Learning the Classification of Traffic Accident Types

Tibebe Beshah, Dejene Ejigu
IT Doctoral Program
Addis Ababa University
Addis Ababa, Ethiopia
Email: tibebe.beshah@gmail.com, ejigud@yahoo.com

Pavel Krömer, Václav Snášel, Jan Platoš, Ajith Abraham
Department of Computer Science
FEECS, VŠB Technical University of Ostrava
Ostrava, Czech Republic
Email: {pavel.kromer,vaclav.snasel,jan.platos}@vsb.cz,
ajith.abraham@ieee.org

***Abstract*—This paper presents an application of evolutionary fuzzy classifier design to a road accident data analysis. A fuzzy classifier evolved by the genetic programming was used to learn the labeling of data in a real world road accident data set. The symbolic classifier was inspected in order to select important features and the relations among them. Selected features provide a feedback for traffic management authorities that can exploit the knowledge to improve road safety and mitigate the severity of traffic accidents.**

I. Introduction

Labeling of data is a common task in data mining and data analysis. In this research, we use genetic programming to evolve a symbolic fuzzy classifier to classify traffic accidents according to their severity. Further, we select the attributes that are most important for accident severity classification.

Fuzzy classifiers constitute a class of tools and systems that exploit the fuzzy set theory to mine, label, and generally process data. There are simple fuzzy classifiers as well as complex rule-based fuzzy classification systems that usually build and maintain sophisticated rule bases. The popularity of fuzzy classifiers can be attributed to their ability to perform soft classification, to assign multiple labels to data samples, and to the ease of their interpretation.

Genetic programming is a nature inspired search and optimization method that was designed to evolve tree-like structures in an automated manner. As such, it is a good tool to evolve symbolic expressions such as the fuzzy predictor.

In this work, we use the genetic programming to evolve a fuzzy predictor inspired by the area of information retrieval [16], [12], [13]. When compared to a more complex fuzzy classifier systems, it can be seen as a single fuzzy rule that maps data features onto a real value from the range $[0, 1]$. This evolutionary classification method is used to learn the severity of traffic accidents in a data set with the records of real traffic accidents in Ethiopia. The main aim of the research is the discovery of important attributes and the relations among them.

~~A.~~ *Fuzzy classification systems evolved by evolutionary algorithms*

The design of fuzzy classifiers and fuzzy rule-based systems has been successfully aided by the nature inspired methods in the recent years. In this section we summarize few examples of such an evolution or more generally nature inspired fuzzy classifier design. For a comprehensive survey on the automated evolution of fuzzy classification tools see e.g. [2].

Multi-objective evolutionary algorithms were used for the evolution of linguistic fuzzy rule-based classification systems in the work of Cordón et al. [3]. Another multi-objective evolutionary approach to the evolution of fuzzy rule-based systems was proposed by Ishibuchi and Nojima [7]. They used a hybrid 2-stage approach that combined an initial heuristic stage to select fuzzy rules and evolutionary stage to optimize and tune the system.

Wang et al. [17] used the genetic algorithms to integrate fuzzy rule sets and membership functions learned from various information sources. In [6], Freischlad et al. used an evolutionary algorithm to generate fuzzy rules for knowledge representation. Zhou and Khotanzad [20] used a genetic algorithm to learn various parameters of a fuzzy classification system from a training data set.

The usage of another nature inspired method - the particle swarm optimization - for fuzzy classification system design was studied recently in [15].

II. Fuzzy predictor

In this work we develop a single fuzzy classifier heavily inspired by the area of information retrieval (IR). In the IR, the extended Boolean IR model employs fuzzy set theory and fuzzy logic to facilitate flexible and accurate search [4], [11]. In essence, a search query that contains search terms, operators, and weights is evaluated against an internal document representation that is modeled as a fuzzy set of index terms, i.e. document terms with weights assigned.

In the framework of the fuzzy classifier, we use similar data structures, basic concepts, and operations as in the fuzzy IR and we apply them to general data processing such as classification, prediction, and so forth.

The data base used by the fuzzy classifier is a real valued matrix. Each row of the matrix corresponds to a single data record which is interpreted as a fuzzy set of features. Such a general real valued matrix $\boldsymbol{D}$ with m rows (data records) and n columns (data attributes) can be mapped to an IR index that describes a collection of documents.

The fuzzy predictor has the form of a weighted symbolic expression roughly corresponding to an extended Boolean query in the fuzzy IR analogy. The predictor consists of weighted feature names and weighted aggregation operators.

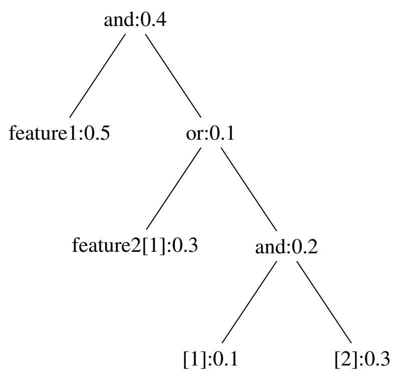

Fig. 1: Tree form of a fuzzy predictor

The evaluation of such an expression assigns a real value from the range $[0,1]$ to each data record. Such a valuation can be interpreted as an ordering, labeling, or a fuzzy set induced on the data records.

A. *Fuzzy predictor structure*

The fuzzy predictor is a symbolic expression that can be parsed into a tree structure. The tree structure consists of nodes and leafs (i.e. terminal nodes). In our fuzzy predictor, we recognize three types of terminal nodes:

- *feature* node - which represents the name of a feature (a search term in the IR analogy). It defines a requirement on a particular feature in the currently processed data record.
- *past feature* node - which defines a requirement on certain feature in a previous data record. The index of the previous data record (current - 1, current - 2 etc.) is a parameter of the node.
- *past output* node - which puts a requirement on a previous output of the predictor. The index of the previous output (current - 1, current - 2) is a parameter of the node.

An example of fuzzy classifier is shown in fig. 1. Written down using a simple infix notation, a predictor shown in fig. 1 would look like:

> *feature1:0.5 and:0.4 (feature2[1]:0.3 or:0.1 ([1]:0.1 and:0.2 [2]:0.3))*

The syntax we use for linear representation of the predictor is very simple. The feature node is defined by feature name and its weight (e.g. *feature1:0.5*), past feature node is defined by feature name, index of previous record, and weight (e.g. *feature2[1]:0.3*), and past output node is defined by the index of previous output and weight (e.g. *[1]:0.5*).

Different node types can be used when dealing with different data sets. For example, the past feature node and past output node are useful for the analysis of time series and data sets where the ordering of the records matters, but their usage is pointless for the analysis of regular data sets. The feature node is the basic building block of predictors developed for any type of data.

The operator nodes supported currently by the fuzzy predictor are *and*, *or*, and *not* node but more general or domain specific operators can be defined. Both nodes and leafs are weighted to soften the criteria they represent.

B. *Fuzzy predictor evaluation*

The fuzzy predictor evaluation is inspired by the fuzzy IR in a similar manner as the data structures it uses. The most important part of the predictor evaluation is the matching of feature values in data records to predictor feature weights. In the IR is the result of such a feature value - feature weight matching called the retrieval status value (RSV) and it evaluates how a document satisfies the criteria represented by a single search criterion defined by the terminal node.

Consider a to be the weight of feature f in the predictor (i.e. the predictor contains a terminal node *f:a*) and $F(d,f)$ to be the value of feature f in data record $d \in \boldsymbol{D}$. The terminal node represents a single criterion. To evaluate this criterion the function $g : [0,1] \times [0,1] \to [0,1]$ will be used. The value of $g(F(d,f),a)$ is the actual RSV for data sample d, feature f, and predictor feature weight a. The key point for RSV evaluation is the interpretation of the predictor feature weight a. The most commonly used IR interpretations see the predictor feature weight as an importance weight, a document term weight threshold, or an ideal document feature weight [4], [11].

We are using the threshold interpretation of a and the equation for RSV evaluation for this case is shown in (1) [4], [11]. In (1), $P(a)$ and $Q(a)$ are coefficients used to tune the shape of the threshold curve. An example of $P(a)$ and $Q(a)$ could be e.g. $P(a) = \frac{1+a}{2}$ and $Q(a) = \frac{1-a^2}{4}$. For the threshold interpretation, a node containing feature f with the weight a is a request satisfied by data samples having $F(d,f)$ equal or greater than a. The data samples satisfying this condition will be awarded with high RSV and the data records with $F(d,f)$ smaller than a will be awarded with small RSV.

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

The operators *and*, *or*, and *not* can be evaluated using fuzzy set operations. Fuzzy set operations are extensions of crisp set operations on fuzzy sets [18]. They are defined using the characteristic functions of operated fuzzy sets [8]. In [18] L. Zadeh defined basic formulas to evaluate the complement, union, and intersection of fuzzy sets but besides these standard fuzzy set operations, whole classes of prescriptions for the complements, intersections, and unions on fuzzy sets were defined [5].

In this study, we use the standard t-norm (2) and t-conorm (3) for the implementation of *and* and *or* operators and fuzzy complement for the evaluation of the *not* operator (4).

$$t(x,y) = \min(x,y) \tag{2}$$

$$s(x,y) = \max(x,y) \tag{3}$$

$$c(x) = 1 - x \tag{4}$$

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

The fuzzy predictor presented in this work is a simple version of a fuzzy classifier. In contrast to more complex fuzzy rule-based systems that usually constitute traditional fuzzy classifiers, it consists of a single expression that states soft requirements on data records in terms of data features. Moreover, conditions can be put on past feature values and past output values and therefore allow the predictor to see the data base as an ordered sequence of records similar to a time series.

III. Genetic programming

Genetic programming (GP) is an extension to the popular nature inspired stochastical optimizer, the genetic algorithms [9], [14]. Genetic algorithms perform an artificial (software) evolution of a population of chromosomes representing potential solutions to an investigated problem encoded into a suitable data structures, most often fixed length strings of low cardinality alphabets (e.g. bit strings). The artificial evolution is performed by an iterative application of genetic operators modifying the chromosomes, in order to emulate the principles of the Darwinian evolution, the survival of the fittest, and the Mendelian inheritance.

The GP extends the genetic algorithms by enabling work with hierarchical, often tree-like, chromosomes with an uneven and unlimited length [9], [10]. The GP was introduced as a tool to evolve simple computer programs and represented a step towards adaptable computers that could solve problems without being programmed explicitly [1]. Moreover, the GP can be used to develop solutions in the field of machine learning, symbolic processing, or any other domain that can formulate its solutions by means of parseable symbolic expression. GP allows the efficient evolution of such symbolic expressions with well-defined syntax and grammar. GP 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 execution of instructions corresponding to tree nodes [1]. 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 on the nodes of the 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 sub-tree
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

TABLE I: Random query generation an mutation probabilities.

((a)) Probabilities of generating random nodes.

Event	Probability
Generate feature node	0.17
Generate past feature node	0.17
Generate past output node	0.17
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

The GP facilitates an efficient evolution of symbolic expressions. In this work, we use the GP for an automated fuzzy classifier development.

~~A.~~ *GP for fuzzy predictor evolution*

To use the GP for fuzzy predictor learning, we need to define the encoding, genetic operators, and the fitness function. The encoding is straightforward because the fuzzy predictor is in essence a tree (see fig. 1). We create a random population of such trees (candidate predictors) and apply the GP to evolve the population. The generation of random predictors is done with respect to the probabilities summarized in .

The implementation of the crossover operator is also simple: for every two trees, we swap randomly selected branches. Such an operation results in valid fuzzy predictors. The mutation operator is more complex because it has to reflect the domain of the problem and properties of each node that should be mutated. The mutation types that were implemented in this work and the probabilities of their application are shown in table I(b). The goal of the fuzzy predictor evolution is to find such a predictor that would describe the same fuzzy set of data records as indicated in the training data base.

The similarity of two fuzzy sets can be defined as:

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

where $\|A\|$ is the $\Sigma-$count, i.e. the sum of the values of characteristic function for all members of the fuzzy set A [19]:

$$\|A\| = \sum_{x \in A} \mu_A(x) \tag{6}$$

Precision P and recall R are two measures that can be used to evaluate the effectiveness of an IR system and we use them to determine the suitability of a candidate fuzzy predictor. In the IR, precision corresponds to the probability of retrieved document to be relevant and recall can be seen as the probability of retrieving relevant document. We use precision P and recall R to evaluate the similarity of two fuzzy sets:

$$P = \rho(t(\boldsymbol{D})|p(\boldsymbol{D})) \qquad R = \rho(p(\boldsymbol{D})|t(\boldsymbol{D})) \tag{7}$$

where $t(\boldsymbol{D})$ stands for the target fuzzy set and $p(\boldsymbol{D})$ for the fuzzy set generated by the predictor f. For an easier 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:

$$F = \frac{(1+\beta^2)PR}{\beta^2 P + R} \tag{8}$$

We use the F-score F as a fitness function when evolving the fuzzy predictor. A good fuzzy predictor that generates fuzzy set that is similar to the training fuzzy set will yield high precision, recall, and F-score.

IV. Learning the classification of traffic accident types

We have used the fuzzy classifier to learn the labeling of different classes of data describing traffic accidents in Ethiopia. We have used a manually created data set obtained from the Addis Ababa Traffic Office. The data set captures more than 12336 traffic accidents that happened in Ethiopia between the years 1998 and 2000. The record about each accident contains information about accident time (e.g. year, month, week, day, hour), involved vehicle (e.g. vehicle type, technical condition), driver of the vehicle (e.g. driver age, driving experience), accident location and local conditions (e.g. road condition, road orientation, weather situation), and accident victim (e.g. type of victim, type of collision, injury severity). Several attributes in the data set are numerical (e.g. accident year) and the rest is categorial (e.g. vehicle type). The accidents are classified according to the severity into four classes: accidents with no injury, accidents resulting in an injury, fatal accidents, and accidents with an unknown severity.

The goal of this research was to develop a fuzzy classifier that would label data in the data set. The classifier was not intended for real-time (e.g. online) labeling of traffic accidents (especially because the severity of the accident is usually already known at the time when the data is available) but rather to see the attributes and the relations among attributes that were used for accident severity classification. The knowledge of the important attributes is intended to help in defining new road traffic regulations, preparing new road safety measures, and prioritizing traffic management and constructions.

The evolutionary learning was executed independently several times to verify the ability of the stochastic genetic programming to learn a classifier that would divide the records to appropriate classes. The settings used for the GP are summarized in table II. The mutation operator was implemented according to table I(b). We have executed the evolution for 5000 and 10000 generations. On average, the 10000 generations were processed in 2 and a half minutes. The best classifier found by the 5000 runs had the fitness 0.973079 (the maximum was 1), the best classifier produced by the 10000 runs had the fitness 0.973842. The best classifier found during the experiments is shown in fig. 2. Only the features shown in fig. 2 are considered for the classification. They were found important during the evolution of the classifier. The results of classification by the best classifier are shown in table III. We can see that the attributes taken into account for the accident record labeling were *road surface*, *victim health status*, *driver sex*, *accident sub city*, *accident hour*, *accident month*, *driver age*, *vehicle type*, *vehicle technical status*, *vehicle service year*, *driving licence*, *driving experience*, and *road orientation*. Some of the attributes are intuitively important (e.g. the type of vehicle can have great influence on the severity of the accident) but some of them are rather surprising (e.g. accident month, road orientation). The results are not bad considering the incompleteness of the data set and unbalanced size of the classes. We can see that the evolved classifier was able to separate classes no injury and injury but it failed to ~~assign~~ the fatal accident severity class ~~its own label.~~ However, the majority of fatal accidents were classified as an injury, i.e. they were recognized as more dangerous than accidents with no injury. Even with the whole fatal class misslabeled, the precision of the classifiaction reached 94.2%. The result suggests that a separate classifier for each class should be evolved to reach better precision and sensitivity of the labeling.

TABLE II: GP settings.

Parameter	Value
Population size	100
Crossover probability	0.8
Mutation probability	0.2
F-score β	1.0
No. of. generation	5000 or 10000

TABLE III: Classification results.

Class	Size	Correct	Incorrect	Note
Unknown	3	0	3	All unknown records were classified as no injury.
No injury	9550	9466	84	
Injury	2232	2155	77	
Fatal	551	0	551	3 fatal records were classified as unknown, 71 were classified as no injury, and 474 were classified as injury

V. Conclusions

We have used genetic programming to evolve a fuzzy classifier to label records in a data set describing traffic accidents. A real world data set from Ethiopia was used to train the classifier. Thanks to the symbolic nature of the classifier, the attributes that were used for data labeling can be easily extracted and explored. The most successful classifier used only 13 attributes out of 43 available attributes. The number of important attributes, their relations and weights were selected automatically and they represent an important knowledge that can be exploited to improve road safety.

In the future, we want to improve the data set and the

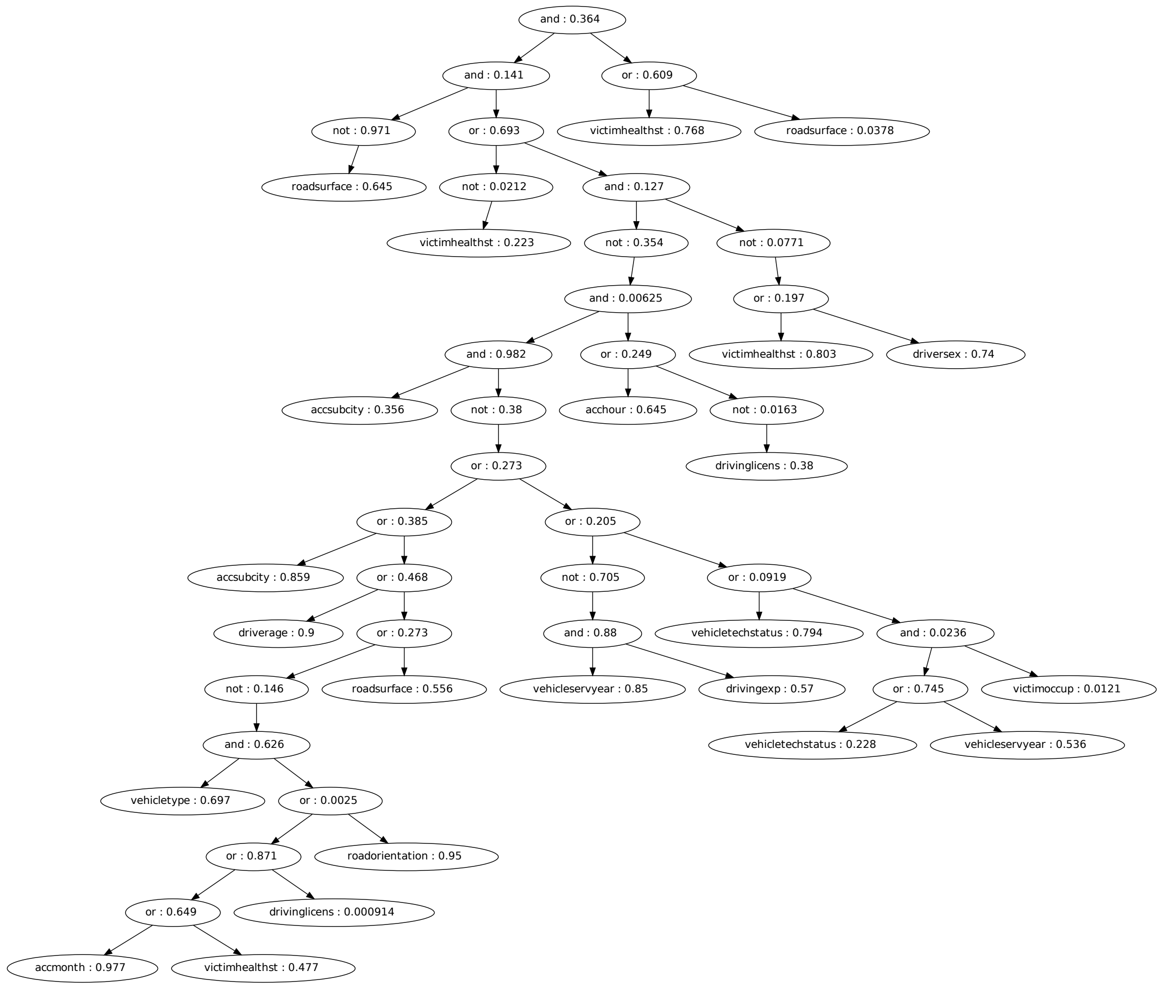

Fig. 2: Best classifier

precision of the classification. We will eliminate the noise (i.e. records with unknown class) from the data, remove the most incomplete records and balance the accident severity classes. Next, we will evolve classifiers for other attributes that represent accident results (attributes with categories such as property damage etc.). Moreover, another variant of the RSV might be evaluated for categorial data.

REFERENCES

[1] Michael Affenzeller, Stephan Winkler, Stefan Wagner, and Andreas Beham, *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*, Chapman & Hall/CRC, 2009.

[2] Oscar Cordón, 'A historical review of evolutionary learning methods for mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems', *International Journal of Approximate Reasoning*, **52**(6), 894 – 913, (2011).

[3] Oscar Cordón, María José del Jesus, and Francisco Herrera, 'Genetic learning of fuzzy rule-based classification systems cooperating with fuzzy reasoning methods', *International Journal of Intelligent Systems*, **13**, 1025–1053, (1998).

[4] Fabio Crestani and Gabriella Pasi, 'Soft information retrieval: Applications of fuzzy set theory and neural networks', in *Neuro-Fuzzy Techniques for Intelligent Information Systems*, eds., N. Kasabov and R. Kozma, 287–315, Springer Verlag, Heidelberg, DE, (1999).

[5] Thomas Feuring, *Fuzzy-systeme*, Institut fr Informatik, Westflische Wilhelms Universitt, Mnster, 1996.

[6] Mark Freischlad, Martina Schnellenbach-Held, and Torben Pullmann, 'Evolutionary generation of implicative fuzzy rules for design knowledge representation', in *EG-ICE*, ed., Ian F. C. Smith, volume 4200 of *Lecture Notes in Computer Science*, pp. 222–229. Springer, (2006).

[7] Hisao Ishibuchi and Yusuke Nojima, 'Multiobjective formulations of fuzzy rule-based classification system design.', in *EUSFLAT Conf.*, eds., Eduard Montseny and Pilar Sobrevilla, pp. 285–290. Universidad Polytecnica de Catalunya, (2005).

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

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

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

[11] 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*, eds., E. Sanchez, T. Shibata, and L.A. Zadeh, Singapore, (1997). World Scientific.

[12] Pavel Krömer, Jan Platoš, Václav Snášel, and Ajith Abraham, 'Fuzzy classification by evolutionary algorithms', in *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 313 – 318. IEEE System, Man, and Cybernetics Society, (2011).

[13] Pavel Krömer, Jan Platoš, Václav Snášel, Ajith Abraham, Lukáš Prokop, and Stanislav Mišák, 'Genetically evolved fuzzy predictor for photovoltaic power output estimation', in *2011 Third International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, pp. 41 – 46. IEEE, (2011).

[14] Melanie Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, 1996.

[15] R.P. Prado, S. Garcia-Galán, J. Exposito, and A.J. Yuste, 'Knowledge acquisition in fuzzy-rule-based systems with particle-swarm optimization', *Fuzzy Systems, IEEE Transactions on*, **18**(6), 1083 –1097, (dec. 2010).

[16] Václav Snásel, Pavel Krömer, Jan Platos, and Ajith Abraham, 'The evolution of fuzzy classifier for data mining with applications', in *SEAL*, eds., Kalyanmoy Deb, Arnab Bhattacharya, Nirupam Chakraborti, Partha Chakroborty, Swagatam Das, Joydeep Dutta, Santosh K. Gupta, Ashu Jain, Varun Aggarwal, Jürgen Branke, Sushil J. Louis, and Kay Chen Tan, volume 6457 of *Lecture Notes in Computer Science*, pp. 349–358. Springer, (2010).

[17] Ching-Hung Wang, Tzung-Pei Hong, and Shian-Shyong Tseng, 'Integrating membership functions and fuzzy rule sets from multiple knowledge sources', *Fuzzy Sets Syst.*, **112**, 141–154, (May 2000).

[18] L. A. Zadeh, 'Fuzzy sets', *Information and Control*, **8**, pp. 338–353, (1965).

[19] Lotfi A. Zadeh, *Empirical Semantics*, volume 1 of *Quantitative Semantics, Vol. 12*, chapter Test-score semantics dor natural languages and meaning representation via Pruf, 281–349, Studienverlag Brockmeyer, Bochum, 1981.

[20] Enwang Zhou and Alireza Khotanzad, 'Fuzzy classifier design using genetic algorithms', *Pattern Recogn.*, **40**, 3401–3414, (December 2007).