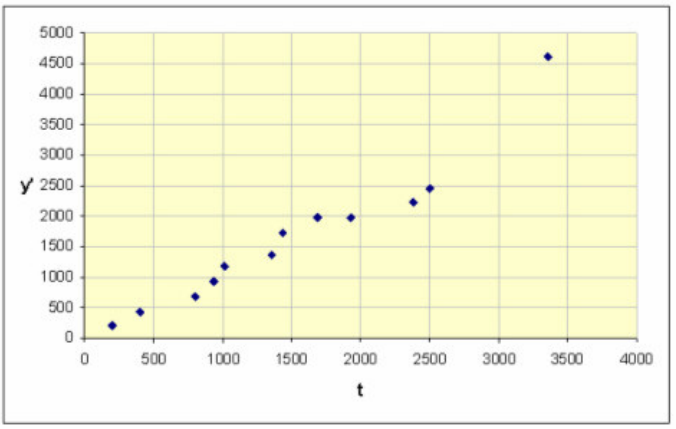
$$y_j' = 244{,}2\, y_j + 26617{,}75 \tag{18}$$

Wyznaczanie parametrów przekształcenia (17) opiera się na metryce L_1. Wybór metryki jest arbitralny i wynika z możliwości zastosowania w przypadku tej metryki minimalizacji zmodyfikowanej funkcji kryterialnej typu CPL (11), podobnie jak przy wyznaczaniu parametrów modelu (16).

Rysunki 3 i 4 obrazują wyniki eksperymentu. W górnej części tabeli na rysunku 3, nad kreską, umieszczono dane dotyczące obserwacji nieuciętych. Wartości otrzymane na podstawie modelu y_j' powinny być zbliżone do danych *a priori* rzeczywistych czasów przeżycia t_j. W dolnej części umieszczono dane dotyczące obserwacji uciętych. W tym przypadku wartości otrzymane na podstawie modelu y_j' są prognozą czasu przeżycia. W ten sposób w momencie przerwania badania z innego powodu niż śmierć pacjenta możemy określić przypuszczalną długość jego życia. Wykres na rysunku 4 przedstawia zależność rzeczywistego czasu przeżycia do czasu przeżycia wygenerowanego przez model (17), (18) dla obserwacji nieuciętych. Im bardziej zależność ta zbliżona jest do funkcji liniowej $y_j' = t_j$, tym lepszy jest model (17), (18).

tj	yj	yj'
198	-108,188	198,2751
400	-107,188	442,4751
799	-106,188	686,6751
930	-105,188	930,8751
1012	-104,188	1175,075
1360	-103,372	1374,248
1434	-101,914	1730,452
1690	-100,914	1974,652
1925	-100,914	1974,652
2386	-99,9136	2218,852
2503	-98,9136	2463,052
3358	-90,1398	4605,608
2272	-94,9121	3440,207
1615	-93,7519	3723,548
2255	-92,2662	4086,336
3099	-72,8111	8837,27
1592	-98,9656	2450,354
2318	-74,7733	8358,118
2294	-89,8265	4682,117
3069	-95,9002	3198,92
3297	-98,0101	2683,689
1701	-82,4123	6492,676
3255	-90,4258	4535,769
2944	-96,5152	3048,734
2468	-98,6327	2531,644
1614	-92,9708	3914,269
1702	-85,9536	5627,88
2033	-52,5628	13781,9
737	-77,7601	7628,724
1735	-76,7314	7879,937

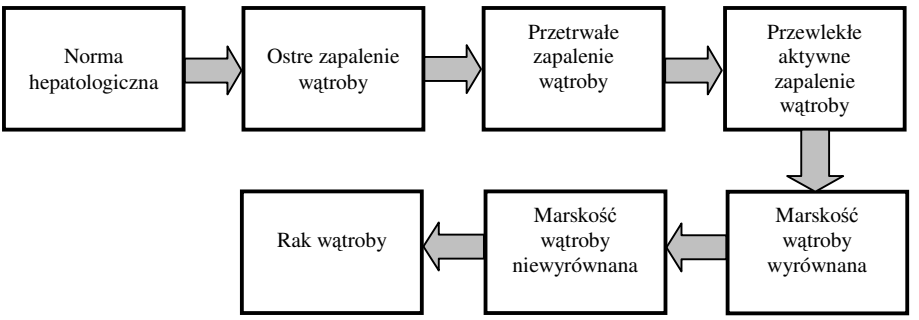
Rys. 3. Wyniki eksperymentu na zbiorze danych PBC



Rys. 4. Wyniki eksperymentu na zbiorze danych PBC, wykres zależności rzeczywistego czasu przeżycia do czasu przeżycia wygenerowanego przez model

6.2. Eksperyment 2

W drugim eksperymencie użyto danych zgromadzonych w systemie „Hepar". System ten zbudowano w Instytucie Biocybernetyki i Inżynierii Biomedycznej PAN. Baza danych systemu zawiera opisy pacjentów z przewlekłymi chorobami wątroby, leczonych w Klinice Gastroenterologii Instytutu Żywności i Żywienia w Warszawie. Opis każdego pacjenta składa się z około 200 atrybutów. Są to odpowiedzi uzyskane od pacjenta podczas wywiadu lekarskiego, symptomy z badania przedmiotowego oraz rezultaty diagnostycznych testów laboratoryjnych. Z powodu braków w danych do eksperymentu wybrano 62 atrybuty, których wartości są ustalone dla wszystkich pacjentów. Czynnikiem czasowym jest stopień zaawansowania choroby η. Uporządkowanie stopni zaawansowania choroby przedstawia rysunek 5 (Norma hepatologiczna $\eta = 1$, Ostre zapalenie wątroby $\eta = 2$, Przetrwałe zapalenie wątroby $\eta = 3$, Przewlekłe aktywne zapalenie wątroby $\eta = 4$, Marskość wątroby wyrównana $\eta = 5$, Marskość wątroby niewyrównana $\eta = 6$, Rak wątroby $\eta = 7$) [6].



Rys. 5. Kolejne etapy chorób wątroby według systemu „Hepar"

Dane użyte w eksperymencie zawierały opisy 272 pacjentów O_j. Zbiory C^+ i C^- wektorów różnicowych $\mathbf{r}_{jj'} = (\mathbf{x}_{j'} - \mathbf{x}_j)$ zbudowano na podstawie wszystkich możliwych 28769 dipoli $\{\mathbf{x}_j, \mathbf{x}_{j'}\}$ ($j < j'$). Dipol tworzą dwa wektory $\mathbf{x}_j$ i $\mathbf{x}_{j'}$ opisujące pacjentów z różnym stopniem zaawansowania choroby. Otrzymany w wyniku minimalizacji funkcji kryterialnej $\Phi_r(\mathbf{w})$ (11) model rangowy ma następującą postać.

$$\begin{aligned} y_j = [&0{,}721\ -0{,}186\ -0{,}16\ 0{,}89\ 0{,}249\ -0{,}081\ 0{,}051\ -0{,}274\ -0{,}511 \\ &-0{,}3\ 0{,}358\ 0{,}627\ 0{,}038\ -0{,}406\ 0{,}393\ 0{,}55\ -0{,}09\ 0{,}297\ -0{,}367\ 0{,}064 \\ &-0{,}338\ -0{,}156\ 0{,}048\ 0{,}147\ 0{,}566\ 0{,}602\ -0{,}354\ -2{,}553\ -0{,}1\ 0{,}42 \\ &-0{,}973\ 0{,}968\ -0{,}727\ -0{,}412\ 0{,}115\ 0{,}09\ -0{,}697\ 0{,}196\ 0{,}638\ -2{,}468 \\ &0{,}895\ -0{,}275\ -0{,}58\ 0{,}11\ -0{,}041{,}125\ -0{,}064\ 0{,}082\ -0{,}196\ 0{,}259 \\ &-0{,}504\ 0{,}292\ 0{,}68\ -0{,}109\ 0{,}767\ 0{,}333\ -0{,}042\ 1{,}048\ 0{,}374\ -0{,}448]^{\mathrm{T}} \cdot \\ &\cdot [x_{j1},...,x_{j51},x_{j54},...,x_{j62}] \end{aligned} \tag{19}$$

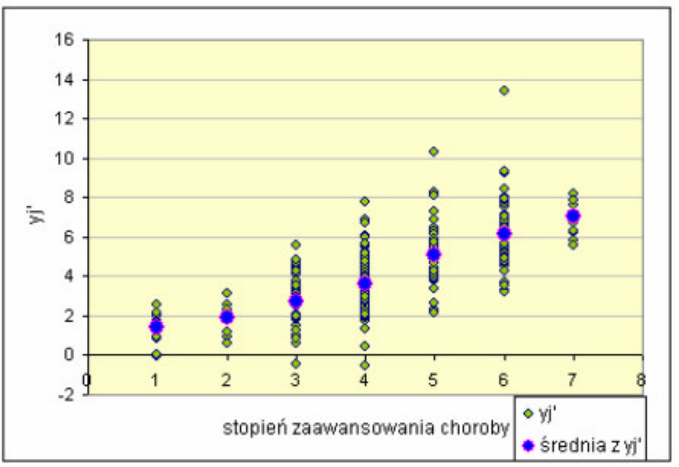
Podobnie jak w opisanym wcześniej eksperymencie 1 zastosowano przekształcenie skalujące. Parametry α i β wyznaczono poprzez minimalizację sumy różnic $|\eta - \alpha y_j - \beta|$ dla wszystkich obiektów ze zbioru danych.

$$y_j' = 0{,}58539\, y_j + 3{,}50484 \tag{20}$$

Wartość funkcji kryterialnej $\Phi(\mathbf{w}^*) > 0$. Model (19) nie zachowuje wszystkich informacji o uporządkowaniu obiektów O_j. Niedoskonałość modelu wynika z dużej liczby obiektów w stosunku do liczby opisujących je atrybutów. W zadanej

przestrzeni atrybutów nie ma możliwości znalezienia hiperpłaszczyzny liniowo separującej zbiory C^+ i C^-. Jednak, jak widać na rysunku 6, w ujęciu średnich wartości y_j' tendencja jest zachowana.

η	Liczność klasy	$\overline{y_j'}$
1	16	1,413581
2	8	1,928993
3	44	2,751939
4	95	3,649423
5	38	5,081494
6	60	6,143924
7	11	7,030914



Rys. 6. Wyniki eksperymentu na zbiorze danych „Hepar"

Na podstawie otrzymanego modelu (19), (20) możliwe jest określenie z pewnym prawdopodobieństwem stopnia zaawansowania choroby w przypadku nowego pacjenta.

7. Podsumowanie

W artykule przedstawiono metodę budowy modelu rangowego, opierając się na minimalizacji wypukłej i odcinkowo linowej (CPL) funkcji kryterialnej (11). Zagadnienie regresji rangowej zostało sprowadzone do problemu znalezienia hiperpłaszczyzny (9) optymalnie rozdzielającej zbiory C^+ i C^-, zbudowane na podstawie dipoli utworzonych z wektorów cech, dla których określony jest *a priori* porządek czasowy. Minimalizacja funkcji kryterialnej (11) może być efektywnie przeprowadzona poprzez zastosowanie algorytmu wymiany rozwiązań bazowych [2], techniki zbliżonej do programowania liniowego.

Transformację rangową stosuje się między innymi w celu uzyskania poprawy procesu wspomagania podejmowania decyzji klasyfikacyjnych. Model rangowy zbudowany na podstawie zbioru uczącego pozwala na późniejsze klasyfikowanie obiektów niebiorących udziału w procesie uczenia. Takie zastosowanie modelu rangowego może być rozszerzeniem opisanego eksperymentu 2.

Model rangowy może być także używany przy prognozowaniu nieznanych wartości określonych parametrów, np. czasu życia dla obserwacji uciętych w analizie przeżyć. To zastosowanie jest przedmiotem zaprezentowanego eksperymentu 1.

Literatura:

[1] Blake, C., Merz, C.: *UCI Repository of machine learning databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html] Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[2] Bobrowski, L.: *Eksploracja danych oparta na wypukłych i odcinkowo-liniowych funkcjach kryterialnych*, Białystok: Wydaw. Politechniki Białostockiej, 2005.

[3] Bobrowski L.: *Ranked modeling with feature selection based on the CPL criterion functions*, s.218-227 w: Machine learning and data mining in pattern recognition, eds. Petra Perner, Atsushi Imiya, Lecture Notes in Computer Science, vol.3587, 2005.

[4] Bobrowski L., Łukaszuk T.: *Ranked linear modeling in survival analysis*, s.61-67 w: [Sixth International Seminar] Statistics and Clinical Practice, Warsaw, June, 2005, ed by L. Bobrowski, J. Doroszewski, C. Kulikowski, N.Victor, Lecture Notes of the ICB Seminars 70, Warsaw, 2005.

[5] Bobrowski L, Łukaszuk T.: *Selection of the linearly separable feature subsets*, s.544-549 w: Artificial intelligence and soft computing : ICAISC'2004, eds. Leszek Rutkowski, Jörg Siekmann, Ryszard Tadusiewicz, Lotfi A. Zadeh, Lecture Notes in Computer Science, vol.3070, 2004.

[6] Bobrowski L., Łukaszuk T, Wasyluk H.: *Ranked modeling of liver diseases sequence*, wysłane do European Journal of Biomedical Informatics.

[7] Bobrowski L., Wasyluk H.: *Diagnosis supporting rules of the HEPAR system*, s.1309-1313 w: MEDINFO 2001: Proceedings of the 10th World Congress on Medical Informatics. P.2, London, September 2-5, 2001, ed. by V. L. Patel, R. Rogers, R. Haux, Amsterdam: IOS Press, 2001.

[8] Duda, O.R., Heart, P.E., Stork D.G.: *Pattern Classification*, Wydanie drugie, zmienione, John Wiley & Sons, 2001.

[9] K. Fukunaga: *Statistical Pattern Recognition*, Academic Press, Inc., San Diego, 1990.

[10] Klein J. P., Moeschberger M. L.: *Survival Analysis, Techniques for Censored and Truncated Data*, Springer, NY 1997.

TEMPORALITY IN RANKED MODELS

Abstract: A known temporal order between selected objects in a data set is given. We assume the ranked model is such a linear transformation, which preserve in the most possible manner the a priori knowledge of the order between objects. The procedure of the ranked models design which is based on the minimisation of the convex and piecewise linear (CPL) criterion functions is presented in the paper. The task of the ranked model design is boiled down to the problem of searching an optimal hyperplane separated the sets constructed on the basis of the elements created from the arithmetic substractions of the vectors – the pairs with the given temporal order.

Keywords: ranked model, convex and piecewise linear (CPL) criterion functions, linear separability of data sets

Artykuł zrealizowano w ramach projektu badawczego „Temporalna reprezentacja wiedzy i jej implementacja w informatycznych systemach wspomagania postępowania medycznego”, nr 3 T11F 011 30.

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2007
Informatyka – Zeszyt 2

Joanna Olbryś[1]

SIEĆ BAYESOWSKA JAKO NARZĘDZIE POZYSKIWANIA WIEDZY Z EKONOMICZNEJ BAZY DANYCH

Streszczenie: Proces decyzyjny w inwestowaniu rozpoczyna się od percepcji i przetwarzania napływających informacji. Podłoże decyzji stanowią przekonania dotyczące prawdopodobieństwa zajścia określonego zdarzenia. Jednostki racjonalne posługują się narzędziami teorii prawdopodobieństwa i statystyki, rozumując zgodnie z prawem Bayesa, czyli aktualizując wyobrażenia o prawdopodobieństwie zdarzenia wraz z ujawnianiem wszelkich nowych informacji, zarówno ilościowych, jak i jakościowych. Wydaje się zatem, że bardzo dobrym narzędziem wspomagającym decyzje inwestycyjne może być odpowiednio skonstruowany model sieci bayesowskiej (*Bayesian Network*). W artykule postawiono za cel główny prezentację możliwości zastosowania modelu sieci bayesowskiej do pozyskiwania wiedzy z ekonomicznej bazy danych, z uwzględnieniem informacji jakościowych oraz preferencji i subiektywnych ocen analityka finansowego, podejmującego decyzje w warunkach niepewności.

Słowa kluczowe: sieć bayesowska, system wspomagający decyzje inwestycyjne, diagram wpływu

1. Wstęp

Proces decyzyjny w inwestowaniu rozpoczyna się od percepcji i przetwarzania napływających informacji. Podłoże decyzji stanowią przekonania dotyczące prawdopodobieństwa zajścia określonego zdarzenia. Przekonania zaś to rezultat procesu wnioskowania o nieznanym, którego przesłankami są napływające informacje [5]. W ekonomii i finansach przyjmuje się, że jednostka racjonalna (inwestor):

- przetwarza wszelkie możliwe informacje, odpowiednio uaktualniając swoje przekonania,

[1] Wydział Informatyki, Politechnika Białostocka, Białystok

- na podstawie przekonań formułuje preferencje i podejmuje decyzje w taki sposób, aby maksymalizować swoją oczekiwaną użyteczność.

Podejście to zakłada, że jednostki racjonalne posługują się narzędziami teorii prawdopodobieństwa i statystyki, rozumując zgodnie z prawem Bayesa, czyli aktualizując wyobrażenia o prawdopodobieństwie zdarzenia wraz z ujawnianiem wszelkich nowych informacji, zarówno ilościowych, jak i jakościowych. Założenie bayesowskiego myślenia nie oznacza, że każdy osąd poprzedzony jest mozolnymi obliczeniami. Chodzi raczej o przeciętny rezultat wnioskowania, który jest zgodny z zasadami prawdopodobieństwa [5].

Biorąc pod uwagę powyższe rozważania można mniemać, że bardzo dobrym narzędziem wspomagającym decyzje inwestycyjne może być odpowiednio skonstruowana sieć bayesowska, która umożliwia uwzględnienie w procesie decyzyjnym informacji o zróżnicowanym charakterze. Pozwala również na wprowadzanie nowych informacji w postaci obserwacji (*evidence*) oraz na modyfikacje modelu zgodnie z preferencjami i subiektywnymi ocenami inwestora w warunkach niepewności.

Sieci bayesowskie (*Bayesian Networks*) [19], nazywane również probabilistycznymi modelami graficznymi, sieciami przekonań lub sieciami przyczynowo – skutkowymi, stały się ostatnio popularnym narzędziem do reprezentacji wiedzy w warunkach niepewności. Sieć bayesowska jest acyklicznym grafem skierowanym i zawiera część jakościową, która stanowi zbiór zmiennych – węzłów grafu wraz z probabilistycznymi zależnościami pomiędzy nimi oraz część ilościową, reprezentującą rozkład prawdopodobieństwa łącznego dla tych zmiennych. Z punktu widzenia inżynierii wiedzy, sieć bayesowska może odzwierciedlać strukturę przyczynowo – skutkową, która pozwala pełniej zrozumieć modelowany problem, zarówno ekspertom jak i użytkownikom systemu [6]. Głównym etapem w budowaniu sieci bayesowskiej jest określenie jej struktury oraz parametryzacja. Podstawową zaletą sieci bayesowskich jest to, że pozwalają one na zintegrowanie wiedzy eksperta z danymi statystycznymi. Jeżeli odpowiednia liczba danych jest dostępna, sieci bayesowskie, zarówno ich struktura jak i parametry, mogą być nauczone z wykorzystaniem danych.

Wnioskowanie w sieci bayesowskiej sprowadza się do wyznaczenia rozkładu prawdopodobieństwa *a posteriori,* pod warunkiem zaobserwowania wartości zmiennych modelu. Rozkład tego prawdopodobieństwa może być bezpośrednio wykorzystany we wspomaganiu decyzji inwestycyjnych.

Głównym celem, jaki w artykule stawia sobie autor, jest prezentacja możliwości zastosowania modelu sieci bayesowskiej do pozyskiwania wiedzy z ekonomicznej bazy danych, przy uwzględnieniu informacji jakościowych oraz preferencji i subiektywnych ocen analityka finansowego, podejmującego decyzje w warunkach niepewności. Dane ekonomiczne wykazują dużą zmienność, zatem ekono-

miczna baza danych powinna być często aktualizowana, aby jak najpełniej odzwierciedlała bieżącą sytuację rynkową.

2. Sieć bayesowska jako narzędzie pozyskiwania wiedzy z ekonomicznej bazy danych

W ekonomii i finansach definicja jednostki racjonalnej (*homo oeconomicus*) obejmuje dwie cechy tej jednostki: konsekwencję działania oraz dążenie do podnoszenia poziomu osobistego dobrobytu [5]. Prawie wszystkie decyzje inwestycyjne podejmowane są na podstawie oszacowanych wartości prawdopodobieństwa realizacji określonego scenariusza zdarzeń [10]. Prawdopodobieństwo jest miarą stopnia przekonania podmiotu co do prawdziwości zajścia określonego zdarzenia. Z formalnego punktu widzenia podejście bayesowskie polega na traktowaniu wszystkich wielkości, których wartości nie są znane przed dokonaniem obserwacji, jako zmiennych losowych i na konsekwentnym stosowaniu we wnioskowaniu prostych reguł rachunku prawdopodobieństwa [18]. W szczególności stosuje się dwie zasady probabilistyki: warunkowanie (względem danych statystycznych) oraz wyznaczanie rozkładów brzegowych. Dane historyczne służą do uzyskania rozkładu *a priori*. Przyjęcie rozkładu *a priori* jest cechą wyróżniającą podejście bayesowskie. W problemach decyzyjnych rozkład *a priori* ma możliwie najpełniej odzwierciedlać całą wstępną wiedzę o zmiennej [18]. Nowe informacje ilościowe, jakościowe, opinie ekspertów są odpowiednie do rozwinięcia rozkładu *a posteriori*. Proces aktualizacji informacji przebiega zgodnie z regułą Bayesa:

$$P(A|I) = \frac{P(A) \cdot P(I|A)}{P(I)} \tag{1}$$

gdzie:

$P(A|I)$ - prawdopodobieństwo *a posteriori* zdarzenia A po uzyskaniu informacji I,

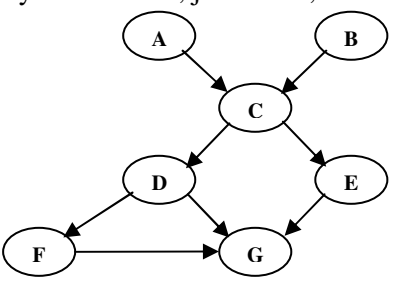
$P(A)$ - prawdopodobieństwo *a priori* (bazowe) zdarzenia A przed pojawieniem się informacji I,

$P(I|A)$ - prawdopodobieństwo pojawienia się informacji I, jeśli A jest prawdą,

$P(I)$ - prawdopodobieństwo całkowite zdarzenia I.

Sieci bayesowskie są szczególnym przypadkiem graficznych modeli probabilistycznych. Zwykle są one wykorzystywane do modelowania złożonych systemów o niepewnych lub niekompletnych danych. Sieć bayesowska jest skierowanym, acyklicznym grafem, którego topologia opisuje zależności (lub ich brak) między

zmiennymi modelu. Pojawienie się łuku łączącego dwa węzły jest interpretowane jako dowód istnienia bezpośredniego związku przyczynowo–skutkowego między węzłami. Zgodnie z definicją sieci bayesowskiej według Jensena [14] graf sieci zawiera zarówno węzły z rodzicami, jak i takie, które nie mają rodziców.



Rys. 1. Acykliczny graf skierowany. Prawdopodobieństwa, które należy wyznaczyć to: $P(A), P(B), P(C|A,B), P(E|C), P(D|C), P(F|D), P(G|D,E,F)$

Na rys.1 węzły, które nie mają rodziców, to są węzły A i B. Z nimi związane są brzegowe rozkłady prawdopodobieństwa. Pozostałe węzły C, D, E, F oraz G posiadają rodziców i odpowiadają im warunkowe rozkłady prawdopodobieństwa.

Bayesowska sieć zależności składa się z dwóch podstawowych części: jakościowej, czyli graficznej struktury zależności w modelu, oraz ilościowej, reprezentowanej przez rozkłady prawdopodobieństwa związane z grafem.

Trzy najważniejsze etapy tworzenia sieci bayesowskiej to:

- graficzna prezentacja zależności w modelu;
- specyfikacja numerycznych zależności między zmiennymi;

Rezultaty uzyskane w wyniku działania sieci w wysokim stopniu zależą od postaci graficznej modelu. Można stwierdzić, że najlepsze efekty uzyskujemy na ogół z bardzo dobrej reprezentacji graficznej problemu decyzyjnego, podczas gdy rozkłady prawdopodobieństwa związane z wierzchołkami grafu mogą być oszacowane w przybliżeniu [6]. Struktura sieci może mieć postać drzewa, polidrzewa lub grafu. Na przykład, algorytm Chow-Liu [4] nie buduje sieci przyczynowo-skutkowej, a jedynie niezorientowane drzewo zależności. Jeżeli sieć Bayesa określonego rozkładu ma postać drzewa, to ten algorytm powinien poprawnie odtworzyć jego kształt. Z kolei algorytm Pearla [19] jest rozszerzeniem algorytmu Chow-Liu. Otrzymana struktura nie jest drzewem skierowanym, ale ma postać polidrzewa. Najbardziej kosztowne natomiast jest uczenie grafu o dużej liczbie wierzchołków. Strukturę sieci można zbudować z wykorzystaniem wiedzy eksperta lub na podstawie danych statystycznych, w sposób automatyczny. Jest to zada-

nie złożone obliczeniowo, lecz wykonalne za pomocą szerokiej gamy skutecznych algorytmów (m.in. [3], [7], [8], [9], [11], [14]).

Tym, co odróżnia sieci bayesowskie od innych metod reprezentowania wiedzy, jest wielość możliwości wnioskowania. Skupiając się na opisie jakościowym (czyli graficznej strukturze modelu), możemy identyfikować warunkowe zależności między zmiennymi. Uwzględniając opisy ilościowe (parametryczne modele przypisane węzłom) możemy, po wprowadzeniu do modelu nowej obserwacji (*evidence*), uzyskać rozkłady prawdopodobieństwa *a posteriori* pojedynczych zmiennych modelu lub rozkład łączny zbioru zmiennych. Możemy aktualizować, na podstawie opinii ekspertów, prawdopodobieństwa stanów zmiennych lub wartości zmiennych. Możemy też znaleźć najbardziej prawdopodobną (w świetle dostępnych obserwacji) konfigurację zmiennych nieobserwowalnych, jak również oszacować prawdopodobieństwo hipotezy, biorąc pod uwagę konkretne obserwacje. W pracy [15] podano następujące rodzaje wnioskowania w sieciach bayesowskich:

- O wiarygodności poszczególnych hipotez dla zadanych obserwacji.
- Poszukiwanie uzasadnienia dla zadanej hipotezy i obserwacji.
- Znalezienie prawdopodobieństwa prawdziwości wyrażenia logicznego przedstawionego w postaci koniunkcji wyrażeń elementarnych atrybut=wartość.
- Znalezienie odpowiedzi na kwerendę złożoną.
- O brzegowej i warunkowej niezależności zmiennych.

Podsumowując należy stwierdzić, że sieć bayesowska wydaje się bardzo użytecznym narzędziem wspomagającym zarządzanie portfelem papierów wartościowych ([6], [20]). Kombinacja czynników makroekonomicznych i mikroekonomicznych, tworzących odpowiednie węzły sieci, pozwala oszacować wartości oraz ryzyko portfela. Umożliwia też aktualizację wyceny portfela poprzez uwzględnienie informacji jakościowych, napływających z rynku oraz preferencji i subiektywnych ocen analityka finansowego - eksperta, podejmującego decyzje w warunkach niepewności.

3. Model diagnostyczny wspomagający decyzje inwestycyjne w warunkach niepewności – etap wstępny

Tradycyjne modele wyceny dóbr kapitałowych, takie jak model CAPM (*Capital Asset Pricing Model*), czy też model APT (*Arbitrage Pricing Theory*) opisują relacje między zmiennymi makroekonomicznymi, mikroekonomicznymi oraz stopami zwrotu z papierów wartościowych i mogą być wykorzystane do modelowania relacji między zmiennymi w sieci bayesowskiej ([6], [20]).

Przykładowy model sieci bayesowskiej, zaprezentowany w dalszej części pracy, zbudowany został na podstawie ekonomicznej bazy danych z rynku polskiego i w założeniu ma być częścią większego modelu wspomagającego proces wyceny oraz szacowania wartości zagrożonej Value-at-Risk portfeli akcji na Giełdzie Papierów Wartościowych w Warszawie. W modelu zaproponowano następujące zmienne makroekonomiczne (na podstawie [1], [2], [6]): inflacja, stopa bezrobocia, deficyt budżetowy, produkcja przemysłowa sprzedana, saldo handlu zagranicznego, średnia rentowność bonów skarbowych 52-tygodniowych. Substytutem portfela rynkowego jest Warszawski Indeks Giełdowy (WIG). Sektory na giełdzie reprezentują indeksy branżowe: WIG–BANKI, WIG-BUDOW (Budownictwo), WIG-INFO (Informatyka), WIG-SPOZYW (Przemysł Spożywczy), WIG-TELKOM (Telekomunikacja). Pozostałe indeksy branżowe, czyli WIG-MEDIA oraz WIG-PALIWA nie były uwzględnione w modelu z powodu zbyt krótkiego okresu funkcjonowania na giełdzie (WIG-MEDIA od 2004r., natomiast WIG-PALIWA od 2005r.). Próba statystyczna obejmuje dane miesięczne z okresu marzec 1998 – czerwiec 2006 (po 100 obserwacji dla każdej zmiennej). Szeregi czasowe zmiennych makroekonomicznych pochodzą z Biuletynów Statystycznych wydawanych przez Główny Urząd Statystyczny (www.stat.gov.pl) oraz Biuletynów Informacyjnych publikowanych przez Narodowy Bank Polski (www.nbp.pl). Źródłem danych dotyczących indeksów giełdowych były portale finansowe, m.in. http://bossa.pl, www.bankier.pl, www.wp.pl, www.gpw.com.pl.

Tabela 1

Symbole i definicje zmiennych makroekonomicznych oraz indeksów

l.p.	Symbol zmiennej	Nazwa zmiennej	Opis zmiennej
1.	ZPI_t	Zmiana poziomu inflacji r / r [2]	Indeks cen towarów i usług konsumpcyjnych r / r (PI_t), pomniejszony o 100
2.	ZPB_t	Zmiana poziomu bezrobocia [3]	Miesięczna stopa wzrostu poziomu bezrobocia (PB_t) $ZPB_t = \frac{PB_t - PB_{t-1}}{PB_{t-1}} \cdot 100$
3.	ZDB_t	Zmiana poziomu deficytu budżetowego [4]	Miesięczna stopa wzrostu deficytu budżetowego (DB_t) $ZDB_t = \frac{DB_t - DB_{t-1}}{DB_{t-1}} \cdot 100$

[2] Analogiczny miesiąc poprzedniego roku = 100

[3] Stopa bezrobocia jako zarejestrowani bezrobotni w stosunku do cywilnej ludności aktywnej zawodowo (w Polsce: 16-60 lat dla kobiet oraz 16-65 lat dla mężczyzn)

[4] Deficyt budżetowy = dochody budżetu państwa – wydatki budżetu państwa

4.	ZPP_t	Zmiana poziomu produkcji przemysłowej sprzedanej [5]	Indeks łańcuchowy produkcji przemysłowej sprzedanej m/m (PP_t), pomniejszony o 100
5.	ZSH_t	Zmiana deficytu bilansu handlu zagranicznego [6]	Miesięczna stopa wzrostu salda handlu zagranicznego (SH_t) $ZSH_t = \frac{SH_t - SH_{t-1}}{SH_{t-1}} \cdot 100$
6.	ZBS_t	Zmiana poziomu rentowności bonów skarbowych 52-tygodniowych	Miesięczna stopa wzrostu średniej rentowności bonów skarbowych 52- tygodniowych (BS_t) $ZBS_t = \frac{BS_t - BS_{t-1}}{BS_{t-1}} \cdot 100$
7.	$ZWIG_t$	Zmiana poziomu Warszawskiego Indeksu Giełdowego	Miesięczna stopa wzrostu indeksu giełdowego (WIG_t) $ZWIG_t = \frac{WIG_t - WIG_{t-1}}{WIG_{t-1}} \cdot 100$
8.	$ZWBANKI_t$	Zmiana poziomu indeksu branżowego $WIG-BANKI$	Miesięczna stopa wzrostu indeksu branżowego sektora bankowego ($WBANKI_t$) $ZWBANKI_t = \frac{WBANKI_t - WBANKI_{t-1}}{WBANKI_{t-1}} \cdot 100$
9.	$ZWBUDOW_t$	Zmiana poziomu indeksu branżowego $WIG-BUDOW$	Miesięczna stopa wzrostu indeksu branżowego sektora budownictwo ($WBUDOW_t$) $ZWBUDOW_t = \frac{WBUDOW_t - WBUDOW_{t-1}}{WBUDOW_{t-1}} \cdot 100$
10.	$ZWINFO_t$	Zmiana poziomu indeksu branżowego $WIG-INFO$	Miesięczna stopa wzrostu indeksu branżowego sektora informatyka ($WINFO_t$) $ZWINFO_t = \frac{WINFO_t - WINFO_{t-1}}{WINFO_{t-1}} \cdot 100$
11.	$ZWSPOZYW_t$	Zmiana poziomu indeksu branżowego $WIG-SPOZYW$	Miesięczna stopa wzrostu indeksu branżowego sektora przemysł spożywczy ($WSPOZYW_t$) $ZWSPOZYW_t = \frac{WSPOZYW_t - WSPOZYW_{t-1}}{WSPOZYW_{t-1}} \cdot 100$

[5] Dynamika produkcji sprzedanej jest indeksem mierzącym comiesięczne zmiany wartości produkcji sprzedanej w cenach bazowych

[6] Saldo handlu zagranicznego = eksport - import

12.	$ZWTELKOM_t$	Zmiana poziomu indeksu branżowego $WIG-TELKOM$	Miesięczna stopa wzrostu indeksu branżowego sektora telekomunikacja ($WTELKOM_t$) $ZWTELKOM_t = \frac{WTELKOM_t - WTELKOM_{t-1}}{WTELKOM_{t-1}} \cdot 100$

Źródło: opracowanie własne

Tabele 3 oraz 4 (Dodatek) zawierają wartości podstawowych statystyk zmiennych makroekonomicznych oraz indeksów giełdowych, natomiast tabela 5 – współczynniki korelacji miesięcznych stóp zwrotu indeksów branżowych z indeksem WIG. Należy zauważyć, że stopy zwrotu z indeksu WIG są dość silnie skorelowane dodatnio ze stopami zwrotu wszystkich indeksów branżowych.

Po uporządkowaniu wartości zmiennych w postaci bazy danych zostały one poddane procesowi dyskretyzacji[7]. Utworzono dla każdej zmiennej przedziały o jednakowej liczebności, tworząc trzy stany reprezentujące niskie (*Low*), średnie (*Medium*) oraz wysokie (*High*) wartości, z prawdopodobieństwami odpowiednio: 34%, 33%, 33%. Jedynie dla zmiennej ZPB odpowiednie prawdopodobieństwa wynoszą: 35%, 32%, 33%. Powstałe w ten sposób rozkłady brzegowe przedstawia tabela 2.

Tabela 2

Brzegowe rozkłady prawdopodobieństwa zmiennych makroekonomicznych [%]

Stan	ZPI	ZPB	ZDB	ZPP	ZSH	ZBS
Low	[0,30; 1,75]	[-4,07; -0,52]	[-96,50; 6,89]	[-19; -1,80]	[-1480; -35,72]	[-14,76; -3,46]
Medium	[1,75; 6,45]	[-0,52; 0,94]	[6,89; 22,24]	[-1,80; 2,75]	[-35,72; 14,60]	[-3,46; 0,70]
High	[6,45; 13,90]	[0,94; 14,86]	[22,24; 550,10]	[2,75; 21,80]	[14,60; 3000]	[0,70; 14,57]

Źródło: opracowanie własne

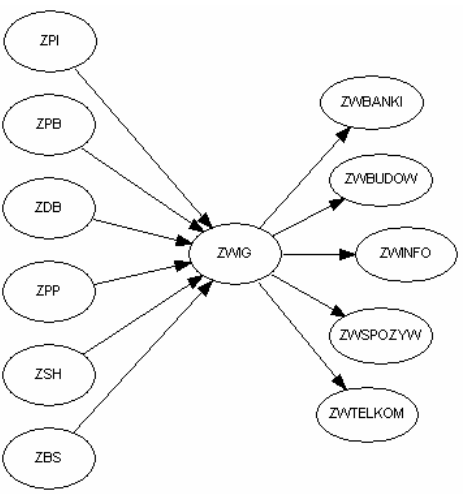
Zakładamy, że w budowanym modelu zmienne makroekonomiczne, tworzące rozkłady brzegowe, są wzajemnie niezależne [6], natomiast wartość zmiennej ZWIG, reprezentującej indeks giełdowy (substytut portfela rynkowego) zależy od wszystkich zmiennych makroekonomicznych. Jest to zgodne z przyjętymi w teorii portfela założeniami, że poziom ważonego wartościami rynkowymi indeksu giełdowego odzwierciedla już efekt oddziaływania zmiennych makroekonomicznych. Oznacza to, że węzły sieci bayesowskiej, odpowiadające zmiennym makroekonomicznym ZPI, ZPB, ZDB, ZPP, ZSH i ZBS są rodzicami węzła

[7] Z wykorzystaniem narzędzia **GeNIe** (http://genie.sis.pitt.edu)

ZWIG, natomiast same nie mają rodziców. Z węzłem ZWIG związany jest warunkowy rozkład prawdopodobieństwa $P(ZWIG / ZPI, ZPB, ZDB, ZPP, ZSH, ZBS)$ w $729 = 3^6$ możliwych stanach. Z kolei wartości indeksów branżowych są uzależnione od głównego indeksu giełdowego WIG (Tabela 5), na który mają wpływ, zdyskontowane przez inwestorów, informacje płynące z makroekonomicznego otoczenia rynku papierów wartościowych. Zatem w tworzonej sieci bayesowskiej węzłom ZWBANKI, ZWBUDOW, ZWINFO, ZWSPOZYW oraz ZWTELKOM powinny odpowiadać warunkowe rozkłady prawdopodobieństwa:

$$P(ZWBANKI / ZWIG),\ P(ZWBUDOW / ZWIG),\ P(ZWINFO / ZWIG),$$
$$P(ZWSPOZYW / ZWIG),\ P(ZWTELKOM / ZWIG)$$

Pierwsza część modelu sieci bayesowskiej, zawierająca zmienne makroekonomiczne oraz indeksy giełdowe (rys. 2), była utworzona z wykorzystaniem modułu uczącego narzędzia **GeNIe**. Metoda *Greedy Trick Thinning*, kryterium oceny modeli *BDeu (Bayesian marginal likelihood with uniform Dirichlet prior)* [3] umożliwiła automatyczne zbudowanie grafu zgodnego z oczekiwaniami odnośnie do związków przyczynowo - skutkowych pomiędzy zmiennymi wchodzącymi w skład modelu (uczenie z wykorzystaniem wiedzy wstępnej). Wybrane tablice warunkowych rozkładów prawdopodobieństwa zmiennych reprezentujących indeksy branżowe WIG–BANKI oraz WIG–INFO przedstawiają rysunki 3 oraz 4.



Rys. 2. Graficzny model zależności między zmiennymi makroekonomicznymi oraz zmiennymi reprezentującymi indeks główny i indeksy branżowe (opis zmiennych w Tabeli 1)

ZWIG	Low	Medium	High
Low	0.814103	0.184818	0.00660066
Medium	0.179487	0.481848	0.333333
High	0.00641026	0.333333	0.660066

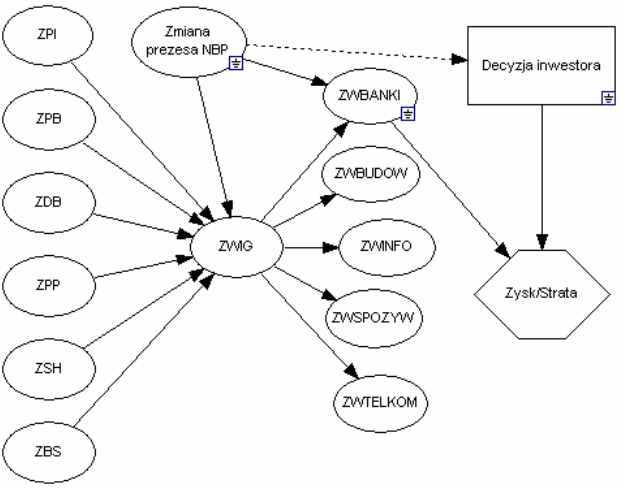
Rys. 3. Tablica warunkowego rozkładu prawdopodobieństwa $P(ZWBANKI \mid ZWIG)$

ZWIG	Low	Medium	High
Low	0.785256	0.184818	0.0363036
Medium	0.179487	0.630363	0.184818
High	0.0352564	0.184818	0.778878

Rys. 4. Tablica warunkowego rozkładu prawdopodobieństwa $P(ZWINFO \mid ZWIG)$

Kolejnym etapem tworzenia systemu wspomagającego decyzje inwestycyjne na polskim rynku będzie rozbudowanie sieci z rys. 2 poprzez włączenie do modelu zmiennych mikroekonomicznych, specyfikujących spółki wchodzące w skład poszczególnych indeksów branżowych. W przyszłości głównym celem wykorzystania sieci będzie wycena portfeli akcyjnych oraz szacowanie wartości zagrożonej Value-at-Risk portfeli [16]. Ze względu na ograniczoną objętość artykułu będzie to tematem kolejnego opracowania.

Równie istotnym zagadnieniem, z punktu widzenia inwestora, jest możliwość rozbudowania sieci i stworzenie diagramu wpływu (*Influence Diagram* [13]), wspomagającego proces podejmowania optymalnych decyzji, z uwzględnieniem informacji jakościowych oraz subiektywnych ocen użytkownika modelu. Uproszczony schemat przykładowego diagramu wpływu przedstawiono na rys. 5. Informacja (*evidence*) o zmianie prezesa Narodowego Banku Polskiego może mieć, w ocenie inwestora, bezpośredni wpływ na wartość indeksów giełdowych oraz pośredni na stopę zwrotu (zysk lub stratę) z portfela akcyjnego, złożonego z akcji banków.



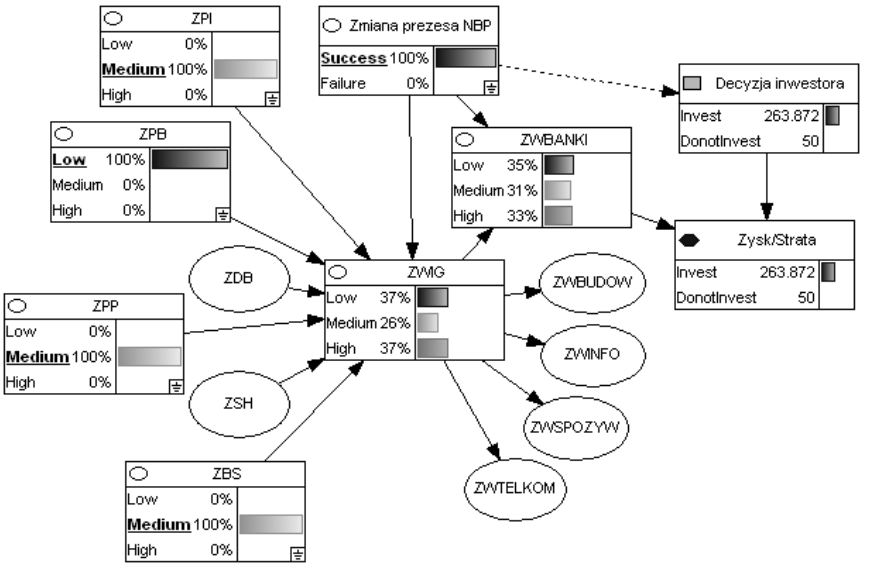
Rys. 5. Diagram wpływu (rozbudowana sieć bayesowska z rys. 2)

Zmienna *Decyzja inwestora* jest reprezentowana przez węzeł typu *Decision node*. Tablicę wartości zmiennej *Zysk/Strata*, reprezentowanej przez węzeł typu *Value node,* przedstawia rys. 6.

Decyzja inwest...	Invest			DonotInvest		
ZWBANKI	Low	Medium	High	Low	Medium	High
Value	100	200	500	50	50	50

Rys. 6. Tablica wartości zmiennej Zysk/Strata (Rys. 5.)

Na rys. 7 przedstawiono diagramu wpływu z rys. 5 po uwzględnieniu nowych obserwacji oraz aktualizacji modelu. Po zaobserwowaniu średnich wartości zmiennych makroekonomicznych ZPI, ZPP, ZBS oraz niskiej wartości zmiennej ZPB (tabela 1), jak również pozytywnej oceny (w opinii eksperta - analityka finansowego) wpływu zmiany prezesa Narodowego Banku Polskiego na sytuację rynkową, otrzymujemy rozkłady prawdopodobieństwa *a posteriori* interesujących nas zmiennych ZWIG oraz ZWBANKI. Tablica węzła *Zysk/Strata* przedstawia oczekiwaną wartość zysku z podjętej inwestycji.



Rys. 7. Diagram wpływu po uwzględnieniu obserwacji dotyczących wybranych węzłów sieci.

Zastosowania diagramów wpływu na rynku finansowym wymagają oddzielnego opracowania ze względu na bardzo istotną rolę preferencji i subiektywnych ocen analityków finansowych w procesie podejmowania decyzji w warunkach niepewności [5].

Zaprezentowany model diagnostyczny z rys. 2 był zmodyfikowany i rozbudowany w pracy [17] oraz wykorzystany do wspomagania wyceny akcji na Giełdzie Papierów Wartościowych w Warszawie, na przykładzie akcji Banku BPH. Włączenie do modelu kolejnych papierów wartościowych pozwoli w przyszłości stworzyć większy system wspomagający decyzje inwestycyjne.

4. Dodatek

Tabela 3

Podstawowe statystyki zmiennych makroekonomicznych [%]

l.p.	Statystyka	ZPI	ZPB	ZDB	ZPP	ZSH	ZBS
1.	Średnia	4,94	0,44	26,9	0,8	11,23	-1,55
2.	Odchylenie standardowe	3,86	2,59	86,58	6,82	399,71	5,3
3.	Mediana	4,15	0	11,76	-0,25	-16,97	-1,08
4.	Min	0,3	-4,07	-96,5	-19	-1480	-14,76
5.	Max	13,9	14,86	550,1	21,8	3000	14,57

Źródło: opracowanie własne

Tabela 4

Podstawowe statystyki indeksów giełdowych [%]

l.p.	statystyka	ZWIG	ZWBANKI	ZWBUDOW	ZWINFO	ZWSPOZYW	ZWTELKOM
1.	Średnia	0,75	1,14	1,02	0,27	0,74	0,16
2.	Odchylenie standardowe	7,36	7,49	8,42	12,34	7,22	12,01
3.	Mediana	1,92	2,14	0,98	-0,98	-0,04	-0,09
4.	Min	-27,09	-26,25	-32,63	-25,89	-20,27	-25,88
5.	Max	18,67	19,05	24,01	46,01	28,31	46,01

Źródło: opracowanie własne

Tabela 5

Współczynniki korelacji miesięcznych stóp zwrotu indeksów branżowych z indeksem WIG

	ZWBANKI	ZWBUDOW	ZWINFO	ZWSPOZYW	ZWTELKOM
ZWIG	0,882	0,784	0,796	0,685	0,821

Źródło: opracowanie własne

Podziękowania

Model przedstawiony w pracy został utworzony i przetestowany przy użyciu biblioteki klas C++:**SMILE**© i narzędzia **GeNIe**, służącego do tworzenia i wnioskowania w graficznych modelach probabilistycznych. Narzędzia powstały w Laboratorium Systemów Decyzyjnych (*Decision Systems Laboratory*) Uniwersytetu Pittsburgh'skiego i są dostępne na stronie http://genie.sis.pitt.edu.

Szczególne podziękowanie za pomoc w przygotowaniu artykułu oraz cenne wskazówki składam dr Agnieszce Oniśko z Wydziału Informatyki Politechniki Białostockiej.

Literatura:

[1] Adamczak, A.: *Empiryczna weryfikacja modelu arbitrażu cenowego w warunkach Giełdy Papierów Wartościowych w Warszawie*, [w:] T.Trzaskalik (red.) Modelowanie preferencji a ryzyko'99, Katowice, 1999, str. 31 – 46.

[2] Berry, M.A., Burmeister, E., McElroy, M.B.: *Sorting Out Risk Using Known APT Factors*, Financial Analysts Journal, vol. 44, no. 2, 1988, pp. 29-42.

[3] de Campos, L.M.: *A Scoring Function for Learning Bayesian Networks based on Mutual Information and Conditional Independence Tests,* Journal of Machine Learning Research 7 (2006), pp. 2149 – 2187.
[4] Chow, C.K.: *Approximating Discrete probabilisty Distributions with Dependence Teres,* IEEE Transactions on Information Theory, 14(3), 1968, pp. 462-467.
[5] Cieślak, A.: *Behawioralna ekonomia finansowa. Modyfikacja paradygmatów funkcjonujących w nowoczesnej teorii finansów,* Materiały i Studia Zeszyt nr 165, NBP, Warszawa, 2003.
[6] Demirer, R., Mau, R.R., Shenoy, C.: *Bayesian Networks: A Decision Tool to Improve Portfolio Risk Analysis*, Working Paper, University of Kansas, School of Business, 2006.
[7] Druzdzel, M.J., Oniśko, A., Wasyluk, H.: *Uczenie parametrów sieci bayesowskich z danych z wykorzystaniem bramek Noisy–OR* [w:] Z.Bubnicki, O.Hryniewicz, R.Kulikowski (red.) Problemy współczesnej nauki. Teoria i zastosowania. Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2002, str. 19 – 26.
[8] Druzdzel, M.J., Cheng, J.: *AIS-BN: An Adaptive Importance Sampling Algorithm for Evidential Reasoning in Large Bayesian Networks,* Journal of Artificial Intelligence Research, 13, 2000, pp. 155-188.
[9] Druzdzel, M.J., Yuan, Ch.: *An Importance Sampling Algorithm Based on Evidence Pre-propagation,* Proceedings of the Nineteenth Annual Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, Inc., San Francisco, CA, 2003, pp. 624-631.
[10] Hagstrom, R.G.: *The Warren Buffett Portfolio*, Wiley & Sons, New York, 1999.
[11] Heckerman, D.: *A Tutorial on Learning with Bayesian Networks,* technical Report, MSR-TR-95-06, 1996.
[12] Heckerman, D.: *Bayesian Networks for Data Mining,* Data Mining and Knowledge Discovery, 1, 1997, pp. 79 – 119.
[13] Howard, R.A., Matheson, J.E.: *Influence Diagrams*, [in:] Howard R.A and Matheson J.E. (eds.) Applications of Decision Analysis, vol. 2, 1984, Menlo Park, Calif.: Strategic Decisions Group, pp. 721 – 762.
[14] Jensen, F.V.: *Bayesian Networks and Decisions Graphs*, Springer-Verlag, 2001.
[15] Kłopotek, M.A.: *Inteligentne wyszukiwarki internetowe,* Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2001.
[16] Olbryś, J.: *Estymatory miar Expected Shortfall i Value-at-Risk: przykłady zastosowania do pomiaru ryzyka walutowego*, Inwestycje finansowe i ubez-

pieczenia. Tendencje światowe a rynek polski. Prace Naukowe Akademii Ekonomicznej im. O. Langego, Wrocław, Nr 1088, 2005, str. 65 – 72.
[17] Olbryś, J.: *Model sieci bayesowskiej wspomagający decyzje inwestycyjne w warunkach niepewności,* Szósta Ogólnopolska Konferencja Naukowa „Modelowanie Preferencji a Ryzyko'07", Ustroń, 2007, w trakcie recenzji.
[18] Osiewalski, J.: *Ekonometria bayesowska w zastosowaniach,* Wydawnictwo Akademii Ekonomicznej w Krakowie, 2001.
[19] Pearl, J.: *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, 1988.
[20] Shenoy, C., Shenoy, P.P.: *Bayesian Network Models of Portfolio Risk and Return*, [in:] Abu-Mostafa, Y.S., LeBaron, B., Lo, A.W., Weigend, A.S. (eds) Computational Finance, MIT Press, 1999, pp. 87-106.

BAYESIAN NETWORK AS A TOOL OF EXTRACTING KNOWLEDGE FROM AN ECONOMIC DATABASE

Abstract: Making a decision in investment starts from perception and analysis of incoming information. Rational investors reason according to Bayes formula and try to develop posterior probabilities after new evidence has been added. Virtually all decisions that investors make are exercises in probability. Bayesian networks have been used in different decision support system contexts that combine qualitative and quantitative information. Main goal of this paper is to present Bayesian network as a tool of extracting knowledge from an economic database, with respect to historical quantitative information, uncertain qualitative information, incomplete knowledge and evidence.

Keywords: bayesian networks, investment decision support system, influence diagram

Artykuł zrealizowano w ramach pracy badawczej statutowej S/II/1/03.

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2007
Informatyka – Zeszyt 2

Agnieszka Oniśko[1]

KNOWLEDGE ACQUISITION FROM HUMAN EXPERTS FOR BUILDING BAYESIAN NETWORK MODELS

Abstract: Knowledge acquisition from experts is a costly and time-consuming task. While domain experts have the necessary knowledge and expertise, they rarely have the experience needed to translate this knowledge into the model. This paper describes typical problems that are encountered by knowledge engineers when building Bayesian network models and illustrates some practical techniques to overcome them. The presented examples capture the problems that occurred during elicitation the numerical parameters of the model for diagnosis of liver disorders.

key words: knowledge acquisition, Bayesian network, parameter elicitation

1. Introduction

Elicitation of probabilities is often pointed out as the major obstacle in building Bayesian networks [1,2]. In addition to tediousness of elicitation, the resulting numbers are not always reliable because of various factors that can skew them. The problems related to biases and poor calibration during elicitation of judgemental probabilities from human experts are well known [7]. There are several techniques that facilitate the process of the elicitation of numerical parameters from human experts.

Probability elicitation essentially involves posing a set of questions that require an expert to either provide direct probability estimates, or to choose between simple alternatives or bets. In case of direct response methods, experts give their estimates either numerically as cumulative probabilities, graphically by plotting density functions, or verbally by expressing their estimates by terms *fifty-fifty*, *certain*, *improbable*, etc. Quantitative interpretation of these different kinds of descriptors are then encoded numerically. Indirect response methods are more sophisticated. Experts are asked here to make choices between simple alternatives in gambles related to the events in question. The subjective probabilities

[1] Wydział Informatyki, Politechnika Białostocka, Białystok

are inferred from these choices. Two classical approaches to indirect probability elicitation are betting and reference lottery [5,14]. However, it is known that in laboratory setting experts tend bet and take a risk more easily than in a real-world setting [8]. Additionally, these methods tend to be infeasible for models that include thousands of probabilities. Other alternatives include a use of ranking, relative likelihood, and interval techniques [10].

Spiegelhalter *et al.* [15] introduced several techniques for assessment, refinement, and improvement of imprecise probabilities that were elicited from human experts. The method involved measuring quality of assessments by scoring rules. The authors observed that reliable probability assessments can be obtained from experts, although the experts tended to be too extreme in their judgements. The method presented by van der Gaag *et al.* [4] allowed to elicit from a domain expert around 150-200 probabilities per hour. The approach was based on the assessment of both verbal probability expressions and numbers.

This paper presents several knowledge acquisition techniques for building Bayesian network models, particularly, for elicitation of numerical parameters from human experts. The paper is structured as follows: Section 2 describes briefly HEPAR II, a Bayesian network model for diagnosis of liver disorders, Section 3 summarizes the interactions between knowledge engineer and human expert, Sections 4 and 5 present the techniques for elicitation of numerical parameters from human experts. Finally, section 6 concludes the paper.

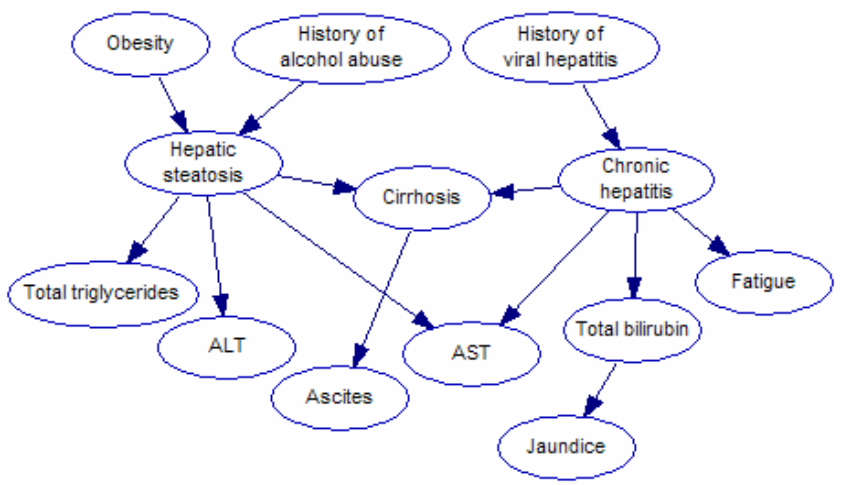


Fig 1. A simplified fragment of the HEPAR II network

2. HEPAR II

HEPAR II is a Bayesian network model for diagnosis of liver disorders [11]. The model consists of 70 variables and covers 11 different liver diseases and 61 medical findings, such as patient self-reported data, signs, symptoms, and laboratory tests results. The structure of the model, (i.e., the nodes of the graph along with arcs among them) was built based on medical literature and conversations with our domain expert, a hepatologist Dr. Hanna Wasyluk and two American experts, a pathologist, Dr. Daniel Schwartz, and a specialist in infectious diseases, Dr. John N. Dowling. The elicitation of the structure took approximately 50 hours of interviews with the experts, of which roughly 40 hours were spent with Dr. Wasyluk and roughly 10 hours spent with Drs. Schwartz and Dowling. This includes model refinement sessions, where previously elicited structure was reevaluated in a group setting. The structure of the model consists of 121 arcs and the average number of parents per node is equal to 1.73. There are on the average 2.24 states per variable. Figure 1 shows a simplified fragment of the HEPAR II network.

There are two versions of HEPAR II: (1) the data model and (2) the expert model. In the former version of the model conditional probability distributions were learned from the HEPAR database.[2] In the latter version, the parameters of HEPAR II were elicited from human expert.

3. Interactions with experts

Building the structure of a Bayesian network typically involves interaction of the knowledge engineer with domain experts. In case of the HEPAR II model, regular, short sessions with the expert worked well. In between these sessions, the knowledge engineer focused on refining the model and preparing questions for the expert. The refinement consisted of analyzing positive and negative influences in the model when the model was fully quantified, i.e., when the numerical parameters of the network were already specified. There are several tools that can be useful in debugging a Bayesian network (e.g., Elvira [16], GeNIe [17]). It helps when the knowledge engineer understands the domain at least at a basic level. It is a good idea to read at least a relevant section of a medical textbook on the topic of the meeting with the expert, so the knowledge engineer is familiar

[2] The HEPAR database was created in 1990 and is thoroughly maintained at the Gastroentorogical Clinic of the Institute of Food and Feeding in Warsaw. Each hepatological case is described by over 160 different medical findings and by a histopathologically verified diagnosis. The version of the HEPAR data set used in HEPAR II consisted of 699 patient records.

with the terminology, the variables, and interaction among them. It is also recommended to record the sessions with the expert because it is often hard to process all the medical knowledge that is provided by a domain expert during a meeting. It is also recommended to organize brainstorming sessions with a participation of knowledge engineers and medical experts who are not directly involved in building the model. With respect to HEPAR II, there were a few such sessions, and they addressed important issues and raised questions about the model.

4. Elicitation of conditional probability distributions

This section presents a set of techniques that were applied during the elicitation of conditional probability distributions of HEPAR II from the domain expert. The expert participating in parameter elicitation was familiar with elementary probability theory, e.g., the expert knew that probability ranges between 0 and 1, and that for a set of mutually exclusive and exhaustive set of events, the probabilities should sum exactly to 1. With respect to probability elicitation for HEPAR II, direct approach was applied.

There were 70 nodes to quantify and 370 independent probabilities to elicit. The elicitation of numerical parameters took around 10 hours, composed of five sessions of roughly two hours each.

4.1 What probability?

Before the process of elicitation begins, it is important to make sure that a common framework between the expert and the knowledge engineer has been established. First of all, the expert should know for which population the probabilities are provided. It is natural that medical experts think in terms of the setting that they work in. Therefore, they often give numerical parameters that reflect their experience in a particular clinic. Since probabilities must refer to a general population, an expert may argue:

> *If I am going to use a model at a hospital, why should I use general population frequencies?*

Hence, it is crucial to agree for which population the probabilities are elicited. Otherwise, the parameters and the entire model can be very wrong [3]. Figure 5 captures risk factors of *Functional hyperbilirubinemia* that were considered in the HEPAR II model. During the elicitation of parameters for the node *Functional hyperbilirubinemia*, one of the values provided by the expert was

initially equal to 0.8. This value was supposed to indicate the probability of suffering from *Functional hyperbilirubinemia* by men below 30. However, it appeared that this values represented a hospital population instead of a general population. After clarifying this issue, the value was changed to 0.1.

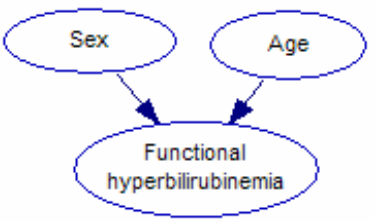


Fig 2. Modeling risk factors of *Functional hyperbilirubinemia*

An example that has been often applied during the quantification of HEPAR II was related to a *tram population* (we assumed here that a tram population is representative of a general population in Poland), i.e., when the knowledge engineer asked the expert about a particular probability, for example, a prevalence of the variable *Alcohol abuse*, the following question was posed:

> *Imagine a tram with 100 people. How many of them would suffer from alcohol abuse?*

The expert found such framing of the question helpful and easy to follow.

4.2 How to get it?

The following section provides the examples of elicitation of numerical parameters for HEPAR II from a human expert. It introduces several types of questions that were posed to the expert during the elicitation sessions for HEPAR II. It was challenging to quantify some of the nodes that were modeled as binary variables. For example, it was hard to estimate the conditional probability distribution for *Steatosis* given *Alcohol abuse*, since the latter was modeled in HEPAR II as a binary variable. It would be easier to quantify this node if alcoholism was represented in the model by several states indicating different degrees of alcoholism. So, during the elicitation of the probability that *Steatosis* occurs given that a patient abuses alcohol, the expert had to average over several groups of patients representing different stages of alcoholism. Similar situation was observed during the quantification of nodes that represented children of the variable *Gallstones* (modeled as a binary variable). It happens that gallstones

present with different stages, hence, the expert again needed to average over several groups of patients representing different stages of the disease.

An advisable practice in probability elicitation is verification of the elicited numbers. Monti *et al.* [9] noticed that confronting the expert with inconsistencies in the assessments, after the expert had already gone through the whole elicitation process, is ineffective. With respect to HEPAR II, the expert was asked to confirm a particular probability in terms of a percentage rather than a probability value. For example, when the expert provided a value equal to 0.3, representing a probability that *Amylase* is elevated in patients with *Gallstones*, the following question was posed:

> *Does it mean that 30% of the patients presenting with gallstones have elevated amylase?*

The expert often re-thought the situation and refined the value. It appears that in some cases, it is easier to specify the probabilities for a node that is conditioned on another node. For example, the expert found it difficult to provide the prevalence for the node *Reactive hepatitis* (representing a rare liver disease). However, she found it much easier to quantify this node when it had *Hepatotoxic medications* as a direct predecessor (*Hepatotoxic medications* is a risk factor of *Reactive hepatitis*).

It is challenging to quantify a node with several parents, especially when they consist of several states. For binary nodes with two parents, it is helpful to draw a table that simplifies the process of elicitation. Table 1 captures the table that was introduced to the expert in order to quantify the node *Bleeding*. Each value in this table represents independent probabilities that *Bleeding* occurs given a combination of values of *Platelet* and *INR*. In my experience, experts prefer to start the elicitation from situations that are extreme: either from a normal state or the most abnormal. For example, with respect to elicitation of parameters for the node *Bleeding*, the expert specified the values in bold font first (see Table 1). Similar tables were created for the variables *PBC* and *Functional hyperbilirubinemia*. The expert found these tables useful.

Table 1

Elicitation of probabilities for the node *Bleeding*

Platelet/INR	normal	low	very low
normal	**0.0**	0.01	0.01
low	0.01	0.1	0.2
very low	0.01	0.2	0.5
extremely low	0.01	0.4	**0.9**

Bayesian networks allow to combine different sources of knowledge. For example, they allow to combine expert knowledge with existing clinical data. However, the constructors of Bayesian network models should be aware of biases that can occur during combining different sources of knowledge [3].

5. Assessment of Noisy-OR parameters

Some types of conditional probability distributions can be approximated by canonical interaction models that require fewer parameters. Very often such canonical interactions approximate the true distribution sufficiently well and can reduce the model building effort significantly. One type of canonical interaction, widely used in Bayesian networks, is known as Noisy-OR gate [6,12]. Noisy-OR gates are usually used to describe the interaction between n causes X_1, X_2, …., X_n and their common effect Y. The causes X_i are each assumed to be sufficient to cause Y in absence of other causes and their ability to cause Y is assumed independent of the presence of other causes.

Elicitation of numerical parameters for HEPAR II involved also the assessment of parameters for Noisy-OR gates. This section describes details related to obtaining Noisy-OR parameters. There were 25 nodes identified by the expert that could be approximated by Noisy-OR gates. To obtain the numerical parameters for these nodes, direct approach for elicitation was applied. There was a total of 189 parameters and the assessment took a total of about four hours of expert time.

There were several types of questions posed to the expert to investigate whether a node can be approximated by a parametric distribution. Figure 3 captures the causes of *Cirrhosis* that were modeled in HEPAR II. To check whether *Cirrhosis* can be approximated by a Noisy-OR gate, the following question was posed:

> *Is Cirrhosis the effect of combination of both causes Fibrosis and Steatosis?*

If the expert answered that *Cirrhosis* is the effect of the two modeled causes, the interaction between the node and its parents cannot be modeled by a Noisy-OR gate.

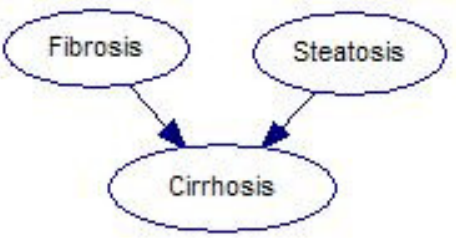


Fig 3. HEPAR II: Modeling the node *Cirrhosis*

Another question that was posed by the knowledge engineer was related to the interaction of mechanisms. Figure 4 captures two causes of the node *Carcinoma* modeled in HEPAR II. There were two mechanisms considered here. The first mechanism represented *PBC* leading to *Carcinoma*. The second mechanism was related to *Cirrhosis* causing *Carcinoma*. In such cases the following question was posed:

Do mechanism-1 and mechanism-2 interact with each other in causing carcinoma?

If the expert is not aware of any interaction between the mechanisms, it is reasonable to assume that such interactions do not exist or are not too strong and the interaction among *Cirrhosis*, *PBC*, and *Carcinoma* can be modeled by a Noisy-OR gate.

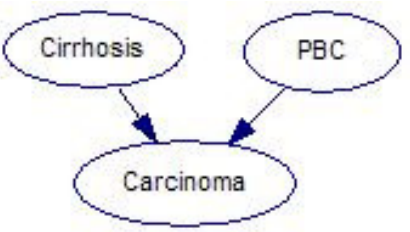


Fig 4. HEPAR II: Modeling the node *Carcinoma*

Another example involved modeling the causes of the variable *Nausea* (see Figure 5). In this case the following question was posed:

Imagine that there are two patients: one is abusing alcohol and the other is not. Both of them receive hepatotoxic medications. Will the patient who drinks alcohol have a significantly different probability of being affected by these medications?

If this probability is reported by the experts as significantly higher or lower, then there is a significant synergy between the causes and *Nausea* should not be modeled as a Noisy-OR gate.

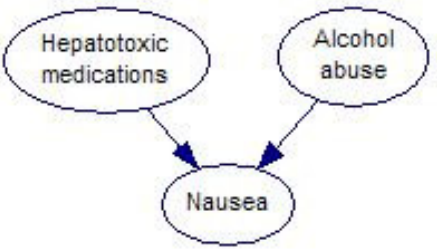


Fig 5. HEPAR II: Modeling the node *Nausea*

Table 2

Noisy-OR parameters for the node Total cholesterol

Cholesterol/parent	Steatosis	ACH	PCH	PBC	Leak
very high	0.02	0.02	0	0	0
high	0.6	0.4	0.25	0.1	0.01
normal	0.38	0.58	0.75	0.9	0.99

Similarly to the CPT (conditional probability table) elicitation, the expert preferred to provide first the values for either normal or the most abnormal states. Table 2 captures the Noisy-OR parameters for the node *Total cholesterol*.[3] With respect to this node, the expert indicated first the probabilities for the states *normal* and *very high*. Then, she specified the values for intermediate range of cholesterol level. The expert often provided the initial values and then adjusted them until reasonably satisfied with them (this behavior is known as "anchoring and adjustment heuristic" [6]). The expert preferred providing a Noisy-OR probability within a particular state, i.e., for a normal state the parameters for each of the auses were specified first (the values in the fourth row of Table 2). The expert often found it difficult to provide a value of the leak probability, i.e., the expert tended to provide this value for a hospital population instead of referring to the general population. I have also observed that the expert preferred giving a leak probability at the very end of the elicitation.

[3] The abbreviations used in Table 2: ACH and PCH, stand for Active Chronic Hepatitis and Persistent Chronic Hepatitis respectively.

6. Conclusions

This paper presented typical problems that are encountered by the knowledge engineers during building Bayesian network models. The author provided examples of problems that have been occured during elicitation of numerical parameters from human experts and then illustrated practical techniques to overcome these problems.

I would like to acknowledge funding from the MNiI grant number 3T10C03529. Contents of this paper is based in part on one of the chapters of my PhD dissertation [11].

References:

[1] Marek J. Druzdzel and Linda C. van der Gaag. Elicitation of probabilities for belief networks: Combining qualitative and quantitative information. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 141-148, San Francisco, CA, 1995. Morgan Kaufmann Publishers, Inc.

[2] Marek J. Druzdzel and Linda C. van der Gaag. Building probabilistic networks: "Where do the numbers come from?" guest editors' introduction. *IEEE Transactions on Knowledge and Data Engineering*, 12(4):481-486, 2000.

[3] Marek J. Druzdzel and F. Javier Díez. Combining knowledge from different sources in causal probabilistic models. *Journal of Machine Learning Research*, 4:295-316, 2003.

[4] Linda van der Gaag, Silja Renooij, Cilia Witteman, Berthe Aleman, and Babs Taal. How to Elicit Many Probabilities, In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, Morgan Kaufmann Publishers, San Francisco, CA, 1999, 647-654.

[5] J. Hampton, P. Moore, and H. Thomas. Subjective probability and its measurements. *Journal of the Royal Statistical Society*, 21-42, 1973.

[6] Max Henrion. Some practical issues in constructing belief networks. W L.N. Kanal, T.S. Levitt, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence 3*, 161-173. Elsevier Science Publishers B.V., North Holland. 1989.

[7] Daniel Kahneman, Paul Slovic, and Amos Tversky. Judgment Under Uncertainty: Heuristics and Biases, Cambridge University Press, Cambridge, 1982.

[8] R. Duncan Luce and Howard Raiffa. Games and Decisions: Introduction and Critical Survey. Dover Publications, 1989.
[9] Stefano Monti and Giuseppe Carenini. Dealing with the expert inconsistency in probability elicitation. *IEEE on Knowledge and data engineering*, 12(4):499-508, 2000.
[10] M. Granger Morgan and Max Henrion. Uncertainty: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis, Cambridge University Press, Cambridge, 1990.
[11] Agnieszka Oniśko, Marek J. Druzdzel, and Hanna Wasyluk. Learning Bayesian network parameters from small data sets: Application of Noisy-OR gates. *International Journal of Approximate Reasoning*, 27(2):165-182, 2001.
[12] Agnieszka Oniśko. *Probabilistic Causal Models in Medicine: Application to Diagnosis of Liver Disorders*. Ph.D. dissertation, Institute of Biocybernetics and Biomedical Engineering, Polish Academy of Science, Warsaw, April 2003.
[13] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, CA. 1988.
[14] L. Savage. The foundations of statistics. Wiley, New York, 1954.
[15] J. Spiegelhalter, David, C. G. Franklin, Rodney, and Kate Bull. Assessment, criticism and improvement of imprecise subjective probabilities for a medical expert systems. *Uncertainty in Artificial Intelligence* 5, pages 285-294. Elsevier Science Publishers B.V., North Holland, 1990.
[16] http://www.ia.uned.es/~elvira
[17] http://genie.sis.pitt.edu

POZYSKIWANIE WIEDZY OD EKSPERTÓW W BUDOWANIU MODELI SIECI BAYESOWSKICH

Streszczenie: Pozyskiwanie wiedzy od ekspertów jest kosztownym i czasochłonnym zadaniem. Pomimo ogromnej wiedzy i doświadczenia, jakie posiadają eksperci, niejednokrotnie nie potrafią ich przenieść na tworzony model. Poniższy artykuł opisuje przykłady problemów, z jakimi może się zetknąć inżynier wiedzy w trakcie budowania modeli sieci bayesowskich, jak również proponuje rozwiązania tych problemów. Prezentowane przykłady dotyczą problemów, jakie pojawiły się w trakcie pozyskiwania od eksperta parametrów numerycznych modelu sieci bayesowskiej do diagnozowania chorób wątroby.

Keywords: pozyskiwanie wiedzy, inżynieria wiedzy, sieci bayesowskie

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2007
Informatyka – Zeszyt 2

Tomasz Rybak[1]

PROBLEMS WITH STORING TEMPORAL DATA

Abstract: Article shows problems with storing temporal data. It describes tree structures used to store it, and problems with them. Then it describes using relational databases for storing temporal data, and how features provided in relational databases can be used to overcome problems present when trees are used to store temporal data.

Keywords: temporal data, relational databases, data structures, B-Tree

1. Introduction

Thanks to advances in computer technology each user can store more data each year. As Adelstein noted in [2], in last fifteen years disk space available to single person increased over 1000 times. So we store more and more data using enormous disk spaces. But storing large amounts of data makes it harder to retrieve necessary files when needed. We may either remember where each file is located, what is impossible as soon as number of files is more than few dozens, or we may search for them. As noted by Cutrell et al. in [6], users use search programs when they are easy to use yet provide users with enough functionality. When searching, over 60% of searches results were sorted by date. Authors noticed, that we remember files by associating them with time of activities we were doing when creating those files. So being able to access temporal data is more and more necessary to be able to cope with enormous amounts of stored data.

Also, we store not only personal data but also data associated with observed processes, which we try to understand. We want to be able to make assumptions on data, to get information and then knowledge from it. To do it, we must be able to observe it's values, both currently and in the past and on those premises discover what are dependencies in data and predict future values or make theory explaining values present in set. To be able to do it we must store facts together with time it held its values.

Some of problems of storing temporal data are described by Snodgrass in [17] and partially by Gray in [10]. Gray mentioned that introduction of magnetic tapes and

[1] Faculty of Computer Science, Bialystok Technical University, Bialystok

disks as computer memory make storing history of data more difficult. Previous types of persistent memory were able to be written once and then couldn't be changed, so it was natural to store all previous values of data. It was true both in computers, and in human crafts, like accounting, medical records, and so on. But magnetic memory was expensive and there was temptation to use it again by overwriting old values with new ones.

So now we need to return to storing old values. It puts much request on computers, algorithms, memory and computing power. We need to discover efficient ways of processing large and larger amounts of data.

This article present some of the ways of dealing with problems with storing and accessing temporal data. Section 2 describes data structures used to store temporal data. Section 3 discusses using relational databases to store temporal data. Section 4 describes ways of solving performance problems in databases.

2. Data types

Adelstein noted in [2] that we store so many data, that it is impossible to load it at once into memory, or manually search. Even automatic operation on files which size is measured in terabytes takes time. So we need to use data structures and algorithms that are efficiently operating on data.

Salzberg and Zhang in [15] shown four properties that must be met for a system to be able to efficiently operate on data. They called them *properties of good access methods*, I will call them *principles*. They describe both operations and way of storing data:

1. Data should be clustered on disk according to anticipated queries. Usually data is stored in order governed by their domain. This property is called clustering
2. Additional data (i.e. auxiliary data, used to describe stored values) should use the least possible space on disk.
3. During operations we should minimize number of accessed disk pages with data not relevant to performed operation.
4. Operations performed on local tuples should not change many pages, i.e. operation on small amount of tuples should never require accessing large amount of disk pages.

In other words, principles require that amount of stored data and number of operations should be minimal. Also, algorithms should minimize number of accessed pages during any operation.

Of course these rules, especially 1st principle connected with 4th, assume that there is difference in cost of accessing local and non-local disk pages. It is true when data is stored on magnetic tapes and magnetic and optical disk drives. However when using Solid State Memory disks cost of accessing any disk page is constant and does not depend on previously read page. However magnetic memory offers more capacity and is cheaper than disks produced in different technology so they will remain main storage for data for long time.

2.1 Storing changes of state

Whether we try to store state of entire system or we use data structures to store state, we need a way of saving values and changes that occurred in time. Gray proposed ([10]) two solutions for that problem. Both of them are being used today.

First is to store values together with time of their validity. It allows for easy access to each value at any given time, as it requires accessing appropriate memory fragment representing requested time. This solution is used in temporal databases.

The second one is to store initial values, and then save all changes with time of their occurrence. As this solution stores only changes, usually it requires less memory than first solution. But if one wants to get value at one particular moment he or she has to redo all changes up to this moment.

Implementation of second solution is described in [9], and it's approach for storing data in databases in [14].

In some situations these two solutions are used together. In regular intervals full state is stored, and between them only differences are saved. It joins advantages of both solutions; data takes less space, and fast forwarding in time is possible without computational overhead. It is mostly used in multimedia; in the movie every few seconds there is so called primary frame, which contains information about all pixels, and then there are differential frames, containing only information about differences to the next/previous one. This solution shows most benefits when there is not too many differences between frames.

2.2 B-Tree

B-trees are described in Chapter 19 of [5]. It stores structures (sometimes called tuples), where attributes are used to name it's parts. One or more attributes are grouped together to serve as key. Key can be used to distinguish two tuples. Tuples are stored in order given by their keys, so function able to compare two keys and tell which of two keys is greater is necessary, especially if key is made from more than one

attribute. Sometimes all keys must be unique, but it is not required by algorithms operating on B-tree.

We try to minimize number of operations according to principles. Number of operations is measured as function of number of tuples stored in tree, noted as N. Ways of calculating and describing cost is described in Chapter 2 of [5]. The most common operations are:

- inserting tuple into tree
- removing tuple from tree
- finding tuple in the tree
- finding elements whose key holds some property (related to previous one)

Each node in B-tree contains between $B/2-1$ and $B-1$ objects stored in order according to key values. Each node except leafs has from $B/2$ to B child nodes. Each child node contains tuples with key values between greater than key before this child, and less than following key in parent node.

In B-tree all operations for only one element are proportional to height of the tree. It's height is $O(log_B N)$, where base of logarithm B depends on number of keys stored in each node. So the more keys are stored in the node, the less tall is tree, and the less is the cost of operation. To make sure that tree has regular size, i.e. all leafs have the same distance from the root, operations on nodes that will ensure this are performed during inserting and removing values. But changes in tree are done only on one path from root to leaf, so their cost is $O(log_B N)$. If inserting values results in node having more than $B-1$ children, it has to be split into two nodes. After split each of nodes has $B/2-1$ objects, so they are divided between new and old node. If, as result of removing object, node has less than $B/2-1$ objects, it is merged with other node.

Because all values are ordered, searching for keys in range may be implemented by finding first and last key in range and then traversing through all tuples between them. It requires less operations than accessing all keys. However it is not always possible to use such optimization. In many cases there is need to find some tuples when exact key value is not known, but only conditions to be met by key will require going through all tuples.

B-tree is used to store large amounts of data, usually so large that it cannot be put in main memory and must be stored on disk. As each node to be processed must be read from disk and each disk operation is much slower than CPU operation, we try to minimize number of disk pages read. Because reading entire disk block takes the same amount of time as reading fragment of page, in B-tree node size is equal to disk block size. As noted earlier, time of most operations is logarithmic with base

equal to the number of tuples is node. As size of node is constant, number of tuples depends only on size of tuple. So there is tendency to have more tuple stored in one node.

B+-tree allows for storing more values in node. It stores full tuples only in leafs; inner nodes store only keys. It allows for even bigger values of B, so tree has less height, hence less reads from disk. But such tree uses more disk space, as keys are repeated. They are stored in inner nodes, and also in child nodes, together with the rest of object. When keys have the same length it is possible use binary search instead of linear search to make searching for particular key faster. It is the more efficient the more keys are in one node. Sometimes, to be able to put as many keys in one node, keys with the same prefix are compressed, and only their different part is stored, and common prefix is stored once for node. But this attitude makes algorithms more complicated and resulting tree are not so regular. Those changes are described in [15]

In some cases hashing structure, described in details in Chapter 12 of [5] and partially in [15] is used for storing data. It is, however, not used as frequently as B-tree. It uses special functions, called hashing functions and because of way of using it, is extremely slow when there is too many tuples stored. To be faster it would require another hash function, and it would require rebuilding entire hash. But main reason why it is not so frequently used is because it stores data in unordered fashion. It can be used for searching only for equal values, not for ranges. Also, because of properties of hash functions, similar values are stored in different disk fragments, so principle 1 is not met. More detailed description of reasons that hashes are less popular is in [15].

2.3 R-Tree

Adding time to data set adds one or more attributes to stored structure. This additional attributes are used to search data (e.g. find values of all tuples from given moment) so it can be seen as key. In result we have two keys: a main key, previously present in data, and a temporal key, being time in which tuple had those particular values. So we have got a multidimensional key, and using B-tree is not enough as it is able to deal only with one-dimensional key. It will be shown, that meeting four Principles is very hard when having data that cannot be ordered according to only one key.

R-Tree, described in [15], can be used to store tuples that can be accessed by using many keys; it doesn't prefer any of keys and treat each of them in the same way. Because of that it is mainly used to store multidimensional, spatial data. But even though R-trees are not used to store temporal data, they present interesting problems similar to those present in storing time-oriented data.

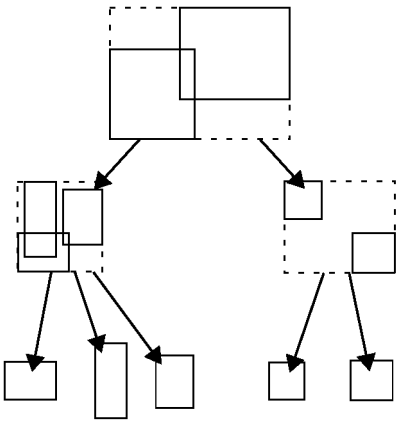


Fig. 1. Sample R-Tree

R-tree assumes that key is point or rectangle (box in higher dimensions) in M-dimensional space, and each point can be described by M numbers. Each node of R-tree contains multi-dimensional rectangle (hypercube) in which all tuples, both belonging to this node and it's all descendant nodes, can be put. If one of keys is non-numerical, it must be made possible to perform the same operations as for numerical values, in particular defining range of values, sum and product of two sets and determining if value lies in particular range. Nodes located higher in a tree has their bounding boxes computed basing on their child nodes. Bounding boxes of child nodes can overlap, which means that some fragments of space are described by more than one node. It makes searching not so efficient as it was in B-trees. Sample 2-dimensional R-Tree with overlapping bounding boxes is shown in Figure 1.

During inserting tuple bounding boxes of all nodes up to root must be updated, so 4th principle (locality of changes) is violated. During removing tuple hypercubes should also be updated, but this is often omitted for performance reasons. It makes removing faster, but later searching is slower, especially when searched keys values are not found, as more nodes has to be inspected. As in B-trees, inserting into full node requires splitting. But unlike B-tree, there is more than one way of dividing keys to nodes.

One of algorithms is described in [15]. There are two seed tuples chosen. One remains in old node, one is moved to new node. Each of the remaining tuples is tested to two nodes, and is places in this, in which it requires less increase of volume of bounding box. If one of nodes reached minimal number of elements, remaining

elements are put into the other one. This algorithm's main disadvantage is that when tuple is added to one node, it's part may belong to bounding box of another node. This complicates searching, as many nodes must be visited during search.

R*-trees try to solve this problem. They choose nodes and seed points in a way ensuring that nodes will share minimal volume of space. However, this requires visiting more nodes in a tree, so locality principle is violated again. It is price for speeding searching, but even in this case there can be necessity to search many nodes when tuple is not found.

Another genre of R-tree are hR^{Π}-trees, very complicated in implementation. They ensure that bounding boxes do not share common points. Inserting is done in three steps, and there can be return from inserting function after first phase. Two other, tiding steps can be made later, which can be faster if they are done for more that one inserted tuple. But even in those trees searching for element not existing in a tree is slow, as more than $O(log_B N)$ nodes must be searched to be sure that tuple is not in tree.

All R-trees store the more additional information the more dimensions space have. Because each node must store a bounding box, and each object must store its space coordinates, memory requirements are $O(N^M)$, where M is the dimension of space. Additional memory usage means that each of nodes can store less objects, so a tree has more height.

Different way of storing spatial data is presented by structure called z-order, similar to hash. It takes bits from binary representation of key attributes, and from this bits a integer number is created. This number is used as a new key to store object in ordinary B-Tree. Advantage of this solution is that the key size does not depend on number of dimensions. But there are many disadvantages. In many cases, especially when tuples are located near each other, many of bits will be the same, so many keys will be very similar. As z-ordering uses attitude similar to hashing, tuples located near in space may be located in different places on a tree, and one node can hold tuples located in different places in space. This depends on bits used for creation of key, and they cannot be changed after creating a tree. Such change would require rebuilding the entire tree and during this operation no other operations on data can be performed. Searching for ranges of values is inefficient, as is in hashes. However, authors of [15] noted that there is one algorithm allowing for searching for range, but it requires reading more disk pages that is necessary, thus violating 3rd principle. But, as z-order uses B-tree, there is only one order. So z-order is not popular and none of popular databases uses it.

2.4 Temporal structures

There are many structures created to efficiently store temporal data, described in [15]. They are created under an assumption that time is another key, but it is very special key. Some of temporal operations that require using this special key are:

- find all tuples from time T
- find a record with particular key at time T
- find records with keys in range at time T
- find all values of tuple at any time in past (searching for history).

One of temporal structures is WOBT, Write-Once B-Tree. This B-Tree contains additional temporal key, so operations can be performed both on main and temporal key. It assumes that any given disk block can be written only once, and then is accessed in read-only mode. So it can be used for reliable logging, if it is required by law. Even on ordinary magnetic drives each disk block will be written only once, so historical data will be always available. Of course rewriting unused pages would lessen disk space requirements, but ability of accessing all nodes as they were in the past has more advantages than less memory usage.

In WOBT each page stores range of keys from some time slice. Searching is done as in an ordinary B-tree but taking time value into consideration.

Inserting key (and also changing it, as changing a value is equivalent to removing the old value and inserting a new one) requires writing at least one new page. When split is needed, only one key is used to split at any given time. If splitting is required for both key and time, first key-split is performed before time-split. Splitting is done in a way requiring upgrading only one parent node, storing information from last time period. This means that only youngest parent is upgraded, and older parent nodes are not updated. They still can point to active data, so they cannot be moved to archive; all pages must be treated in the same way, because without traversing through the entire tree there is no way of knowing if each of nodes points to current data.

As result of splitting there can exist many nodes pointing to the same tuples, and each node can have many parents. So it is not a tree, but rather a directed acyclic graph (DAG).

This problem is not present in Time-Split B-tree (TSB). This tree is split by using only keys. Unlike in WOBT, time chosen for a new and an old node can be any time, not only the current one like in case of WOBT. But it must be the moment before any of currently active transactions has started. Splitting according to any time, not only current moment is unique for TSB. TSB allows for creating disk pages (and nodes) with only inactive data (data from the past, changed or deleted afterwards). It

allows for moving such pages to an archive, for example to other drive, which makes management of disk space easier.

Another structure for storing temporal data is Partially Persistent R-tree (PPR). It is an R-Tree with time as an additional key. It has the same performance as R-Tree during searching, but inserting and deleting nodes is much more complicated. It is a directed acyclic graph (DAG) storing differences between trees that have been done in time. There are many trees, where each stores data at each moment. Some of trees share nodes, so there can even be many roots. Both during inserting and deleting there can be a split, always according to the current time. The most recent root points to the tree with active ("alive") data.

3. Databases

Temporal data can be stored in relational databases (described by Codd in [4], an article introducing relational databases); one can manage them using all features that can be used when dealing with non-temporal data, like access through Structured Query Language, transactions (introduced by Jim Gray in [10]), database constraints, etc.

There are three types of temporal values, as described by Snodgrass in [17].

instant something happened at a particular moment
interval length of time
period length of time, but the anchored one, with a starting moment defined

Similarly to data structures described in Section 2., adding time to relational database requires extending primary key. Usually we add two additional columns, the one with start of period of validity and the second with end of such a period.

This means that there can exist many rows with the same values of columns of the old primary key. So now we must add time to a primary key, to be able to distinguish rows. This approach will allow for easy retrieving all historical values. Simply we ask for all rows with a given value of the primary key, without using time, and DBMS returns all values, the current and the historical ones.

3.1 Standards

As relational databases was becoming more and more populer, different communities tried to extend SQL by adding custom types and keywords to better serve them. To prevent from creating many incompatible SQL variants, International Standards Organization (ISO) created SQL Multimedia and Applications Packages (SQL/MM), standard extending SQL with types usefull in different fields. It consists of six parts:

1. Framework
2. Full-text
3. Spatial
5. Still Image
6. Data Minint
7. History

Currently different databases implement different parts of SQL/MM. Most commonly supported is full-text part, whether by native types (Microsoft SQL Server, Oracle), or by extensions (contrib/tsearch2, full-text search suite in PostgreSQL). Spatial standards are managed by OpenGIS consortium, and extensions for different DBMS are available: MsSqlSpatial for Microsoft SQL Server, Oracle Spatial for Oracle, PostGIS for PostgreSQL. Data mining solutions are available by using extensions.

Description of SQL/MM and it's purposes, and more detailed description of different parts can be found in [13]. Here only "Part 7: History" ([1]), describing temporal data, is discussed.

Part 7 describes recommended way of creating history tables. It presents types, functions, and stored procedures for working with temporal data. Its idea and presented solutions are very similar to described by Snodgrass ([17]), but more formalized and presenting more details of used functions. Snodgrass focused of ideas and theory, ISO on implementation.

Names of all objects described in standard start with letters HS and underscore (HS_). As SQL/MM is using object-oriented SQL as base, used types can contain attributes.

For each table additional table storing historical data is created. Historical table contains all columns that originaln table contains, plus additional column with type HS_Hist, consisting of two attributes, HS_HistoryBeginTime and HS_HistoryEndTime. Those attributes desribe start and end time of period when row was valid. History and original tables have also triggers that are used to update history table whenever original is changed. Additional triggers preventing any changes in history table other than inserting new rows and changing HS_HistoryEndTime attribute from NULL to other value are created. However those triggers disallow correcting any mistakes, so wrong values remain in history tables forever. System similar to this, but without triggers disallowing changes in history tables, was created and desribed ([14]) by Author.

Three main parts of standards are Chapter 5, describing contcepts and presenting examples, Chapter 6 describing procedures used to implement it, and Chapter 7 describing neccessary types. Described procedures are used to create all objects, like

tables and triggers, neccessary to manage historical data. Functions are able to cast temporal types to other types supported by database, and provide input and output methods. Also, functions for checking temporal predicates (described by Snodgrass, e.g. checking if moment is in particular period) are described, for example HS_Intersects for checking whether two periods have common subperiod.

SQL/MM Part 7 is yet to be fully implemented by any commercially available database.

3.2 Implementation

As all operations are executed inside transactions, their implementation is crucial to assuring validity of data. The most important property from point of view of data consistency is Isolation. It requires that each transaction sees version of row that existed at its beginning and all changes made by itself even if there is many transactions accessing this row. If it is not, many problems with consistency of data (like non-repeatable and phantom reads, described in [12]) may occur.

The easiest implementation of transactions can be, that each row (or table, or any other object) has its own lock. When transaction wants to read data, it gets read-lock; when it wants to write data it gets write-lock. There can be many read-locks for any object, but only one write-lock. So many transactions can concurrently read data, but only one can write. This solution is easy, but very slow. In this solution there is only one state of each row at any given moment.

The other possible solution, called Multi Version Concurrency Control (MVCC), is present in namy different databases. PostgreSQL stores data in this form natively since it's beginning, Microsoft SQL Server uses MVCC as primary way of storing data since version 2005, other databases (like Berkeley DB, Firebird, ZODB) offer this technique as option. DBMS has monotonic counter of transactions. Each transaction is given different number and counter is increased. So transactions that started later has bigger number identifying them. Each row has two additional columns, one with number of transaction before which row isn't valid, and one with number of transaction after which row isn't valid. Each transaction only sees rows with appropriate values of these columns.

MVCC allows for coexistence of many versions of each row; there is as many versions as many transactions changed it. But each transaction sees only one row, one with the largest number of transaction in first column; of course this number must be less than number of this transaction. Example is shown in Figure 2.

MVCC allows for fast read access, as there is no need for locks. Also when one transaction wants to write and other ones want to read, there is no conflict. Only

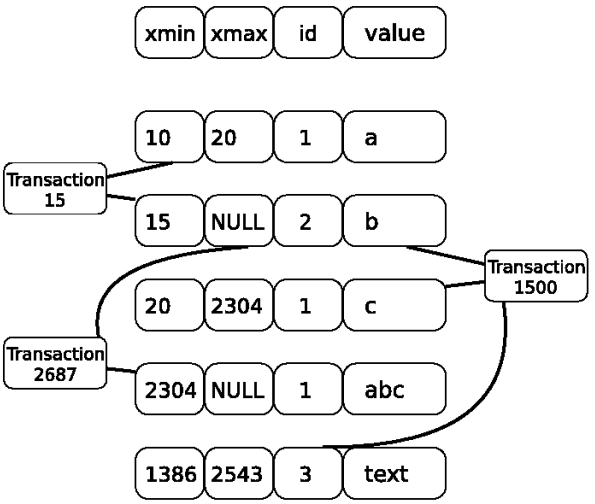


Fig. 2. Multiversion Concurrency Control table

if there are two transactions wanting to change the same row, DBMS uses locks to resolve conflict on write. One of transactions (the later one) is locked until the other finishes, either by commit or by rollback.

However there is cost associated with using MVCC. Each operation changing data creates new, active rows and leaves old, inactive and invisible for all new transactions rows. This is similar to WOBT, where number of taken disk pages also was growing during performing operations on data. But databases using MVCC allow for removing old rows by using vacuum command. This operation takes number of the oldest transaction (it is the lowest number of active transaction) and removes all rows that are not visible by this transaction. These rows can be deleted, because if they aren't visible by oldest transaction (with lowest transaction number) they aren't visible by younger ones (because new transactions have bigger transaction numbers). Vacuuming frees space taken by dead rows, but requires time, because it must check all rows on disk drive. Also it works better when there is no long-lasting transactions. If there are such ones, not every row that should be freed can be. But long-lasting transactions are also problem in lock-based DBMS, because they take many resources, especially locks.

To implement temporal database, we need to take care about managing two additional time columns. When we add new row, we must insert current time as value

of column describing start of validity period. When removing row, it is not removed, but only current time is inserted into column with end of validity period. Changing value of row is really removing old row and inserting new one.

As shown in [14], this changes can be done in existing database schema, without making changes to any of programs using database. Old programs will function without using temporal features, and new ones will be able to use them.

Snodgrass suggested ([17]) that second time column should not be empty in active rows, but should contain moment from distant future. It is consistent with suggestion of Date [7] that no column should contain NULL value. But choosing distant moment is very hard, so in most implementations rows with current data has NULL in their second time column, even though queries that must deal with this must contain additional conditions and aren't so nice-looking as those which are in systems implemented according to Snodgrass' suggestion. Choosing too early end moment means that in some time in future this "future" moment will be in the past, and entire database will be useless as source of temporal data. Also, different databases has different ranges of possible date values, so there is no one maximum date that can be used in many system similar to INT_MAX in C language. So, ISO standard regarding tempral and historical data ([1]) uses NULL as value indicating that particular row is still active.

Sample implementation of temporal database, called timetravel, is distributed together with PostgreSQL in module contrib/spi. Contrib part contains additional functions and modules that can be used together with database, but that are not as popular or has not yet been enough tested to be included into main server.

Timetravel requires adding two columns, date_on and date_off, both with type abstime, for storing period of validity. Default value for column date_on is currabstime(), and for date_off is infinity (PostgreSQL has infinite time, which is moment very far away). This module adds triggers which are fired when current (active, one with infinity in date_off) row is changed or deleted.

For managing time-enabled tables there exist functions set_timetravel and get_timetravel. It enables and disables trigger responsible for operations on data in tables.

3.3 Alternate data values

Most of the commercially available databases (IBM DB2, Oracle, Microsoft SQL Server) allow for creating hot-swap solutions by using master server as souce of events and one or more slave servers as recipients of these ovents. This way each of slaves has the same state as master, so in case of master's failure it can be promoted to

be new master. This technique is using the same mechanisms that are used to recovery after failure in case of single server ([8]). But according to author's knowledge, only PostgreSQL and Oracle 10g allow for database administrator to go back to any moment in the past. Oracle calls this ability Oracle Flashback Query, PostgreSQL calls this very interesting way of creating backup is called Point In Time Recovery (PITR).

To create backup it is required to save initial state of database by simply copying entire data directory. Then all changes in database will be saved in the backup. Instead of saving new values commands that made those changes will be saved, in order they were executed by the database. PITR is done by using provided by user command for dealing with files saved by backup subsystem. This can be any command, such as compressing, copying files, transferring them using network, as long as it is possible to check if command succeeded. Command cannot require any data from user entered using keyboard.

This approach is very interesting. During creating initial backup state database needn't be in consistent state. All changes, and transactions that are active during creating initial state will be saved later.

This implementation allows using PITR as simple replication system (system for storing data in more than one DBMS, described in [8]), even though it's main function is creating backup. But it is very simple system, and requires manual applying changes; for each changeset an operator must "recover" system as there was an error.

If there is need to restore data, all data files must be removed, old files must be copied back to data directory and then all files saved by PITR must be played again. To use those files, user must provide system with a command to recover them. This command is reverse of the command used to backup files. All saved files are played in chronological order, i.e. in order they were saved.

The most interesting feature is ability to stop recovering at any moment between initial backup and last issued command. As PostgreSQL uses transaction numbers to identify time, and PITR uses time, it is possible to identify time to which replay commands either as number of last transaction or as time.

Ability to stop recovery at any given moment is very interesting, but PostgreSQL uses it to provide user with more interesting functionality. If recovery files were not played to the end and then in the database were done changes, PostgreSQL creates a new "timeline". Each timeline has it's own identifier, so it is possible to choose a timeline during recovery. It means that during PITR activity it is possible to change data in the database, than go to earlier point in time, and make other changes. It is

even possible to switch from one timeline to an other. Even though it requires restart of the server, it is a very interesting possibility.

Using both PITR and temporal databases can be problematic. When we stop recovering at any point earlier that current, and then resume operations in data there will be a gap in time. Time values will look like there was the end of some period, and then from some time there is activity again. It will look like there is a "silent period". It may or may not be problem, but analyzing such data may lead to wrong assumptions or results.

4. Performance

There are two types of usage of databases mentioned in [18], called OLTP and OLAP.

OLTP OnLine Transactional Processing
OLAP OnLine Analytical Processing

OLTP is a normal relational database, where there is equilibrium between reading and writing operations. It usually contains current data, without many historic records, to be able to process data very fast. It is normalized, so there is no redundancy, and is tuned to serve as many queries per second as possible, and to be responsive even if it is serving many clients.

OLAP is a database used for analysis of data. Data warehouse contains historical records, used in data mining processes to allow for quick answers for questions asked during analysis of business trends.

OLAP database is used differently than OLTP one. It is used by one or few clients, who run long-lasting jobs to calculate report about trends in data; mostly requiring long and complicated computations.

Trend analysis requires analysing large amounts of data, usually from different periods, for different clients or regions, and comparing those disparate fragments of database. Each variable that can be used to distinguish fragment of database important for business analysis (usually relating to some real world concept, like region or client) is called dimension. So set of all those variables (dimensions) creates multidimensional space. When we imagine that all dimensions are located on perpendicular axis, and values coming from database are distributed along them, we will get multidimensional cube. Main usage of OLAP is to divide this cube into fragments and to compare one fragment to another in search for trends and differences important from business standpoint.

As we do not know what is the scale of needed fragments, database must be able to cope with the largest (entire company) and smallest (single customer or shop)

fragments. And on each of those levels we must be able to get all data from all other dimensions appropriate for this level.

But relational databases are normalized and contain minimal amount of indexes to be able to respond fast to usual OLTP needs. This means that queries required by data mining system will be executed very slowly, especially when switching between different variables and levels of detail, because it would require calculating all aggregates for another distribution of variables.

So usually OLAP databases store denormalized, precomputed values for all dimensions and all possible levels of detail. Such denormalized database contains many repeated records, columns and attributes. Usually its data comes from OLTP database (for example every Sunday data from previous week is imported) and during importing there are many changes in structure of database, records, and many values are precomputed to allow faster queries later. However it makes non-read access very hard, if not impossible, as to change anything would require changing values of variables in many places of cube. Such precomputed cube may take much more disk space than ordinary relational database, as data is repeated for each level and each variable. As data size may grow exponentially, it is possible that there will be phenomenon called database explosion. It occurs when there is many dimensions containing sparse data. In such case precomputing creates many empty cubes (but each such record takes portion if disk to write) filling disk with information that there is no information. Moreover, creating such cube will take long time, as all calculations must be done during importing data into OLAP database.

OLAP cubes store higly-dimentional data, so they can be implemented by using previously described data structures, for example R-Trees. But this would require changing algorithms operating on those structures. On each level of tree we would have stored data being aggregation of data stored on lower levels of the tree. Each level of tree would be particuler level of detail for variable. Also algorithms changing trees would not be used, as OLAP database is used only as source of data, and no operations changing data are executed. Even importing new fragment of data only adds few new nodes, without changing shape of the tree.

Temporal databases try to join OLTP and OLAP approach. They contain both current and historical data. The longer temporal database exists, the more historical rows it contains, so current rows become lesser fraction of entire table. So if a query requires accessing all rows, it becomes less and less efficient in time.

This problem can be solved by moving historical data to other place, but it must remain in the database. In current RDBMS it is possible by moving data to an other table or by partitioning existing table. Problem with moving to other table is that analysis will be harder as it will require accessing more tables. PostgreSQL solves it

by allowing for table inheritance. One table may be used as a template for an other. When both of them contain data, reading data from the original table will also read from other tables. This way data can be divided into many tables which may reside on different disk drives, and use the same queries for accessing data.

4.1 Indexing

Shasha and Bonet point in [16] that using indexes makes read operations on data in DBMS faster. Usually for indexes are used B-trees and hashes. As noted in Section 2. and in [15], hashes allow only for eqality comparisons, and B-trees can be used in operations on ranges, so B-trees are used more frequently, as they can be used to optimize more queries.

All available databases use some way of indexing. But different databases offer different levels when trying to index custom types. The most advanced solution is offered by PostgreSQL. This DBMS allows for creating custom indexing structures. User can create two types of indexes, either Generalized Search Tree (GiST) described in [3] and in Chapter 50 of PostgreSQL documentation ([11]), and Generalized Inverted Index (GIN), described in Chapter 51 of [11]. PostgreSQL's contrib module contains sample implementation of an R-tree done using GiST. Implementing GiST index requires providing library with six functions; GIN requires implementing four functions.

GiST allows for creating indexes similar in structure to B-Tree indexes. But it can be used to implement custom comparison functions or additional conditions that should be met by a row to be chosen by a query ([11]):

> This extensibility should not be confused with the extensibility of the other standard search trees in terms of the data they can handle. For example, PostgreSQL supports extensible B-trees and hash indexes. That means that you can use PostgreSQL to build a B-tree or hash over any data type you want. But B-trees only support range predicates ($<$, =, $>$), and hash indexes only support equality queries.
> So if you index, say, an image collection with a PostgreSQL B-tree, you can only issue queries such as "is imagex equal to imagey", "is imagex less than imagey" and "is imagex greater than imagey"? Depending on how you define "equals", "less than" and "greater than" in this context, this could be useful. However, by using a GiST based index, you could create ways to ask domain-specific questions, perhaps "find all images of horses" or "find all over-exposed images".

On the other hand, GIN stores keys together with lists of rows where the key value appears, so it allows for speeding up queries searching for rows with particular value of column. It also can allow for building statistics of data stored in a database, as it stores list of all values of column.

A code can be built together with a domain specialist. It is next step on a way proposed by Codd ([4]). He proposed to split responsibility for storing and operating on a data between a database and a program. Now we can let database experts to build storage engine, and domain experts to build functions operating on data and put those functions into database, to make queries more efficient ([11]):

> One advantage of GIN is that it allows the development of custom data types with the appropriate access methods, by an expert in the domain of the data type, rather than a database expert. This is much the same advantage as using GiST.

As shown in [15] and [12], relational databases can use many indexes concurrently to optimize query. These indexes can be of different types (B-trees, hashes, own types); also they can involve one or many columns. Moreover, one column can be taken into consideration in many indexes. DBMS cas then build partial results using those indexes and then use so called bitmap index to join them to receive a final result.

But having many indexes can worsen performance during changing data, as all of them must be updated. Also, they require space. But it is often good trade-off to sacrify some space for efficient data operation.

DBMS is free to choose a way of performing query (inter alia choose one of indexes) because SQL is a declarative language, To do this, it creates many equivalent queries using the relational algebra, and then checks what will be the most efficient one. Entire process of optimizing queries is described in [12].

To be able to choose the best plan, DBMS needs information about data and system. The database either computes different performance estimators, or allows for administrator to set them. For example, PostgreSQL allows for tuning a database to hardware it is located on by using different configuration variables, described in Chapter 17 of [11]. Some of them (enable_bitmapscan, enable_hashjoin, enable_nestloop, enable_seqscan) allow turning on and off usage of particular types of indexes (hash index, B-Tree index etc.). Other allow for setting cost of different operations in the system (seq_page_cost and random_page_cost allow for setting cost of reading one disk page id different modes, cpu_tuple_cost is cost of processing one row, cpu_operator_cost is cost of applying one operator, effective_cache_size is size of memory used by operating system to cache disk pages).

As it can be seen, system administrator can tune database to be as optimal as possible.

5. Summary

Temporal data must be stored in special structures, which are complicated in description and implementation. These structures, however they are enriched b-trees, present less performance during operations on temporal data, and require more disk space to store it. Data structures like trees offer only one way of performing operations on data stored in them.

On the other hand database optimization allows changes of "query plans" (ways of performing operation on data in database), based on available indexes, number of rows, even values of columns. Thanks to that DBMS can choose optimal plan, which helps with solving problems with efficient operations on temporal data.

So it is preferable to use relational databases instead of own data structures to store temporal data.

References

[1] ISO/IEC 13249-7:2005. Information technology – database languages – sql multimedia and application packages – part 7: History, 2005.

[2] Frank Adelstein. Live forensics: diagnosing your system without killing it first. *Communications of ACM*, 49(2):63–66, 2006.

[3] Oleg Bartunov and Teodor Sigaev. Introduction to gist. Technical report, Moscow, Russia. http://www.sai.msu.su/ megera/postgres/gist/doc/intro.shtml.

[4] E. F. Codd. A relational model of data for large shared data banks. *Communications of ACM*, 13(6):377–387, 1970.

[5] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Wprowadzenie do algorytmów*. Wydawnictwa Naukowo–Techniczne, Warszawa, wydanie trzecie edition, 2000.

[6] Edward Cutrell, Susan T. Dumais, and Jaime Teevan. Searching to eliminate personal information management. *Communications of ACM*, 49(1):58–64, 2006.

[7] Chris Date. *Database In Depth. Relational Theory for Practitioners*. O Reilly Media Inc., Sebastopol, CA, USA, 2005.

[8] Michael J. Franklin. Concurrency control and recovery. In Tucker [19]. Published in Cooperation with ACM, The Association for Computing Machinery.

[9] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns. Elements of Reusable Object–Oriented Software*. Addison–Wesley, 1995.
[10] Jim Gray. The transaction concept: Virtues and limitations. Technical report, Tandem Computers Incorporated, Cupertino, CA, USA, 1981.
[11] The PostgreSQL Global Development Group. Postgresql 8.2.4 documentation, 2006. http://www.postgresql.org/docs/8.2/static/index.html.
[12] Yannis E. Ioannidis. Query optimization. In Tucker [19]. Published in Cooperation with ACM, The Association for Computing Machinery.
[13] Jim Melton and Andrew Eisenberg. Sql multimedia and application packages (sql/mm). *SIGMOD Rec.*, 30(4):97–102, 2001.
[14] Tomasz Rybak. Using object/relational mapping system for analysis of program execution. In *Forum-Conference on Computer Science*, Smardzewice, Poland, 2006.
[15] Betty Salzberg and Donghui Zhang. Access methods. In Tucker [19]. Published in Cooperation with ACM, The Association for Computing Machinery.
[16] Dennis Shasha and Philippe Bonnet. Tuning database design for high performance. In Tucker [19]. Published in Cooperation with ACM, The Association for Computing Machinery.
[17] Richard T. Snodgrass. *Developing Time-Oriented Database Applications in SQL*. Morgan Kaufmann Publishers, San Francisco, California, USA, 2000.
[18] Alexander Thomasian. Transaction processing. In Tucker [19]. Published in Cooperation with ACM, The Association for Computing Machinery.
[19] Allen B. Tucker, editor. *Computer Science Handbook, Second Edition*. Chapmann & Hall/CRC, 2004. Published in Cooperation with ACM, The Association for Computing Machinery.

PROBLEMY ZE SKŁADOWANIEM DANYCH TEMPORALNYCH

Streszczenie: Artykuł przedstawia problemy implementacyjne związane z przechowywaniem danych temporalnych. Opisuje struktury drzewiaste, które mogą być użyte do przechowywania temporalnych danych oraz problemy z nimi związane. Przedstawia sposób użycia relacyjnych baz danych do przechowywania danych temporalnych, jakie wiążą się z tym problemy, jak można wykorzystać możliwości oferowane przez bazy danych do ułatwienia implemenacji. Opisuje problemy z wydajnością oraz sposób ich rozwiązania w relacyjnych bazach danych.

Słowa kluczowe: struktury danych, drzewa, dane temporalne, bazy danych, relacyjne bazy danych

Artykul zrealizowano w ramach pracy badawczej W/WI/7/06.

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2007
Informatyka – Zeszyt 2

Marta K. Smolińska, Zenon A. Sosnowski[1]

PODSUMOWANIA LINGWISTYCZNE Z GRUPOWANIEM ROZMYTYM

Streszczenie: W pracy przedstawiono zastosowanie podsumowania lingwistycznego jako predykatu rozmytego do wyznaczania obiektów z typową wartością atrybutu lub zbioru atrybutów. W rozmytym algorytmie grupującym wykorzystana jest populacja z wyznaczoną ze względu na dany atrybut typowością obiektów. Wyniki działania tego algorytmu oraz jego zmodyfikowanej postaci zostały przedstawione na przykładzie populacji, której obiektami są piksele obrazu.

Słowa kluczowe: podsumowania lingwistyczne, grupowanie rozmyte

1. Wstęp

W [1] został opisany język SummarySQL, który umożliwia definiowanie zapytań z podsumowaniami lingwistycznymi. Tam też, miedzy innymi, opisano algorytm wyznaczania typowych wartości (ang. *Typical Values*) oraz typowych klastrów (ang. *Typical Clusters*). W [2] przedstawiono algorytm znajdowania przybliżonych wartości typowych.

W niniejszej pracy analizowaną bazą obiektów był obraz. Obraz potraktowany jako zbiór obiektów, z których każdy reprezentowany jest przez położenie oraz kolor, jest bazą rzeczywistych danych, przy czym efekt klasteryzacji jest widoczny i łatwy w ocenie przez człowieka. Poza tym atrybut kolor może być traktowany jako pojedyncza wartość lub jako obiekt składający się z trzech atrybutów składowych RGB, co daje możliwość zastosowania różnych funkcji podobieństwa.

W rozdziale drugim opisaliśmy podsumowanie lingwistyczne, a w rozdziale trzecim algorytm znajdowania typowych wartości. W rozdziale czwartym przedstawiono algorytm grupowania typowych klastrów oraz jego modyfikacji dającej lepsze efekty grupujące w zastosowaniu do obrazów. W rozdziale piątym opisano

[1] Wydział Informatyki, Politechnika Białostocka, Białystok

algorytm znajdowania przybliżonych wartości typowych oraz jego postać zmodyfikowaną w zastosowaniu do kilku atrybutów.

2. Podsumowania lingwistyczne

Podsumowanie lingwistyczne jest zkwantyfikowanym rozmytym wyrażeniem postaci *„większość ludzi jest wysoka”* lub *„niewielu wysokich ludzi jest lekkich”*, gdzie *(wysocy)* ludzie są związani ze specyficzną populacją obiektów.

W zapisie ogólnym:

$$Q\,(obiektów)\,w\,(populacji)\,C_f \;\; jest \;\; S \tag{1}$$

C_f jest populacją rozmytą, czyli rozmytym zbiorem obiektów (każdemu obiektowi przypisany jest stopień przynależności do zbioru).

S jest rozmytym wyrażeniem lingwistycznym, zwanym sumatorem.

Q jest kwantyfikatorem lingwistycznym, np. *większość, niewiele, około połowa*.

Z podsumowaniem lingwistycznym związana jest wartość prawdy $\tau \in [0;1]$ nazywana miarą istotności podsumowania. Jest ona stopniem przynależności proporcji obiektów należących do populacji C_f, które spełniają S ($r \in [0;1]$), do zbioru rozmytego Q. Informuje ona, w jakim stopniu podsumowanie lingwistyczne zgodne jest z populacją C_f.

$$\tau = \mu_Q(r) \tag{2}$$

gdzie

$$r = \frac{card_f\left(S \cap C_f\right)}{card_f\left(C_f\right)} \tag{3}$$

a $card_f$ jest mocą zbioru rozmytego, zdefiniowaną jako:

$$card_f(B) = \sum_{o_i \in B} \mu_B(o_i) \tag{4}$$

Natomiast przecięcie $\cap$ jest zdefiniowane jako agregacja minimum lub inna dowolna t-norma.

Złożoność obliczenia stopnia prawdy τ podsumowania lingwistycznego w wypadku prostego zbioru rozmytego jest złożonością rzędu $O(|C_f|)$, gdzie $|C_f|$ jest liczbą obiektów w C_f.

Formalnie podsumowanie lingwistyczne zapisywane będzie jako predykat rozmyty:

$$\sum_Q \left(\mu_{C_f}(o_j) \middle| \mu_S(o_j)\right) \tag{5}$$

lub krócej

$$\sum_Q \left(\mu_{C_f} \middle| \mu_S\right) \tag{6}$$

gdzie $\sum_Q$ reprezentuje podsumowanie z kwantyfikatorem rozmytym Q. Predykat μ_{C_f} jest funkcją przynależności klasy rozmytej C_f, którą chcemy podsumować, a μ_S jest funkcją przynależności sumatora S.

3. Typowe wartości

Podsumowanie lingwistyczne, tak jak predykat rozmyty, związane jest z wartością prawdy z przedziału jednostkowego $[0;1]$. Na podstawie tej obserwacji możemy używać podsumowań lingwistycznych jako predykatów rozmytych, co możemy wykorzystać w wyznaczaniu typowej wartości.

Jeśli chcemy zapytać o *„osoby z typową wagą”*, to myślimy o osobach z populacji *C*, z wagą podobną do wagi u większości osób. Musimy, więc zdefiniować funkcję $\approx_w$ podobieństwa wagi, np.: $x \approx_w y = \max\left(1 - \frac{|x-y|}{\delta}, 0\right)$, gdzie δ jest stałą definiowaną przez użytkownika (np. 10% dziedziny atrybutu). Nową populację $C_{typical}$ zawierającą osoby z typową wagą wraz ze stopniem przynależności możemy opisać jako:

$$\mu_{C_{typical}}(o_i) = \mu_C(o_i) \wedge \sum_{most}\left(\mu_C(o_j) \middle| o_j.Weight \approx_w o_i.Weight\right) \tag{7}$$

Wartość prawdy podsumowania $\sum_{most}\left(\mu_C(o_j) \middle| o_j.Weight \approx_w o_i.Weight\right)$ definiuje rozmyty predykat dla *typowej wagi* w populacji C, a predykat $\mu_C(o_i)$ zapewnia, że typowy obiekt o_i nie może być bardziej typowy niż jego stopień przynależności do C.

3.1. Algorytm znajdowania typowych wartości

Każdy obiekt o_i z populacji C staje się prototypem obiektu, z typową wartością rozpatrywanego atrybutu a. Dla każdego z tych prototypów obliczane jest podsumowanie „większość obiektów w populacji ma wartość atrybutu a podobną do wartości tego atrybutu w obiekcie o_i". Wynikiem jest populacja $C_{typical}$, zawierająca obiekty wraz ze stopniem przynależności. Algorytm może znajdować typowe obiekty ze względu na kilka atrybutów, wtedy stosujemy iloczyn stopnia podobieństwa poszczególnych atrybutów. W [1] wynikiem tego algorytmu jest populacja zawierająca wartości atrybutów wraz z ich stopniem typowości. W obiektowych bazach danych [3] danymi wejściowymi algorytmu są: rozmyta populacja C oraz funkcja sim – podobieństwa między obiektami, ze względu na atrybut a (lub zbiór atrybutów). Jako wynik natomiast otrzymujemy populację $C_{typical}$, zawierającą obiekty wraz z ich stopniem typowości. Pseudokod tego algorytmu przedstawiono na rysunku 1.

1) **znajdowanie_typowych_obiektow**
2) foreach (object $o_i \in C$)
3) $sum = \sum_{most} \left(\mu_C(o_j) \middle| sim(o_j.a, o_i.a) \right)$
4) $C_{typical}.Add(o_i)\ with\ \mu_{C_{typical}}(o_i) = \min(sum, \mu_C(o_i))$;
5) end
6) return $C_{typical}$

Rys. 1. Algorytm znajdowania typowych obiektów ze względu na atrybut *a*.

4. Typowe klastry

Rysunek 2 przedstawia algorytm grupowania rozmytego przy użyciu typowej wartości. Przy zastosowaniu tego algorytmu w obrazie nie dawał on zbyt dobrych rezultatów, gdyż do każdego kolejnego klastra trafiały obiekty mniej do siebie podobne.

Lepsze efekty klasteryzacji uzyskaliśmy po zmodyfikowaniu punktu 4 do postaci przedstawionej na rysunku 3.

Tak zmodyfikowany algorytm umieszcza w klastrach elementy bardziej podobne niż algorytm oryginalny, ma jednak bardzo dużą wadę – jego złożoność obliczeniowa jest bardzo duża.

1. Znajdujemy typowe obiekty w populacji C ze względu na kryteria klastra. W wyniku tej operacji otrzymujemy populację $C_{typical}$. Kryteria klastra w wypadku obrazu to podobieństwo między dwoma kolorami.
2. Znajdujemy najbardziej typowy obiekt o'. Jest to obiekt z najwyższym stopniem typowości (najwyższy stopień przynależności w $C_{typical}$).

   $$\forall o \in C_{typical} : o'.\mu \geq o.\mu$$

   Zapisujemy ten obiekt w nowej populacji będącej *i*-tym klastrem U_i i usuwamy go z $C_{typical}$.
3. Poszukiwanie typowego klastra:
   Wśród obiektów, z $C_{typical}$, których stopień typowości jest większy niż α, znajdujemy obiekt podobny do przynajmniej jednego obiektu w klastrze U_i, dodajemy go do tego klastra i usuwamy z $C_{typical}$. Znajdowanie to kontynuujemy do momentu, w którym żaden obiekt nie zostanie dodany do klastra.
   Obiekty traktujemy jako podobne, jeśli podobieństwo między nimi jest większe niż α, która jest stałą ustaloną przez użytkownika. Na przykład:
   α - 75% z najbardziej typowej wartości ($\alpha = o'.\mu \cdot 75\%$).
   Użycie parametru α powoduje, ze klaster zawiera obiekty z $\alpha - przekroju$. Dzięki temu do klastra nie trafiają obiekty ze zbyt małym stopniem typowości (mniejszym niż α) lub zbyt mocno oddalone od elementów klastra.
4. Powtarzać od punktu 2 do osiągnięcia kryterium stopu, np. dopóki najbardziej typowy obiekt z $C_{typical}$ będzie miał stopień typowości większy lub równy β ($o'.\mu \geq \beta$). β jest stałą definiowaną przez użytkownika, np. 50% typowości najbardziej typowej wartości znalezionej w pierwszej iteracji.
   Kryterium stopu można też zdefiniować jako osiągnięcie zadanej liczby klastrów lub opróżnienie całej populacji $C_{typical}$.

Rys. 2. Algorytm grupowania rozmytego przy użyciu typowych wartości – typowe klastry.

Przykład 4.1.

W celu dokonania prezentacji została utworzona syntetyczna populacja prostych obiektów składających się z nazwy (indeksu) oraz wartości całkowitej z zakresu $[0;250]$. Każdy z obiektów należy do populacji ze stopniem przynależności 1. Tabela 1 przedstawia populację oraz efekt klasteryzacji algorytmem „ty-

powe klastry" z rysunku 2 oraz wersją zmodyfikowaną (rys. 3). W eksperymencie tym wartość $\alpha = 75\% \cdot o'.\mu$, przy czym $o'.\mu$ jest najwyższym stopniem przynależności w populacji $C_{typical}$. Natomiast relacja podobieństwa między dwoma obiektami została zdefiniowana jako podobieństwo między wartościami całkowitymi (wartościami drugiego atrybutu), opisane wzorem 8.

$$s(x, y) = \max\left(1 - \frac{|x - y|}{50}, 0\right) \tag{8}$$
$$x, y \in [0;250]$$

4. Zmodyfikować populację $C_{typical}$ w sposób następujący: stopień przynależności każdego elementu będącego w $C_{typical}$ zamienić na stopień przynależności, jaki ten obiekt miał w oryginalnej populacji, i ponownie obliczyć typowość dla wszystkich elementów w zmodyfikowanej $C_{typical}$.

Powtarzać od punktu 2 do osiągnięcia kryterium stopu.

Rys. 3. Modyfikacja punktu czwartego w algorytmie grupującym.

Przy takich parametrach oba algorytmy znalazły 6 klastrów, z tym że pierwszy algorytm umieścił w drugim klastrze więcej obiektów, które są mniej do siebie podobne. Klastry uzyskane przy użyciu drugiego algorytmu wydają się bardziej spójne, tzn. bardziej zwarte pod względem wartości atrybutu, na którego dziedzinie określona jest funkcja podobieństwa. Elementy do klastra dodawane są ze stopniem przynależności równym stopniowi typowości obiektu w populacji $C_{typical}$.

Tabela 1

Przykładowa populacja oraz klastry uzyskane algorytmem typowe klastry i wersją zmodyfikowaną

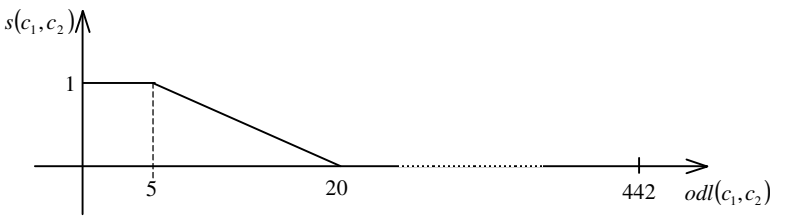
Populacja			Klastry			Klastry uzyskane algorytmem zmodyfikowanym		
1	200	1	1	200	0,26	1	200	0,26
2	220	1	2	220	0,24	2	220	0,24
3	10	1	5	190	0,24	5	190	0,24
4	20	1	8	210	0,26	8	210	0,26
5	190	1	9	200	0,26	9	200	0,26
6	100	1	19	190	0,24	19	190	0,24

7	150	1	14	30	0,21	11	70	0,55
8	210	1	3	10	0,18	10	50	0,48
9	200	1	4	20	0,2	12	50	0,48
10	50	1	10	50	0,19	14	30	0,54
11	70	1	11	70	0,17	3	10	0,47
12	50	1	12	50	0,19	4	20	0,47
13	250	1	16	110	0,18			
14	30	1	20	100	0,18			
15	240	1	6	100	0,18			
16	110	1	15	240	0,16	16	110	0,75
17	130	1	13	250	0,12	6	100	0,67
18	0	1				17	130	0,6
19	190	1				20	100	0,67
20	100	1	17	130	0,15	7	150	0,48
			18	0	0,14	13	250	0,75
						15	240	0,75
			7	150	0,11	18	0	1

Na rysunku 5 przedstawiono efekt działania obu algorytmów na obrazie. Rysunek 5.b przestawia obraz po grupowaniu bez wyznaczania typowości po znalezieniu każdego klastra. Przy zastosowaniu miary podobieństwa między kolorami, opartej na odległości euklidesowej, zdefiniowanej wzorem 9, z reprezentacją graficzną przedstawioną na rysunku 4.

$$s(c_1,c_2)=\begin{cases}1, & odl <= 5\\ -\dfrac{odl-5}{15}+1, & 5 < odl <= 20\\ 0, & odl > 20\end{cases} \tag{9}$$

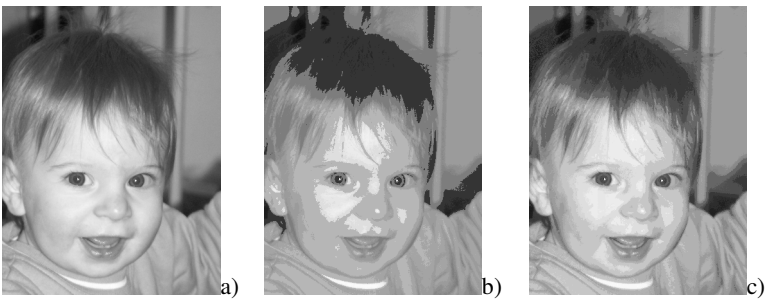
$$odl(c_1,c_2)=\sqrt{(c_1.R-c_2.R)^2+(c_1.G-c_2.G)^2+(c_1.B-c_2.B)^2} \tag{10}$$



Rys. 4. Funkcja podobieństwa między dwoma kolorami, zależna od odległości między tymi kolorami

Algorytm znalazł 30 klastrów z 247 oryginalnych kolorów, i $340 \times 449 = 152660$ obiektów (pikseli). Na rysunku klaster to jeden kolor. Do reprezentacji klastra wybierany jest kolor, z najwyższym stopniem przynależności w klastrze

Zmodyfikowany algorytm znalazł 56 klastrów, przy czym obraz wynikowy jest bardziej podobny do oryginału (rys. 5.c).



a) b) c)

Rys. 5. Wynik działania algorytmu typowego grupowania a) obraz oryginalny, b) algorytm typowej klasteryzacji, c) zmodyfikowany algorytm typowego grupowania.

5. Algorytm znajdowania przybliżonych wartości typowych

Algorytm typowej klasteryzacji ma bardzo dużą złożoność, bo sam algorytm znajdowania typowych wartości ma złożoność $O\left(|C|^2\right)$, gdzie $|C|$ jest liczbą obiektów w populacji C. W [2] autorzy przedstawili algorytm znajdowania przybliżonych wartości typowych. Pseudokod tego algorytmu przedstawiliśmy na rysunku 6.

Danymi wejściowymi dla tego algorytmu są: rozmyty zbiór obiektów O, A — atrybut, dla którego poszukiwana jest typowa wartość oraz zbiór rozłącznych przedziałów $L = \{L_1, ..., L_n\}$ zdefiniowanych w dziedzinie atrybutu A, takich że

$L_j \cap L_k = \varnothing$ jeśli $j \neq k$

i suma przedziałów definiuje całą dziedzinę atrybutu A, a więc

$L_1 \cup ... \cup L_n = range(A)$

1) proc findTypicalValapprox
2) cardArray[n] = [0,...,0]
3) forall object $o_i \in O$ do
4) j = findIntervalIndex($o_i.A$, *L)*
5) cardArray[j] = cardArray[j]+ $\mu_o(a_i)$
6) end
7) forall intervals $L_j \in L$ do
8) $r_j = \sum_k \left(mean(L_j) \approx mean(L_k)\right) * cardArray[k],\ k \in [1,n]$
9) $\mu_j = \dfrac{r_j}{fcard(O)}$
10) extend $typical_A$ with $\{\mu_j / mean(L_j)\}$
11) end
12) return $typical_A$
13) end

Rys. 6. Algorytm znajdowania przybliżonych wartości typowych

Funkcja $mean(L_j)$ zwraca medianę (element, który w uporządkowanej próbce znajduje się na jej środku) przedziału o indeksie j.

Złożoność tego algorytmu jest złożonością rzędu $O(|O|+n^2)$ $|O|$ - liczba obiektów w *O, n* – liczba przedziałów.

Złożoność tę można zmniejszyć do, $O(|O|+n\cdot \max_j(K_j))$, jeśli r_j obliczamy tylko dla indeksu $k \in K_j$, gdzie $mean(L_{jk}) \in [mean(L_j)-\delta, mean(L_j)+\delta]$ i δ jest odległością, dla której relacja podobieństwa jest różna od 0.

Algorytm ten zmodyfikowany na nasz użytek przedstawiono na rysunku 7. Danymi wejściowymi są: populacja C zawierająca piksele opisane przez położenie w obrazie $(p.x, p.y)$ oraz kolor składający się z 3 składowych (R,G,B). Każda ze składowych ma wartość z zakresu $[0;255]$. Jako zbiór przedziałów będziemy traktować zbiór sześcianów wyodrębnionych z sześcianu RGB, poprzez podział każdego z boków na x równych części. Liczba powstałych w ten sposób przedziałów to y^3 ($y = 255/x$). W wyniku otrzymujemy populację $C_{typicalIntervals}$, zawierającą przedziały wraz z ich stopniem typowości.

1. **TypicalValue_Approximated**
2. cardArray[y,y,y] = {0…0; 0…0; 0…0}
3. foreach (object $o_i \in C$)
4. i = $o_i.R / x$; j = $o_i.G / x$; k = $o_i.B / x$;
5. cardArray[i, j, k] += $\mu_C(o_i)$;
6. if ($L_{i,j,k} \notin C_{typicalIntervals}$)
7. $C_{typicalIntervals}.Add(L_{i,j,k})$;
8. $L_{i,j,k}.Add(o_i)$;
9. end
10. foreach (interval $L_{i,j,k} \in C_{typicalIntervals}$)
11. $r_{i,j,k} = 0$;
12. foreach(interval $L_{ii,jj,kk} \in C_{typicalIntervals}$)
13. $r_{i,j,k} += sim(L_{i,j,k}, L_{ii,jj,kk}) \cdot cardArray[i,j,k]$
14. $L_{i,j,k}.\mu = \frac{r_{i,j,k}}{fcard(C)}$;
15. end
16. end
17. return $C_{typicalIntervals}$

Rys. 7. Algorytmu znajdowania przybliżonych wartości kolorów w obrazie

Grupowanie przebiega tak samo jak przy algorytmie z rysunku 2 lub 3, z tym że w tym wypadku grupowane są przedziały. Grupowaniu poddajemy tylko te przedziały, do których trafiły jakieś obiekty, bo zależy nam na znalezieniu grup obiektów. Kolor w obrazie wynikowym jest kolorem przedziału z najwyższym stopniem przynależności w klastrze, do którego należy przedział zawierający obiekt. W przedstawionych obrazach długość przedziału wynosiła 1. Maksymalna liczba niepustych przedziałów dla obrazów w odcieniach szarości wynosi 256. Jednak w wypadku obrazów kolorowych długość przedziału musi być większa, gdyż przy przedziale jednostkowym ich liczba może wynieść $256^3 = 16\,777\,216$. Bardzo dobre rezultaty uzyskaliśmy przy długości boku równej 8 i 16.

a) b) c)

Rys. 8. Wynik działania algorytmów grupujących z przybliżonym wyznaczaniem typowości i funkcją podobieństwa opartą na euklidesowej odległości między kolorami a) obraz oryginalny (213 kolorów) b) algorytm grupujący (34 klastry) c) zmodyfikowany algorytm grupujący (18 klastrów)[2]

Wykonane zostały eksperymenty z zastosowaniem dwóch różnych relacji podobieństwa. Podobieństwo określane dla kolorów będących środkami przedziałów według wzoru 9 (rysunek 8) oraz funkcja podobieństwa oparta na reprezentacji wartości atrybutu jako liczby rozmytej.

5.1. Miara podobieństwa oparta na liczbach rozmytych

Jeśli każdą ze składowych koloru potraktujemy jako trójkątną liczbę rozmytą, reprezentowaną jako $A = (m_A, \alpha_A, \beta_A)$

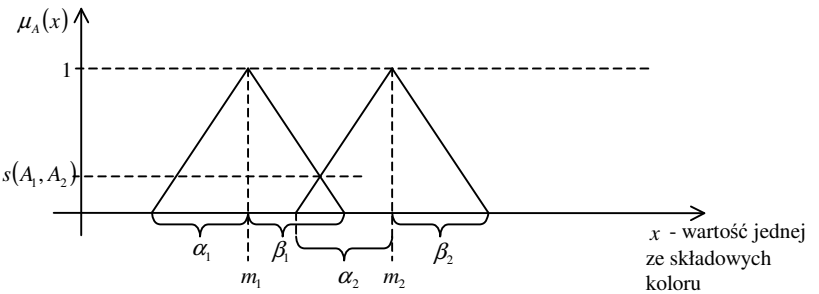
$$\mu_A(x) = \begin{cases} 0 & x \le m_A - \alpha_A \\ \dfrac{x - (m_A - \alpha_A)}{\alpha_A} & m_A - \alpha_A < x \le m_A \\ \dfrac{m_A + \beta_A - x}{\beta_A} & m_A < x < m_A + \beta_A \\ 0 & x \ge m_A + \beta_A \end{cases} \tag{11}$$

to podobieństwo między liczbami rozmytymi $A_1 = (m_1, \alpha_1, \beta_1)$ i $A_2 = (m_2, \alpha_2, \beta_2)$ możemy określić wzorem (12). Graficzną reprezentację tego podobieństwa ilustruje rysunek 9.

[2] Obrazy przestawione na rysunku 8 i 11 pochodzą z bazy [4]

$$s(A_1, A_2) = \begin{cases} 1, & m_1 = m_2 \\ \dfrac{\beta_1 + m_1 - m_2 + \alpha_2}{\alpha_2 + \beta_1}, & (m_1 < m_2) \wedge (m_2 - m_1 \le \beta_1 + \alpha_2) \\ \dfrac{\beta_2 + m_2 - m_1 + \alpha_1}{\alpha_1 + \beta_2}, & (m_2 < m_1) \wedge (m_1 - m_2 \le \beta_2 + \alpha_1) \\ 0, & ((m_1 < m_2) \wedge (m_2 - m_1 > \beta_1 + \alpha_2)) \vee \\ & \vee ((m_2 < m_1) \wedge (m_1 - m_2 > \beta_2 + \alpha_1)) \end{cases} \quad (12)$$

$$s(c_1, c_2) = s(c_1.R, c_2.R) \cdot s(c_1.G, c_2.G) \cdot s(c_1.B, c_2.B) \quad (13)$$



Rys. 9. Podobieństwo między dwiema liczbami rozmytymi

Dla tak zdefiniowanej miary podobieństwa z wartościami $\alpha = \beta = 8$ algorytm grupowania z przybliżonym znajdowaniem typowej wartości dał rezultaty przedstawione na rysunku 10 i 11. W obrazie na rysunku 10 kolor został potraktowany jako trzy oddzielne atrybuty, a miarą podobieństwa obiektów był iloczyn podobieństwa trzech składowych R, G i B, obliczonych wg wzoru 13. Takie potraktowanie koloru zdaje się bardziej zasadne przy obrazach kolorowych (których ze względu na czarno-biały druk nie możemy zaprezentować). Przy odcieniach szarości można sprawdzać tylko jedną ze składowych i obliczać podobieństwo dla tej jednej składowej, bo wszystkie mają taką samą wartość dla danego obiektu. Wyniki z uwzględnieniem takiego podobieństwa prezentuje rys. 11.

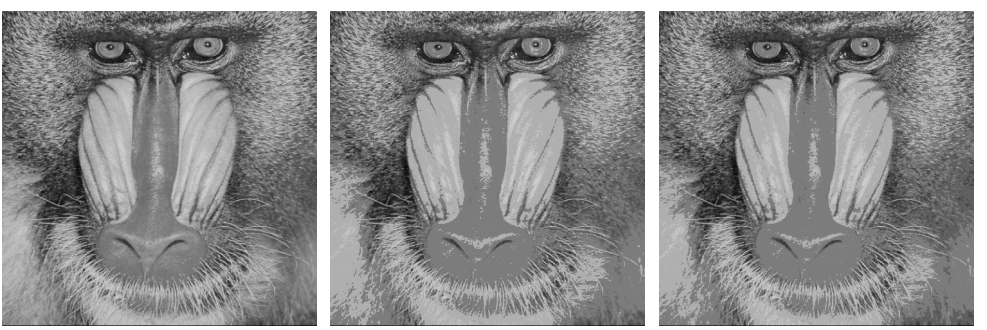
6. Podsumowanie

We wszystkich przedstawionych eksperymentach kryterium stopu algorytmu grupującego było opróżnienie populacji $C_{typical}$, tak aby pogrupowane były wszystkie obiekty. Algorytm zmodyfikowany daje lepsze efekty grupowania. W obrazach

czarno-białych tworzy mniej klastrów, przy czym obraz jest bardziej podobny do oryginału. W eksperymentach prowadzonych na obrazach kolorowych algorytm zmodyfikowany przeważnie tworzył większą liczbę klastrów, cały czas jednak lepiej odwzorowując oryginał. Oczywistą wadą tej modyfikacji jest złożoność, która przy dużych populacjach lub też (w przypadku algorytmu wyznaczającego przybliżone wartości typowe) szerokich dziedzinach atrybutów, których podział na przedziały nie daje wcale mniejszej ich liczby niż liczebność populacji, jest nie do przyjęcia. W przeciwnym wypadku użycie algorytmu wydaje się zasadne w celu uzyskania lepszych efektów.

a) b) c)

Rys. 10. Wynik działania algorytmu grupującego z przybliżonym znajdowaniem typowości a) obraz oryginalny (247 kolorów), b)algorytm grupujący 30 klastrów, c) zmodyfikowany algorytm grupujący 21 klastrów.



a) b) c)

Rys. 11. Wynik grupowania z przybliżonym wyznaczaniem typowości z funkcją podobieństwa opartą na liczbach rozmytych. a) obraz oryginalny kolorów 218, b) algorytm grupowania (34 klastry), c) zmodyfikowany algorytm grupowania (16 klastrów)

Literatura

[1] Rasmussen D.: *Application of the Fuzzy Query Language – Summary SQL*, DATALOGISKE SKRIFTER, Roskilde University, 1997
[2] Rasmussen D., Yager R. R.: *Introdution of Fuzzy Characteristic Rules by Typical Values*, DATALOGISKE SKRIFTER, Roskilde University, 1997
[3] Smolińska M. K., Sosnowski Z. A.: *Podsumowania lingwistyczne w obiektowych bazach danych*, XIII Warsztaty Naukowe PTSK, Symulacja w Badaniach i Rozwoju, Wrzesień 2006
[4] *The USC_SIPI Image Database,* [http://sipi.usc.edu/database/index.html] University of Southern California

LINGUISTIC SUMMARIES WITH FUZZY CLUSTERING

Abstract: This paper presents linguistic summary as a fuzzy predicate, which is used to find, objects with typical values of an attribute or a set of attributes. In the fuzzy clustering algorithm we use population with given typicality of objects for selected attribute. We present the results of this algorithm and its modification basing on an example with population of pixels in image.

Keywords: linguistic summaries, fuzzy clustering

Artykuł zrealizowano w ramach prac badawczych S/WI/2/03 i 3T11F01130.

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2007
Informatyka – Zeszyt 2

Paweł Tadejko[1]

ADDITIONAL DATA PREPROCESSING AND FEATURE EXTRACTION IN AUTOMATIC CLASSIFICATION OF HEARTBEATS

Abstract: The paper presents the classification performance of an automatic classifier of the electrocardiogram (ECG) for the detection abnormal beats with new concept of feature extraction stage. Feature sets were based on ECG morphology and RR-intervals. This paper compares two strategies for classification of annotated QRS complexes: based on original ECG morphology features and proposed new approach - based on preprocessed ECG morphology features. The mathematical morphology filtering and wavelet transform is used for the preprocessing of ECG signal. Within this framework, the problem of choosing an appropriate structuring element in mathematical morphology filtering in signal processing was studied. Configuration adopted a Kohonen self-organizing maps (SOM) and Support Vector Machine (SVM) for analysis of signal features and clustering. In this study, a classifiers was developed with LVQ and SVM algorithms using the data from the records recommended by ANSI/AAMI EC57 standard. The performance of the algorithm is evaluated on the MIT-BIH Arrhythmia Database following the AAMI recommendations. Using this method the results of identify beats either as normal or arrhythmias was improved.

Key words: ECG, preprocessing, mathematical morphology, ECG filtering, wavelet approximation, feature extraction, heartbeat classification

1. Introduction

The analysis of heart beat cycles in ECG signal is very important for long-term monitoring of heart patients. Automatic classification of cardiac rhythms remains a vital problem in clinical cardiology, especially when it is performed in real time. Several researchers have addressed the problem of automatic classification of cardiac arrhythmias [1-7]. Annotation of ECG recording requires the detection of various types of heartbeats. This is a pattern recognition task. Very often, a

[1] Technical University of Bialystok, Faculty of Computer Science, Bialystok

classifier is to be trained to recognize different types of beats. The training set of the classifier is usually a large database, which consists of the ECG beats from a large pool of patients. However, these classifiers suffer from the problem of poor generalization because there are usually some variations in the "normal" range among human beings. Even doctors may experience difficulty in assessing abnormal ECG beats if only considering the reference values based on the general patient population.

There are two general approaches to the training process: building general heartbeat classifier [3-7] and patient-adapting classifier [1,2]. However, most of the approaches proposed in the literature deal with a limited number of arrhythmic types and process the entire ECG signal extracting several features from it, such as the P wave, which is an extremely time-consuming process and sometimes difficult due to the presence of noise. Other researches suggested that there is a need to incorporate local information of a specific patient to improve the recognition of abnormal ECG beats and thus help to improve the generalization.

This work proposes a method for the unification of variations in ECG beats, based on mathematical morphology and resampling model, which can be easily applied to the ECG signal.

2. Mathematical morphology

By "morphological signal processing" we mean a broad and coherent collection of theoretical concepts, mathematical tools for signal analysis [12]. Originally mathematical morphology (MM) was applied to analyzing images from geological or biological specimens. However, its rich theoretical framework, algorithmic efficiency, easy implementability on special hardware, and suitability for many shape-oriented problems have propelled its widespread diffusion and adoption by many academic and industry groups in many countries as one among the dominant image analysis methodologies [8-11].

As a result, MM nowadays offers many theoretical and algorithmic tools to and inspires new directions in many research areas from the fields of signal processing, image processing and machine vision, and pattern recognition.

2.1. Mathematical morphology transformations

Morphological filters are nonlinear signal transformations that locally modify geometric features of signals. They stem from the basic operations of a set-theoretical method for signal analysis, called mathematical morphology, which was introduced by Serra [12].

In morphological filtering [8-11], each signal is viewed as a set, and its geometrical features are modified by morphologically convolving the signal with a structuring element (SE), which is another set of simple shape and size. By varying the structuring element we can extract different types of information from the signal.

A structuring element is characterized by its shape, width, and height. Its width, or length, is largely determined by the duration of the major waves and the sampling rate. The values of the structuring element determine the shape of the output waveform.

2.2. Elementary mathematical morphology operators

In the sequel we use definitions of grey-level morphology basic operators in the same form as in [12]. Let us recall that *erosion* Θ of a function $f : R \to R$ by a structuring element $b : R \to R$ can be defined as

$$(f \Theta b)(s) = \min_x \{ f(s+x) - b(x) : (s+x) \in D_f \wedge x \in D_b \} \tag{2}$$

where $D_f = \text{supp}\, f$, $D_b = \text{supp}\ b$. In a similar way, *dilation* $\oplus$ is an operator given by

$$(f \oplus b)(s) = \max_x \{ f(s-x) + b(x) : (s-x) \in D_f \wedge x \in D_b \} \tag{3}$$

Two other operators: *closing* $\bullet$ and *opening* $\circ$ are defined with help of (2) and (3), i.e.

$$f \bullet b = (f \oplus b) \Theta b, \quad f \circ b = (f \Theta b) \oplus b \tag{4}$$

3. SOM and LVQ

3.1. Self-Organization Map

The Self-organizing Map (SOM) is an artificial neural network architecture based on unsupervised, competitive learning [14]. It provides a topology preserving, smooth mapping from a high-dimensional input space to the map units usually arranged as a two-dimensional lattice of neurons (nodes). Thus, the SOM can serve as a tool for cluster analysis of complex, high-dimensional data.

A parametric reference vector *m*, is associated with every node. A data vector *x* is compared to all reference vectors in any metric and the best matching node is defined, e.g., by the smallest Euclidean distance between the data vector and any of the reference vectors. During learning, those nodes that are topographically close

in the array up to a certain distance will activate each other to learn from the same input:

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)] \tag{5}$$

where t is an integer representing time, and $h_{ci}(t)$ is the so-called neighbourhood kernel describing the neighbourhood that is updated around the best-matching node. Several suitable kernels can be used, e.g. a so-called bubble kernel or a gaussian kernel, relating to different ways of determining the activating cells. The kernel also includes the learning rate parameter $\alpha(t)$. With time, the size of the neighbourhood and the learning rate are diminished. The described learning process leads to a smoothing effect on the weight vectors in the neighbourhood and by continued learning to global ordering of the nodes [14,15].

The SOM consists of a two-dimensional lattice that contains a number of neurons. These neurons are usually arranged in a rectangular or hexagonal way. The position of the units in the grid, especially the distances between them and the neighborhood relations, are very important for the learning algorithm. A prototype vector (also "model" or "codebook" vector) is associated with each neuron, which is a vector of the same dimension as the input data set. This prototype vector approximates a subset of the sample vectors. The dimension of the sample is called input dimension, and is usually larger than 2, the dimension of the lattice, which is called output dimension. For training and visualization purposes, the sample vectors are assigned to the most similar prototype vector, or best-matching unit (BMU), formally

$$c(x) = \arg\min_i \{|x - m_i(t)|\} \tag{6}$$

The learning process itself gradually adapts the model vectors to match the samples and to reflect their internal properties as faithfully as possible, which means that input vectors which are relatively close in input space should be mapped to units that are relatively close on the lattice.

3.2. Learning Vector Quantization

Learning Vector Quantization (LVQ) [14, 22] is a supervised, clustering-based classification technique which classifies a feature vector according to the label of the cluster prototype (code word) into which is clustered. Classification error occurs when the feature vectors within the same cluster (hence, assigned to the same class label) are actually drawn from different classes. To minimize classification error, the LVQ algorithm fine tunes the clustering boundary between clusters of different class labels by modifying the position of the clustering center

(prototype or code word). This method is called "learning vector quantization" because this clustering based classification method is similar to the "vector quantization" method used for signal compression in the areas of communication and signal processing.

Assume that a number of 'codebook vectors' $m_i(t)$ (free parameter vectors) are placed into the input space to approximate various domains of the input vector *x* by their quantized values. Usually several codebook vectors are assigned to each class of *x* values, and *x* is then decided to belong to the same class to which the nearest mi belongs. According to (6) we defined the nearest $m_i(t)$ to *x*, denoted by *mc*. Values for the $m_i(t)$ that approximately minimize the misclassification errors in the above nearest-neighbor classification can be found as asymptotic values in the following learning process. Starting with properly defined initial values, the following equations define the basic LVQ1 process:

$$mc(t+1) = mc(t) + \alpha(t)[x(t) - mc(t)], \tag{7}$$

if *x* and *mc* belong to the same class,

$$mc(t+1) = mc(t) * \alpha(t)[x(t) - mc(t)], \tag{8}$$

if *x* and *mc* belong to different classes,

$$m_i(t+1) = m_i(t) \tag{9}$$

for *i* not in *c*,

Here $0 < \alpha(t) < 1$, and $\alpha(t)$ may be constant or decrease monotonically with time. In the above basic LVQ1 it is recommended that alpha should initially be smaller than 0.1

4. Support Vector Machine

Support Vector Machine is a method for finding a hyperplane in a high dimensional space that separates training samples of each class while maximizes the minimum distance between the hyperplane and any training samples [26]. SVM approach can apply any kind of network architecture and optimization function.

We are given a set of *N* data points $\{x_i, y_i\}_{i=1}^{N}$, where $x_i \in \mathbf{R}^n$ is the *i*-th input data, and $y_i \in \{-1, +1\}$ is the label of the data. The SVM approach aims at finding a classifier of form:

$$y(x) = sign\left[\sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b\right] \tag{10}$$

Where α_i are positive real constants and b is a real constant, in general, $K(x_i, x) = \langle \phi(x_i), \phi(x) \rangle$, $\langle \bullet, \bullet \rangle$ is inner product, and $\phi(x)$ is the nonlinear map from original space to the high dimensional space.
In the high dimensional space, we assume the data can be separated by a linear hyperplane, this will cause:

$$y_i \left[w^T \phi(x_i) + b \right] \geq 1, \quad i = 1, \ldots, N \tag{11}$$

In case of such separating hyperplane does not exist, we introduce a so called slack variable ξ_i such that

$$\begin{cases} y_i \left[w^T \phi(x_i) + b \right] \geq 1 - \xi_i, & i = 1, \ldots, N \\ \xi_i \geq 0, & i = 1, \ldots, N \end{cases} \tag{12}$$

According to the structural risk minimization principle, the risk bound is minimized by the following minimization problem:

$$\min_{w,\xi} J_1(w, \xi) = \frac{1}{2} w^T w + c \sum_{i=1}^{N} \xi_i \tag{13}$$

subject to (12). One constructs the Lagrangian function as

$$L_1(w, b, \xi, \alpha, \beta) = J_1(w, \xi) - \sum_{i=1}^{N} \alpha_i \left\{ y_i \left[w^T \phi(x_i) + b \right] - 1 + \xi_i \right\} - \sum_{i=1}^{N} \beta_i \xi_i \tag{14}$$

where $\alpha_i \geq 0$, $\beta_i \geq 0$ $(i = 1, \ldots, N)$ are the Lagrangian multipliers. The optimal point will in the saddle point of the Lagrangian function, i.e.

$$\max_{\alpha,\beta} \min_{w,b,\xi} L_1(w, b, \xi, \alpha, \beta) \tag{15}$$

Substitute (10), we will get the following quadratic programming problem:

$$\max_{\alpha} Q_1(\alpha) = -\frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) + \sum_{i=1}^{N} \alpha_i \,. \tag{16}$$

where $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$ is called the kernel function. Solving this QP problem subject to constrains in (13), we will get the hyperplane in the high dimensional space and the classifier in the original space as in (10).

5. Evaluation method

The proposed method consists of four steps: (a) preprocessing (denoising, baseline drift removal and feature extraction), (b) additional data filtering and wave form transformation (feature extraction), (c) building feature vector (feature selection) (d) arrhythmic episode classification. The MIT-BIH arrhythmia database [17] is used for evaluation of the method.

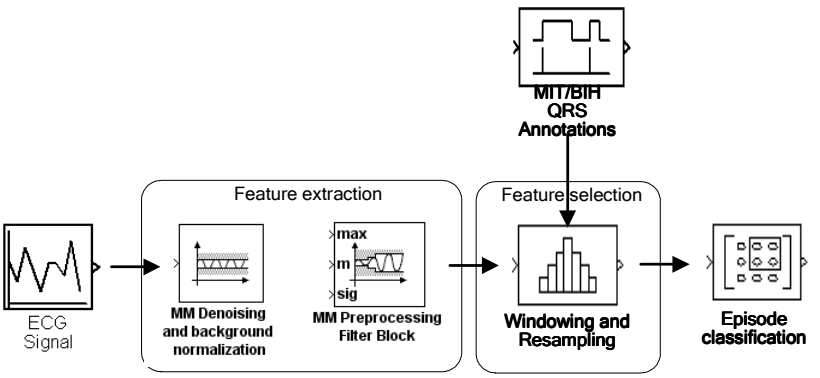


Fig. 1. Block diagram of the proposed morphological filtering based transformation.

We evaluate various combinations of morphological filters and conduct experiments for different structuring elements [18,19] in feature extraction steps. It turns out that in results are strong depends on shape and size of structuring element. Since the opening and closing operations are intended to remove impulses, the structuring element must be designed so that the waves in the ECG signal are not removed by the process.

5.1. Datasets

To evaluate the performance of our approaches, we used the 48 tapes of the MIT-BIH arrhythmia database, which comes along with a very detailed annotation for each beat. The Association for the Advancement of Medical Instrumentation (AAMI) has summarized those detailed classes to four classes of clinical relevance, as shown in table 1 [20]. Four records (102, 104, 107, and 217), including paced beats, are excluded from the study in compliance with the standards recommended for reporting performance results of cardiac rhythms by the AAMI.

The original signals in the MIT-BIH arrhythmia database are two-leads, sampled at 360 Hz. The ECG signal of Lead 1 is used in this study. Accompanying

each record in the database is an annotation file in which each ECG beat has been identified by expert cardiologists. These labels are referred to as 'truth' annotations and are used in developing the classifiers and also to evaluate the performance of the classifiers in the testing phase.

Table 1

Beat classes according to MIT-BIH arrhythmia database and AAMI recommended practice

MIT-BIH heartbeat types	AAMI heartbeat class	Notation used in this work
normal beat (NOR),	NORMAL (N) Any heartbeat not in the other classes	NL
left bundle branch błock beat (LBBB),		LB
rignt bundle branch block beat (RBBB),		RB
atrial escape beats (AE),		AE
nodal (junctional) escape beat (NE)		NE
atrial premature beat (AP),	SVEB (S) Supraventricular ectopic beat	AP
aberrated atrial premature beat (aAP),		AA
nodal (junctional) premature beat (NP),		NP
supraventricular premature beat (SP)		SP
premature ventricular contraction (PVC),	VEB (V) Ventricular ectopic beat	PV
ventricular escape beat (VE)		VE
fusion of ventricular and normal beat (fVN)	FUSION (F) Fusion beat	FS
paced beat (P),	NOTQRS (Q) Unknown beat	PB
fusion of paced and normal beat (fPN),		PF
unclassified beat (U)		NQ

5.2. Feature extractions

Many experiments have been done to test the performance of morphological filters used in ECG signal preprocessing [18,19]. The experiments show that nonstandard filter block construction, especially combination of elementary morphology operators (as sequence operations), has very big impact for characteristic of output signal. The morphological filter preprocessing stage was implemented using the MATLAB programming languages.

We propose a feature-preserved transformation for signal processing of ECG data, based on mathematical morphology (MM) filtering. Despite the fact that MIT-BIH [17] database consist of wide spectrum of ECG signals we obtained similar results for all data. For extraction and unification of the ECG features, the

length of the SE should be less than QRS interval. Therefore, we chose 70 ms as the length of the SE. Here, we take the flat SE to illustrate preprocessing effect.

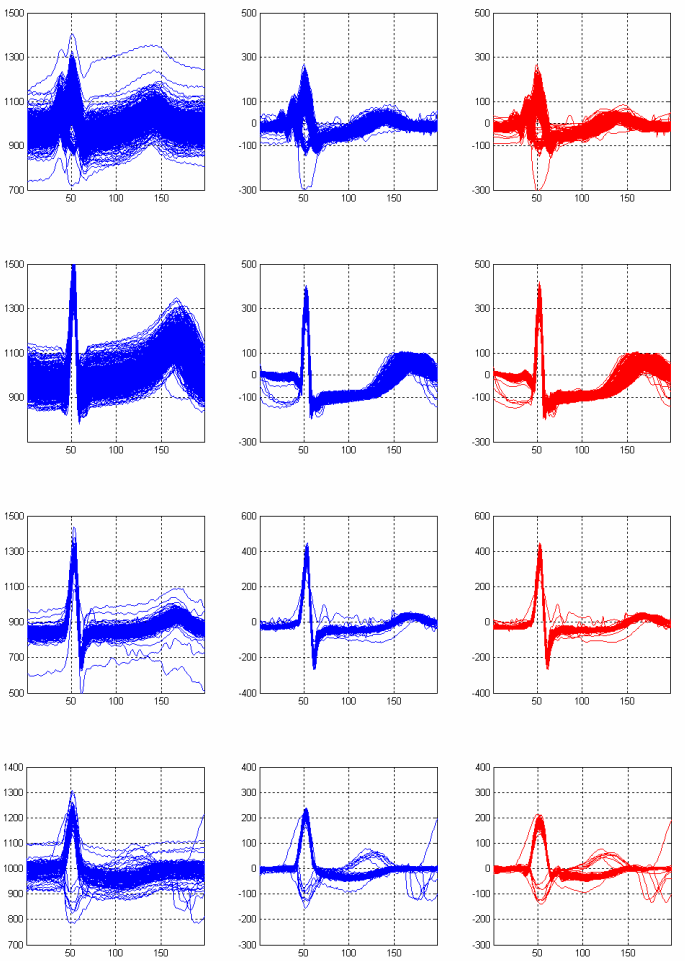


Fig. 2. Feature extraction stages combination: A – column 1) without preprocessing, B – column 2) only with MM denoising, baseline drift removal, C – column 3) MM denoising, baseline drift removal (1-st stage) and additional MM filtering block (2-nd stage). Record e.g. 111, 113, 123, 210 (top-down) from MIT-BIH Arrhythmia Database.

Also we evaluate various wavelets (WT) in feature extraction steps. Wavelets are well suited to approximate such signals when nonlinear approximation is allowed [25]. The compression features of a given wavelet basis are primarily linked to the relative scarceness of the wavelet domain representation for the signal. The notion behind compression is based on the concept that the regular

signal component can be accurately approximated using the following elements: a small number of approximation coefficients (at a suitably chosen level) and some of the detail coefficients.

To study compression behavior we done many experiments to test the performance of wavelet approximation used in ECG signal preprocessing. Here, we take the two wavelets: daubechies and symlet to illustrate preprocessing effect. For our experiments, we choose the decomposition of the signals at level *n* (where *n=5..7*) and forced to zero detail coefficients for level *1..n-2*.

5.3. Feature selection

Here we investigate the use of raw amplitude of the time domain ECG signals after noise suppression, baseline drift removal and additional preprocessing as feature vectors to represent the ECG beats. After the R-peak is found (using MIT-BIT database annotations), the ECG signal in a window of 550 ms is taken as an ECG beat. The lengths of the signal before and after the R-peak in each beat are 140 ms and 410 ms, respectively, such that the window covers most of the characterization of the ECG beat. The signal in each window is then resampled to form a feature vector of 20-dimensions. The R-R interval (the interval between two consecutive R-peaks) is also used in this study by appending it to the 20-dimensional feature vector.

5.4. Feature classification

In this study, a classifiers was developed with SOM/LVQ and SVM algorithms.

5.4.1. Clustering by SOM

The issue of SOM quality is a complicated one [16]. Typically two evaluation criterias are used: resolution and topology preservation. There are many ways to measure them. The ones used here were chosen for their simplicity:

- qe (quantization error) - Average distance between each data vector and its BMUs (best matching units). Measures map resolution,

The average quantization error is measure used for this purpose. In order to study the behaviour of these factors we chose hexagonal topology map with automatic determine map size.

We used the method to allocate manually detected heartbeats to one of the five or fifteen beat classes (Table 1) recommended by ANSI/AAMI EC57:1998 standard, i.e., normal beat (NORMAL), ventricular ectopic beat (VEB),

supraventricular ectopic beat (SVEB), fusion of a normal and a VEB (FUSION), or unknown beat type (NOTQRS) and second group with notation used in this work.

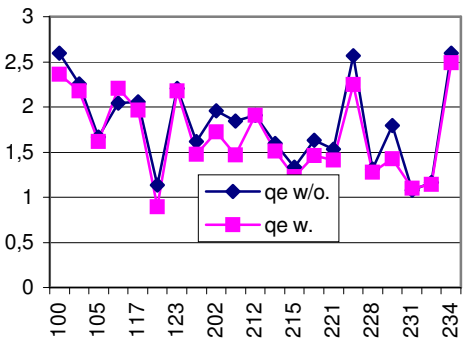


Fig. 3. Quantization error for each recording of the test set.

Figure 2 shows the quantization error for all MIT/BIH datasets. Average distance between each data vector and its BMUs is smaller for almost all records.

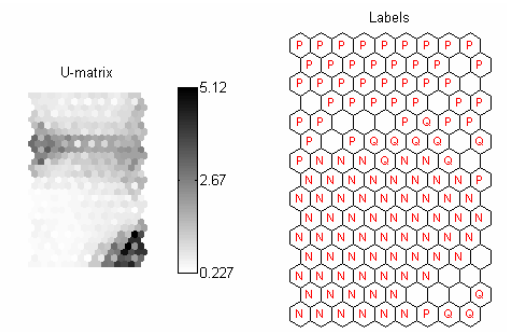


Fig. 4. Distance matrix U-matrix and labes for clustering of record 200 (MIT/BIH) without preprocessing; i.e. NORMAL (N), VEB (V), SVEB (S), FUSION (F), NOTQRS (Q)

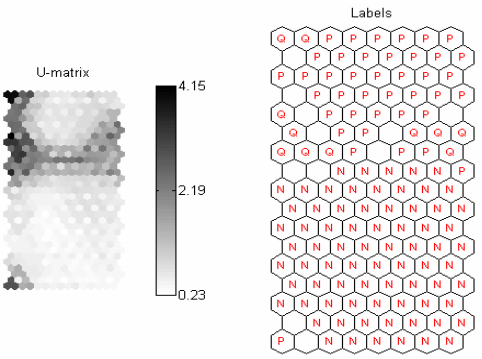


Fig. 5. Distance matrix U-matrix and labes for clustering of record 200 (MIT/BIH) with preprocessing; i.e. NORMAL (N), VEB (V), SVEB (S), FUSION (F), NOTQRS (Q)

Clusters on the map are typically visualized using the unified distance matrix (U-Mmatrix) method. The U-matrix of the example system is shown in Fig. 3 (without preprocessing) and Fig 4 (with additional preprocessing). The lighter the color of a unit, the, the closer it is to its neighbors.

It is demonstrated that the map direction is changing smoothly on the SOM. Moreover, the fully trained SOM did not include any meaningless or random maps. Of all teaching runs, the SOM that resulted in the lowest quantization error of the test set was chosen for further classification.

5.4.1. LVQ training and classification

According to Kohonen, there are three different LVQ algorithms, called LVQ1, LVQ2, and LVQ3 developed at subsequent stages to handle classification problems with different natures. In this study, the learning-rate LVQ1 algorithm was used for the training and fine-tuning of the code book respectively. This stage is based on applying the training and accuracy classification using the MATLAB SOM Toolbox [21] and LVQ_PAK [22].

5.4.2. Support Vector Machine (SVM)

We have performed many experiments to determine best classification performance. We have used Gaussian radial kernels with $\gamma = 1$. The parameter C used in experiments was set to *1000*. Our experiments confirmed that classification ratio strongly depends on this parameters.

Waikato Environment for Knowledge Analysis (WEKA) [29] workbench was used for the experiments. We ran experiments with SVM algorithm that is

available as Weka's implementation with SMO approach. Sequential Minimal Optimization (SMO) is a fast method to train SVM. SMO breaks this large QP problem into a series of smallest possible QP problems. These small QP problems are solved analytically, which avoids using a time-consuming numerical QP optimization as an inner loop. The amount of memory required for SMO is linear in the training set size, which allows SMO to handle very large training sets.

6. Results and discussion

The method allocates manually detected heartbeats (using MIT/BIT database annotations) to one of the classes showed in Table 1 (MIT-BIH heartbeat types). The labels in the annotation files of MIT-BIH database made by cardiologists are used as the ground truth in evaluating the classifier.

Classifier performance has been estimated using MIT-BIH heartbeat types showed in Table 1. Because SMO algorithm and memory limitations all recordings were divided into two datasets with each dataset containing ECG data from 22 recordings with the same approximate proportion of beat types. Both datasets contain a mixture of the routine and complex arrhythmia.

Dataset in this study comprises data from recordings 100, 103, 105, 111, 113, 117, 121, 123, 200, 202, 210, 212, 213, 214, 219, 221, 222, 228, 231, 232, 233, 234. The available data was used as training and testing data. Training stage containing the data from 22 records and then we test classifier for each record. To determine the classification performance of our method, on next step each record (LVQ) and ten test sets (SVM) was processed as testing data.

6.1. Evaluation of results

Table 2 shows classification performance using LVQ and SVM classifier respectively. Some interesting trends emerge from these results. The results shows that the classification performance for PV (premature ventricular contraction - PVC) preprocessed by hybrid wavelet-MM filter are notably higher than the same resulting without wavelet compression stage. Specifically, accurate detection of premature ventricular contractions (PVCs) is imperative to prepare for the possible onset of lifethreatening arrhythmias.

According to AAMI recommendations classes, we see that also aggregate classification performance for normal beats (NL, LB, RB, AE, NE) from any others heartbeat may be very high performance with this method for both classifier (LVQ and SVM).

Table 2
Classification performance of MIT-BIH heartbeat type each recording using the LVQ and SVM algorithms on MM preprocessed and MM+WT preprocessed ECG data, (better results in gray).

Performance [%]	MIT-BIH Arrhythmia Database heartbeat types												
	NL	AP	NQ	PV	PB	LB	AA	FS	VE	RB	AE	NE	NP
total beats	9175	432	306	788	3	1021	8	106	1	881	35	43	12
w/o preproc.	95,99	86,34	92,81	80,71	0,00	49,56	0,00	56,60	0,00	87,63	97,14	58,14	16,67
only 1st MM (LVQ)	96,70	85,90	97,70	92,80	0,00	97,40	0,00	78,30	0,00	95,00	94,30	44,20	16,70
and 2nd MM (LVQ)	93,65	82,87	90,85	81,47	0,00	51,32	0,00	70,75	0,00	94,10	97,14	39,53	0,00
and 2nd WT^1 (LVQ)	97.20	79.86	42.16	85.41	0,00	92.65	0,00	69.81	0,00	87.74	91.43	41.86	0,00
and 2nd WT^2 (LVQ)	94.87	85.88	93.46	88.32	0,00	51.03	0,00	70,75	0,00	85.81	97,14	69.77	0,00
only 1st MM (SVM)	96,70	85,90	97,70	92,80	0,00	97,40	0,00	78,30	0,00	95,00	94,30	44,20	16,70
and 2nd MM (SVM)	94.87	85.88	93.46	88.32	0,00	51.03	0,00	70,75	0,00	85.81	97,14	69.77	0,00
and 2nd WT (SVM)	89.17	85.65	90.20	82.11	0,00	51,22	12.50	66.04	0,00	86.04	97,14	37.21	0,00

For our experiments, we choose Deubechies wavelets (Table 2, db2 as $WT^1$and db6 as WT^2). The meaning of the labels in Table 2 are as follow:

- w/o preproc - without preprocessing,
- only 1st (MM) - only with MM denoising, baseline drift removal,
- and 2nd (MM) - MM denoising, baseline drift removal (1st stage) and additional MM filtering block (2nd stage),
- and 2nd (WT) - MM denoising, baseline drift removal (1st stage) and additional wavelet transformation (2nd stage).

6.2. Comparison with other works

Reporting performance results of cardiac rhythm algorithms has been standardized by the Association for the Advancement of Medical Instrumentation (AAMI) [20]. However, despite the existence of this standard very few studies have reported performance results in own format.

Due to memory limitation of SMO implementation of SVM algorithm we have used dataset containing 12811 ECG beats from MIT-BIH Arrhythmia Database for learning and testing. In the numerical experiments we have used ECG data of 13 heart rhythm types corresponding to the normal sinus rhythm (N) and 12 types of arrhythmias.

Chazal *et al.* [2] achieved a 97.4% accuracy in separating VEBs from non-VEBs (Table I) and 94.6% accuracy in separating SVEBs from non-SVEBs, used the same 22 records but the heartbeat P-QRS-T points times (e.g. QRS onset and offset and T-wave offset times; a boolean value indicating the presence/absence of

a P-wave and, if present, the P-wave onset and offset) provided with the MIT-BIH were manually verified.

Lagerholm *et al.* [23] described a method for clustering ECG heartbeats from a recording into 25 clusters and determined that on average 98.5% of the heartbeats in any one cluster were from the same heartbeat class. However Lagerholm's system used 25 clusters, and need annotating of the dominant beat of a cluster by an expert before the classification of a cluster can be made to one of 16 types of MIT-BIH beat types.

Osowski *et al.* [28] achieved a 95.9% sensitivity in separating the same 13 heart rhythm types like ours. Despite that beats come from 52 patients of the database as a result of such strategy for data selection, we don't know how the records were subdivide. The total number of data used in learning neural networks was equal 6690 and 6095 data points have been left for testing. According to [28] 12785 beats used in learning and testing.

Table 3

Classification performance of MIT-BIH heartbeat type each recording using the LVQ and SVM algorithms on MM preprocessed and MM+WT preprocessed ECG data, (better results in gray).

Method	Perf. [%]	No. of classes	Descriptions
Present work (only MM)	95,53	13	used the same 22 from MIT-BIH, 12811 ECG beats for
Present work (MM / WT)	97,82	13	training and cross-testing
Chazal *et al.* [1] V-nV	97,40	2	used the same 22 records but the heartbeat P-QRS-T
Chazal *et al.* [1] SV-nSV	94,60	2	points times provided with the MIT-BIH were manually verified
Lagerholm *et al.* [2]	98,50	25	used 25 clusters, but need annotating of the dominant beat of a cluster by an expert before the classification of a cluster can be made to one of 16 types of MIT-BIH beat types
Osowski *et al.* [3]	95,90	13	12785 beats used in learning and testing, come from 52 patients – but don't know strategy for data selection

Additionally feature vectors for all these approaches was totally different: characteristic P-QRS-T points times, Hermite functions coefficients and resampled P-QRS-T ECG morphology window.

7. Conclusions

Results showed that the proposed hybrid algorithm leads to an improvement in the heartbeat classification using MIT-BIH arrhythmia database, especially good recognition rates have been achieved for those beats, for which the large number of cases were available in the database. The efforts to improve the performance of the algorithm and to investigate its effects on the subject are currently continuing.

The experiment results show that using mathematical morphology and wavelet model together can result in high accuracy of classification. This confirms that a highly reliable classifier can be obtained by combining a number of classifiers.
A more comprehensive study is necessary to assess the utility of this method. This is our preliminary work for applying hybrid wavelet-MM filtering to ECG data for normalization patient-specific features.

The future research will be oriented on the improvement of the performance of the presented algorithm, e.g. build more suitable feature vector containing RR-interval features, heartbeat interval features and ECG morphology features and training and testing on all records from MIT-BIH database.

Research for this paper was supported in part by grant from Technical University of Bialystok, no. W/WI/13/07.

References:

[1] Hu, Y.H., Palreddy, S., Tompkins, W.J.: *A patient-adaptable ECG beat classifier using a mixture of experts approach,* IEEE Transactions on Biomedical Engineering, Volume 44, Issue 9, pp. 891-900, 1997.

[2] de Chazal, P., Reilly, R.B.: *A patient-adapting heartbeat classifier using ECG morphology and heartbeat interval features*, IEEE Transactions on Biomedical Engineering, Volume 53, Issue 12, pp. 2535-2543, 2006.

[3] Zhang, Z.G., Jiang, H.Z., Ge, D.F., Xiang, X.J.: *Pattern recognition of cardiac arrhythmias using scalar autoregressive modeling*, Fifth World Congress on Intelligent Control and Automation WCICA, Volume 6, pp. 5545-5548, 2004.

[4] Shyu, L.Y., Wu, Y.H., Hu, W.: *Using wavelet transform and fuzzy neural network for VPC detection from the holter ECG*, IEEE Transactions on Biomedical Engineering, Volume 51, Issue 7, pp.1269-1273, 2004.

[5] de Chazal, P., O'Dwyer, M., Reilly, R.B.: *Automatic classification of heartbeats using ECG morphology and heartbeat interval features*, IEEE

Transactions on Biomedical Engineering, Volume 51, Issue 7, pp. 1196-1206, 2004.

[6] Mitra, S., Mitra, M., Chaudhuri, B.B.: *A rough-set-based inference engine for ECG classification*, IEEE Transactions on Instrumentation and Measurement, Volume 55, Issue 6, pp. 2198-2206, 2006.

[7] Khadra, L., Al-Fahoum, A.S., Binajjaj, S.: *A quantitative analysis approach for cardiac arrhythmia classification using higher order spectral techniques*, IEEE Transactions on Biomedical Engineering, Volume 52, Issue 11, pp. 1840-1845, 2005.

[8] Chu-Song, C., Ja-Ling, W., Yi-Ping, H.: *Theoretical aspects of vertically invariant gray-level morphological operators and their application on adaptive signal and image filtering*, IEEE Trans. on Signal Processing, Volume 47, no. 4, 1999.

[9] Maragos, P., Schafer, R.W.: *Morphological systems for multidimensional signal processing*, Proceedings of the IEEE, Volume 78, no. 4, 1990.

[10] Maragos, P., Schafer, R.W.: *Morphological filters-part I: Their set-theoretic analysis and relations to linear shift-invariant filters*, IEEE Trans. on Acoustic, Speech and Signal Processing, Volume ASSP-35, no. 8, 1987.

[11] Maragos, P.: *Morphological Signal and Image Processing*, CRC Press, 2000.

[12] Serra, I.: *Image Analysis and Mathematical Morphology*, New York Academic, 1982.

[13] Köhler, B-U., Hennig, C., Orglmeister, R.: *The Principles of Software QRS Detection*, IEEE Engineering in Medicine and Biology, 2002.

[14] Kohonen, T.: *Self-organizing Maps*, Springer-Verlag, 1995.

[15] Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J.: *SOM_PAK - The Self-organizing Map Program Package*, 1995.

[16] Kivimoto, K.: *Topology Preservation in SOM*, Proc. of International Conference on Neural NetWorks. Washington, DC, Volume 1, pp. 294-300, 1996.

[17] Harvard-MIT Division of Health Sciences and Technology, *MIT/BIH Arrhythmia Database: The MIT/BIH arrhythmia database CD-ROM*, [http://www.physionet.org/physiobank/database/mitdb/], 1992.

[18] Tadejko, P., Rakowski, W.: *Matlab Simulink in modeling morphological filters for electrocardiogram signal processing*, Simulation in Research and Expansion, XIII Science Work-shop, PTSK, 2005.

[19] Piekarski, K., Tadejko, P., Rakowski, W.: *Properties of Morphological Operators Applied to Analysis of ECG Signals*, Biometrics, Computer Security Systems and Artificial Intelligence Applications, Springer US, p. 279-288, 2006.

[20] Mark, R., Wallen, R.: *AAMI-recommended practice: Testing and reporting performance results of ventricular arrhythmia detection algorithms*, Association for the Advancement of Medical Instrumentation, Arrhythmia Monitoring Subcommittee, AAMI ECAR, 1987.
[21] J. Vesanto, J. Himberg, E. Alhoniemi, J. Parhankangas, *SOM Toolbox for Matlab,* Som Toolbox Team, Helsinki University of Technology, Finland, http://www.cis.hut.fi/projects/somtoolbox/ ver. 2, 2007
[22] T. Kohonen, J. Kangas, J. Laaksonen, K. Torkkola, *LVQ_PAK: A program package for the correct application of Learning Vector Quantization algorithms*, In Proceedings of the International Joint Conference on Neural Networks, Baltimore, IEEE, vol. 1, pp. 725-730, 1992
[23] M. Lagerholm, C. Peterson, G. Braccini, L. Edenbrandt, L. Sornmo, *Clustering ECG complexes using Hermite functions and self-organizing maps*, IEEE Trans. on Biomed. Eng., vol. 47, pp. 838-848, 2000
[24] I. Daubechies, *Ten lectures on wavelets*, In: SIAM CBMS–NSF Regional Conference Series in Applied Mathematics, 1992.
[25] S. Mallat, *A theory for multiresolution signal decomposition: The wavelet representation*, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 11, pp. 674-693, 1989.
[26] C. Cortes, V. Vapnik, *Support vector machines*, Machine Learning, vol. 20, pp. 273-297, 1995
[27] C.J.C. Burges, *A Tutorial on Support Vector Machines for Pattern Recognition*, Data Mining and Knowledge Discovery, vol. 2, pp. 121-167, 1998
[28] S. Osowski, L.T. Hoai, T. Markiewicz, *Support vector machine-based expert system for reliable heartbeat recognition*, IEEE Trans. on Biomed. Eng., vol. 51, pp. 582-589, 2004
[29] I.H. Witten, E. Frank, *Data Mining: Practical machine learning tools and technique, 2nd Edition*, Morgan Kaufmann, San Francisco, 2005
[30] O. Zhao, L. Zhang, *ECG Feature Extraction and Classification Using Wavelet Transform and Support Vector Machines*, International Conference on Neural Networks and Brain, ICNN&B, vol.2, pp. 1089-1092, 2005

DODATKOWE PRZETWARZANIE WSTĘPNE I EKSTRAKCJA CECH W PROCESIE AUTOMATYCZNEJ KLASYFIKACJI RYTMU SERCA

Streszczenie: Artykuł prezentuje nowe podejście do problemu klasyfikacji zapisów ECG w celu detekcji zachowań chorobowych. Podstawą koncepcji fazy ekstrakcji cech jest proces przetwarzania wstępnego sygnału ECG z wykorzystaniem morfologii matema-

tycznej oraz innych transformacji. Morfologia matematyczna bazując na teorii zbiorów, pozwala zmienić charakterystyczne elementy sygnału. Dwie podstawowe operacje: dylatacja i erozja pozwalają na uwydatnienie lub redukcję wielkości i kształtu określonych elementów w danych. Parametry charakterystyki zapisów ECG stanowią bazę dla wektora cech. Do klasyfikacji przebiegów ECG w pracy wykorzystano samoorganizujące się mapy (SOM) Kohonena z klasyfikatorem LVQ oraz algorytm Support Vector Machines (SVM). Eksperymenty przeprowadzono klasyfikując sygnały pomiędzy trzynaście kategorii rekomendowanych przez standard ANSI/AAMI EC57, to jest: prawidłowy rytm serca i 12 arytmii. Zaproponowany w artykule algorytm opiera się na wykorzystaniu elementarnych operacji morfologii matematycznej i ich kombinacji. Ocenę wyników eksperymentów przeprowadzono na sygnałach z bazy MIT/BIH. Na tej podstawie zaproponowano wyjściową architekturę bloku filtrów morfologicznych dla celów ekstrakcji cech oraz unifikacji wejściowego sygnału ECG jako danych wejściowych do budowy wektora cech.

Słowa kluczowe: ECG, przetwarzanie wstępne, morfologia matematyczna, filtrowanie ECG, ekstrakcja cech, klasyfikacja rytmu serca

Artykuł zrealizowano w ramach pracy badawczej Wydziału Informatyki Politechniki Białostockiej o numerze W/WI/13/07.