# The Evolution of Fuzzy Classifier for Data Mining with Applications

Václav Snášel[1,2], Pavel Krömer[1], Jan Platoš[1], and Ajith Abraham[2]

[1] Department of Computer Science
Faculty of Electrical Engineering and Computer Science
VŠB – Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava – Poruba, Czech Republic
{vaclav.snasel,pavel.kromer,jan.platos}@vsb.cz
[2] Machine Intelligence Research Labs (MIR Labs),
Washington 98071, USA
ajith.abraham@ieee.org

**Abstract.** Fuzzy classifiers and fuzzy rules can be informally defined as tools that use fuzzy sets or fuzzy logic for their operations. In this paper, we use genetic programming to evolve a fuzzy classifier in the form of a fuzzy search expression to predict product quality. We interpret the data mining task as a fuzzy information retrieval problem and we apply a successful information retrieval method for search query optimization to the fuzzy classifier evolution. We demonstrate the ability of the genetic programming to evolve useful fuzzy classifiers on two use cases in which we detect faulty products of a product processing plant and discover intrusions in a computer network.

**Keywords:** genetic programming, information retrieval, classifier evolution, fuzzy systems.

## 1 Introduction

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, symbolic expressions, decision trees, and recently to infer search queries from relevance ranked documents in a fuzzy information retrieval system.

The last application is interesting for the general data mining as well. It can be directly applied in the data mining domain. Extended Boolean queries (i.e. fuzzy queries) 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 subsequently be used for efficient classification of new data samples and e.g. predict quality of products or detect harmfull actions in computer network. Artificial evolution of search expressions is a promising approach to data mining because genetic programming yields very

K. Deb et al. (Eds.): SEAL 2010, LNCS 6457, pp. 349–358, 2010.

good ability to find symbolic expressions in other problem domains. The general process of classifier evolution can be used to evolve custom classifiers for many data sets with different properties.

## 2 Fuzzy Information Retrieval

The evolution of fuzzy classifiers for data mining is implemented within the framework for search query optimization designed for efficient information retrieval. Data records are interpreted as documents and data features are mapped to index terms.

The area of information retrieval (IR) is a branch of computer science dealing with storage, maintenance, and searching in large amounts of data [1]. It defines and studies IR systems and models.

An IR model is a formal background defining the document representation, query language, and document-query matching mechanism of an IR system. The proposed classification algorithm builds on the extended Boolean IR model, which is based on the fuzzy set theory and fuzzy logic [1,2]. Documents are interpreted as fuzzy sets of indexed terms, assigning to every term contained in the document a particular weight from the range $[0, 1]$ expressing the degree of significance of the term for document representation. A formal description of a document collection in the extended Boolean IR model is shown in Eqn. (1) and Eqn. (2), where $d_i$ represents $i-$th document and $t_{ij}$ $j-$th term in $i-$th document. An index matrix of the entire document collection is denoted $\boldsymbol{D}$.

$$\boldsymbol{d}_i = (t_{i1}, t_{i2}, \ldots, t_{im}),\ \forall\ t_{ij} \in [0, 1] \tag{1}$$

$$\boldsymbol{D} = \begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ t_{21} & t_{22} & \cdots & t_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & \cdots & t_{nm} \end{pmatrix} \tag{2}$$

Each IR model defines a query language to specify search requests. The query language in the extended Boolean model allows weighting search terms in order to attribute them different levels of importance. Moreover, the aggregation operators (most often AND, OR and NOT) can be weighted to parameterize their impact on query evaluation [1,2].

In this study, we adopt the threshold interpretation of the query term weights. In the threshold interpretation, an atomic query (i.e. a query with one weighted term representing single search criterion) containing term $t_i$ with the weight $a$ is a request to retrieve documents having index term weight $t_i$ equal or greater than $a$ [1].

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 $\|A\|$ [3]:

$$\rho(X|Y) = \begin{cases} \frac{\|X \cap Y\|}{\|Y\|} & \|Y\| \neq 0 \\ 1 & \|Y\| = 0 \end{cases} \tag{3}$$

$$P = \rho(REL|RET) \qquad R = \rho(RET|REL) \tag{4}$$

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 $F = \frac{2PR}{(P+R)}$ [4] is among the most used scalar combinations of $P$ and $R$.

## 3 Genetic Programming for the Evolutionary Query Optimization

Genetic programming (GP) [5,6], an offshoot of genetic algorithms [7], facilitates the efficient artificial evolution of symbolic expressions. In this work, we use the genetic programming originally developed for search query optimization (see e.g. [8,9]) to evolve general fuzzy classifiers for data mining. Genetic programming was chosen due its ability to evolve symbolic tree-like expressions that correspond to search queries. The symbolic nature of the algorithm output also allows good possiblility of verification and feedback from the side of system users.

It was shown that the GP 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 collection of all documents and the extended Boolean queries that would describe them were evolved.

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 [10] was used for document indexing and the threshold interpretation of query term weights was implemented. 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 the evolutionary query optimization. The GP evolved tree representations of the search queries with the 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 parameters of the random query generator showing the probabilities of generating a particular query node are summarized in Table 1a.

The crossover operator was implemented simply as a mutual exchange of two randomly selected branches of parent tree chromosomes. The mutation operator

**Table 1.** Random query generation an mutation probabilities

(a) Probabilities of generating random query nodes

| Event | Probability |
|---|---|
| Generate term | 0.5 |
| Generate operator AND | 0.24 |
| Generate operator OR | 0.24 |
| Generate operator 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 |

selected a node from the processed chromosome at random and performed one of the mutation operations summarized in Table 1b.

The query mutation types that were implemented included:

- change of selected node weight
- replacement of selected node type by a compatible node type (e.g. operator OR replaced by an AND, term replaced by another term)
- insertion of the NOT operator before selected node
- removal of the NOT operator if selected
- replacement of selected node by a randomly generated branch

The F-Score was used as a fitness function. An experimental evaluation of such an information retrieval system showed that the GP can find search expression describing fuzzy sets of relevant documents [8,9].

The extended queries evolved by the algorithm can be seen as fuzzy classifiers describing the fuzzy set of relevant documents, or, more generally, a fuzzy set of data records. The fuzzy classifier evolved over a training data set can be easily used to classify new data samples.

## 4 Applications

The algorithm for evolutionary query optimization was applied to the evolution of a general symbolic fuzzy classifier. In this work, we have evolved a fuzzy classifier for quality prediction in an industrial manufacturing process and for intrusion detection in an intrusion detection system.

**Table 2.** Product features data set

(a) Normalized product features

| Id | Feat. 1 | Feat. 2 | ... | Feat. 839 | Prod. 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) Product features data set as an IRS index matrix $\boldsymbol{D}$.

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

### 4.1 Genetic Evolution of Fuzzy Classifier

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 algorithm 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.

We have obtained a test dataset from a product processing plant. The dataset contained 204 samples with 839 features each. 200 samples described flawless products (class 0) and 4 samples described deffective 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 2a. The mapping of normalized data onto an IRS index matrix is demonstrated in Table 2b. The goal of the optimization algorithm was to find a search expression (fuzzy classifier) that would describe the set of defective products as good as possible. As the data set is very small and contains only 4 samples of deffective products, the results of presented experiment should be seen as a proof of concept rather than a rigorous evaluation of the algorithm.

We have implemented the GP for the evolution of fuzzy classifiers. 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 3.

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

**Table 3.** 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 | 0.8 |
| Independent runs | 12 |

**Table 4.** 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 |

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 4.

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

The symbolic nature of the GP output gives us valuable information about the features that indicate product defectiveness. From Q1, 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.

### 4.2 Genetic Evolution of Fuzzy Classifier for Intrusion Detection

Next, we have implemented an evolution of fuzzy classifier for intrusion detection. In this experiment, larger-scale data were processed by the algorithm.

To investigate the ability of discussed algorithm to find useful classifiers, a test sytsem implementing evolution of fuzzy expressions was implemented. 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.

**Intrusion detection classifier evolution.** We interpret data samples in intrusion detection data set as documents and its features as terms. The normalized feature value corresponds to the index weight of a term in a document (feature weight in a data sample) while the class of the record corresponds to document relevance. In the testing data, there are only 2 crisp product classes: normal traffic (class 0) and attack (class 1). The goal of the algorithm was to find an expression (fuzzy classifier) that will describe the set of records describing an attack.

The settings for the GP are summarized in Table 5.

**Table 5.** GA parameters used for fuzzy classifier evolution

| Parameter | Value |
|---|---|
| Population size | 100 |
| Generations limit | 5000 |
| Fitness | F-Score |
| Mutation probability | 0.8 |
| Crossover probability | 0.8 |

The F-Score parameter $\beta$ was set in different experiments to 1, 0.5 and 5 to see detection capabilities of evolved classifiers with different priorities of precision and recall in fitness function. We have observed overall accuracy of the classification (OA) as the percent of correctly classified records in the test collection, detection rate (DR), e.g. the percent of correctly classified attacks and

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

**Table 6.** Calssification results for different attack classes

(a) Results for attack class DoS

| | $\beta$ | | |
|---|---|---|---|
| | 0.5 | 1 | 5 |
| OA | 93.95 | 99.31 | 95.22 |
| DR | 99.42 | 99.27 | 94.04 |
| FP | 28.07 | 0.53 | 0.05 |

(b) Results for attack class U2R

| | $\beta$ | | |
|---|---|---|---|
| | 0.5 | 1 | 5 |
| OA | 99.95 | 99.96 | 99.95 |
| DR | 50 | 34.34 | 50 |
| FP | 0.02 | 0 | 0.02 |

(c) Results for attack class R2L

| | $\beta$ | | |
|---|---|---|---|
| | 0.5 | 1 | 5 |
| OA | 93.95 | 98.87 | 99.09 |
| DR | 99.42 | 38.17 | 31.07 |
| FP | 28.07 | 0.43 | 0.12 |

(d) Results for attack class Probe

| | $\beta$ | | |
|---|---|---|---|
| | 0.5 | 1 | 5 |
| OA | 94.02 | 98.46 | 98.34 |
| DR | 90.34 | 63.2 | 59.11 |
| FP | 5.83 | 0.05 | 0.01 |

false positives (FP), e.g. the percent of regular records missclassified as attacks. Obviously, good classifier would feature high OA, high DR and low FP.

The results of experiments are summarized in Table 6.

We can see that the evolved classifier reached in all cases and for all attack types good accuracy higher than 93 percent. However, DR and FP are for some attack classes not so good. The best combination of high DR and low FP was reached for DoS and $\beta = 1$. The classifier managed to detect 99.27 percent of attacks and misclassified only acceptable 0.53 percent of harmless connections. For the U2R attack class, the best classifiers managed to detect 50 percent of the attacks. In R2L experiment, the classifier evolved with $\beta = 0.5$ reached DR 99.42 percent, but it also misclassified close to 30 percent of harmless connections. The classifiers with low FP managed to detect only 38 and 31 percent of attacks. Finally, the classifiers evolved for Probe attacks managed to detect fair 90 percent of attacks at the cost of 5.83 percent of false positives for $\beta = 0.5$ and around 60 percent of attacks with FP percent below 0.05 for $\beta = 1$ and $\beta = 5$.

The different results for different attack classes suggest that the nature of the fetaures describing the attacks varies and different settings for GP (e.g. the value of $\beta$) needs to be used. Moreover, we have seen that high overall acuuracy of classification does not imply good detection rate and low misclassification of legitimate traffic.

## 5 Conclusions

We have implemented a genetic programming to evolve fuzzy classifiers for data minig. In contrast to previous efforts in this area (see e.g. [11]), 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.

## 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. Crestani, F., Pasi, G.: Soft information retrieval: Applications of fuzzy set theory and neural networks. In: Kasabov, N., Kozma, R. (eds.) Neuro-Fuzzy Techniques for Intelligent Information Systems, pp. 287–315. Springer, Heidelberg (1999)
2. Kraft, D.H., Petry, F.E., Buckles, B.P., Sadasivan, T.: Genetic Algorithms for Query Optimization in Information Retrieval: Relevance Feedback. In: Sanchez, E., Shibata, T., Zadeh, L. (eds.) Genetic Algorithms and Fuzzy Logic Systems. World Scientific, Singapore (1997)
3. Larsen, H.L.: Retrieval evaluation. In: Modern Information Retrieval Course, Aalborg University Esbjerg (2004)
4. Losee, R.M.: When information retrieval measures agree about the relative quality of document rankings. Journal of the American Society of Information Science 51(9), 834–840 (2000), `http://citeseer.ist.psu.edu/losee00when.html`
5. Koza, J.: Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Dept. of Computer Science, Stanford University, Technical Report STAN-CS-90-1314 (1990)
6. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
7. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge (1996)
8. Húsek, D., Owais, S.S.J., Snášel, V., Krömer, P.: Boolean queries optimization by genetic programming. Neural Network World, 359–409 (2005)
9. Snasel, V., Abraham, A., Owais, S., Platos, J., Kromer, P.: Emergent Web Intelligence: Advanced Information Retrieval. In: User Profiles Modeling in Information Retrieval Systems. Advanced Information and Knowledge Processing, Springer, London (2010), `http://www.springerlink.com/content/p74163276h776457/`, doi:10.1007/978-1-84996-074-8_7 iSBN 978-1-84996-073-1 (Print) 978-1-84996-074-8 (Online)

10. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management 24(5), 513–523 (1988)
11. Carse, B., Pipe, A.G.: A framework for evolving fuzzy classifier systems using genetic programming. In: Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference, pp. 465–469. AAAI Press, Menlo Park (2001)