# Evolving Classifiers – Evolutionary Algorithms in Data Mining

Thomas Weise, Stefan Achler, Martin Göb, Christian Voigtmann, and Michael Zapf

University of Kassel, Wilhelmshöher Allee 73, D-34121 Kassel, Germany
{weise|zapf}@uni-kassel.de, {achii|martingoeb|voigtmannc}@web.de
http://www.vs.uni-kassel.de

**Abstract.** Data mining means to summarize information from large amounts of raw data. It is one of the key technologies in many areas of economy, science, administration and the internet. In this report we introduce an approach for utilizing evolutionary algorithms to breed fuzzy classifier systems. This approach was exercised as part of a structured procedure by the students Achler, Göb, and Voigtmann as contribution to the 2006 Data-Mining-Cup contest, yielding encouragingly positive results.

## 1 Introduction

Data mining is a means of extracting information from huge, unprocessed data. Today, data mining methods are applied in almost all areas of economy, science, the internet, and administration. Yet, it is still an active field of research where gradually new methods, like the support vector machine [1] approach that gained importance in the last ten years, are developed. Existing methods are continuously improved, like learning classifier systems, where in the same time span new systems have branched off [2,3].

In this report, we will discuss a simple general approach to data mining. This approach has been exercised on Data-Mining-Cup contest. In order to cope with the especially high complexity of the 2007 tasks in this biggest international student data mining contest, we developed a small and efficient fuzzy type of non-learning classifier systems. Similar to the Pittsburgh Learning Classifier Systems approach [4,5,6], these systems can be created as a whole with a genetic algorithm.

In the following text, we give a short summary on data mining and the Data-Mining-Cup, to which the work discussed in this report was contributed. This contest contribution is mainly based on a combination of evolutionary algorithms, genetic algorithms, and learning classifier systems which are introduced as related work in Section 3. Section 4 subsequently defines some advisable steps, which define a structured approach applicable to all kinds of data mining problems. This methodology is then carried out in Section 5 where a fuzzy form of classifier systems is developed especially suitable for the DMC 2007 contest. Finally, Section 6 sums up the experiences and conclusions that can be drawn from our work and ideas for future improvements. These will on one hand be applied in next year's contest, but are also generally practical values on the other hand.

## 2 Data Mining

Data mining can be defined as the nontrivial extraction of implicit, previously unknown, and potentially useful information from data [7] and the science of extracting useful information from large data sets or databases [8].

Today, gigantic amounts of data are collected in the web, in medical databases, by enterprise resource planning (ERP) and customer relationship management (CRM) systems in corporations, web shops, by administrative and governmental bodies, and in science projects. These data sets are way to large to be incorporated directly into a decision process or to be understood as-is by a human being. Instead, automated approaches have to be applied that extract the relevant information, to find underlying rules and patterns, or to detect time-dependent changes. Data mining subsumes methods and techniques capable to perform this task. It is very closely related to estimation theory [9,10] in stochastic [11,12] – the simplest digest of data sets is still the arithmetic mean. Data mining is also strongly related to artificial intelligence [13,14], which includes learning algorithms that can generalize the given information. Some of the most wide spread and most common data mining techniques are:

- (artificial) neural networks (ANN) [15,16],
- support vector machines (SVM) [17,18,19,20],
- logistic regression [21],
- decision trees [22,23],
- learning classifier systems as introduced in Section 3.3 on page 5, and
- naïve Bayes Classifiers [24,25].

The Data-Mining-Cup has been established in the year 2000 by the prudsys AG and the Technical University of Chemnitz. It aims to provide an independent platform for data mining users and data analysis tool vendors and builds a bridge between academic science and economy. Today, it is one of Europe's biggest and most influential conferences in the area of data mining.

The DMC Contest is the biggest international student data mining competition. In the spring of each year, students of national and international universities challenge to find the best solution of a data analysis problem. It provides an excellent platform to test new approaches or to apply established methods in a challenging, inspiring environment.

## 3 Related Work

### 3.1 Evolutionary Algorithms

Evolutionary algorithms (EA) [26,27,28,29] are generic, population-based meta-heuristic optimization algorithms that use biology-inspired mechanisms like mutation, crossover, natural selection and survival of the fittest.

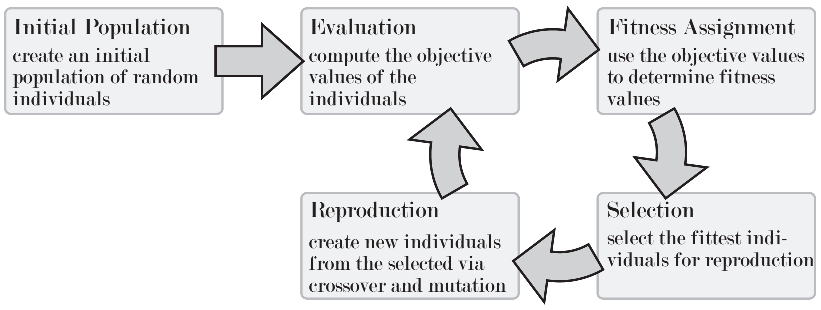

Initial Population
create an initial population of random individuals
Evaluation
compute the objective values of the individuals
Fitness Assignment
use the objective values to determine fitness values
Selection
select the fittest individuals for reproduction
Reproduction
create new individuals from the selected via crossover and mutation

Fig. 1: The basic cycle of evolutionary algorithms.

The family of evolutionary algorithms encompasses genetic algorithms (see Section 3.2), genetic programming, evolution strategy, evolutionary programming, and learning classifier systems (see Section 3.3).

The advantage of evolutionary algorithms compared to other optimization methods is that they make only few assumptions about the underlying fitness landscape and therefore perform consistently well in many different problem categories.

All evolutionary algorithms proceed in principle according to the scheme illustrated in Figure 1:

1. Initially, a population of individuals with a totally random genome is created.
2. All individuals of the population are tested. This evaluation may incorporate complicated simulation and calculations.
3. With the tests, we have determined the utility of the different features of the solution candidates and can now assign a fitness value to each of them.
4. A subsequent selection process filters out the individuals with low fitness and allows those with good fitness to enter the mating pool with a higher probability.
5. In the reproduction phase, offspring is created by varying or combining these solution candidates and integrated it into the population.

6. If the termination criterion is met, the evolution stops here. Otherwise, it continues at step 2.

In the context of this report, especially genetic algorithms and learning classifier systems are of interest.

## 3.2 Genetic Algorithms

Genetic algorithms are a subclass of evolutionary algorithms that employs two different representations for each solution candidate. The genotype is used in the reproduction operations whereas the phenotype is the form of the individual which can be used for the determining its fitness [30,31,32,33,34,29].

### Mutation

Mutation is an important method of preserving individual diversity. In fixed-length string chromosomes it can be achieved by modifying the value of one element of the genotype, as illustrated in Figure 2. More generally, a mutation may change $0 \leq n < |g|$ locations in the string. In binary coded chromosomes for example, the elements are bits which are simply toggled.

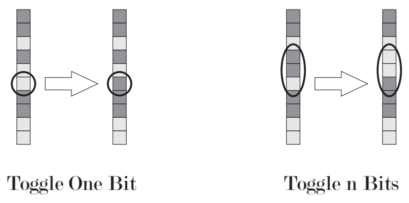


Fig. 2: Bit-toggling mutation of string chromosomes.

### Crossover

Figure 3 outlines the recombination of two string chromosomes. This operation is called crossover and is done by swapping parts of the genotypes between the parents.

When performing crossover, both parental chromosomes are split at randomly determined crossover points. Subsequently, a new child chromosome is created by appending the first part of the first parent with the second part of the second parent. This method is called one-point crossover. In two-point crossover, both parental genotypes are split at two points, constructing a new offspring by using parts number one and three from the first, and the middle part from the second ancestor. The generalized form of this technique is $n$-point crossover.

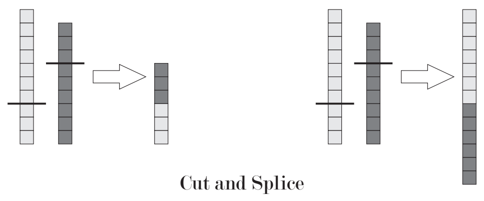


Fig. 3: Crossover (recombination) of variable-length string chromosomes.

### 3.3 Learning Classifier Systems

In the late 1970s, Holland invented and patented a new class of cognitive systems, called classifier systems (CS) [35,36,29]. These systems are a special case of production systems [37,38] and consist of four major parts:

1. a set of interacting productions, called *classifiers*,
2. a performance algorithm that directs the action of the system in the environment,
3. a simple learning algorithm that keeps track on each classifier's success in bringing about rewards, and
4. a more complex algorithm that modifies the set of classifiers so that variants of good classifiers persist and new, potentially better ones are created in a provably efficient manner [39].

Figure 4 illustrates Holland's original idea of the structure of a Michigan-style learning classifier system. A classifier system is connected via detectors *(b)* and effectors *(c)* to its environment *(a)*. The input in the system, coming from the detectors, is encoded in binary messages that are written into a message list *(d)*. On this list, the classifiers, simple `if-then` rules *(e)*, are applied. The result of a classification is again encoded as a message and written to the message list. These new messages may now trigger other rules or are signals for the effectors [40]. The payoff of the performed actions is distributed by the credit apportionment system *(f)* to the rules. Additionally, a rule discovery system *(g)* (normally a genetic algorithm) is responsible for finding new rules and adding them to the classifier population [41].

## 4 A Structured Approach to Data Mining

Whenever any sort of problem should be solved, a structured approach is always advisable. This goes for the application of optimization methods like evolutionary algorithms as well as for deriving classifiers in a data mining problem. In this section we discuss a few simple steps which should be valid for both kinds of tasks and which have been followed in our approach to the 2007 DMC.

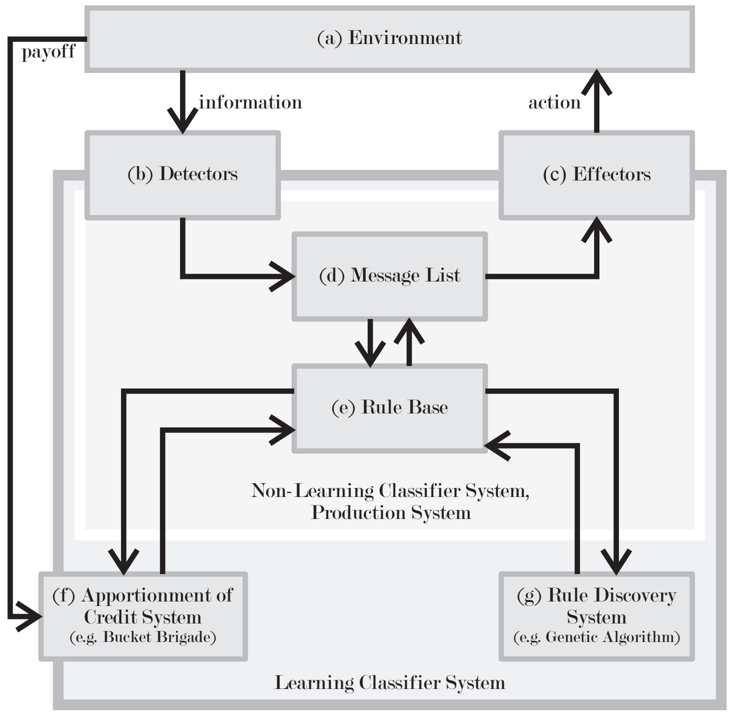


Fig. 4: The structure of a Michigan style learning classifier system.

The first step is always to clearly specify the problem that should be solved. Parts of this specification are possible target values and optimization criteria as well as the semantics of the problem domain. The optimization criteria tell us how different possible solutions can be compared with each other. If we were to sell tomatoes, for example, the target value (subject to maximization) would be the profit. Then again, the semantics of the problem domain allow us to draw conclusions on what features are important in the optimization or data mining process. Again, when selling tomatoes, the average weight of the vegetables, their color, and maybe the time of the day when we open the store are important. The names of our customers on the other hand are probably not. The tasks of the DMC 2007 Contest, outlined in Section 5.1 on the next page, are a good example for such a problem definition.

Before choosing or applying any data mining or optimization technique, an initial analysis of the given data should be performed. With this review and the problem specification, we can filter the data and maybe remove unnecessary features. Additionally, we will gain insight in the data structure and hopefully can already eliminate some possible solution approaches. It is of course better to exclude some techniques that cannot lead to good results in the initial phase instead of wasting working hours in trying them out to avail. We have now to decide on one or two solution approaches that are especially promising for the problem defined. We have performed this step for the DMC 2007 Contest data in Section 5.2 on page 9.

The next step is to apply these approaches. Of course, running an optimizer on all known sample data at once is not wise. Although we will obtain a result with which we can solve the specified problem for all the known data samples, it is possible not a good solution. Instead, it may be overfitted or overspecialized and can *only* process the data we are given. Normally however, we are only provided with fraction of the "real data" and want to find a system that is able to perform well also on samples that are not yet known to us. Hence, we need to find out whether or not our approach generalizes. Therefore, it is sufficient to derive a solution for a subset of the available data samples, the training data. This solution is then tested on the test set, the remaining samples not used in its creations. The system we have created generalizes well if it is rated approximately equally good by the optimization criterion for both, the training and the test data. Now we can repeat the process by using all available data. We have evolved classifier systems that solve the DMC 2007 Contest according to this method in Section 5.3 on page 11.

The students Achler, Göb, and Voigtmann have participated in the 2007 DMC Contest and proceeded according to this pattern. In order to solve the challenge, they chose for a genetic algorithm evolving a fuzzy classifier system. The results of their participation are discussed in Section 5.4 on page 13.

The following sections are based on their experiences and impressions, and reproduce how they proceeded.

## 5 Applying the Structured Approach

### 5.1 The Problem Definition

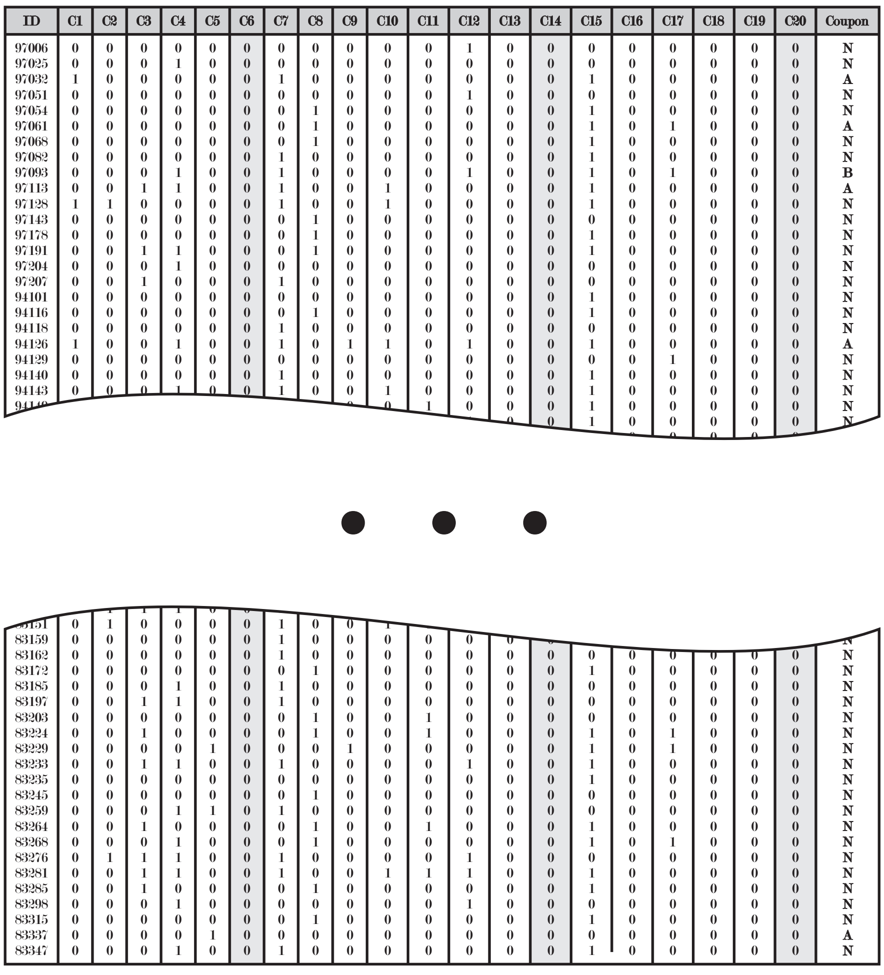
Rebate systems are an important means to animate customers to return to a store in classical retail. In the 2007 contest, we consider a check-out couponing system. Whenever a customer leaves a store, at the end of her bill a coupon can be attached. She then can use the coupon to receive some rebate on her next purchase. When printing the bill at the checkout, there are three options for couponing:

**Case N:** attach no coupon to the bill,
**Case A:** attach coupon type **A**, a general rebate coupon, to the bill, or
**Case B:** attach coupon type **B**, a special voucher, to the bill.

The profit of the couponing system is defined as follows:

- Each coupon which is not redeemed costs 1 money unit.
- For each redeemed coupon of type **A**, the retailer gains 3 money units.
- For each coupon of type **B** which is redeemed, the retailer gains 6 money units.

It is thus clear that simply printing both coupons at the end of each bill makes no sense. In order to find a good strategy for coupon printing, the retailer has initiated a survey. She wants to find out which type of customer has an affinity to cash in coupons and, if so, which type of coupon most likely. Therefore the behavior of 50000 customers has been anonymously recorded. For all these customers, we know the customer id, the number of redemptions of 20 different coupons and the historic information whether coupon type **A**, coupon type **B**, or none of them has been redeemed. Cases where both have been cashed in are omitted.



| ID | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | Coupon |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 97006 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N |
| 97025 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N |
| 97032 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | A |
| 97051 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N |
| 97054 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 97061 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | A |
| 97068 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 97082 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 97093 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | B |
| 97113 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | A |
| 97128 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 97143 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N |
| 97178 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 97191 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 97204 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N |
| 97207 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N |
| 94101 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 94116 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 94118 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N |
| 94126 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | A |
| 94129 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | N |
| 94140 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 94143 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| [illegible] |  |  |  |  |  |  |  |  |  | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |

● ● ●

| ID | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | Coupon |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 83151 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |  |  |  |  |  |  |  |  |  |  |  |
| 83159 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  | N |
| 83162 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N |
| 83172 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 83185 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N |
| 83197 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N |
| 83203 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N |
| 83224 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | N |
| 83229 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | N |
| 83233 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 83235 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 83245 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N |
| 83259 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N |
| 83264 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 83268 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | N |
| 83276 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N |
| 83281 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 83285 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 83298 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N |
| 83315 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |
| 83337 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | A |
| 83347 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | N |

Fig. 5: A few samples from the DMC 2007 training data.

Figure 5 shows some samples from this data set. The task is to use it as training data in order to derive a classifier $C$ that is able to decide from a record of the 20 features whether a coupon **A**, **B**, or none should be provided to a customer. This means to maximize the profit $P(C)$ of retailer gained by using the classifier $C$ which can be computed according to

$$P(C) = 3 * AA + 6 * BB - 1 * (NA + NB + BA + AB) \tag{1}$$

where

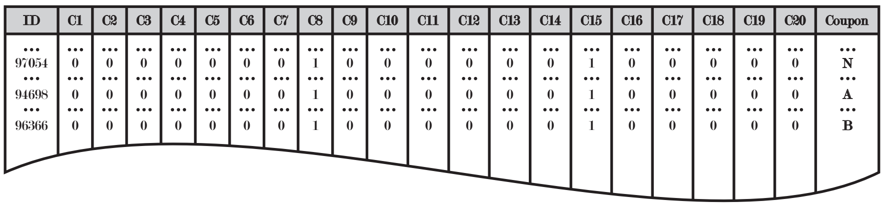
- $AA$ is the number of correct assignments for coupon **A**.
- $BB$ is the number of correct assignments for coupon **B**.
- $NA$ is the number of wrong assignments to class **A** from the real class **N**.
- $NB$ is the number of wrong assignments to class **B** from the real class **N**.
- $BA$ is the number of wrong assignments to class **A** from the real class **B**.
- $AB$ is the number of wrong assignments to class **B** from the real class **A**.

Wrong assignments from the classes **A** and **B** to **N** play no role.

The classifier built with the 50000 training data sets is then to be applied to another 50000 data samples. There however, the column *Coupon* is missing and should be the result of the classification process. Based on the computed assignments, the profit score $P$ is calculated for each contestant by the jury and the team with the highest profit will win.

## 5.2 Initial Data Analysis

The test dataset have some properties which make it especially hard for learning algorithms to find good solutions. Figure 6 for example shows three data samples with exactly the same features but different classes. In general, there is some degree of fuzzyness and noise, and clusters belonging to different classes overlap and contain each other. Since the classes cannot be separated by



| ID | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | Coupon |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 97054 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | **N** |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 94698 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | **A** |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 96366 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | **B** |

Fig. 6: DMC 2007 sample data – same features but different classes.

hyper-planes in a straightforward manner, the application of neural networks and support vector machines becomes difficult. Furthermore, the values of the

Table 1: Feature-values in the 2007 DMC training sets.

| value | number of ocurences |
|---|---|
| 0 | 837′119 |
| 1 | 161′936 |
| 2 | 924 |
| 3 | 21 |

features take on only four different values and are zero to 83.7%, as illustrated in Table 1. In general, such a small number of possible feature values makes it hard to apply methods that are based on distances or averages.

At least one positive fact can easily be found by eyesight when inspecting the training data: the columns *C6*, *C14*, and *C20*, marked gray in Figure 5, are most probably insignificant since they are almost always zero and hence, can be excluded from further analysis. The same goes for the first column, the customer *ID*, by common sense.

## The Solution Approach: Classifier Systems

From the initial data analysis, we can reduce the space of values a feature may take on to `0`, `1`, and `>1`. This limited, discrete range is especially suited for learning classifier systems (LCS) discussed in Section 3.3 on page 5.

Since we already know the target function, $P(C)$, we do not need the learning part of the LCS. Instead, our idea was to use the profit $P(C)$ defined in equation 1 directly as objective function for a genetic algorithm.

Very much like in the Pitt-approach [4,5,6] in LCS, the genetic algorithm would base on a population of classifier systems. Such a classifier system is a list of rules (the single classifiers). A rule contains a classification part and one condition for each feature in the input data. We used a two bit alphabet for the conditions, allowing us to encode the four different conditions per feature listed in Table 2. The three different classes can be represented using

Table 2: Feature conditions in the rules.

| condition (in genotype) | condition (in phenotype) | corresponding feature value |
|---|---|---|
| `00` | 0 | must be 0 |
| `01` | 1 | must be $\geq 1$ |
| `10` | 2 | must be $> 1$ |
| `11` | 3 | don't care (i.e. any value is ok) |

two additional bits, where `00` and `11` stands for **A**, `01` means **B**, and `10` corresponds to **N**. We leave three insignificant features away, so a rule is in total $17 * 2 + 2 = 36$ bits small. This means that we need less memory for a classifier system with 17 rules than for 10 double precision floating point numbers, as used by a neural network, for example.

When a feature is to be classified, the rules of a classifier system are applied step by step. A rule fits to a given data sample if none of its conditions is violated by a corresponding sample feature. As soon as such a rule is found, the input is assigned to the class identified by the classification part of the rule. This stepwise interpretation creates a default hierarchy that allows classifications to include each other: a more specific rule (which is checked before the more general one) can represent a subset of features which is subsumed by a rule which is evaluated later. If no rule in the classifier systems fits to a data sample, **N** is returned per default since misclassifying an **A** or **B** as an **N** at least doesn't introduce a penalty in $P(C)$ according to equation 1.

Since the input data is noisy, it turned out to be a good idea to introduce some fuzzyness in our classifiers too by modifying this default rule. During the classification process, we remember the rule which was violated by the least features. In the case that no rule fits perfectly, we check if the number of these misfits is less than one fifth of the features, in this case $\frac{17}{5} \approx 3$. If so, we consider it as a match and classify the input according to the rules classification part. Otherwise, the original default rule is applied and **N** is returned. Figure 7 outlines the relation of the genotype and phenotype of such a fuzzy classifier system. It shows a classifier system consisting of four rules that has been a real result of the genetic algorithm. In this graphic we also apply it to the second sample of the dataset that is to be classified. As one can easily see, none of the four rules matches fully – which is strangely almost always the case for classifier systems that sprung of the artificial evolution. The data sample however violates only three conditions of the second rule and hence, stays exactly at the $\frac{1}{5}$-threshold. Since no other rule in the classifier system has less misfit conditions, the result of this classification process will be **A**.

### 5.3 Analysis of the Evolutionary Process

As already discussed in the previous section, we want to evolve the classifier systems directly. Therefore we apply two objective functions:

$$f_1(C) = -P(C) \tag{2}$$
$$f_2(C) = \max\{|C|, 3\} \tag{3}$$

Here $f_1$ represents the (negated) profit gained with a classifier system $C$ and $f_2(C)$ is the number of rules in $C$ (cut-off at a size of at 3). Both functions are subject to minimization. Figure 8 illustrates the course of the classifier system evolution. Here we have applied a simple elitist[1] genetic algorithm

[1] Elitist genetic algorithms always preserve the best solutions they have found while non-elitist algorithms may loose them [29].

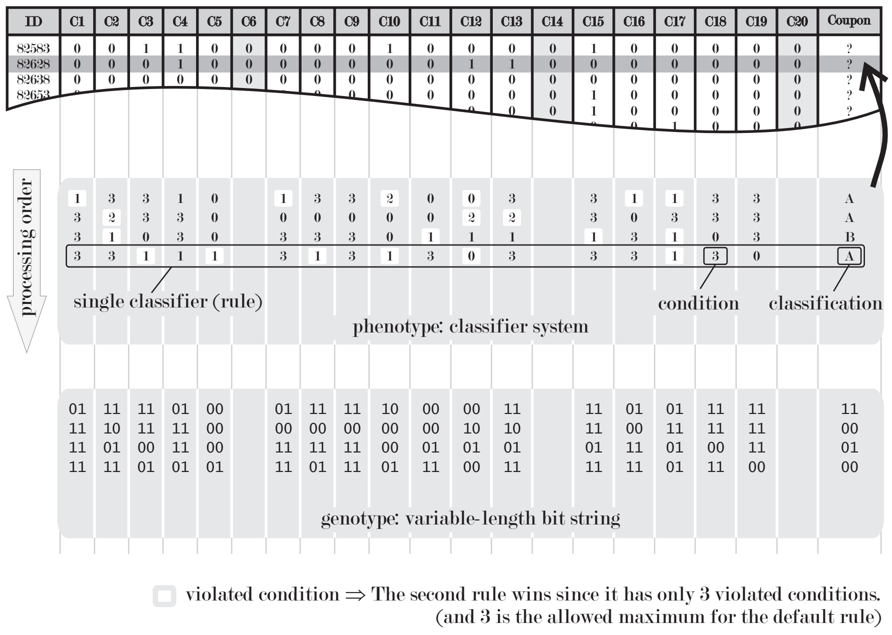

ID
Coupon
processing order
single classifier (rule)
condition
classification
phenotype: classifier system
genotype: variable-length bit string
violated condition ⇒ The second rule wins since it has only 3 violated conditions.
(and 3 is the allowed maximum for the default rule)

Fig. 7: An example classifier for the 2007 DMC.

with a population size of 10240 individuals. We can see a logarithmic growth of the profit with the generations as well as with the number of rules in the classifier systems. A profit of 8800 for the 50000 data samples has been reached. Experiments with 10000 datasets held back and an evolution on the remaining 40000 samples indicated that the evolved rule sets generalize sufficiently well. The cause for the generalization of the results is the second, non-functional objective function which puts pressure into the direction of smaller classifier systems and the modified default rule which allows noisy input data. The result of the multi-objective optimization process is the Pareto-optimal set. It comprises all solution candidates for which no other individual exists that is better in at least one objective value and not worse in any others. Figure 9 displays some classifier systems which are members of this set after generation 1000. $C1$ is the smallest non-dominated classifier system. It consists of three rules which lead to a profit of 7222. $C2$, with one additional rule, reaches 7403. The 31-rule classifier system $C3$ provides a gain of 8748 to which the system with the highest profit evolved, $C4$, adds only 45 to a total of 8793 with a trade-off of 18 additional rules (49 in total).

As shown in Table 1 on page 10, most feature values are `0` or `1`, there are only very few `2` and `3`-valued features. In order to find out how different treatment of those will influence the performance of the classifiers and of the evolutionary process, we slightly modified the condition semantics in Table 3

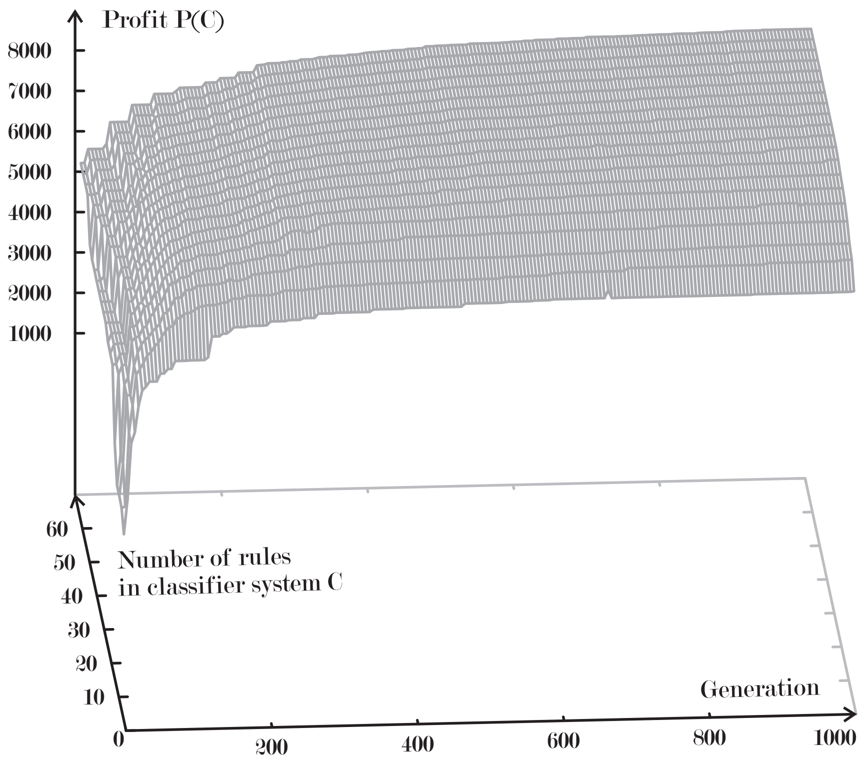


Fig. 8: The course of the classifier system evolution.

by changing the meaning of rule 2 from $> 1$ to $\leq 1$ (compare with Table 2 on page 10).

The progress of the evolution depicted in Figure 10 exhibits no significant difference to the first one illustrated in Figure 8. With the modified rule semantics, the best classifier system evolved delivered a profit of 8666 by utilizing 37 rules. This result is also not very much different from the original version. Hence, the treatment of the features with the values 2 and 3 does not seem to have much influence on the overall result. In the first approach, rule-condition 2 used them as distinctive criterion. The new method treats them the same as feature value 1, with slightly worse results.

### 5.4 Contest Results and Placement

A record number of 688 teams from 159 universities in 40 countries registered for the 2007 DMC Contest, from which only 248 were finally able to hand in results. The team of the RWTH Aachen won place one and two by scoring 7890 and 7832 points on the contest data set. Together with the team from

Table 3: Different feature conditions in the rules.

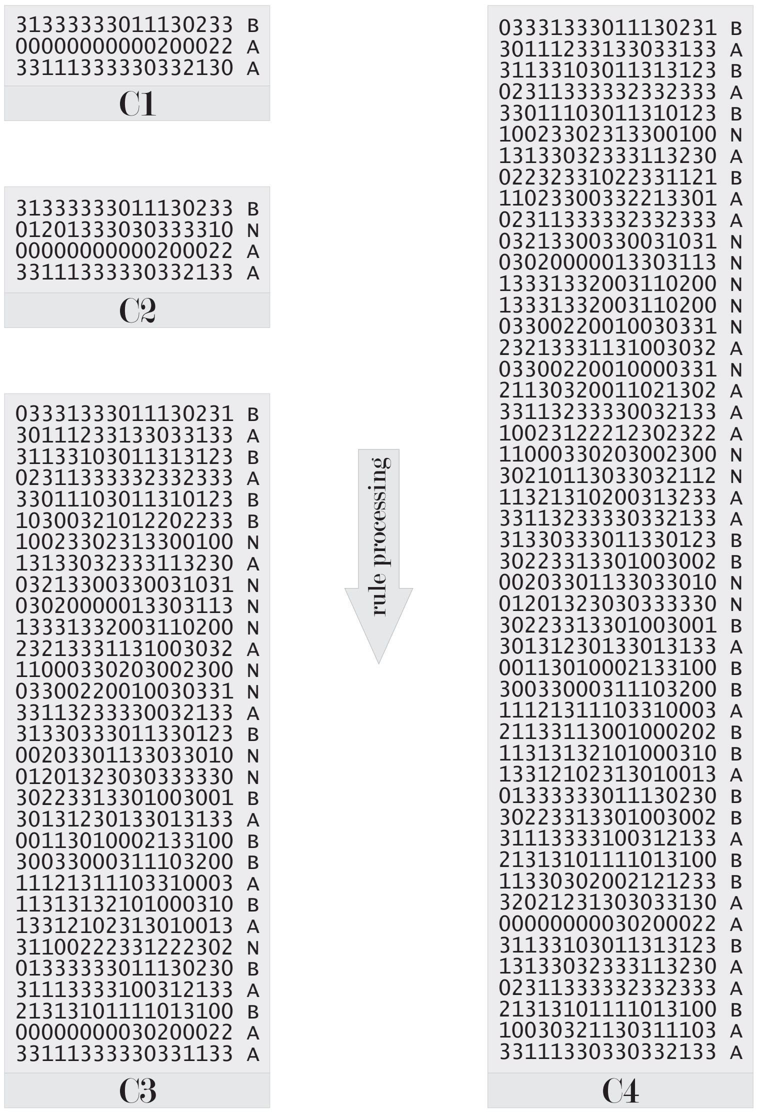
| condition (in genotype) | condition (in phenotype) | corresponding feature value |
|---|---|---|
| `00` | 0 | must be 0 |
| `01` | 1 | must be $\geq 1$ |
| `10` | 2 | must be $\leq 1$ |
| `11` | 3 | don't care (i.e. any value is ok) |

the Darmstadt University of Technology, ranked third, they occupy the first eight placements.

Our team reached place 29 which is quite a good result considering that none of its members had any prior experience in data mining.

Retrospectively one can recognize that the winning gains are much lower than those we have discussed in the previous experiments. They are, however, results of the classification of a different data set – the profits in our experiment are obtained from the training sets and not from the contest data. Although our classifiers did generalize well in the initial tests, they seem to suffer from some degree of overfitting. Furthermore, the systems discussed here are the result of reproduced experiments and not the original contribution from the students. The system with the highest profit that the students handed in also had gains around 8600 on the training sets. With a hill climbing optimizer, we squeezed out another 200, increasing, of course, the risk of additional overfitting. In the challenge, the best scored score of our team, a total profit of 7453 (only 5.5% less than the winning team). This classifier system was however grown with a much smaller population (4096) than in the experiments here, due to time restrictions.

Remarkably we did not achieve the best result with the best single classifier system evolved, but with a primitive combination of this system with another one: If both classifier systems delivered the same result for a record, this result was used. Otherwise, `N` was returned, which at least would not lead to additional costs (as follows from equation 1 on page 9).



```
31333333011130233 B
00000000000200022 A
33111333330332130 A
```

C1

```
31333333011130233 B
01201333030333310 N
00000000000200022 A
33111333330332133 A
```

C2

```
03331333011130231 B
30111233133033133 A
31133103011313123 B
02311333332332333 A
33011103011310123 B
10300321012202233 B
10023302313300100 N
13133032333113230 A
03213300330031031 N
03020000013303113 N
13331332003110200 N
23213331131003032 A
11000330203002300 N
03300220010030331 N
33113233330032133 A
31330333011330123 B
00203301133033010 N
01201323030333330 N
30223313301003001 B
30131230133013133 A
00113010002133100 B
30033000311103200 B
11121311103310003 A
11313132101000310 B
13312102313010013 A
31100222331222302 N
01333333011130230 B
31113333100312133 A
21313101111013100 B
00000000030200022 A
33111333330331133 A
```

C3

```
03331333011130231 B
30111233133033133 A
31133103011313123 B
02311333332332333 A
33011103011310123 B
10023302313300100 N
13133032333113230 A
02232331022331121 B
11023300332213301 A
02311333332332333 A
03213300330031031 N
03020000013303113 N
13331332003110200 N
13331332003110200 N
03300220010030331 N
23213331131003032 A
03300220010000331 N
21130320011021302 A
33113233330032133 A
10023122212302322 A
11000330203002300 N
30210113033032112 N
11321310200313233 A
33113233330332133 A
31330333011330123 B
30223313301003002 B
00203301133033010 N
01201323030333330 N
30223313301003001 B
30131230133013133 A
00113010002133100 B
30033000311103200 B
11121311103310003 A
21133113001000202 B
11313132101000310 B
13312102313010013 A
01333333011130230 B
30223313301003002 B
31113333100312133 A
21313101111013100 B
11330302002121233 B
32021231303033130 A
00000000030200022 A
31133103011313123 B
13133032333113230 A
02311333332332333 A
21313101111013100 B
10030321130311103 A
33111330330332133 A
```

C4

Fig. 9: Some elements of the Pareto-optimal set of evolved classifier systems.

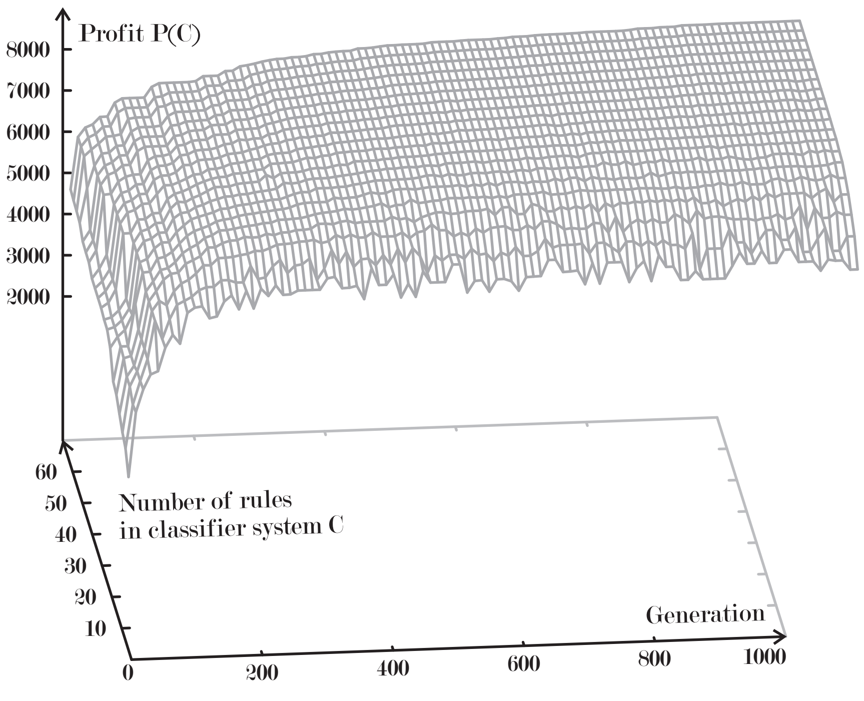


Fig. 10: The course of the modified classifier system evolution.

## 6 Conclusion and Future Work

In order to solve the 2007 Data-Mining-Cup contest we exercised a structured approach. After reviewing the data samples provided for the challenge, we have adapted the idea of classifier systems to the special needs of the competition. As a straightforward way of obtaining such systems, we have chosen a genetic algorithm with two objective functions. The first one maximized the utility of the classifiers by maximizing the profit function provided by the contest rules. The section objective function minimized a non-functional criterion, the number of rules in the classifiers. It was intended to restrict the amount of overfitting and overspecialization. The bred classifier systems showed reasonable good generalization properties on the test data sets separated from the original data samples, but seem to be overfitted when comparing these results with the profits gained in the contest. A conclusion is that it is hard to prevent overfitting in an evolution based on limited sample data – the best classifier system obtained will possibly be overfitted. In the challenge, the combination of two classifiers yielded the best results. Such combinations of multiple, independent systems will probably perform better than each of them alone.

In further projects, especially the last two conclusions drawn should be considered. Although we used a very simple way to combine our classifier systems for the contest, it still provided an advantage.

A classifier system in principle is nothing more but an estimator. There exist many sophisticated methods of combining different estimators in order to achieve better results [42]. The original version of such “boosting algorithms”, developed by Schapire [43], theoretically allows to achieve an arbitrarily low error rate, requiring basic estimators with a performance only slightly better than random guessing on any input distribution. The AdaBoost algorithm [44] additionally takes into consideration the error rates of the estimators. With this approach, even classifiers of different architectures like a neural network and a learning classifier system can be combined. Since the classification task in the challenge required non-fuzzy answers in form of definite set memberships, the usage of weighted majority voting [45,46], as already applied in a very primitive manner, would probably have been the best approach.

## References

1. Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* Information Science and Statistics. Springer-Verlag, second edition, Nov 1999 (first edition: 1995).
2. Stewart W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.
3. Tim Kovacs. *XCS Classifier System Reliably Evolves Accurate, Complete, and Minimal Representations for Boolean Functions*, pages 59–68. Springer-Verlag, Aug 1997.

4. William M. Spears and Kenneth A. De Jong. Using genetic algorithms for supervised concept learning. In *Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence*, number IEEE Cat. No. 90CH2915-7, pages 335–341, Herndon, VA, 6-9 1990. IEEE Computer Society Press, Los Alamitos, CA.
5. Kenneth A. De Jong and William M. Spears. Learning Concept Classification Rules using Genetic Algorithms. In *Proceedings of the Twelth International Conference on Artificial Intelligence IJCAI-91*, volume 2, 1991.
6. Stephen Frederick Smith. *A learning system based on genetic adaptive algorithms*. PhD thesis, University of Pittsburgh, 1980.
7. William J. Frawley, Gregory Piatetsky-Shapiro, and Christopher J. Matheus. Knowledge discovery in databases: An overview. *AI Magazine*, pages 213–228, Fall 1992. Online available at `http://citeseer.ist.psu.edu/524488.html` (version 2007-08-11) and `http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1992.pdf` (version 2007-08-11).
8. David J. Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. MIT Press, Cambridge, MA, Aug 2001.
9. John A. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, third edition, Apr 2006.
10. Steven M. Kay. *Fundamentals of Statistical Signal Processing, Volume I: Estimation Theory*, volume 1. Prentice Hall PTR, us es edition, Mar 1993.
11. Alfréd Rényi. *Probability Theory*. Dover Publications, May 2007.
12. Gregory F. Lawler. *Introduction to Stochastic Processes*. Chapman & Hall/CRC, second edition, May 2006.
13. Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, Dec 2002.
14. Werner Dilger. *Einführung in die Künstliche Intelligenz*. Apr 2006. Lecture notes for the lectures artificial intelligence. Online available at `http://www.tu-chemnitz.de/informatik/KI/skripte.php` (version 2007-08-06).
15. Joseph P. Bigus. *Data Mining With Neural Networks: Solving Business Problems from Application Development to Decision Support*. Mcgraw-Hill (Tx), May 1996.
16. Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, USA, Jan 1996.
17. Sankar K. Pal and Pabitra Mitra. *Pattern Recognition Algorithms for Data Mining*. Chapman & Hall/CRC, May 2004.
18. Lipo Wang, editor. *Support Vector Machines: Theory and Applications*. Studies in Fuzziness and Soft Computing. Springer, first edition, Aug 2005.
19. Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998. Online available at `http://citeseer.ist.psu.edu/burges98tutorial.html` (version 2007-08-08) and `http://research.microsoft.com/~cburges/papers/SVMTutorial.pdf` (version 2007-08-08).
20. Christiaan M. van der Walt and Etienne Barnard. Data characteristics that determine classifier performance. In *Proceedings of the Sixteenth Annual Symposium of the Pattern Recognition Association of South Africa*, pages 160–165, 2006. Online available at `http://www.meraka.org.za/pubs/CvdWalt.pdf` (version 2007-08-08).

21. Alan Agresti. *An Introduction to Categorical Data Analysis*. Wiley-Interscience, first edition, Jan 1996.
22. Michael J. A. Berry and Gordon S. Linoff. *Mastering Data Mining: The Art and Science of Customer Relationship Management*. Wiley, first edition, Dec 1999.
23. Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, first edition, Oct 1999.
24. Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, Nov 1997. Online available at `http://citeseer.ist.psu.edu/domingos97optimality.html` (version 2007-08-11) and `http://www.ics.uci.edu/~pazzani/Publications/mlj97-pedro.pdf` (version 2007-08-11).
25. Irina Rish. An empirical study of the naive bayes classifier. In *Proceedings of IJCAI-01 workshop on Empirical Methods in AI, International Joint Conference on Artificial Intelligence*, pages 41–46. American Association for Artificial Intelligence, 2001. Online available at `http://www.cc.gatech.edu/~isbell/classes/reading/papers/Rish.pdf` (version 2007-08-11).
26. Thomas Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Jan 1996.
27. Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz. *Handbook of Evolutionary Computation*. Computational Intelligence Library. Oxford University Press in cooperation with the Institute of Physics, ringbound edition, Apr 1997.
28. Thomas Bäck, U. Hammel, and Hans-Paul Schwefel. Evolutionary computation: comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1:3–17, Apr 1997.
29. Thomas Weise. *Global Optimization Algorithms – Theory and Application*. July 2007 edition, Jul 2007. Online available at `http://www.it-weise.de/` (version August 17, 2007).
30. John H. Holland. Outline for a logical theory of adaptive systems. *J. ACM*, 9(3):297–314, 1962. Online available at `http://portal.acm.org/citation.cfm?id=321128` (version 2007-07-28).
31. John H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, 1975. Reprinted by MIT Press, April 1992, ISBN-10: 0262581116, ISBN-13: 978-026258111.
32. Jack L. Crosby. *Computer Simulation in Genetics*. John Wiley and Sons Ltd, Jan 1973.
33. David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, first edition, Jan 1989.
34. Jörg Heitkötter and David Beasley, editors. *Hitch-Hiker's Guide to Evolutionary Computation: A List of Frequently Asked Questions (FAQ)*. ENCORE (The EvolutioNary Computation REpository Network), 1998. USENET: comp.ai.genetic. Online available at `http://www.cse.dmu.ac.uk/~rij/gafaq/top.htm` (version 2007-07-03) and `http://alife.santafe.edu/~joke/encore/www/` (version 2007-07-03).

35. John H. Holland and Judith S. Reitman. Cognitive systems based on adaptive algorithms. In D. A. Waterman and F. Hayes-Roth, editors, *Pattern directed inference systems*, pages 313–329. Academic Press, New York, NY, 1978. Reprinted in: Evolutionary Computation. The Fossil Record. David B. Fogel (Ed.) IEEE Press, 1998. ISBN: 0-7803-3481-7.
36. John H. Holland and Arthur W. Burks. *Adaptive computing system capable of learning and discovery*. Number 619349 filed on 1984-06-11. US Patent Issued on September 29, 1987, Current US Class 706/13, Genetic algorithm and genetic programming system 382/155, LEARNING SYSTEMS 706/62 MISCELLANEOUS, Foreign Patent References 8501601 WO Apr., 1985.
37. R. Davis and J. King. An overview of production systems. Technical Report STAN-CS-75-524, Oct 1975.
38. R. Davis and J. King. An overview of production systems. *Machine Intelligence*, 8, 1977.
39. John H. Holland and Judith S. Reitman. Cognitive systems based on adaptive algorithms. *SIGART Bull.*, (63):49–49, 1977.
40. Bart de Boer. Classifier systems: a useful approach to machine learning? Master's thesis, Leiden University, Rijksuniversiteit Leiden, Netherlands, Aug 1994. Internal Report Number IR94-02. Supervisors Ida Sprinkhuizen-Kuyper and Egbert Boers. Online available at `citeseer.ist.psu.edu/deboer94classifier.html` (version 2007-08-08).
41. Andreas Geyer-Schulz. Holland classifier systems. In *APL '95: Proceedings of the international conference on Applied programming languages*, pages 43–55, New York, NY, USA, 1995. ACM Press.
42. Ran Avnimelech and Nathan Intrator. Boosting regression estimators. *Neural Computation*, 11(2):499–520, 1999.
43. Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.
44. Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.
45. Yoav Freund. Boosting a weak learning algorithm by majority. In *COLT: Proceedings of the Workshop on Computational Learning Theory*. Morgan Kaufmann Publishers, 1990.
46. Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proceedings 14th International Conference on Machine Learning*, pages 322–330. Morgan Kaufmann, 1997.