computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

# CLASSIFICATION DATA EXPLORATION METHODS IN MODERN REAL-TIME DATA WAREHOUSE

**Jakub Chłapiński, Piotr Mazur, Jan Murlewski, Marek Kamiński, Bartosz Sakowicz**

[1)] Department of Microelectronics and Computer Science,
Technical University of Lodz, Poland,
al. Politechniki 11, 90-924 Łódź, Poland,
{jchlapi, pmaz, murlewski, kamiński, sakowicz}@dmcs.pl
http://www.dmcs.p.lodz.pl

**Abstract:** *The goal of this article is to introduce problems that may arise during analysis of classification methods used in data mining applications. In the following sections some of the most common classification techniques are described along with several proposed extensions which allow these methods to be used in incremental data warehouses. The primary focus was aimed at the problem of performing incremental learning methods that may be used in near real-time data warehousing applications.*

**Keywords:** *Data Mining, Data warehouse, Classification method, Neural networks, Decision tree.*

## 1. INTRODUCTION

Data Mining is a term commonly used to describe a group of methods which purpose is to extract hidden dependencies between the data stored in data warehouses, data marts and other relational database systems. The end-user may even not be aware of these dependencies, although the knowledge gained from extracting them may prove to be extremely useful. Data Mining methods may be divided into several basic groups, each dealing with specific problems. Among the most used method types in data warehouses are classification methods [2].

Classification methods aim to provide certain model, which can be helpful in deducing a specific group that one of the database object belongs to, based on its characteristics (Fig.1). One of the common uses of classification methods include determining future actions of customers so that a company could alter its customer care plan more precisely.

Classification methods can also be divided into two groups: methods that require a set of data that is used for learning and others.

K-NN method is one of the most popular methods used in classification [1]. Its function is to find k-nearest (using some predefined metrics) neighbors of the subject, and assign it to a certain class that is dominant in all successfully found subjects.

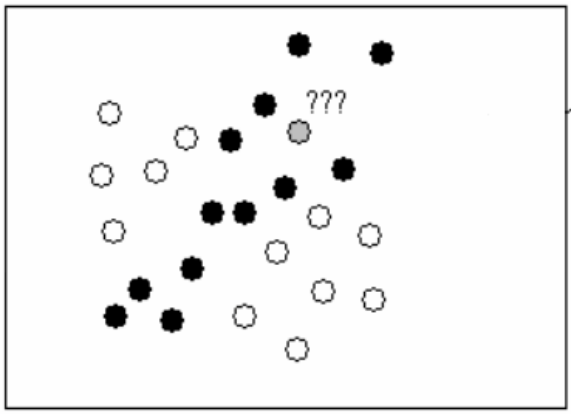



**Fig.1 – Cassification method – (i.e. k-NN).**

Another commonly used method which has advantage of being relatively simple and efficient is Naive Bayes Classifier.

Methods that require a learning set form a numerous group which includes: neural networks [5], simple and oblique decision trees and SVM (Support Vector Machines) method [1]. All of these methods are based on a similar principles that consists of choosing a structure (for example multi layered perception for neural networks and core function for the SVM method), and assigning the best parameters that allow to minimize erroneous classifications on the given learning set (for example: using the error back propagation method, optimization methods, or evaluating GNI indexes values). The last step is to verify the resulting

structure, which can be performed by evaluating results for a given set of values also known as the verify set. If the verification process is unsuccessful, then appropriate changes in the structure should be made (for example: trimming the decision tree), and the whole process has to be repeated. The process of learning may prove to be time-consuming, and may also lead to under or over learning phenomena.

## 2. INCREMENTAL LEARNING METHODS

Taking into account the fact that changing any relationship within a set of data (for example: accounting for market fluctuations and quick market changes) may change the classification model unexpectedly, the knowledge model should not be generated once, based on a predetermined set of data because the resulting model may loose its properties over time and become less precise. On the other hand performing the full learning process with every change in input data set can be time consuming and inefficient. Therefore to reflect the changes more efficiently a new set of methods needs to be devised.

These algorithms should take into account only recent changes in data, and reflect them on a model in an incremental fashion, without re-analyzing the whole data set. Using incremental learning methods may lead to developing a dynamic model that adapts itself every time new data is added into it which greatly improves quality and accuracy of the results.

Applying of incremental learning methods to data sets may also lead to a few model inconsistencies. Taking the client credit standing for example: after successful classification of a client as an positive candidate the data set would be updated to reflect the changes, but if the classification process was unsuccessful no data would be updated in the data set. New data about erroneous positive classifications that are reflected in the data set may lead to a state where every new classification would be considered as erroneous, and no changes in the model would be performed. Therefore, before using an incremental learning method these negative effects should be considered, and the usefulness of the resulting data has to be evaluated. To counteract these effects usually a separate process of recognizing negative classifications and finding their inner relationships is performed, followed by full regeneration of a model when existing model proves not to reflect the data in a satisfactory manner.

Some of the common methods used in data mining may not have their incremental learning versions. A good example may be the k-nearest neighbor method where new elements have to be updated in the data warehouse and used along with remaining elements while performing classification process of unknown data.

## 3. NAIVE BAYES CLASSIFIER

One of the most popular classification methods is the Naive Bayes algorithm [2]. It is considered to be relatively simple in implementation and to achieve better results (assuming all its requirements are met) than other methods. Naive Bayes Classifier is based on the Bayes theorem, and uses comparisons of the statistical indicators described by formula 1.

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)} = \frac{P(X \mid C_i)\frac{s_i}{n}}{P(X)} \qquad (1)$$
$$P(X \mid C_i) = P\prod_{j=1}^{p} P(x_j \mid C_i) = P\prod_{j=1}^{p} \frac{s_{ij}}{s_i}$$

where: $P(C_i|X)$ – probability of belonging to a class $C_i$, given attributes X, $P(X|C_i)$ – probability of having attributes X given a class Ci, P(X) – occurrence probability of an element with attributes X, n – total count of the elements in a set, $s_i$ – count of elements belonging to class $C_i$, $P(x_j|C_i)$ – probability of having attribute x on j-th position when belonging to class $C_i$, $s_{ij}$ – total count of the elements that belong to class $C_i$ and have attribute x on j-th position

The element is classified as belonging to a certain class based on the highest probability score of belonging to that class.

The basic problem while dealing with the Bayes method is to calculate the probability factor $P(X|C_i)$. To achieve this a set of kp probability factors need to be calculated where p is the number of attributes and k is the total count of values that these attributes can have. One of the most popular approaches is to assume that all attributes are independent of each other, then the overall probability can be simplified and expressed with formula 1. It is important to mention that this assumption may not be correct in some cases.

It is also important that in the process of Bayes classification the absolute values of calculated probabilities are not as important as their relative values used for comparing one to another, which means that all constant factors (n, P(x)) can be omitted in all formulas as having no real effect on the outcome. Using this simplification the Bayes Method can be reduced to calculating the result of comparison of multiplied $s_{ij}$ elements.

The incremental learning Bayes method operates by recording these values and updating attributes of any new element that is added to the data set, followed by an increase of the associated coefficients. There is no need to analyze old data.

Incremental learning version of the Bayes method requires at most $(k^p + m)$ – sized memory

region, where m stands for the total count of all classes. Developing a specified repository to hold these values can be relatively simple.

When using the Naive Bayes Classifier method there is a need to store the certain algorithm information in the underlying database. The information consists of metadata describing each discovered class that will have elements assigned to, and additional data describing every single element and the segment that it is assigned to having a certain value of the given attribute. To represent the Naive Bayes Classifier algorithm in a database a new database table needs to be created:

**Table 1. bayes_classes (SQL table)**

| Field name | Field type | Description |
|---|---|---|
| id | numeric | Entry identifier |
| analysis_id | numeric | Analysis identifier |
| class_id | numeric | Class identifier |
| class_count | numeric | Element count in the given class |

To improve data access and retrieval creation of an B-Tree index on the analysis_id field is recommended, which will allow for faster access to all classes belonging to a certain analysis. Also the class_id field should be defined as a reference to the class_id field in the bayes_classes table described later in this document. The use of a foreign key with the cascade delete option set is preferred.

Results obtained for the Naive Bayes Classification method are presented in table 2 and Fig. 2. These results are based on an assumption that 10 percent of the database objects are object which were incrementally added to the source database, and the whole data set is to be divided into four different classes. It was also assumed that all database objects consist of eight different binary attributes.

**Table 2. Influence of size of the data set on analysis times**

| Object count | Incremental method [ms] | Classic method [ms] |
|---|---|---|
| 1000 | 0 | 15 |
| 10000 | 31 | 156 |
| 100000 | 141 | 1 562 |
| 1000000 | 1 468 | 16 516 |
| 5000000 | 7 703 | 80 547 |
| 10000000 | 15 781 | 168 297 |

Based on the test results it is clearly visible that the proposed incremental version of the Naive Bayes Classifier is more efficient than the standard method.

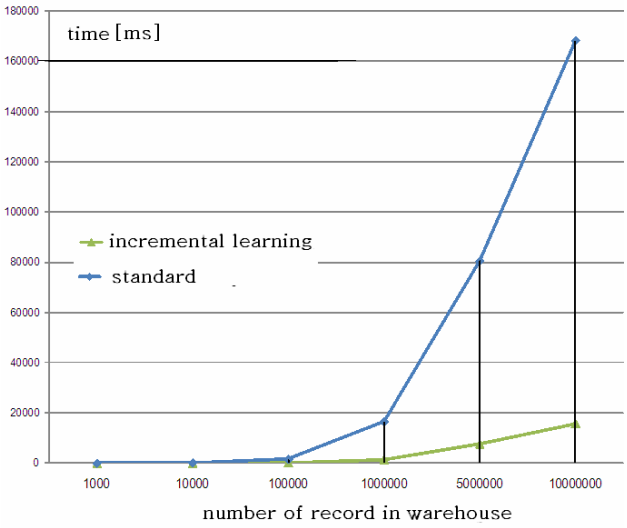



**Fig.2 – Influence of the data set size on timing of the incremental classification method.**

Table 3 presents the results of performance analysis of the Bayes classification for the standard and incremental method depending on the number of attributes used in classification process.

**Table 3. Influence of number of attributes on time of the classification process**

| Attributes | Incremental method [ms] | Classic method [ms] |
|---|---|---|
| 2 | 797 | 5 109 |
| 4 | 1 110 | 11 328 |
| 8 | 1 594 | 13 453 |
| 20 | 3 375 | 36 015 |

Total number of objects used in the test was set to 1 000 000, including 10 % of objects that were added incrementally. Total number of desired classes was set to 10.

The test results indicate that, as in previous cases, incremental method performs much faster than classic method.

## 4. DECISION TREES

Among many methods that require using the learning process decision trees are probably the most popular [3, 8]. Decision Tree algorithm is based on a graph, where vertices are described by tests (i.e.: comparing the values of attributes), arcs are represented by test results, and leaf nodes by classification classes. The process of building a Decision Tree is based on a recurrent division of the training set to partitions up to the point where every partition is sufficiently small or contains only

elements belonging to one class. The division process is driven by the value of one specified attribute. The "quality" of the resulting decision tree depends on the proper choice of attributes and division criteria. There are many indicators that evaluate these attributes: the correlation index X (CHAID algorithm), GNI indexes (CART algorithm), information gain (ID3 and C4.5 algorithms [4]). Decision Trees may also be divided into two groups: simple and oblique (Fig. 3).

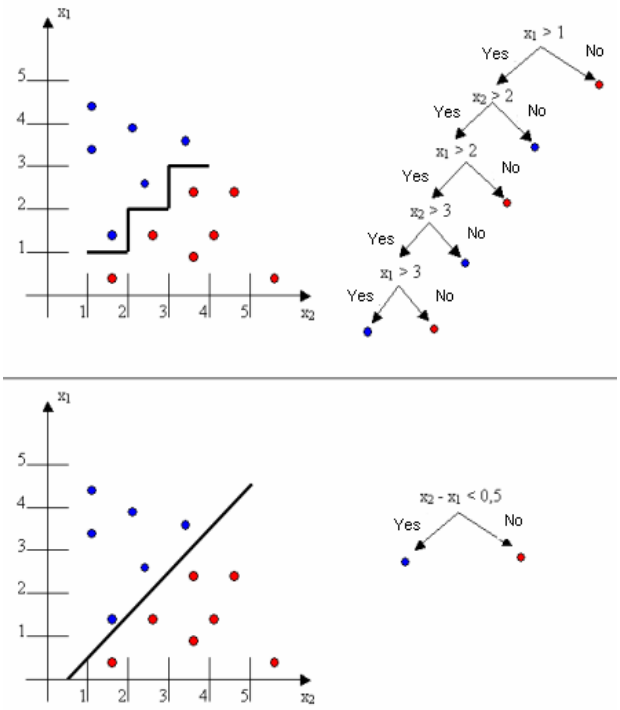



**Fig.3 – Simple and oblique decision trees.**

When dealing with oblique decision trees (OC1) to divide a set a linear combination of many more attributes is required. The resulting combinations form a climbing method that uses probabilistic as well as heuristic algorithms. The main advantages of oblique decision trees are their smaller size, while their bigger disadvantages include complex and hard implementation of division methods.

One of the characteristics of all classification methods that require learning is the fact, that they always use a part of the data set to perform the learning process. Some subset of data is needed to verify the result of the learning stage. In some cases also new incoming data is used to verify learning and, assuming the base model is correct, should give accurate results. In other cases when the classification process produces incorrect results the source data set may be moved to a special repository which will be used in the next instance of the learning process.

When this solution fails it is also possible to apply incremental learning, assuming some prerequisites are met. When performing incremental learning one of two methods may be used:

Cut – which require that for every branch of a tree additional attribute is stored. The attribute represents confidence. If the confidence attribute values for a specific branch of a tree are significantly low, the branch should be deleted and replaced by a leaf node. There is also a possibility of regenerating a branch of a tree, and replacing invalid branch with new one.

Tree Growth – in a set of incorrectly classified elements some association rules may be found [7,9] (the common attribute combination resulting in incorrect classification). If there is no support for an association rule then the rule may be safely incorporated into a tree as a new branch located near the root node. If the new branch is strengthened in the following process of classification its members may be moved to the learning set, so that standard classification rules can learn new trends.

In the case of methods based on decision tree algorithms it is crucial to store the tree structure in the database, along with additional data that identifies the set of elements used in learning and verification processes that it was based on.

**Table 4. dec_tree_sets (SQL)**

| Field name | Field type | Description |
|---|---|---|
| id | Numeric | Field identifier |
| analysis_id | Numeric | Analysis |
| set_type | Enumeration | Data set identifier: learning or verification set |

**Table 5. dec_tree_elements (SQL)**

| Field name | Field type | Description |
|---|---|---|
| set_id | Numeric | Entry identifier |
| element_id | Numeric | Element identifier |

The description of a decision tree structure may be represented by a metadata database table describing each tree with the corresponding analysis.

**Table 6. dec_tree_defs (SQL)**

| Field name | Field type | Description |
|---|---|---|
| id | Numeric | Tree identifier |
| analysis_id | Numeric | Analysis identifier |

The tree structure itself can be represented as a set of nodes with references to their corresponding parent nodes.

**Table 7. dec_tree_nodes (SQL)**

| Field name | Field type | Description |
|---|---|---|
| id | Numeric | Entry identifier |
| tree_id | Numeric | Decision tree identifier |
| parent_id | Numeric | Parent node identifier |
| trust_level | Floating point | Trust level of aa node |

Fields: id, tree_id and parent_id should be indexed to provide faster data access when building the decision tree structure in memory. To optimize the data access further materialized views [10] may be used based on tables dec_tree_defs and dec_tree_nodes because the references between these tables are not subjects to frequent changes. To improve the algorithm performance it is also advised to preload the whole structure to memory, and process the data in grouped data sets periodically updating the tree structure whenever one of its attributes changes.

## 5. NEURAL NETWORKS

The process of incremental learning of a neural network applies only when certain conditions are met. New data added to the system cannot oppose the data that was already used in the learning stage. In case of such scenario the incremental learning process may have a negative influence on the classification giving false results (Fig.4).

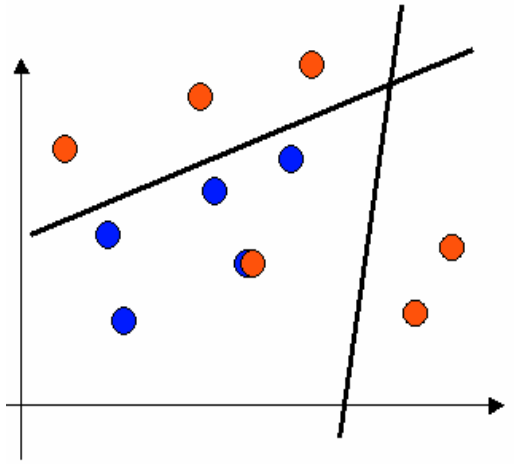


**Fig.4 – Neural network requireing another learning.**

Another characteristic of Neural Network is the fact that during the initial learning there is a need to minimize the size of the network. That requirement may negatively influence the network when new, unseen data is added to the data set, because there may exist no efficient way of representing the data set using the same network structure. The case when new data influences only the weight coefficients is depicted in (Fig. 5), while the situation when a network structure change is required is shown in (Fig 6).

In the case depicted in (Fig.5) only a small adjustment to the weight coefficient is necessary. The back propagation algorithm (along with its modifications) can be run sequentially for the chosen elements of a learning data set. In this case the algorithm is run only to check new incoming data. After adjusting the weight coefficients the algorithm is run once more to examine older elements of the data set to correct errors that could be introduced in the first run. Considering the fact that the network is already in semi-learned state another run of the algorithm is performed more efficiently than in the case that requires the whole learning process to begin from the start.

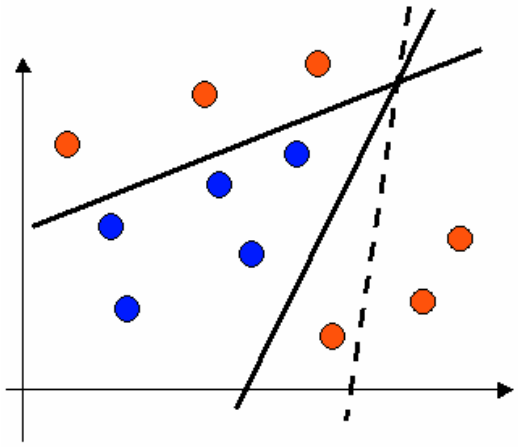


**Fig. 5 – Neural network requiring weights correction.**

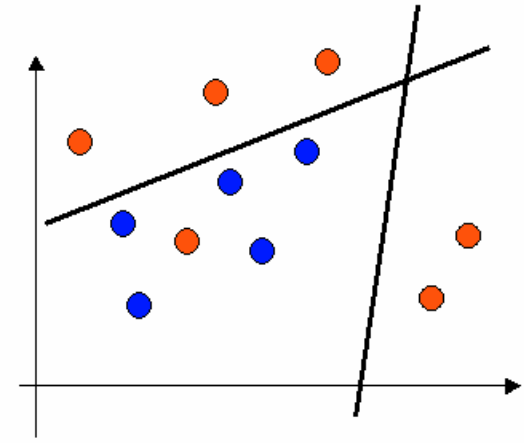


**Fig.6 - Neural network requiring structure expanding.**

It's also worth mentioning that dampening of the weight coefficient change may be performed while the error propagation process is run for newly inserted elements. That should prevent rapid coefficient changes, which may result in loss of network efficiency when dealing with old elements.

In case of the situation depicted in (Fig.6) incremental learning using the sequential method is not advised, because the network cannot acquire

more information without changing its structure. Neural Network growth may also be performed incrementally [6] but in the case of a working data warehouse it is not advised. Automatic growth of the neural network with every new, incorrectly classified element may lead to the effect of an over trained network (or at least result in poor classifier efficiency). The methods that implement network growth should be run on an already verified set of data. Incremental learning algorithms should be used only as temporary means of correcting the classification results before the actual full learning process that can be run in a later time.

Example algorithm of a neural network growth:

1) Every weight coefficient of the network is being kept in its original state.

2) A new neuron is added to the location of the outermost layer (more neurons may be added when necessary and other layers may be used). Its weight coefficients are set only by the elements belonging to the same class and newly added elements (Fig.7)

3) New output layer is build which makes choice of the score function between old and new part of the network (AND gate (Tab. 8))

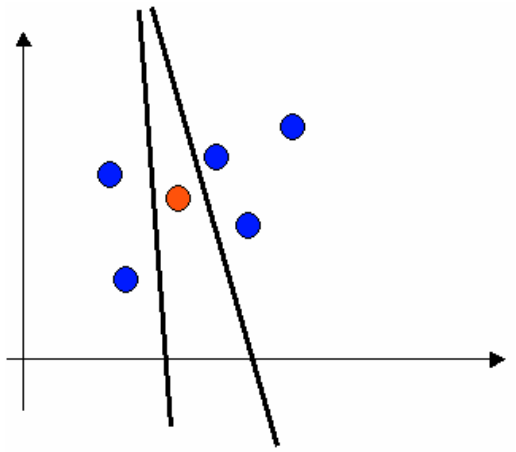


**Fig.7 - Functionality of new built network.**

**Table 8. New output layer**

| Learning element type | Output neuron state in the preceding network. | Output neuron state in the joined network. | Neuron state in the new output layer. |
|---|---|---|---|
| negative, classified correctly | 0 | X | 0 |
| positive, classified correctly | 1 | 1 | 1 |
| negative, classified incorrectly (incremental) | 1 | 0 | 0 |

Newly formed neural network performs the classification process taking into consideration old and new sets of data.

## 6. SUMMARY

In this article a few of the classification methods were introduced along with their incremental learning methods which can perform efficiently in near real-time data warehousing environments. Mentioned algorithms are presently being incorporated in the project of building an advanced database system for real-time data warehousing which can be incrementally updated. In this project the 10 minute analysis update limit is being enforced on every new data that is added to the database.

## 7. REFERENCES


[1] Burges C.J.C. A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery* 2(2).

[2] Langey P., Iba W., Thompson K. "An analysis of Bayesian classifiers", *In Proc. of 10th National Conference on Artificial Intelligence*, San Jose, CA, 1992, AAAI Press, – pp. 223-228.

[3] L. Breiman, J. H. Friedman, A. Olshen, C. J. Stone. *Classification and regression trees.* Wadsworth, Belmont, CA, 1984.

[4] Quinlan J.R. *C4.5: Programs for machine learning.* Morgan Kaufman, 1993.

[5] Bigus J.P. *Data mining with neutral networks*, McGraw Hill, 1996.

[6] S.E. Fahlman, C. Lebier, "The Cascade-Corelation Learning Architecture", *Technical Report CMU-CS-90-100,* School of Computer Science, Carnegie Mellon University, August 1991.

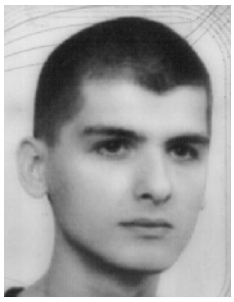
[7] Agrawal R., Srikant R. Fast Algorithms for Mining Association Rules, *Proc. of 1994 International Conference on Very Large Databases VLDB*, Santiago de Chile, September 12-15, Morgan Kaufman, 1994. –pp. 487-499.

[8] Quinlan J.R. Induction of decision trees. *Machine Learning 1(1),* – pp.81-106.

[9] Agrawal R., Imielinski T., Swami A. Mining association rules between sets of items in large databases, *Proc. of 1993 ACM SIGMOD International Conference on Management of Data*, Washington D.C., May 26-28, ACM Press 1993, – pp. 207-216.

[10] Gupta, H.; Mumick, I.S. Selection of views to materialize in a data warehouse, *IEEE Transactions Knowledge and Data Engineering*, Volume 17, Issue 1, Jan 2005, – pp.24-43.

***Jakub Chlapinski*** *received MSc degree in 2003 in Computer Science from the Technical University of Lodz in Poland. He is a PhD student with the Department of Microelectronics and Computer Science. His research area covers distributed computing, software engineering, database applications as well as signal and image processing.*



***Piotr Mazur*** *is a PhD candidate with the Department of Microelectronics and Computer Science on Lodz, Poland. His current research focuses on areas concerning relational database management systems, operating systems and voice transmission using VoIP technology.*

***Jan Murlewski*** *received MSc degree (2004) in Computer Science from the Technical University of Łódź. He is currently a PhD candidate with the Technical University of Łódź. His primary research interests are object-oriented databases, real-time data warehouses, distributed computing and software engineering.*

***Marek Kaminski*** *received a Masters Degree and Ph. D. Degree in Electronic from the Technical University of Lodz in Poland in 2002 and 2006 respectively. During Ph. D. studies he joined the Department of Microelectronics and Computer Science (DMCS TUL). He is now an Assistant Professor. His research activities include internet applications, distributed computing, data warehouses and stochastic optimization.*

***Bartosz Sakowicz*** *received a Masters Degree and Ph. D. Degree in Computer Science from the Technical University of Lodz in Poland in 2001 and 2007 respectively. During Ph. D. studies he joined the Department of Microelectronics and Computer Science (DMCS TUL). He is now an Assistant Professor. His research activities include internet applications, distributed computing, data warehouses, game theory and stochastic optimization.*