Fuzzy Classification by Evolutionary Algorithms

Pavel Krömer, Jan Platoš, Václav Snášel, Ajith Abraham
Faculty of Electrical Engineering and Computer Science
VSB-Technical University of Ostrava
17. listopadu 12, Ostrava-Poruba, Czech Republic
Email: {pavel.kromer,jan.platos, vaclav.snasel}@vsb.cz; ajith.abraham@ieee.org

***Abstract*—Fuzzy sets and fuzzy logic can be used for efficient data classification by fuzzy rules and fuzzy classifiers. This paper presents an application of genetic programming to the evolution of fuzzy classifiers based on extended Boolean queries. Extended Boolean queries are well known concept in the area of fuzzy information retrieval. An extended Boolean query represents a complex soft search expression that defines a fuzzy set on the collection of searched documents. We interpret the data mining task as a fuzzy information retrieval problem and we apply a proven method for query induction from data to find useful fuzzy classifiers. The ability of the genetic programming to evolve useful fuzzy classifiers is demonstrated on two use cases in which we detect faulty products in a product processing plant and discover intrusions in a computer network.**

***Index Terms*—fuzzy sets, data mining, evolutionary computing**

I. Introduction

Fuzzy sets and fuzzy logic provide means for soft classification of data. In contrast to crisp classification, which states crisp decisions about data samples, fuzzy classification allows us to analyze the data samples in a more sensitive way [1]. Fuzzy decision trees and if-then rules are examples of efficient, transparent, and easily interpretable fuzzy classifiers [1], [2].

Genetic programming is a powerful machine learning technique from the wide family of evolutionary algorithms. In contrast to the traditional evolutionary algorithms, it can be used to evolve complex hierarchical tree structures and symbolic expressions. It has been used to evolve Lisp S-expressions, mathematical functions, various symbolic expressions including crisp and fuzzy decision trees, and recently to infer search queries from relevance ranked documents in a fuzzy information retrieval system.

The last approach can be used for general data mining as well. It can be directly applied in the data mining domain. Extended Boolean queries, that is softened Boolean search expressions, can be interpreted as symbolic fuzzy classifiers that describe a fuzzy subset of some data set by means of its features. Moreover, a fuzzy classifier evolved over a training data set can later be used for efficient and fast classification of new data samples and e.g. predict quality of products, detect harmful actions in computer network, and generally assign labels to data samples.

Artificial evolution of search expressions is a promising approach to data mining because genetic programming yields very good ability to find symbolic expressions in various problem domains. The general process of classifier evolution can be used to evolve custom classifiers for different data classes and various data sets with different properties and with different internal structure. The resulting classifiers can be used as standalone data labeling tools or participate in collective decision in an ensemble of data classification methods.

In the remainder of this paper, we describe fuzzy information retrieval as the background of the proposed fuzzy classifier. Then we present genetic algorithms and genetic programming as the tool to infer the classifiers from data. Later on, the construction of fuzzy classifier and experiments with real-life data sets are presented and conclusions are drawn.

II. Fuzzy information retrieval

The proposed fuzzy classification algorithm builds on the extended Boolean IR model, which is based on the fuzzy set theory and fuzzy logic [3], [4]. In the extended Boolean IR model, documents are interpreted as fuzzy sets of indexed terms. In each document, every indexed term has a weight from the range $[0, 1]$ expressing the degree of significance of the term for document representation. Many different weighting approaches can be used to assign weights to index terms, e.g. the $tf \cdot idf_t$ term weighting scheme [5].

The entire document collection can be represented by a real valued index matrix $\boldsymbol{D}$, where each row d_i represents $i-$th document and value d_{ij} $j-$th term in $i-$th document. The query language in the extended Boolean model of IR is improved with the possibility of weighting query terms in order to attribute different levels of importance to those in a search request and by weighting (parameterizing) aggregation operators (most often AND, OR, and NOT) to soften or blur their impact on query evaluation [3], [4].

Consider Q to be the set of user queries over a collection; then the weight of term t in query q is denoted as $a(q, t)$ satisfying $a : Q \times T \rightarrow [0, 1]$. To evaluate the atomic query of one term representing single search criterion the function $g : [0, 1] \times [0, 1] \rightarrow [0, 1]$ will be used. The value of $g(F(d, t), a)$ is called the retrieval status value (RSV). For RSV evaluation the interpretation of the query term weight a is crucial. The most commonly used interpretations see the query term weight as the importance weight, threshold or ideal document description [3], [4].

The theorem for the evaluation of RSV in the case of threshold interpretation is shown and (1) respectively [3], [4], where $P(a)$ and $Q(a)$ are coefficients used for tuning the threshold curve. An example of $P(a)$ and $Q(a)$ could be as follows: $P(a) = \frac{1+a}{2}$ and $Q(a) = \frac{1-a^2}{4}$. The RSV formula from (1) is illustrated in Fig. 1. Adopting the threshold interpretation, an atomic query containing term t of the weight a is a request to retrieve documents having $F(d,t)$ equal or greater to a. Documents satisfying this condition will be rated with high RSV and contrariwise documents having $F(d,t)$ smaller than a will be rated with a small RSV.

$$g(F(d,t),a) = \begin{cases} P(a)\frac{F(d,t)}{a} & \text{for} \quad F(d,t) < a \\ P(a) + Q(a)\frac{F(d,t)-a}{1-a} & \text{for} \quad F(d,t) \geq a \end{cases} \quad (1)$$

The operators AND, OR, and NOT can be evaluated with the help of fuzzy set operations. Fuzzy set operations are extensions of crisp set operations on fuzzy sets [6]. A characteristic function uniquely defines a fuzzy set and hence fuzzy set operations are defined using characteristic functions [7]. In [6] L. Zadeh defined basic methods for the complement, union and intersection of fuzzy sets. Next to these standard (Zadeh's) fuzzy set operations, whole classes of prescriptions for defining the complements, intersections and unions on fuzzy sets were later designed [8].

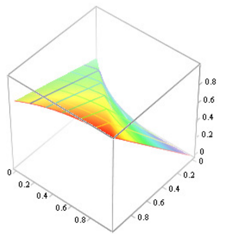

Fig. 1: $g(F(d,t),a)$ according to (1).

In this study, we use the threshold interpretation of RSV and standard t-norm (2) and t-conorm (3) for the implementation of AND and OR operators and fuzzy complement for the evaluation of NOT operator (4).

$$c(x) = 1 - x \quad (2)$$

$$t(x,y) = \min(x,y) \quad (3)$$

$$s(x,y) = \max(x,y) \quad (4)$$

However, the use of other common t-norm and t-conorm pairs is possible.

A. IR evaluation

The effectiveness of an information retrieval system can be evaluated using the measures precision P and recall R. Precision corresponds to the probability of retrieved document to be relevant and recall can be seen as the probability of retrieving relevant document.

Precision and recall in the extended Boolean IR model can be defined using the $\Sigma-$count $\|A\|$ [9]:

$$\rho(X|Y) = \begin{cases} \frac{\|X \cap Y\|}{\|Y\|} & \|Y\| \neq 0 \\ 1 & \|Y\| = 0 \end{cases} \quad (5)$$

$$P = \rho(REL|RET) \qquad R = \rho(RET|REL) \quad (6)$$

where REL stands for the fuzzy set of all relevant documents and RET for the fuzzy set of all retrieved documents.

For an easier IR effectiveness evaluation, measures combining precision and recall into one scalar value were developed. The F-score F is among the most used scalar combinations of P and R:

$$F = \frac{(1+\beta^2)PR}{\beta^2 P + R} \quad (7)$$

The index matrix $\boldsymbol{D}$ can be seen as a general data matrix with m rows (data samples) and n columns (data features). The evaluation of Extended Boolean query over the document collection generates an ordering of the documents (i.e. it assigns a real value from the range $[0,1]$ to each document). The ordering can be also interpreted as a fuzzy set of documents. If we abandon the IR terminology, we can call the extended Boolean query a general fuzzy classifier and use it to describe fuzzy sets or fuzzy sub sets of data by its features.

III. Genetic algorithms and genetic programming

The evolution of fuzzy classifiers for data mining utilizes genetic programming. In this section, we provide brief introduction into the area of evolutionary computing and genetic programming in particular. Moreover, we describe the application of genetic programming to evolutionary query optimization.

A. Genetic algorithms and genetic programming

Genetic algorithms are a popular member of the wide chapter of evolutionary algorithms. They are based on the programmatic implementation of genetic evolution and they emphasize selection and crossover as the most important operations in the evolutionary optimization process [10], [11]. Genetic algorithms evolve a population of chromosomes representing potential problem solutions encoded into suitable data structures. The evolution is performed by genetic operators modifying the chromosomes, i.e. the encoded forms of problem solutions. Genetic programming (GP) is an extension to genetic algorithms, allowing work with hierarchical, often tree-like, chromosomes with an unlimited length [10], [12]. In GP, the chromosomes take the form of hierarchical variably-sized expressions, point-labeled structure trees. The trees are constructed from nodes of two types, terminals and functions. The chromosomes are evaluated by the recursive execution of instructions corresponding to tree nodes [13]. Terminal nodes are evaluated directly (e.g. by reading an input variable) and functions are evaluated after left-to-right depth-first evaluation of their parameters.

Genetic operators are applied to the nodes in tree-shaped chromosomes. A crossover operator is implemented as the mutual exchange of randomly selected sub-trees of the parent chromosomes. Mutation has to modify the chromosomes by pseudo-random arbitrary changes in order to prevent premature convergence and broaden the coverage of the fitness landscape. Mutation could be implemented as:

i) removal of a sub-tree at a randomly chosen node
ii) replacement of a randomly chosen node by a newly generated subtree

iii) replacement of node instruction by a compatible node instruction (i.e. a terminal can be replaced by another terminal, a function can be replaced by another function of the same arity)
iv) a combination of the above

Genetic programming facilitates the efficient evolution of symbolic expressions, even whole computer programs. In this work, we use genetic programming for fuzzy classifier optimization.

B. Evolutionary query optimization

Genetic programming has been recently used for the optimization of extended Boolean queries [14], [15]. It was shown that genetic programming was able to optimize search queries so that they described a set of relevant documents. In the fuzzy information retrieval model, the relevant documents formed a fuzzy subset of the set of all documents and the extended Boolean queries were evolved to describe them.

An information retrieval system based on the extended Boolean IR model was implemented to validate evolutionary query optimization. The $tf \cdot idf_t$ term statistics [5] were used for document indexing and query weights (RSV) were evaluated using (1). The query language in the IRS supported the standard Boolean operators AND, OR, and NOT.

The information retrieval system served as a test bed for evolutionary query optimization and allowed genetic programming over extended Boolean queries. The GP evolved tree representations of search queries with Boolean operators as function nodes and terms as leaves. Both operator nodes and term nodes were weighted. In order to generate a random initial population for the GP, the system was able to generate random queries. The particular settings of the random query generator showing the probabilities of generating a particular query node are summarized in Table Ia.

The implementation of a crossover operator for GP is straightforward. In the experimental information retrieval system, it was implemented as a mutual exchange of two randomly selected branches of parent tree chromosomes. The mutation operator in query GP aims to perturb the content and structure of the chromosomes randomly. In our implementation, it selects a node from the processed chromosome at random and performs one of the mutation operations summarized in Table Ib.

The query mutation types that were implemented included:

i) change of selected node weight.
ii) replacement of selected node type by a compatible node type (i.e. operator OR replace by operator AND, term replaced by another term).
iii) insertion of NOT operator before selected node.
iv) removal of NOT operator if selected.
v) replacement of selected node by a randomly generated branch.

The IR measure F-Score (7) was used as a fitness function.

This study extends the framework for genetic evolution of extended Boolean queries to the evolution of general fuzzy classification rules. For test each data set, a fuzzy rule that would describe known classes can be found. Rules found using such a supervised learning procedure can be subsequently used to classify new data samples. The learning of a classifier can be long time process (depending on the dimension of the data). On the other hand the classification (i.e. evaluation of the classifier) of each data sample is very fast and can be used also in real time.

TABLE I: Random query generation an mutation probabilities.

(a) Probabilities of generating random query nodes.

Event	Probability
Generate term	0.5
Generate op. AND	0.24
Generate op. OR	0.24
Generate op. NOT	0.02

(b) Probabilities of mutation operations.

Event	Probability
Mutate node weight	0.5
Insert or delete NOT node	0.1
Replace with another node or delete NOT node	0.32
Replace with random branch	0.08

IV. Some use cases of fuzzy rule evolution

We have evaluated the evolution of fuzzy rules in two real world application areas. Genetic programming was used to evolve fuzzy rules describing faulty products in a product processing plant and rules detecting malicious traffic in a computer network. The latter classifier can be used for instance in an intrusion detection system.

A. Genetic evolution of fuzzy classifier for quality prediction

In heavy industry, a product is created. During its processing, a number of product features are measured and recorded. The features include the chemical properties of the raw material, density, temperature at several processing stages, and many other indicators that are recorded several times during the production. At the end, the product is classified as either flawless or defective. The data and classification for a number of product samples are known and the goal of the genetic programming is to find a fuzzy classifier that could be used for product quality prediction during product processing.

The problem differs from the query optimization task only semantically. We interpret products as documents and product features as terms. The product feature value then corresponds to the index weight of a term in a document (feature weight in a product). The product class corresponds to document relevance.

1) Initial evaluation: We have obtained a test data set from a product processing plant. The data set contained 204 samples with 839 features each. 200 samples described flawless products (class 0) and 4 samples described defective products (class 1). The raw product features values were normalized to the interval [0, 1]. A sample of product features data after normalization is shown in Table IIb. The mapping of normalized data onto an IRS index matrix is demonstrated in Table IIc. The goal of the optimization algorithm was to find a fuzzy classifier that would describe the set of defective products as well as possible. As the data set is very small and contains only 4 samples of defective products, the results of presented experiment should be seen as a proof of concept rather than a rigorous evaluation of the algorithm. Besides, it illustrates the mapping of data to IR concepts nicely.

	Feat.	Feat.	…	Feat.	Prod.
Id	1	2		839	class
1	122.42	120.8	…	31	1
2	122.61	120.5	…	32	1
3	123.13	121.7	…	32	0
⋮	⋮	⋮		⋮	⋮
204	117.99	116.1	…	21	0

(a) Product features.

	Feat.	Feat.	…	Feat.	Prod.
Id	1	2		839	class
1	0.846	0.951	…	0.148	1
2	0.856	0.9452	…	0.160	1
3	0.882	0.968	…	0.160	0
⋮	⋮	⋮		⋮	⋮
204	0.618	0.861	…	0.025	0

(b) Normalized product features

$$\boldsymbol{D} = \begin{pmatrix} 0.846 & 0.951 & \cdots & 0.148 \\ 0.856 & 0.9452 & \cdots & 0.160 \\ \vdots & \vdots & \ddots & \vdots \\ 0.618 & 0.861 & \cdots & 0.025 \end{pmatrix}$$

(c) Product features data set as an IRS index matrix $\boldsymbol{D}$.

TABLE II: Product features data set.

The fuzzy classifier that was evolved by the algorithm corresponds to a search expression that describes the class of defective products in terms of product features. The parameters of the executed GP (found after initial tuning of the algorithm) are shown in Table III.

During 12 independent optimization runs, the GP delivered a best classifier with a fitness of 0.9996 and a worst classifier a with fitness of 0.399872. Every fuzzy classifier reaching a fitness of 0.5 and higher was able to identify all defective products without an error or without false positives (i.e. without flawless products being marked as defective). A fuzzy classifier with a fitness higher than 0.5 was evolved in 10 cases out of 12 independent runs. An example of several evolved fuzzy classifiers is shown in Table IV.

The best classifier found by the algorithm was Q1. It was indeed a perfect expression describing defective products in the available data set. It was superior in terms of its F-Score, but also in terms of precision and recall because it describes defective products only.

2) 508 features data sets: After the initial success of evolutionary classifier induction, we have generated fuzzy rules describing faulty products in another industrial data set. The data set contained readings from 508 sensors on 5 different assembly lines. For each assembly line, the data was divided into training (20%) and test (80%) collection. We label the data sets according to the number of samples in the test collection, so e.g. D844 represents a collection with 844 test samples. However, since the data comes from different parts of the production plant, their internal structure differs as well and the patterns desrcibing faulty products in such data sets are unique.

TABLE III: GP parameters used to evolve fuzzy classifier for quality prediction.

Parameter	Value
Population size	100
Generations limit	1000
Fitness	F-Score
Mutation probability	0.8
Crossover probability	0.02
Independent runs	12

TABLE IV: An example of evolved fuzzy classifiers for quality prediction.

Label	Query		Fitness
Q1 (Best)	(Feat308:0.79	and:0.95	0.9996
	(Feat295:0.36	or:0.34	
	Feat413:0.99))		
Q2	Feat641:0.998113		0.5759
Q3	(Feat641:0.97	and:0.06	0.6066
	(Feat593:0.76	and:0.81	
	Feat421:0.80))		
Q4 (Worst)	Feat426:0.999203		0.3999

TABLE V: Classification results for different industrial data sets with 508 features.

	Data set				
	D844	D233	D1134	D73226	D3034
OA	97.63	97.00	99.50	96.99	99.60
FP	1.30	3.00	0	0.43	0.07
FN	1.07	0	0.53	2.58	0.33

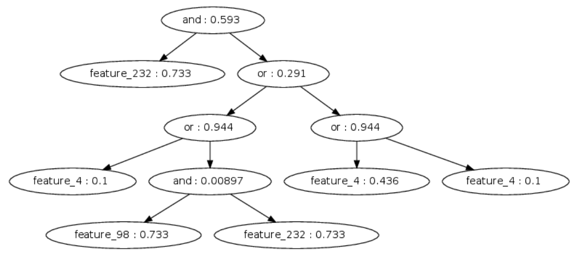

Fig. 2: An example of best fuzzy classifiers found for data set D1134.

In all cases, the evolved classifier reached good accuracy of 96.99 percent and above. However, the data sets were not really comprehensive.

The symbolic nature of the GP output gives us valuable information about the features that indicate product defectiveness. From Q1 in Table IV, we can see that the product can already be classified as faulty or flawless after the value of feature 413 (out of 839 measured product features) was read. Therefore, a defective product can be removed from production at an earlier stage and costs can be saved. Moreover, it is also a good clue telling us what features are really worth measuring. The other sensors can be suspended and savings can be made. Last but not least, the classifier provides also an important feedback on the production process. Production specialists can focus on adjusting the technology so that the reason for the problematic values of identified key features are eliminated in the future.

B. Genetic evolution of fuzzy classifier for intrusion detection

Next, the evolution of fuzzy classifier was applied to the domain of intrusion detection. Fuzzy rules were evolved for the KDD Cup 1999 intrusion detection data set, the de-facto standard benchmark for data mining methods in the intrusion detection area. It was shown that the KDD Cup 1999 data set suffers from a number of problems. The recently published NSL-KDD data set mitigates some of the imperfections found in KDD Cup 1999[16].

1) Experiments with the original KDD Cup 1999 data: The 10% sample of the KDD Cup 1999 intrusion detection dataset[1] was used to evolve classifiers and test their ability to detect illegal actions. It contains 10% of the large intrusion detection data set created in 1998 by the DARPA intrusion detection evaluation program at MIT. The full data set contains 744 MB data with 4,940,000 records with 41 nominal and numerical features. For our experiments, all features were converted to numeric and normalized.

The data describes normal traffic and 4 attack classes called DoS (Denial of Services), U2R (User to Root), R2L (Remote to User), and Probe (Probing). The records for each class are divided into training (40%) and testing (60%) data set. For each class, the training data set was used to evolve the fuzzy classifier and testing data set was used to evaluate the detection capabilities of the classifier. The attack classes contained following number of records: DoS contained 195,494 training and 293,242 testing records, U2R consisted of 38,931 training and 58,399 testing records, R2L included 39,361 training and 59,043 testing records, and finally the Probe class consisted of 40,553 training and 60,832 testing records.

2) Intrusion detection classifier evolution: We have sought for fuzzy classifiers describing the different attack types. For each attack type, there were 2 crisp classes: normal traffic (class 0) and attack (class 1). The goal of the algorithm was to find a fuzzy classifier that would describe the set of records describing an attack. The settings for the GP are summarized in tab:gaFuzzyClass2.

We have observed overall accuracy of the classification (OA) as the percent of correctly classified records in the test collection, false positives (FP) as the percent of regular traffic records classified as attacks, and false negatives (FN) as the percent of attacks classified as regular traffic. Obviously, good classifier would yield high OA, low FP and low FN.

[1]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

TABLE VI: GA parameters used for fuzzy classifier evolution.

Parameter	Value
Population size	100
Generations limit	5000
Fitness	F-Score
Mutation probability	0.8
Crossover probability	0.02
F-Score parameter	$\beta = 1$

The results of experiments are summarized in Table VII. An example of best classifiers found by GP is shown in Fig. 3.

TABLE VII: Classification results for different attack types in KDD Cup 1999.

	Attack class			
	Probe	R2L	U2R	DoS
OA	98.72	99.71	95.58	95.07
FP	0.46	0.07	3.69	0.48
FN	0.81	0.22	0.73	4.46

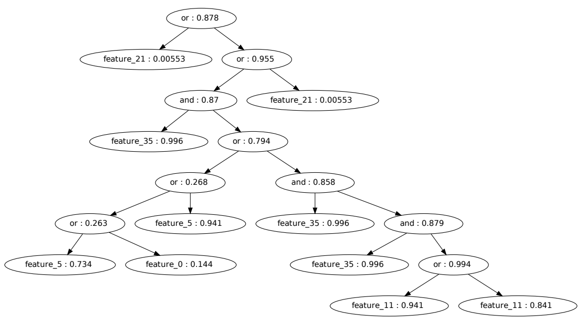

Fig. 3: An example of best fuzzy classifier found for Probe attack type in KDD Cup 1999.

We can see that the evolved classifier reached in all cases and for all attack types good accuracy higher than 95 percent. However, FN and FP are for some attack classes not as low as we would like. The best results were obtained for the R2L attack. The classifier managed to detect 99.71 percent of attacks and misclassified only acceptable 0.22 percent of harmless connections.

The different results for different attack classes suggest that the nature of the features describing the attacks varies and different GP parameters might need to be used.

C. NSL-KDD data set

The original KDD Cup data set, although widely used as benchmark for data minig techniques in the domain of intrusion detection, suffers from a number of problems [17], [16]. The problems include significant overlaps between training and testing data, unclear definitions of attack types, and possible issues caused by overload of data during data collecting.

In a response to some of the problems, a new data collection called NSL-KDD [2] was recently published. The NSL-KDD was created by removal of the redundant records an re-sampling of the records [16]. We have evolved fuzzy classifier over the KDDTrain$^+$ training data set and tested the classifier on the KDDTest$^+$ data set from NLS-KDD. The data sets contain $125,973$ and $22,544$ records classified as regular (class 0) and attack (class 1). The different types of attacks are not distinguished.

The fuzzy classifier found by genetic programming is shown in Fig. 4 and its performance is described in Table VIII.

TABLE VIII: Classification results for NLS-KDD data set.

	NLS-KDD
OA	82.74
FP	3.92
FN	13.30

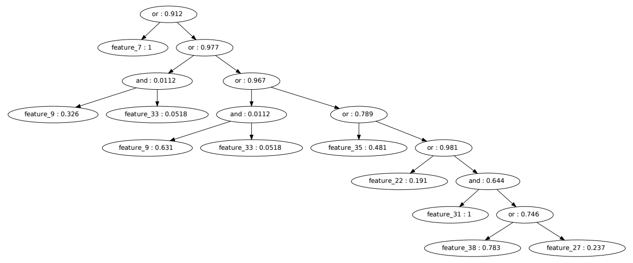

Fig. 4: Best fuzzy classifier found for the NLS-KDD data set.

The classifier found by genetic programming outperforms all methods presented in [16] in terms of classification accuracy. However, it does not yield as good results as some other methods applied on the NLS-KDD data set in the recent time [18]. Moreover, the evolution of the classifier was stopped after 5000 generations while there was still improvement seen in the consecutive populations.

V. Conclusions

We have implemented a genetic programming to evolve fuzzy classifiers for data mining. In contrast to previous efforts in this area (see e.g. [19]), our approach is inspired by information retrieval. We interpret data classes as fuzzy sets and evolve fuzzy search expressions that would describe such sets rather than traditional rule-based fuzzy classifiers. The data mining problems were reformulated as information retrieval tasks and the search query optimization algorithm was used to infer symbolic fuzzy classifiers describing classes of data records.

The evolution of fuzzy classifier for data mining is an ongoing project. We have used the genetic programming originally developed for query optimization and the results are encouraging. However, a number of tasks deserves attention. The choice of the best fitness function (are IR measures really the best fitness function for classifier evolution?) or the interpretation of the fuzzy weights in the classifier (is the IR retrieval status value the optimal choice?) are among the most appealing open questions.

[2] http://nsl.cs.unb.ca/NSL-KDD/

Acknowledgement

This work was supported by the Ministry of Industry and Trade of the Czech Republic, under the grant no. FR-TI1/420.

References

[1] J. C. Bezdek, J. Keller, R. Krisnapuram, and N. R. Pal, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing (The Handbooks of Fuzzy Sets)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[2] A. Verikas, J. Guzaitis, A. Gelzinis, and M. Bacauskiene, "A general framework for designing a fuzzy rule-based classifier," *Knowledge and Information Systems*, pp. 1–19, 2010.

[3] F. Crestani and G. Pasi, "Soft information retrieval: Applications of fuzzy set theory and neural networks," in *Neuro-Fuzzy Techniques for Intelligent Information Systems*, N. Kasabov and R. Kozma, Eds. Heidelberg, DE: Springer Verlag, 1999, pp. 287–315.

[4] D. H. Kraft, F. E. Petry, B. P. Buckles, and T. Sadasivan, "Genetic Algorithms for Query Optimization in Information Retrieval: Relevance Feedback," in *Genetic Algorithms and Fuzzy Logic Systems*, E. Sanchez, T. Shibata, and L. Zadeh, Eds. Singapore: World Scientific, 1997.

[5] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing and Management*, vol. 24, no. 5, pp. pp. 513–523, 1988.

[6] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. pp. 338–353, 1965.

[7] J. Jantzen, "*Tutorial On Fuzzy Logic*," Technical University of Denmark, Dept. of Automation, Technical Report 98-E-868 (logic), 1998.

[8] T. Feuring, *Fuzzy-systeme*. Institut für Informatik, Westfälische Wilhelms Universität, Münster, 1996.

[9] H. L. Larsen, "Retrieval evaluation," in *Modern Information Retrieval course*. Aalborg University Esbjerg, 2004.

[10] J. Koza, "*Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*," Dept. of Computer Science, Stanford University, Tech. Report STAN-CS-90-1314, 1990.

[11] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press, 1996.

[12] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[13] M. Affenzeller, S. Winkler, S. Wagner, and A. Beham, *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*. Chapman & Hall/CRC, 2009.

[14] D. Húsek, S. S. J. Owais, V. Snášel, and P. Krömer, "Boolean queries optimization by genetic programming," *Neural Network World*, pp. 359–409, 2005.

[15] V. Snasel, A. Abraham, S. Owais, J. Platos, and P. Kromer, *Emergent Web Intelligence: Advanced Information Retrieval*, ser. Advanced Information and Knowledge Processing. ch. User Profiles Modeling in Information Retrieval Systems, pp. 169–198, Springer London, 2010.

[16] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *CISDA'09: Proc. of the Second IEEE int. conf. on Computational intelligence for security and defense applications*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 53–58.

[17] M. Sabhnani and G. Serpen, "Why machine learning algorithms fail in misuse detection on kdd intrusion detection data set," *Intell. Data Anal.*, vol. 8, no. 4, pp. 403–415, 2004.

[18] M. Panda, A. Abraham, and M. Patra, "Discriminative multinomial naive bayes for network intrusion detection," in *Inf. Assurance and Security (IAS), 2010 Sixth Int. Conf. on*, 2010, pp. 5 –10.

[19] B. Carse and A. G. Pipe, "A framework for evolving fuzzy classifier systems using genetic programming," in *Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference*. AAAI Press, 2001, pp. 465–469.