Constrained Classification of Large Imbalanced Data by Logistic Regression and Genetic Algorithm

Martin Hlosta, Rostislav Stríž, Jan Kupčík, Jaroslav Zendulka, and Tomáš Hruška

***Abstract*—Imbalance in data classification is a frequently discussed problem that is not well handled by classical classification techniques. We propose a new method that produces highly accurate and easily understandable classification model with regards to user-specified set of class accuracy restrictions. Our method combines logistic regression with a genetic algorithm optimization and has been successfully tested on a large real-world data set from our internet security research. Experiments prove that our method always leads to better results than usage of logistic regression or genetic algorithm alone.**

***Index Terms*—Imbalanced data, classification, genetic algorithm, logistic regression.**

I. Introduction

High emphasis on data collecting and subsequent analysis leads to discovery of new problems related to processing various types of data. In this paper we focus on the imbalanced data learning problem, which has drawn a lot of attention over past years. We present a new approach to handle highly imbalanced binary data classification with accuracy constraints on one of the classes. The example of such a constraint is reaching 95% accuracy on the minority class. Our method combines cost-sensitive logistic regression with the use of genetic algorithm. We provide experiments and evaluation on highly imbalanced real-world data from our internet security research. The presented results clearly show advantages of our proposed method compared to the using logistic regression and genetic algorithm separately. Moreover, the resulting model generated by the method is easily interpretable (compared to, e.g., Neural Network) and is suitable for very fast prediction.

II. Problem Definition and Related Work

In the following paragraphs we describe our problem in greater detail, as well as basic principles and other proposed methods our work is based on.

A. Imbalanced data classification

When we talk about imbalanced data, we consider data with a significant disproportion in the occurrence of instances of each class. Types of data imbalance are widely described in [1]. Experiments performed on imbalanced data sets have shown that such type of data may often have negative impact on the learning phase of most standard classification algorithms. The algorithm is overwhelmed by the number of majority class instances which may lead to not discovering the minority class.

The main problem in dealing with imbalanced data classification is the default inability of most common classifiers to recognize the minority class properly. Another problem is their inability to consider different costs of misclassification. To solve these problems, many methods have been proposed. Basically, there are two most popular approaches that can be used when dealing with imbalanced data classification.

Sampling is the basic approach we can choose to alter the data imbalance. There are two types of sampling - undersampling and oversampling. Majority class undersampling leads to lower between-class imbalance by removing a set of majority class examples from the algorithm training set. The issue here is how to choose the set of majority class examples for the removal. Approaches differ from a random selection to more complex informed methods, which produce generally better results. Minority class oversampling also leads to lower data imbalance but instead of decreasing the number of majority class instances, the minority class occurrence is increased by adding or generating new minority class examples. Techniques for oversampling vary from basic copying of existing tuples to more complex generating methods, like SMOTE [2], which are able to create new synthetic examples fitting to the minority class, and widen the decision region for the classifier. This concept has been explored and extended in [3]. Different informed methods are also presented in [4], [5].

Sampling can solve the problem of data imbalance from the point of the actual data distribution, but it still does not take into account possible costs of misclassification. *Cost-sensitive* methods have been proposed to address this issue and according to various studies focused on certain specific imbalanced learning domains [6], [7], cost-sensitive learning is superior to sampling methods. The basic concept in the context of cost-sensitive learning methodology is the cost-matrix. The cost-matrix essentially represents a numerical penalty of misclassifying an example. The main objective of cost-sensitive methods is to create a model that minimizes the overall cost on the training set. *MetaCost* framework, described in [8], presents a simple and effective way for existing classifiers to adopt a cost-sensitive learning approach in a form of cost-sensitive wrapper, which can be applied onto various data mining systems without prior knowledge of their structure. Another way of creating a cost-sensitive classifier may be integrating such functionality directly inside the classifier itself, or apply misclassification costs to the data set in a form of dataspace weighting.

However, problems with optimal settings may occur while using cost-sensitive or sampling methods. Usually, it is impossible to determine ideal method and its setting for further unknown data classification. Because of this, building classifier ensembles has become a popular practice.

Ensembles are essentially groups of classifiers that are trained on different training sets. These training data sets may vary in cost-matrix values, or in the data themselves when using different sampling settings. Unseen data are then classified by all generated classifiers and the result is decided either by simple majority voting or by other advanced methods (e.g., different voting weights). Studies [9], [10] confirm that classifier ensembles may dramatically improve classification accuracy.

In contrast to previously mentioned techniques, our proposed method combines the use of logistic regression with genetic algorithm. Following sections give an insight on how both of these techniques work.

B. Logistic Regression

Logistic regression *(LR)* is a machine learning model for binary classification. The method can handle both numeric and categorical variables. Given a learned model, the value of the output variable is computed by applying the logistic function to linear combination of attribute values and weight vector. The function is defined as follows:

$$P(Y_i = 1 \mid x_i, w) = \frac{1}{1 + e^{-w^T x_i}} \quad (1)$$

The logistic function converts the input value to interval $[0; 1]$. The result describes a confidence value for a given case being of the class 1. Typically, threshold $t = 0.5$ is applied to determine whether an examined example belongs to class 0 or 1.

In the training phase, the algorithm tries to solve the unconstrained optimization problem of the following objective function:

$$\min_w \; C\sum_{i=1}^{n} \log(1 + e^{-y_i w^T x_i}) + \lambda \quad (2)$$

where λ stands for a regularization. *L1* ($\sum_{i=1}^{n} |w_i|$) and *L2* ($\sum_{i=1}^{n} w_i^2$) are two common regularization used in logistic regression. L1 is usually used when the feature space is sparse, otherwise the use of L2 is recommended.

The logistic regression can handle imbalanced data in two ways: a) The threshold moving technique moves the threshold closer to 0 or 1 resulting in increase of accuracy on the class to which the threshold is further away from, which is typically the rarer class. b) The second technique wraps LR with a cost-sensitive framework. The approach used in [11] sets different C parameter in Equation (2) for examined classes. Usually, the greater value of C is set in the case of the rarer class.

C. Genetic algorithm

Genetic algorithm *(GA)* is a random optimization method based on a principle of natural selection and biological evolution. The examined problem is encoded into a set of special genome-like structures. These structures are essentially data strings representing our original binary encoded data instances. The core part of any genetic algorithm is a *fitness function*. Fitness function is able to rank data genomes, ideally, with regards to the desired result of the algorithm. The set of genomes is called a population. The algorithm works iteratively, creating a new altered population in each step, towards the best (highest ranked by the fitness function) possible population. However, as the algorithm is nondeterministic, some kind of stopping criterion should be implemented.

Generally, genetic algorithm uses selection, crossing and mutation techniques to produce a new population. Pure selection enables the algorithm to simply copy excellent individuals from the old population to the new one. Crossing usually combines two individuals by splitting those into (mostly two) parts and joining them together to create different individuals. Mutation produces new genome by randomly altering parts of the old one with regards to alternation constrains (e.g., random new value has to be valid in the changed attribute's domain). Crossing and mutation are applied randomly and probability of their occurrence is often adjustable by initial algorithm parameters.

Use of the genetic algorithm for data mining has been proposed in [12]. Potential of this approach lies in the search performance of the algorithm, which is very easy to parallelize. There is also no need for dividing data into training and testing sets as the algorithm trains and tests itself on the same data set. Nowadays, genetic algorithm is often used as an optimization technique for other data mining methods, mostly for feature subset selection [13], in hybrid decision structures [14], or for rule induction [15], [16].

D. Evaluation metrics

In classification, accuracy and error rate are commonly used metrics for evaluating classifier. However, for imbalanced learning, these metrics can be misleading due to their emphasis on the influence of the majority class. Because of this, other metrics need to be used. Some of the most used are *AUC* (*Area under ROC Curve*) and *Geometric mean* (*G-Mean*) metric from [18], which is defined as follows:

$$G - mean = \sqrt{TPR \times TNR} = \sqrt{\frac{TP}{TP+FN} \times \frac{TN}{TN+FP}}, \quad (3)$$

where *TP*, *FP*, *TN* and *FN* are values from the confusion matrix, i.e. number of *True/False Positives* and *True/False Negatives*. *TPR* stands for *True Positive Rate*, or sometimes called *Sensitivity* and *Recall*, and it represents the accuracy on positive examples. *TNR* is *True Negative Rate* or *Specificity*, representing the accuracy on negative examples.

III. The Proposed Method

Modern approaches already combine genetic algorithm with logistical regression; however, presented methods use the genetic algorithm mostly just as an optimization technique for adjusting costs in cost-matrix, or reduction of feature space [13], [17]. We propose a different approach where both methods are used subsequently to achieve better results than other techniques.

In this section we present the proposed method for imbalanced classification based on LR and GA. First, the overall method concept is presented and then both LR and GA parts are described in more detail.

A. The LR/GA method

Our method consists of several consequential blocks. The whole concept can be best understood from the method visualization in Fig. 1. The method processes labeled input data and returns optimized model satisfying user accuracy constraints. These constraints can be specified for the accuracy of one, both, or neither of the classes. If such model, satisfying all given constraints, doesn't exist, the method

returns an empty result.

The main task of the LR block is to find a set of logistic regression models, i.e., attribute weights, bias and a threshold. These models are found with respect to a given accuracy constraints and class imbalance in analyzed data.

The GA block utilizes those models and uses them as the initial candidate solution for the genetic algorithm optimization task. Using specific fitness function and mentioned initial candidates, the algorithm starts to find the global optimum, again with respect to given constraints. The main advantage of our approach compared to other methods, described in section II, is speed. We are forced to analyze millions of records in short time periods with very high accuracy constraint on the minority class classification; moreover, the final model application to unseen data has to be swift and as simple as possible. The required speed and simplicity of model application is one of the reasons we didn't choose to use classifier ensembles. Another huge advantage of our proposed method is a comprehensible result model. In comparison with neural networks or SVMs, the method presents an easy-to-read and understandable model that can be easily applied to unseen data, or implemented as a part of another application or system.

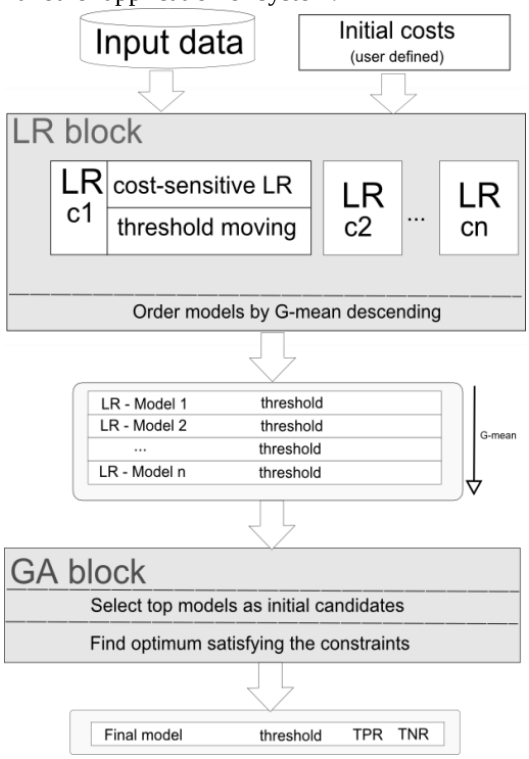

Fig. 1 LR/GA Method schema.

B. *Logistic regression block*

The LR block uses the combination of cost-sensitive LR learning and threshold moving to find an initial solution for the GA block. To perform cost-sensitive learning, the block expects a set of costs for the rarer class to be delivered with the input data. This set defines interval borders. Inside each of these intervals, the local maximum of the G-mean metric is found with respect to specified constraints. This optimization technique is simple and it expects that the objective function is convex or concave on the given interval. The goal is not to find a real maximum but only its rough estimate. Each interval is binarily split until the maximum value increases by given difference. The usage of more initial costs, and thus more intervals where the maximum is searched, allows dealing with local maxima and in addition the learned models for GA can vary widely, which helps the optimization.

The value of the G-mean metric for a given cost is computed as follows: First, the LR model is learned with C parameter set to the selected cost value. Then, the model is applied to testing data to get the confidence vector. The threshold moving technique is then used to find the maximum G-mean metric value. For each evaluated threshold, the confusion matrix and G-mean value are computed. The accuracy constraints can be used to find this solution faster and the interval is divided to bins with an equal size. The final models are then sorted by the G-mean value and they are ready for use in the GA block.

C. *Genetic algorithm block*

On input, GA block is given models from the LR block and its main purpose is to make these models more accurate with regards to given class restrictions. It iteratively creates populations of genomes that, in our case, represent groups of differently weighted models. The core part of the GA block is the fitness function, which ranks all generated models. We can alter the fitness function in such way it reflects all our preferences and restrictions. This technique has great impact on the quality of produced models. Based on our experiments, described in section IV, all tested regression models have been improved by the use of genetic algorithm.

Although, GA is based on random data modification, there are many ways we can alter the optimization process. Custom heuristics can be easily applied to the mutation and crossing methods; moreover, it is possible to experiment with sizes of initial population, crossing and mutation probabilities, or the number of population cycles.

Our fitness function requires the model to fulfill all required restriction rules; subsequently we rank our models based on their overall classifying accuracy while favoring minority class by applying cost-sensitive accuracy computation.

As we aim to make the algorithm as efficient as possible, we use massive parallelization – all operations executed over single genomes are performed in parallel.

IV. EXPERIMENTS

Experiments were performed over a large data set from our internet security research. Examined data are labeled with two highly imbalanced classes. Characteristics of analyzed data are summarized in Table I:

TABLE I: SOURCE DATA CHARACTERISTICS

Data Cases	Attributes	Attribute type	Class ratio
5 000 000	120	Binary	1 : 99

The main goal was to achieve at last 99.0% accuracy when classifying minority class examples. There were no further restrictions regarding the majority class, however, there are no obstacles for defining such restrictions in our method. The final solution was built to maximize the majority class accuracy while still satisfying the minority class accuracy restriction.

Our experiments show the behavior of the proposed method

under several parameter settings and their influence on its accuracy and run time. Performed experiments were divided into two parts, examining the properties of the LR block and GA block separately.

A. The LR block

The main task of the LR block is to provide set of initial solutions that are good enough as starting models for the subsequent GA block.

1) Qualitative metrics

In order to provide best candidates for the GA block, both *AUC* and *G-mean* qualitative metrics were examined. For approximate AUC computation, it is necessary to evaluate the model using several thresholds. On the other hand, G-mean is computed only for one threshold. Both AUC and G-mean of the best threshold for 30 different costs have been compared. The results show that there is high correlation (above 99%) between the two, which can be seen in Fig. 2. Our experiments also showed that the run time for the AUC computation for given thresholds is considerably longer than for finding the threshold with maximal G-mean.

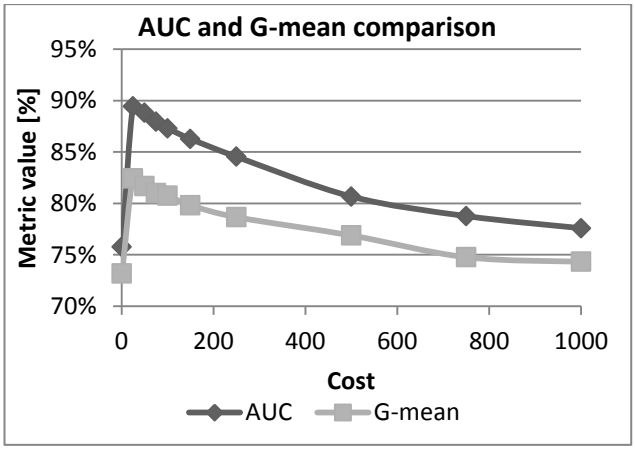

Fig. 2 G-mean and AUC values for different costs.

2) LR block results

We examined two candidate model learning strategies for the LR block. The first one uses constraints, i.e., 99.0% accuracy on the minority class classification. The second one only finds the unconstrained maximum of the G-mean metric, thus making models that don't have to satisfy given constraints. For both strategies we used the same initial costs for the LR learning phase (ranging from 1 to 1000). The final comparison of both strategies is described by Fig. 3. We can see that G-mean values are higher for unconstrained solutions; this is due to the fact that G-mean penalizes solutions where one class achieves very high accuracy at the expense of the other.

Additionally, the best three candidate models are listed in Table II and Table III. These tables reveal that neither of these models satisfies given constraints; however, all of them were successfully provided for our GA block.

In all experiments, we used L2 regularization for learning the model. We also examined L1 regularization with very similar results but longer learning time.

It's worth mentioning that besides LR we tried to deploy similar processes using *SVM*s (*Support Vector Machines*) and *neural networks*, however, on our examined data set, both learning algorithms were unable to finish in a reasonable time frame. In addition, *random forest* with undersampling and naive Bayes algorithms were compared to our solution, with all of them being beaten by cost-sensitive logistic regression.

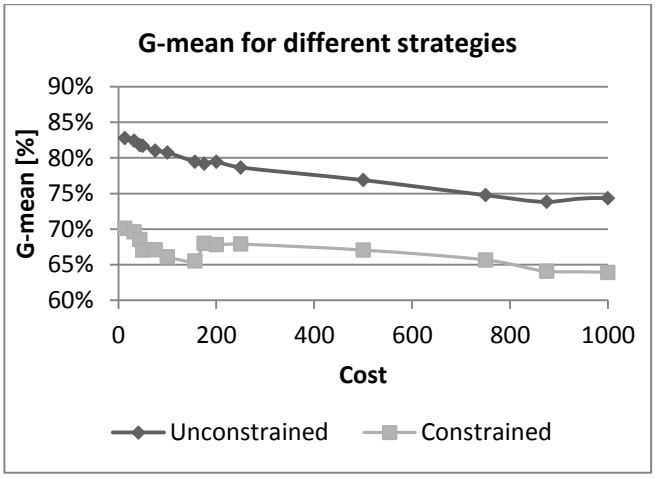

Fig. 3 G-mean metric comparison for constrained and unconstrained learning strategy.

TABLE II: TOP 3 MODELS FOR CONSTRAINED STRATEGY

Cost	Threshold	TPR	TNR	G-mean
32	0.90	0.482	0.991	0.696
225	0.70	0.480	0.990	0.689
175	0.75	0.466	0.466	0.680

TABLE III: TOP 3 MODELS FOR UNCONSTRAINED STRATEGY

Cost	Threshold	TPR	TNR	G-mean
13	0.70	0.777	0.881	0.828
25	0.55	0.779	0.872	0.824
33	0.50	0.777	0.874	0.824

B. The GA block

As previously mentioned, our GA block processes input models generated by the LR block and aims to improve on them with regards to specified constraints. In all performed experiments (over 1000), the use of genetic algorithm improved upon recieved initial models. In Table IV we can see a few real examples of models whose accuracy has been improved by the subsequent use of GA block.

TABLE IV: ORIGINAL MODELS COMPARED WITH OPTIMIZED RESULTS

ID	Orig. TPR	Orig. TNR	GA Opt. TPR	GA Opt. TNR
1	0.479	0.989	0.592	0.990
2	0.783	0.869	0.597	0.990
3	0.480	0.990	0.579	0.990

All optimized models satisfy given minority class accuracy restriction, i.e., at least 99% classification accuracy. When presented with LR model with similar TNR, the GA block always improves both – TNR and TPR values. In the second showed experiment the TNR value is much lower than requested so TPR is naturally higher. Our experiments also confirm the idea from previous section that models generated by logistic regression with high accuracy constraints may lead to worse results than models generated without those restrictions, or with lower constraints.

Regarding the course of the GA itself, introduced to models from LR block, the population improvements are most significant at the beginning of the optimization phase, which

is generally expected. With further iterations, the resulting models converge to their maximum value. An example of such evolution can be seen in Table V.

TABLE V: GA ALGORITHM EVOLUTION

Population no.	TPR	TNR
50	0.562	0.990
100	0.573	0.990
200	0.584	0.990
400	0.590	0.990
1000	0.592	0.990

As we can see, TPR rises dramatically at the beginning of the optimization process, later on the raise is less and less significant and after some time we should find a model with best possible accuracy. We can experimentally determine an approximate number of repopulation cycles in which there is the potential of our optimization maxed out.

V. CONCLUSION

In this paper, we proposed a new hybrid method for classification of large imbalanced data sets in regards to specified accuracy constraints. The method is based on cost-sensitive logistic regression for generating initial models for genetic algorithm optimization. The method was tested on large and highly imbalanced data with more than 5 million records and class imbalance ratio 1:99. Performed experiments showed that proposed method produces better results than both the logistic regression and genetic algorithm separately.

In the future work, we would like to focus on applying different optimization techniques such as *Particle Swarm Optimization*. Also, we would like to add the support for semi-supervised learning, i.e., exploiting data cases with unknown classes, which could extend the final model and increase its accuracy.

REFERENCES

[1] H. He and E.A. Garcia, "Learning from Imbalanced data," *IEEE Trans. on Knowledge and Data Engineering*, vol. 21 no.9, pp. 1263-1284, Sept. 2009.

[2] N.V. Chawla, K.W. Bowyer, L.O. Hall and W.P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique" *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-357, 2002.

[3] H.Han, W.-Y. Wang and B.-H. Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning" in *Proc. Of Advances in Intelligent Computing, International Conference on Intelligent Computing (ICIC '05),* Hefei, China, 2005, pp. 878-887.

[4] X.-Y. Liu, J. Wu and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning" *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 21 no.6, pp. 539-550, April 2009.

[5] J. Zhang and I. Mani, "KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction" in *Proc. Of the Int'l. Conf. on Machine Learning (ICML '03)*, 2003.

[6] K. McCarthy, B. Zabar and G. Weiss, "Does cost-sensitive learning beat sampling for classifying rare classes" in *Proc. 1st Int'l workshop on Utility-based data mining*, 2005, New York, pp. 69-77.

[7] X.-Y. Liu and Z.-H. Zhou, "The Influence of Class Imbalance on Cost-Sensitive Learning: An Empirical Study" in *Proc. 6th Int'l. Conf. on Data Mining (ICDM '06)*, Hong Kong, 2006, pp. 970-974.

[8] P. Domingos, "MetaCost: a general method for making classifiers cost-sensitive" in *Proc. 5th Int'l Conf. on Knowledge discovery and data mining (KDD '99)*, San Diego, 1999, pp. 155-164.

[9] Y.Sun, M.S. Kamel, A.K.C. Wong, Y.Wang, "Cost-Sensitive Boosting for Classification of Imbalanced Data", *Pattern Recognition*, vol. 40 no.12, pp. 3358-3378, Dec. 2007.

[10] T.G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees", *Machine Learning*, vol. 40 no.2, pp. 139-157, Aug. 2000.

[11] D.R. Carvalho and A.A. Freitas, "LIBLINEAR: A Library for Large Linear Classification" *Journal of Machine Learning Research*, vol. 9, pp. 1871-1874, August 2008.

[12] J.S. Tan, W. He and Y.Qing, "Application of Genetic Algorithm in Data Mining" in *Proc. 1st Int'l Workshop on Education Technology and Comp. Science (ETCS '09)*, 2009, pp. 353-356.

[13] J.Yang and V.G. Honovar, "Feature subset selection using a genetic algorithm" *IEEE Trans. on Intelligent Systems*, vol. 13 no.2, pp. 44-49, March 1998.

[14] D.R. Carvalho and A.A. Freitas, "A hybrid decision tree/genetic algorithm method for data mining" *Information Sciences – Special issue: Soft computing data mining*, vol. 163 no.1-3, pp. 13-35, June 2004.

[15] A.A. Freitas, "A Genetic Programming Framework for Two Data Mining Tasks: Classification and Generalized Rule Induction" in *Proc. of the 2nd Annual Conf. in Genetic Programming*, 1997, pp. 96--101.

[16] A.A. Freitas and H.S. Lopes, "Discovering interesting prediction rules with a genetic algorithm" in *Proc. Congress on Evolutionary Computation*, 1999.

[17] D.R. Carvalho and A.A. Freitas, "A genetic algorithm to select variables in logistic regression" *Journal of the American Medical Informatics Association*, vol. 6, pp. 984-988, 1999.

[18] M. Kubat and S.Matwin, "Addressing the Curse of Imbalanced Training Sets: One-Sided Selection", in *Proc. of the 14th Intl. Conf. on Machine Learning*, 1997, pp. 179-186.

Martin Hlosta is a third year PhD student. He received his master's degree in Computer Science in 2010 from Faculty of Information Technology, Brno University of Technology. Both his bachelor and master theses were focused on methods of knowledge discovery in databases. His PhD research focuses on knowledge discovery from malware detection data and learning from imbalanced data.

Rostislav Stríž is a first year PhD student. He received his master's degree in Computer Science in 2012 from Faculty of Information Technology, Brno University of Technology. His PhD research focuses on modern data mining systems and real-world data mining.

Jan Kupčík received his master's degree in Information Systems from Faculty of Information Technology, Brno University of Technology in 2007. He is currently pursuing the PhD degree at the same university. His research interests include database and OLAP technologies, data mining and software engineering.

Jaroslav Zendulka received his M.Sc. degree in computers and Ph.D. in technical cybernetics at the Brno University of Technology, Czech Republic. He is currently an Associate Professor at the Department of Information Systems at the Brno University of Technology. He has participated in several projects and has written tens of papers in international journals and conference proceedings. He is a PC member of several international conferences. His research interests include data and object modeling; database technology and information systems; data mining.

Tomáš Hruška received his Ing. (MSc.) and CSc. (PhD) titles from Brno University of Technology, Czech Republic. Since 1978 he works at the Brno University of Technology, since 1998 as full professor. In 1978-1983, he dealt with research in the area of compiler implementation for simulation languages. He dealt with an implementation of an object-oriented database systems as a tool for modern information systems design and Lissom/Codasip project now. It is focused on the research of the processor description language for transformations of processor models.