Evolutionary Intelligence manuscript No.
(will be inserted by the editor)

Novel Evolutionary Algorithms for Supervised Classification Problems: An Experimental Study

Pu Wang · Thomas Weise · Raymond Chiong

Received: date / Accepted: date

Abstract Evolutionary Algorithms (EAs) are population-based, stochastic search algorithms that mimic natural evolution. Over the years, EAs have been successfully applied to many classification problems. In this paper, we present three novel evolutionary approaches and analyze their performances for synthesizing classifiers with EAs in supervised data mining scenarios. The first approach is based on encoding rule sets with bit string genomes, while the second one utilizes Genetic Programming (GP) to create decision trees with arbitrary expressions attached to the nodes. The novelty of these two approaches lies in the use of solutions on the Pareto front as an ensemble. The third approach, EDDIE-101, is also based on GP but uses a new, advanced fitness measure and some novel genetic operators. We compare these approaches to a number of well-known data mining methods, including C4.5 and Random-Forest, and show that the performances of our evolved classifiers can be very competitive as far as the solution quality is concerned. In addition, the proposed approaches work well across a wide range of configurations, and EDDIE-101 particularly has been highly efficient. To further evaluate the flexibility of EDDIE-101 across different problem domains, we also test it on some real financial datasets for finding investment opportunities and compare the results with those obtained using other classifiers. Numerical experiments confirm that EDDIE-101 can be successfully extended to financial forecasting.

Pu Wang and Thomas Weise
Nature Inspired Computation & Applications Laboratory
University of Science and Technology of China
Hefei, Anhui, China
Emails: wuyou308@mail.ustc.edu.cn, tweise@ustc.edu.cn

Raymond Chiong
Faculty of Information & Communication Technologies
Swinburne University of Technology
Victoria 3122, Australia
Email: rchiong@swin.edu.au

Keywords data mining · evolutionary algorithms · rule-based classifiers · decision trees · EDDIE-101

1 Introduction

Data mining is an important process in many business, science and industrial problems. Its aim is to extract implicit, not yet known information from data [23]. Owing to the huge amount of data generated and stored in databases nowadays, the commercial and research values of effective data mining are tremendous. Numerous application areas, including astronomy, bioinformatics, drug discovery, advertising, customer relationship management, fraud detection, healthcare, manufacturing, targeted marketing, financial transactions, government data and environmental monitoring, among others (see [32, 36]), are relying heavily on powerful data mining techniques to uncover trends and hidden knowledge from large volumes of data.

In the past two decades, a wide variety of different approaches have been proposed in the literature for classification tasks in large datasets, and many of these approaches have been proven to be very effective. However, not all of them are flexible and easily extensible. Evolutionary Algorithms (EAs), a family of metaheuristic optimization methods which comprises Genetic Algorithms (GAs), Genetic Programming (GP), Evolutionary Programming (EP) and Evolution Strategies (ES), are commonly used to address problems with large search spaces. EAs have been successfully applied to many different data mining problems too (see [12, 25, 29]). The strength of EAs is that they require very little domain-specific knowledge of the data to be classified, thus simplifying the classification tasks to some extent. Besides that, EAs also possess great versatility, i.e., they can be used to generate solutions of virtually arbitrary structures.

Motivated by these, the purpose of this paper is to propose some novel evolutionary approaches for supervised classification tasks in data mining which are flexible across different problem domains. Three such approaches, including a rule-based EA and two tree-based GPs, have been presented. The performances of these approaches are compared to a number of well-established classification methods using some popular real-world datasets. Different comparisons on (1) the sensitivity of the approaches to configurations, (2) the accuracy of the approaches, (3) the required runtime, and (4) the convergence have been performed. The results show that these evolutionary approaches are promising and comparable to the well-established algorithms. To further test their flexibility across different problem domains, we also apply them to real datasets from the financial sector. It is confirmed via numerical experiments that the proposed approaches can be extended to financial forecasting.

The rest of this paper is organized as follows. Section 2 starts with some theoretical background on classification in supervised data mining, followed by a brief description of EAs and discussion of some related work. Subsequently, the three proposed approaches are presented in Section 3. In Section 4, we describe the experimental settings, including the datasets used, the classifiers to be compared, and discuss the experimental results. Finally, Section 5 concludes the study.

2 Background

2.1 The Classification Task

Broadly speaking, the task of classification is to assign a class k from a set of possible classes K to vectors (data samples) $t = (t_1, t_2, \ldots, t_n)$ consisting of n attribute values $t_i \in \mathbb{T}_i$. In supervised approaches, the starting point is a training set A including training samples $t \in A$ for which the corresponding classes $class(t) \in K$ are already known. A data mining algorithm is supposed to learn a relation (called *classifier*) $C : \mathbb{T}_1 \times \mathbb{T}_2 \times \cdots \times \mathbb{T}_n \mapsto K$ which can map such attribute vectors t to a corresponding class $k \in K$. The training set A usually contains only a small subset of the possible attribute vectors and may even include contradicting samples $(a = b : a, b \in A \wedge class(a) \neq class(b))$.

The better the classifier C is, the more often it can correctly classify attribute vectors. An overfitted classifier has no generalization capability: it learns only the exact relations provided in the training sample but is unable to classify samples not included in A with reasonable precision. In order to test whether a classifier C is overfitted, not the complete available data $\mathbf{A}$ is used for learning. Instead, $\mathbf{A}$ is divided into the training set A and the test set $\overline{A}$. If C's precision on $\overline{A}$ is much worse than on A, it is overfitted and should not be used in a practical application.

2.2 Evolutionary Algorithms

EAs belong to the family of nature-inspired optimization algorithms [14, 15]. In general, an EA can be schematized as a population-based search which is characterized by an initial creation of a set of candidate solutions and a generation cycle, as depicted in Figure 1. A population of candidate solutions is presumed to evolve over the generation

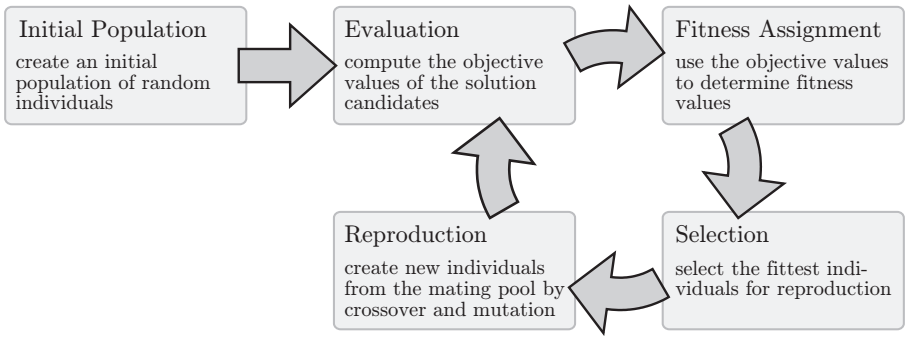

Fig. 1: The common EA cycle.

cycles utilizing some forms of natural processes such as selection and reproduction in order to refine the solutions iteratively [6, 66].

First, a set of randomly configured candidate solutions are created. The cycle itself then starts with the evaluation of the objective values of these solutions. Based on

the results, a relative fitness is assigned to each candidate solution in the population. This fitness value is the criterion on which selection algorithms operate to pick the most promising individuals for further investigation while discarding the less successful ones. The candidate solutions that managed to enter the so-called *mating pool* are then reproduced, i.e., combined via crossover and/or slightly changed by mutation operations. When this is done, the cycle starts again in the next generation.

2.3 Related Work

The necessity to discover patterns and analyze trends in large sets of data has made data mining one of the most active research areas in Machine Learning, Artificial Intelligence, and Optimization along the years. Numerous methods such as probabilistic models, neural networks, support vector machines, rough sets, decision trees and evolutionary computation techniques have been proposed in the literature for various classification tasks, and the majority of these methods have been proven to be very effective (e.g. [3, 7, 16, 19, 28, 31, 42, 44, 48, 49, 50, 52, 57, 70]). Among them, decision tree algorithms are perhaps the most commonly used. This kind of algorithms typically consists of two phases, i.e., a tree-building phase and a tree-pruning phase. Of the many decision tree algorithms that have been used in data mining, C4.5 [49] and Random-Forest [11, 28] are by far the most well-known.

Besides decision tree algorithms, EAs have also been used extensively in data mining. GAs, the best-known type of EAs, are most often applied to classification in the form of Learning Classifier Systems (LCSs). There are two types of LCSs: the Pittsburgh approach [55] and the Michigan approach [33]. The Pittsburgh approach resembles a traditional GA in which each individual in the population is a set of rules representing a complete solution. The Michigan approach, on the other hand, uses the entire population to represent individual rules, where each rule is a partial solution to the overall learning task. Some examples of recently proposed GA-based approaches can be found in [3, 19, 35, 59]. For more information about LCSs in data mining, see [12].

GP, another type of EAs that evolves trees or program-like structures, has found many applications in classification tasks too (see the review by Jabeen and Baig [36]). The earliest related work on tree-based evolutionary data mining dates back to Forsyth's experiments in 1981, in which he evolved single, binary decision rules that did not feature branching [21]. Another set of early results in the area of evolving decision trees has been contributed by Koza [38] as he applied a GP approach to an example problem also used by Quinlan for illustrating the ID3 algorithm – the predecessor of the popular C4.5 algorithm [49].

Freitas [24] used a GP framework to create classification rules composed of comparison and logical operators. In problems with more than two classes ($|K| > 2$), he applied an approach similar to the Michigan-style LCS: a niching method which allows the selection of $|K| - 1$ classification rules, each matching a different class. The need for this niching scheme in the case of multiple-class problems has been discussed in his work.

The evolution of *oblique* decision trees has been pursued by Cantú-Paz and Kamath [13]. The decision expressions attached to the nodes of such trees have the form $\sum_{i=1}^{n} a_i t_i > 0$ where the factors $a_1 \ldots a_n$ are subject to optimization. Their empirical

results suggest that EAs not only can find competitive classifiers very quickly but also scale well with the dimensionality.

In [45], a multi-tree approach for solving classification problems with $|K| \geq 2$ is presented. This approach takes an integrated view of all classes when evolving GP, and uses one distinct tree for deciding whether a data sample belongs to one of the $|K|$ classes or not. Numerical results on several benchmark datasets show that the performance of this approach is highly satisfactory.

While most of the evolutionary classifiers in the literature have been tested on artificial or pure benchmark data, the *Evolutionary Dynamic Data Investment Evaluator*, or EDDIE in short, was developed with real-world problems from the fields of Finance and Economics in mind [60, 61, 62]. EDDIE-1 [60], for example, was applied to horse races in 1998. However, the lack of sufficient datasets for testing in horse races at that time brought about limited performance of EDDIE-1. The subsequent release of EDDIE, Li et al's FGP-1 [40], was used for financial forecasting. In FGP-1, a cost-sensitive fitness method has been adopted to guide the evolution process [41].

A recent work from García-Almanza et al [27] combined EDDIE and a repository method [26] to evolve a set of decision rules for discovering patterns in financial datasets. The repository method builds a repository by extracting rules from decision trees and collecting those with best performances. These rules are then simplified and added to the repository if some conditions are satisfied (such as not being too similar to the rules already stored in the repository). Finally, all rules in the repository are combined to classify the training or test datasets.

Before ending this section, we would like to point out several comparative studies that can be found in the literature. For example, Bernadó et al [9] compared two different LCSs to classical, non-evolutionary methods on several datasets and found that EAs are highly suitable for data mining and classification. Bacardit and Butz [5] compared a Pittsburgh-style LCS called GAssist to XCS, a Michigan-style LCS, using some publicly available datasets and revealed important differences between the two systems. Kharbat et al [37] compared XCS with C4.5 based on breast cancer data and showed that XCS provides significantly higher accuracy. Orriols-Puig et al [47] reviewed several evolutionary based machine learning approaches for pattern recognition and compared them with some of the most influential non-evolutionary supervised learning methods using a collection of twenty real-world datasets. In terms of accuracy, they found that UCS, a competitive Michigan-style LCS derived from XCS, appears to be the best performer. In a separate study, Orriols-Puig and Bernadó-Mansilla [46] further evaluated XCS and UCS for extracting knowledge from imbalanced data using both fabricated and real-world problems. They showed that both LCSs are robust to high imbalance ratios if some critical parameters are correctly configured.

In one of the most recent studies, Tanwani and Farooq [58] performed some extensive experiments using six well-known evolutionary rule-learning algorithms on numerous publicly available biomedical datasets for the purposes of automatic knowledge extraction. Another work by Fernández et al [18] also entailed an exhaustive experimental study comparing evolutionary rule-based approaches with some classical non-evolutionary algorithms using both standard classification problems and imbalanced datasets. None of these studies, however, has compared evolutionary rule-learning classification approaches with evolutionary (decision) tree synthesizing approaches directly. The work presented here, thus, aims to provide new insights into the mutual benefits and demerits of these two classes of data mining approaches by introducing some novel enhancements to them.

3 Evolutionary Data Mining Approaches

3.1 The Rule-Based EA

The first classification approach we propose in this work is similar to the Pittsburgh LCS [17, 55, 56]. Like the Pittsburgh LCS, we use a GA to work on a population of classifiers encoded as bit strings, each of which being a list of rules (i.e., individual classifiers which together form the classifier system). This simple rule design is chosen so that we can test the ability of the evolutionary approach to build good solutions (classifiers) of arbitrary structures. At the same time, the design allows us to encode the rules in an EA.

3.1.1 Classifying with Rules

A rule contains a classification part encoding a class $k \in K$ and a condition for each feature in the input data. Unlike conventional LCSs, the conditions of our rule-based EA are no simple ternary patterns (like `0`, `1`, and `*` for *don't care*) to be matched against data samples but encode a more complex relation.

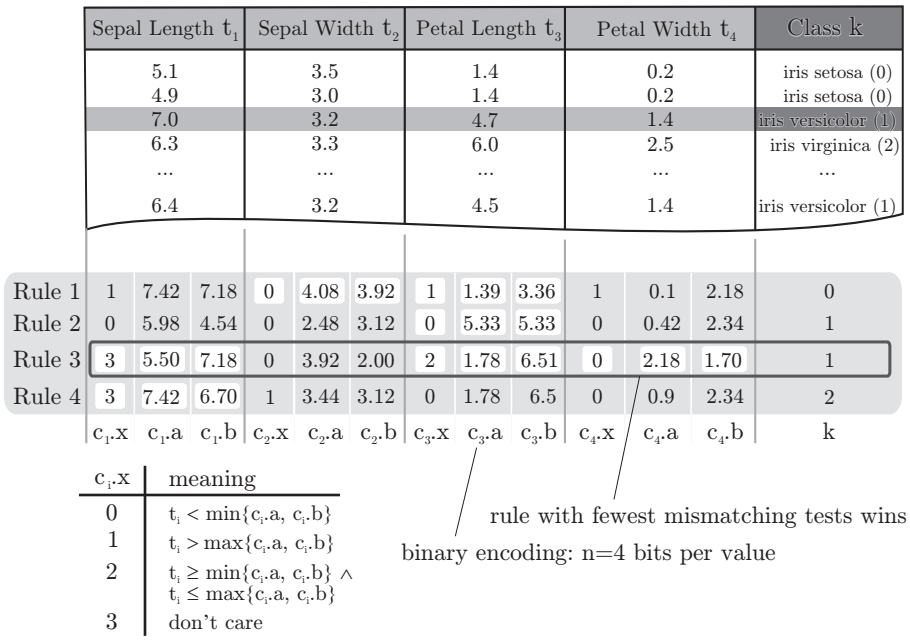

Fig. 2: An example rule-based classifier applied to the *Iris Dataset* [2, 54].

Figure 2 sketches the application of such a rule-based classifier to the well-known *Iris Dataset* [2, 54], where each data sample t stands for a flower and is characterized by four real attributes $(t_1 \dots t_4)$. In our approach, each rule of the classifier therefore consists of four conditions $c_1 \dots c_4$. Each condition c_i, in turn, consists of one operation

selector $c_i.x$ and two real values $c_i.a$ and $c_i.b$. If $c_i.x = 0$, the condition matches if $t_i < \min\{c_i.a, c_i.b\}$ and for $c_i.x = 1$, it is `true` if and only if $t_i > \max\{c_i.a, c_i.b\}$. t_i must fall into the real interval $[\min\{c_i.a, c_i.b\}, \max\{c_i.a, c_i.b\}]$ if $c_i.x = 2$ and $c_i.x = 3$ stands for *don't care* (like the * symbol in LCSs). All rules are matched against a data sample t one after the other, starting with the first rule. The corresponding class of the rule with the least unmet conditions is returned in the end. If there is a tie, the rule at the lower index wins. This allows for rules describing classes which are nested within each other. It is thus possible to describe a big hypercube containing a smaller hypercube of data by simply putting the rule for the smaller area first.

In Figure 2, the application of our rule-based EA to the third instance of the *Iris Dataset* is illustrated. By chance, the third rule has the fewest mismatching conditions (one) and therefore, the data sample is classified with class `1`.

3.1.2 Evolving Rules

Since we use a GA with binary encoding to evolve the rule-based classifiers, we also represent real values $c_i.a$ and $c_i.b$ in the conditions with binary numbers. We use at most four bits for each condition. In cases where an attribute can take on more than sixteen values, the linear scaling mechanism defined in Equation 1 is applied to translate values z (four bits) to the corresponding conditions $c_i.a$ and $c_i.b$ respectively.

$$\min\{t_i : \forall t \in T\} + z\frac{\max\{t_i : \forall t \in T\} - \min\{t_i : \forall t \in T\}}{15} \tag{1}$$

It is necessary to note that our rule set evolution shares some similarities to the work of Corcoran and Sen [16]. In their approach, classification rules are encoded in a real-valued genome where each condition is defined as a range $[c_i.a, c_i.b]$ given by two real values $c_i.a$ and $c_i.b$. A *don't care* situation occurs when $c_i.a > c_i.b$. Different from ours, they used a fixed-length genome where the number of rules is predetermined. They considered only exact matching rules during the classification process. In our approach, the rule that has the least errors wins, if no perfect match could be found. Furthermore, Corcoran and Sen [16] used an internal voting mechanism in case of draws between two rules matching a sample. For us, the rule with the lower index wins. We can use this much simpler approach since our rule-based classifiers are able to permutate rules via crossover and thus, can find their best order during the course of evolution.

3.2 GP with Unconstrained Decision Trees

As mentioned in Section 2.3, the synthesis of decision trees is a very common data mining approach. The leaf nodes of decision trees contain the identifiers of the classes $k \in K$ and each inner node usually has two children. The starting point for the classification of a data sample $t \in A$ is the root of the tree. From there on, a path is followed to one of the leaves which then represents the estimated class of t. At each node on this path, a decision is typically made based on the comparison of one attribute value with a constant. According to this decision, either the left or the right child of the node is visited. Eventually, the control token will reach a leaf node and the class attached to this node is returned.

3.2.1 Classifying with Unconstrained Decision Trees

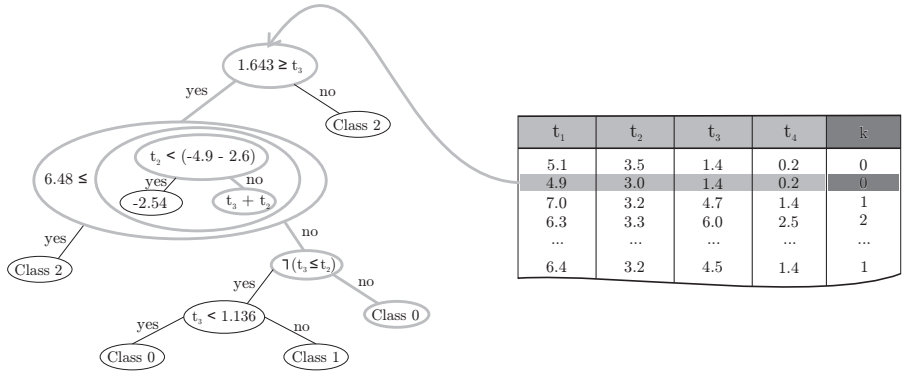

Fig. 3: An evolved decision tree applied to the same data sample as used in Figure 2

In our second classification approach, we extend the decision tree approach by releasing all constraints on the expressions leading to the decisions. In other words, instead of a simple comparison like $t_3 > 1.4$, complex formulas like $(t_3 * t_1 - 4 \leq 2 * t2) \vee (t_3 - t_1 > t_2 * 0.2)$ can be attached to a node in our unconstrained decision trees (UDTs). Furthermore, the shape of the trees is subject to optimization too.

3.2.2 Evolving Unconstrained Decision Trees

GP is a type of EAs where the search space consists of trees and therefore, is predestined for the creation of decision trees. In order to evolve arbitrarily structured decision trees, we apply GP with a function set $F = \{+, -, *, \wedge, \vee, \neg, =, >, <, \leq, \geq, \neq, \texttt{if-then-else}\}$ and a terminal set consisting of constants, references to attributes, and class identifiers.

In Figure 3, an evolved decision tree with unconstrained node expressions is applied to the same data sample used in Figure 2. The control flow is highlighted by marking the nodes and transitions with bold lines.

Our UDT-based GP is different from the related GP approaches discussed in Section 2.3. Unlike [13], for example, the trees evolved in our GP are much less restricted in their form and can, in fact, use arbitrary mathematical expressions (depending on the function set made available to the GP process). The size of the expressions in the nodes of our GP may vary and is being optimized too, which increases the information density in the classifiers.

The UDT-based GP also differs from the multi-tree approach in [45] as we use only one single tree whose leaves are labels which may belong to any of the $|K|$ classes.

3.3 EDDIE-101 – An Advanced Tree-based Classifier

Our third classification approach, known as EDDIE-101, is a new variant of the EDDIE approach. EDDIE-101, like its predecessors, is also based on decision trees but with a set of new, advanced features being incorporated.

3.3.1 Classifying with EDDIE-101

Figure 4 sketches an individual, a so-called Genetic Decision Tree (GDT, [40]), as it may have resulted from a reproduction operation in EDDIE-101. Initially, EDDIE-101 does not assign classes to the output leaf nodes but only allocates one counter in them for each possible class $k \in K$. This occurs in the evaluation and learning phase, during which the GDTs are applied to each training sample $t \in A$.

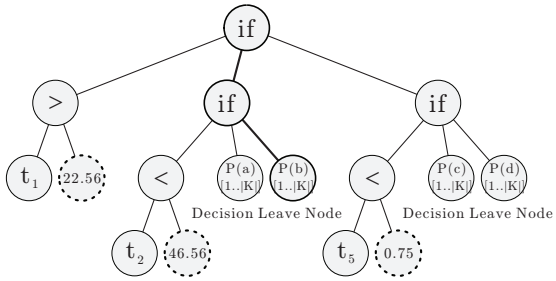

Fig. 4: An example GDT of EDDIE-101

Like normal decision trees, the samples are "put" into the root node of the trees and move along the branches according to the evolved decision expressions. In the learning phase, the leaf nodes record just how many instances of each class have arrived at them. The classification result is then the class arriving in the leaf node with the highest frequency during the training phase.

3.3.2 Evolving Genetic Decision Trees

For synthesizing GDTs, EDDIE-101 adopts the new crossover and mutation operations introduced by Muni et al [45]. The overall algorithm of EDDIE-101 is illustrated in Figure 5.

At the beginning of the algorithm, we initialize a group of individuals as the parents. We use the *ramped half-and-half* method proposed by Koza [39] to initialize the population. This method creates equal amounts of individuals according to two schemes (*full* and *grow*) for each depth. The *full* method which enforces that all leaf nodes have the same depth in one tree and the *grow* method which limits the maximum distance between the leaves and the roots but allows the leaves to be at random depths. We apply tournament selection to weed out individuals with low fitness in each generation with high probability.

Tournament selection is used for parental selection as well. In order to give the surviving individuals with lower fitness a chance to improve, the two losers of a tournament with four contestants are recombined (with probability p_c) and mutated (with probability p_m).[1] This step is repeated until all free slots in the population (opened by survival selection) are filled with fresh offspring. Before inserting the newly created GDTs into the population, however, they are optimized with a local search operation. In this step, the thresholds in their selector nodes (the nodes enclosed with dotted lines in Figure 4) are, one after the other, refined with a simple hill climber.

```
Algorithm EDDIE-101(M)
M is the maximum generation
BEGIN
   Let gen = 0
   Initialize the population using the ramped-half-and-half method
   while (gen < M)
      Evaluate fitness (f4, see Section 3.4.3) of each individual
      Update the best individual
      Survival selection: tournament selection by f4
      Crossover and mutation
      Adjust individuals with hill-climbing
      gen = gen + 1
   end while
Best individual on test dataset
END
```

Fig. 5: The EDDIE-101 algorithm

While EDDIE-101 is related to its predecessors, it is also different in several ways: EDDIE-101 differs from FGP-1 not only in the fitness function utilized and the structure of the GP process, but also in the way the classifiers are initialized. While EDDIE-101 counts the instances of each class arriving at the leaf nodes, FGP-1 assigns random classes to the leaves.

Furthermore, we have not applied the repository method as used by García-Almanza et al [27] because its main search operator is based on subtree mutation which makes it vulnerable to premature convergence. Another reason for not doing so is that it needs a large repository to manage a high number of rules. Maintaining this repository costs much time.

3.4 Objective Functions, Voting, and Information Gain

Our first two approaches, the rule-based EA and GP with UDTs, are driven by three different objective functions, $f_1 \dots f_3$, all subject to minimization. The Pareto-front returned by a multi-objective optimization process is then combined to a single super classifier. EDDIE-101, on the other hand, utilizes an advanced single-objective concept (i.e., the objective function f_4).

[1] A similar approach has been suggested by Muni et al [45].

3.4.1 Objective Functions for the Rule-based EA and GP with UDTs

The objective values of the candidate classifiers are determined by applying them to each sample in the training data A. f_1 corresponds to the number of samples which have been classified correctly. Since our aim is minimization, we negate this value. The value of f_2 is determined by a cost matrix which can be specified *a priori*. It also allows the introduction of different penalties for certain misclassifications. For instance, classifying an *iris setosa* as *iris virginica* might be less critical than classifying it as an *iris versicolor*. If no cost matrix is provided, the identity matrix is used instead and f_1 and f_2 become interchangeable. The third optimization criterion (f_3) is the size of the classifier, i.e., the number of rules in a rule set or the number of nodes in a decision tree. The smaller the classifier is, the less likely it is overfitted. Therefore, it makes sense to add some selection pressure on decreasing the classifier size.

3.4.2 Voting Mechanisms

If the three objective functions are applied in the context of Multi-Objective Evolutionary Algorithms (MOEAs) [63, 71] together with a Pareto-ranking based fitness assignment process [66], the result will be a set of classifiers instead of a single solution. If evolving decision trees, for instance, the Pareto set will contain trees of several different sizes. A classifier C_1 with fewer nodes (better f_3 value) which performs slightly worse (in respect of f_2 or f_1) than another one (C_2) with more nodes, is neither better nor worse. In terms of the Pareto relation, this means that C_1 is not dominated by C_2 and vice versa.

Instead of using only one of the evolved classifiers, we combine the complete Pareto set resulting from the optimization process to a single super classifier. We therefore introduce two alternative voting mechanisms **V1** and **V2**. If a data sample is to be classified, all the members of the Pareto set are applied to it and their results (the suggested classes) are noted. If **V1** is applied, each classifier can cast a number of votes inversely proportional to the costs it has caused (f_2); and with variant **V2**, the number of votes of a classifier is inversely proportional to a combination of f_1 and f_3. The class that receives the most votes is then the result of the classification.

Compared to other related evolutionary data mining methods, our approaches here have one obvious advantage: they utilize the full potential of MOEAs by incorporating the whole Pareto set into one super classifier. With this, none of the evolved solutions, i.e., the knowledge gathered, is lost and higher robustness against overfitting can be achieved. Another advantage of this is that results from multiple runs of the optimizer may also be combined.

Boosting and ensemble learning methods [4, 51] are other approaches that also combine different learners or classifiers in order to increase precision. These methods follow a concept different from our simple voting mechanisms by involving learning on subsets of the training dataset. In our case, all classifiers are trained with the same data and we do not explicitly aim to create different classifiers; they simply result from the nature of multi-objective optimization. Voting is only used to utilize the full information which our EAs extract from their input data.

3.4.3 Information Gain in EDDIE-101

In EDDIE-101, a novel fitness measure f_4 which represents the information gain [8] is introduced according to Equation 3.

$$p(l,k) = \frac{P(l)[k]}{\sum_{i=1}^{|K|} P(l)[i]} \tag{2}$$

$$f_4(x) = \frac{\sum_{\forall leaves\ l \in x} \left(1 + \sum_{k=1}^{|K|} p(l,k) \log_2 p(l,k)\right) \left(-1 + \sum_{k=1}^{|K|} P(l)[k]\right)}{|A| - |leaves \in x|} \tag{3}$$

As already mentioned, a classifier x is applied to all training samples $t \in A$ during the evaluation phase. Each leaf node l has a set of counters $P(l)$, one counter for each of $|K|$ classes $k \in K$, and the leaf node remembers how many instances of different classes have arrived at it. Let $P(l)[k]$ be the number of instances of class $k \in K$ arriving at leaf node l. $p(l,k)$ in Equation 2 is then the frequency of this class in l relative to other classes. The entropy-based fitness $f_4(x) \in [0,1]$ stands for the total discrimination power of the classifier. If it is 1, the classifier has the best possible training result since in each leaf node, only samples belonging to a single class have arrived. If $f_4(x)$ is 0, the classifier x exhibits random behavior in which equally many instances of each class are found at the leaf nodes. f_4 is therefore subject to maximization.

4 Experiments and Results

4.1 Datasets

For the purpose of verifying the performances of our evolutionary approaches, we first applied them to five standard real-world datasets ($\mathbf{A}_1$-$\mathbf{A}_5$) from supervised data mining. Except for $\mathbf{A}_3$ which was already divided into two subsets by default and $\mathbf{A}_5$, where we used 100 samples for testing, 70% of the samples of the datasets $\mathbf{A}_i$ were used as training samples (A) and the remaining 30% as test samples $\overline{A}$. This was done so according to the dataset division utility provided by *Weka* [22, 34].

The first dataset used $\mathbf{A}_1$ is the *Iris Dataset* [2, 54] already introduced in Section 3.1.1. Besides this one, other datasets include the *Wine Dataset*, the *Heart Disease Dataset*, the *Wisconsin Breast Cancer Dataset* and the *Hepatitis Dataset*.

The *Wine Dataset* ($\mathbf{A}_2$) by [20] represents the results of a chemical analysis of different wines. The three examined wines ($|K| = 3$) all grow in the same region in Italy but stem from different cultures. For each of the 178 samples t, thirteen characteristic components $t_1 \ldots t_{13}$ have been measured.

The *Heart Disease Dataset* [69] ($\mathbf{A}_3$) contains cardiological diagnostic samples from Single Proton Emission Computed Tomography (SPECT3) images. Each of the 267 samples (representing a patient) has 22 binary attributes and is classified into one of the two classes $k_1 =$ `normal` and $k_2 =$ `abnormal`.

The *Wisconsin Breast Cancer Dataset* [69] ($\mathbf{A}_4$) consists of documented clinical samples which have 10 attributes, each stands for one measurement from a breast cancer examination with a scale from one to ten. The 699 samples are again to be classified into two classes.

The *Hepatitis Dataset* [30] ($\mathbf{A}_5$) consists of 19 attributes which indicate whether a person suffering from hepatitis will survive this infection or not ($|K| = 2$). There are categorical, integer-valued, and also real attributes. This dataset has 155 samples.

To further evaluate the flexibility of the proposed approaches across different problem domains, we also tested them on much larger real datasets ($\mathbf{A}_6$-$\mathbf{A}_8$) from the financial sector. Here, cross validation (i.e., to test different partitions of the datasets) makes little sense since the financial data basically represent time series that are used to develop forecasting classifiers. Randomly dividing a time series may lead to much worse performance of the classifier and thus biased experimental results. For the sake of consistency, we have decided to use single divisions for both the test and training data in this set of experiments.

The *S&P-500 Dataset* ($\mathbf{A}_6$) consists of 2700 samples. This dataset stems from the Heng Seng Index. Available to us were data from the 2nd of April 1963 to the 25th of January 1974. The attributes of each case are composed by indicators which are derived from financial technical analysis. These indicators are based on the daily closing price. We tried to find classifiers which detect an increase of stock value of 4% in a horizon of 63 days. We divided the data into 1800 cases for training and put 900 cases into the test dataset.

The *DJIA Dataset* was obtained from the closing prices of the Dow Jones Industrial Average. We chose two ranges in DJIA for our experiments, one has 2800 cases (from the 7th of April 1969 to the 5th of May 1980, $\mathbf{A}_7$) while the other consists of 3035 cases (from the 7th of April 1969 to the 9th of April 1981, $\mathbf{A}_8$). In $\mathbf{A}_7$, on one hand we again tried to detect an increase of 4% in stock value during a horizon of 63 days (setup $\mathbf{A}_7^1$). On the other hand, we also tried to classify according to a stock value increase of 2.2% in a horizon of 21 days (setup $\mathbf{A}_7^2$). For $\mathbf{A}_8$, we targeted an increase of 2.2% in a horizon of 21 days.

4.2 Classifiers for Comparison

For comparison, we also applied some well-established decision tree-based classification approaches to the datasets described in Section 4.1. We used the Weka framework [22, 34] which provides default implementations of these conventional classification methods.

The first approach included for comparison is the Random-Forest algorithm by [11] which trains multiple decision trees on sub-samples drawn with the *bagging* [10] approach from the input data. Like our approach, the trees are combined with a voting mechanism. In Weka, we used the default settings for both the tree size as well as the features in our experiments.

The second approach used for comparison is the popular C4.5 algorithm by [49] which creates decision trees where an arbitrary number of branches can follow each node. Its implementation in Weka, called *J48*, has separate tree branches for each value of the attribute used for decision at a given node. We used this approach with a confidence factor of 0.25 and a minimum number of samples per leaf of 2.

Finally, we applied the Weka-specific *RepTree* algorithm which uses information gain/variance reduction and prunes the trees with reduced-error pruning and back-fitting. In the experiments, we used the default settings, i.e., a minimum variance proportion of 0.001 and a minimal sample weight of the leaf nodes of 2.0.

4.3 Experimental Setup

In the experiments with our evolutionary data mining approaches, by default we used EAs with a population size of 2000 individuals, Pareto ranking, tournament selection with three contestants, and a simple convergence prevention (SCP) algorithm from [65, 66]. We ran multiple series for 1000 generations each, with different mutation rates $mr \in \{0.6, 0.7\}$, crossover rates $cr \in \{0.3, 0.4\}$, both voting methods $v \in \{\mathbf{V}_1, \mathbf{V}_2\}$, and either with steady-state ($ss = 1$) or generational ($ss = 0$) population handling strategies. All randomized algorithms, i.e., the EAs and the Random-Forest method, were applied at least ten times to each dataset.

4.4 Experimental Results

With our experiments, we aim to answer three questions:

1. Do different parameter settings of EAs lead to significant differences in the classification accuracy of the evolved classifiers? This would be undesirable, since we aim to provide classifiers which work well regardless of user settings.
2. How are the performances of our approaches compared to the well-known methods like C4.5 and Random-Forest? Obviously, evolutionary classification is only useful if the approaches are at least on-par with classical methods.
3. Can more advanced approaches like EDDIE-101 provide better solutions than straightforward ones like the UDT-based GP? In EDDIE-101, we deviate from default EA structures and introduce a couple of novel operators (as discussed in Section 3.3.2) as well as an advanced fitness assignment procedure (Equation 3). Only if these modifications elevate the performance above the other two evolutionary approaches, the overhead of implementing EDDIE-101 can be justified.

We measured the median cA of the percentage of correctly classified samples from the training data A, the median $c\overline{A}$ of correctly classified samples on the test data $\overline{A}$, and the best result from all runs on the test data $b\overline{A}$.

4.4.1 Robustness across Different Configurations

In Table 1, we present the results achieved with different configurations of our EAs in the form of rule-based ($ap = r$) or tree-based ($ap = t$) ordered by $c\overline{A}$. From the table, it is clear that the best median precision on the test data is always reached by the rule-based EA. In all but one case, it is also the approach with the best $b\overline{A}$ value. The combination of a mutation rate of 30% and a crossover rate of 70% seems to be more effective than a 40%/60% setting. Interestingly, the classifiers with the best precision are always found by the generational EAs ($ss = 0$) while the steady-state EAs ($ss = 1$) always achieve the best median result. Between the two voting mechanisms, no supremacy can be detected.

We further analyzed the influence of each single parameter with two-tailed Sign Tests [53] and the Wilcoxon's Signed Rank Test [68]. We found that no single parameter alone has a significant positive or negative influence when applying the tests with a significance level of 5%. In other words, our EAs are very robust and deliver results with similar qualities for a variety of settings. This is a significant positive finding since

Rank	ap	mr	cr	ss	v	cA	$c\overline{A}$	$b\overline{A}$
Iris Dataset (A_1)								
1	r	0.7	0.3	1	V2	97.98	98.04	98.04
2	t	0.7	0.3	1	V2	97.98	98.04	98.04
3	t	0.6	0.4	1	V2	97.98	98.04	98.04
4	t	0.7	0.3	0	V1	98.48	98.04	98.04
...	...	...	...	...	...	...	...	...
15	r	0.7	0.3	0	V1	100.00	94.12	100.00
...	...	...	...	...	...	...	...	...
Wine Dataset (A_2)								
1	r	0.7	0.3	1	V1	100.00	91.67	96.67
2	r	0.6	0.4	0	V1	100.00	91.67	96.67
3	t	0.6	0.4	1	V2	96.61	90.83	93.33
4	t	0.7	0.3	1	V2	97.03	90.83	93.33
...	...	...	...	...	...	...	...	...
16	t	0.7	0.3	0	V1	97.46	86.67	98.33
Heart Disease Dataset (A_3)								
1	r	0.7	0.3	1	V1	92.41	73.93	76.34
2	r	0.6	0.4	0	V2	91.14	73.12	81.18
3	t	0.7	0.3	1	V1	89.87	72.90	75.81
4	t	0.7	0.3	1	V2	86.71	72.04	77.41
...	...	...	...	...	...	...	...	...
Breast Cancer Dataset (A_4)								
1	r	0.6	0.4	1	V1	99.42	99.00	99.00
2	t	0.7	0.3	0	V1	98.24	98.50	99.00
3	t	0.7	0.3	1	V2	97.99	98.00	99.00
4	t	0.6	0.4	0	V1	98.16	98.00	99.00
...	...	...	...	...	...	...	...	...
7	r	0.7	0.3	0	V2	99.25	98.00	100.00
...	...	...	...	...	...	...	...	...
Hepatitis Dataset (A_5)								
1	r	0.7	0.3	1	V2	98.10	86.28	90.19
2	r	0.7	0.3	1	V1	99.10	84.62	88.26
3	t	0.7	0.3	0	V2	91.43	84.31	86.27
4	t	0.7	0.3	0	V1	96.67	84.31	88.23
...	...	...	...	...	...	...	...	...
7	r	0.6	0.4	0	V2	98.10	83.33	92.16
...	...	...	...	...	...	...	...	...

Table 1: The best configurations of the classifier-evolving EAs.

it means that, indeed, little knowledge about good parameter settings is required. The EAs are hence easy to use.

When comparing the columns cA and $c\overline{A}$, we found that the classifiers are only moderately overfitted except in the *Heart Disease Dataset* where the evolved classifiers are almost 20% better on the training data than on the test samples. From these results, we can deduce that the approaches are able to generalize well, even if excessive runtime in the form of 1000 generations is provided.

a) Rule-EA

	$c\overline{A}$	$b\overline{A}$	b	c	d	e	f
$\mathbf{A}_1$	98.04	98.04	0	0	0	0	0
$\mathbf{A}_2$	89.17	93.33	0	−	−	0	−
$\mathbf{A}_3$	73.39	77.42	0	0	0	−	−
$\mathbf{A}_4$	98.00	100.00	0	−	−	−	+
$\mathbf{A}_5$	84.31	88.24	+	0	−	+	−
$\mathbf{A}_6$	55.56	57.78	0	+	−	+	−
$\mathbf{A}_7^1$	56.67	57.98	+	+	−	+	−
$\mathbf{A}_7^2$	54.47	56.67	−	+	+	+	−
$\mathbf{A}_8$	52.20	54.40	0	0	0	+	−

b) UDT-EA

	$c\overline{A}$	$b\overline{A}$	a	c	d	e	f
$\mathbf{A}_1$	96.08	98.04	0	0	−	0	−
$\mathbf{A}_2$	88.33	95.00	0	−	−	0	−
$\mathbf{A}_3$	75.27	77.42	0	0	0	0	−
$\mathbf{A}_4$	98.00	99.00	0	−	−	−	+
$\mathbf{A}_5$	81.37	84.31	−	0	−	+	−
$\mathbf{A}_6$	52.45	59.62	0	0	0	+	−
$\mathbf{A}_7^1$	50.66	53.93	−	0	−	−	−
$\mathbf{A}_7^2$	57.80	60.71	+	+	+	+	−
$\mathbf{A}_8$	51.82	53.52	0	0	0	+	−

c) Random-Forest

	$c\overline{A}$	$b\overline{A}$	a	b	d	e	f
$\mathbf{A}_1$	98.99	100.00	0	0	0	0	0
$\mathbf{A}_2$	96.67	98.33	+	+	+	+	−
$\mathbf{A}_3$	74.73	74.73	0	0	+	0	+
$\mathbf{A}_4$	99.00	99.00	+	+	0	0	−
$\mathbf{A}_5$	80.39	82.25	0	0	0	+	−
$\mathbf{A}_6$	52.47	54.69	−	0	−	+	−
$\mathbf{A}_7^1$	51.85	54.17	−	0	−	−	−
$\mathbf{A}_7^2$	50.18	51.43	−	−	+	0	−
$\mathbf{A}_8$	52.58	56.26	0	0	0	+	−

d) J48

	$c\overline{A}$	$b\overline{A}$	a	b	c	e	f
$\mathbf{A}_1$	98.04	98.04	0	+	0	+	0
$\mathbf{A}_2$	93.33	93.33	+	+	−	+	−
$\mathbf{A}_3$	73.66	73.66	0	0	−	−	−
$\mathbf{A}_4$	100.00	100.00	+	+	0	0	+
$\mathbf{A}_5$	86.28	86.28	+	+	0	+	0
$\mathbf{A}_6$	56.42	56.42	+	0	+	+	−
$\mathbf{A}_7^1$	57.74	57.74	+	+	+	+	0
$\mathbf{A}_7^2$	45.36	45.36	−	−	−	−	−
$\mathbf{A}_8$	52.20	52.20	0	0	0	+	−

e) RepTree

	$c\overline{A}$	$b\overline{A}$	a	b	c	d	f
$\mathbf{A}_1$	96.08	96.08	0	0	0	−	−
$\mathbf{A}_2$	90.00	90.00	0	0	−	−	−
$\mathbf{A}_3$	75.80	75.80	+	0	0	+	−
$\mathbf{A}_4$	100.00	100.00	+	+	0	0	+
$\mathbf{A}_5$	76.47	76.47	−	−	−	−	−
$\mathbf{A}_6$	50.12	50.12	−	−	−	−	−
$\mathbf{A}_7^1$	52.62	52.62	−	+	+	−	−
$\mathbf{A}_7^2$	49.88	49.88	−	−	0	+	−
$\mathbf{A}_8$	50.22	50.22	−	−	−	−	−

f) EDDIE-101

	$c\overline{A}$	$b\overline{A}$	a	b	c	d	e
$\mathbf{A}_1$	97.74	100.00	0	0	0	0	+
$\mathbf{A}_2$	96.33	98.11	+	+	0	+	+
$\mathbf{A}_3$	81.01	82.28	+	+	+	+	+
$\mathbf{A}_4$	95.85	98.99	−	−	−	−	−
$\mathbf{A}_5$	86.67	91.11	+	+	+	0	+
$\mathbf{A}_6$	62.42	64.15	+	+	+	+	+
$\mathbf{A}_7^1$	57.78	60.43	+	+	+	0	+
$\mathbf{A}_7^2$	63.90	67.66	+	+	+	+	+
$\mathbf{A}_8$	56.00	57.87	+	+	+	+	+

Table 2: The approaches in comparison.

4.4.2 Comparisons of Different Approaches

Based on the highly encouraging preliminary results in Section 4.4.1, we resolved to using a single set of configurations for our evolutionary approaches in this section. Since steady-state populations delivered the best mean result, we applied this strategy and the cost-based voting **V1** in our rule-based EA and GP with UDTs. While the population size of 2000 remained unchanged, both mutation and crossover rates were fixed at $mr = 0.3$ and $cr = 0.7$ respectively. These parameter settings used in EDDIE-101 too.

Moreover, we noticed in the previous experiment that, for most of the problems, exhausting the full 1000 generations were not necessary. Instead, the optimization process converged much faster to one optimum most of the time and thus could have been terminated earlier, as we will show later in Section 4.4.4. For this reason, the comparisons here were done based on one tenth of the generations previously used, i.e., 100 generations. This represents the most active phase of the evolution, as can be seen later in Figure 6.

In Table 2, we show the performances of our approaches (denoted as Rule-EA, UDT-EA, and EDDIE-101) in comparison with the other three decision tree algorithms. The

best values in a performance metric for a given dataset are underlined. Every algorithm is denoted with an alphabetic character from **a** to **f**. We compared the results of each algorithm with others using the two-tailed Mann-Whitney U test with a significance threshold of 5%. The results of these comparisons are displayed in the five additional columns for each approach (with headers of **a** to **f** denoting the algorithm to which the comparison was performed). Three different symbols have been used: a `0` denotes that there is no significant difference between the two algorithms, `+` means that the algorithm has yielded significantly better results than the algorithm in the column, and `-` is used when the algorithm has performed statistically worse than the algorithm in the column.

From the table, three things are obvious:

1. EDDIE-101 is significantly superior than other approaches for most of the datasets.
2. In terms of median results, each of the evolutionary approaches has significantly outperformed or is at least as good as the non-evolutionary approaches in at least one dataset; both the rule-based EA and GP with UDTs, however, are not as good as EDDIE-101.
3. Classification for financial forecasting is much harder than classification on the standard real-world datasets, often can be done only with mediocre accuracy.

The rule-based EA, for instance, is significantly better than Random-Forest on $\mathbf{A}_6$ and $\mathbf{A}_7$, J48 on $\mathbf{A}_7^2$, and RepTree on $\mathbf{A}_5$ to $\mathbf{A}_8$. In contrast, it is statistically not as good as Random-Forest on $\mathbf{A}_2$ and $\mathbf{A}_4$, J48 on several datasets, and RepTree on $\mathbf{A}_3$ and $\mathbf{A}_4$. The UDT-based GP has outperformed all other approaches except EDDIE-101 on $\mathbf{A}_7^2$, but has not done well on $\mathbf{A}_7^1$, where it loses against every approach except Random-Forest. While it has done significantly better than EDDIE-101 on $\mathbf{A}_4$, it loses against Random-Forest on $\mathbf{A}_2$ and $\mathbf{A}_4$. EDDIE-101 has obtained the best median results for all datasets except on $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{A}_4$. A surprisingly interesting observation is that EDDIE-101 has been outperformed by all other classifiers on $\mathbf{A}_4$.

These results indicate that all three evolutionary data mining approaches are indeed useful and may lead to very good results even with small numbers of runs. In particular, EDDIE-101 has been able to produce remarkable performances on both the standard as well as the financial datasets. Especially in the latter, with accuracies of typically below 55%, the conventional classification schemes are not much better than random guessing. EDDIE-101, meanwhile, is able to maintain accuracies of more than 60% in the majority of the test cases. This clearly shows that EDDIE-101 can be a robust approach not only for classifying standard data, but it also possesses great potential in forecasting classification.

4.4.3 Runtime

The runtime requirements of our evolutionary approaches are higher than those used for comparison. 1000 generations of the rule-based EA took between half an hour and 10 hours on an off-the-shelf computer for the experiment in Section 4.4.1. The evolution of UDT-based GP, meanwhile, needed between 1.5 hours and 6 hours. There are two reasons for these higher time requirements:

1. EAs are population-based, iterative optimization algorithms which evaluate all candidate solutions in the population in each generation. With our population size of 2000 individuals, this makes 2 000 000 evaluations in total.

2. Evaluating a classifier means applying it to each of the data samples in the training set. In the Random-Forest method, for instance, its individual classifiers are constructed using subsets of the training data only.

For the experiment in Section 4.4.2, the rule-based EA still required up to 10 hours to complete the reduced generation limit of 100 but GP with UDTs needed only 0.5 to 1 hour. A single run of EDDIE-101 took between 3 and 9 minutes – even if multiplied with 10 to approximate the runtime of 1000 generations, it is still the fastest among the three evolutionary approaches.

The high runtime requirements by these evolutionary approaches can clearly be justified by their performances, especially in the case of EDDIE-101. Furthermore, EAs are very suitable for parallelization, which would reduce the runtime significantly.

4.4.4 Convergence

Figure 6 illustrates the convergence behavior of the evolutionary approaches for synthesizing classifiers in terms of the first objective function f_1.[2] For the rule-based EA and the UDT-based GP, the figures are based on data from the experiment presented in Section 4.4.1. The sub-figures show that most of the progresses in terms of classification accuracy on the training samples are always made in the first 100 generations and the evolution usually converges after 200 generations. In the *Wine Dataset* $\mathbf{A}_2$, the process could have been stopped after this phase without any loss in precision. Generally, significant progresses rarely occur after 400 generations. On a side note, it is observed that the evolution of rule-based classifiers converges faster than the synthesis of decision trees.

5 Conclusion and Future Work

In this paper, we have presented three novel evolutionary approaches for building classifiers in supervised data mining tasks. We showed that all three approaches can perform very competitively compared to some standard decision tree algorithms on five general benchmark and four financial forecasting datasets. Their performances have been robust across a variety of configurations.

We have also demonstrated that even simple classifier structures can be synthesized by EAs in a way which is no worse than the highly sophisticated algorithms such as Random-Forest. EDDIE-101 especially stands up to be the most promising among the three evolutionary approaches, and will be the focal point of our future research. Besides that, the problem of runtime has not been considered in this work. In our experiments, EAs in general took much longer to derive classifiers than the conventional algorithms. The basic principles of evolutionary optimization require creating and evaluating many candidate solutions. Here, the evaluation, i.e., the application of the evolved classifiers to the training datasets, most significantly contributes to the runtime. Determining the fitness of the solutions using only a subset of training data may be one way to speed up the process. Another straightforward method for reducing the runtime would be to use efficient parallelization or distribution techniques [1, 43, 66, 67]. Future work will consider these.

[2] EDDIE-101 does not use f_1 as the objective function but we computed the f_1-values for it for the convergence analysis here.

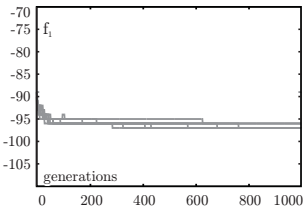

Fig. 6.a: Dataset $\mathbf{A}_1$, rule-based, rank 1.

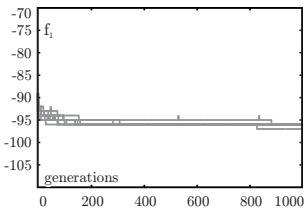

Fig. 6.b: Dataset $\mathbf{A}_1$, tree-based, rank 4.

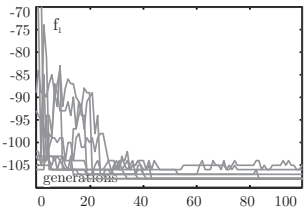

Fig. 6.c: Dataset $\mathbf{A}_1$, EDDIE-101.

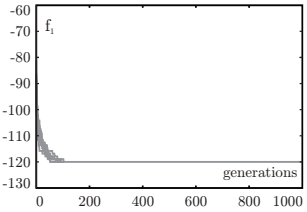

Fig. 6.d: Dataset $\mathbf{A}_2$, rule-based, rank 1.

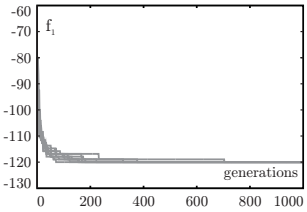

Fig. 6.e: Dataset $\mathbf{A}_2$, tree-based, rank 3.

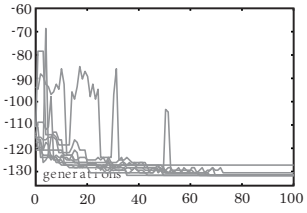

Fig. 6.f: Dataset $\mathbf{A}_2$, EDDIE-101.

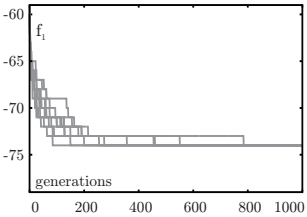

Fig. 6.g: Dataset $\mathbf{A}_3$, rule-based, rank 1.

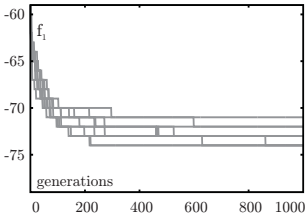

Fig. 6.h: Dataset $\mathbf{A}_3$, tree-based, rank 3.

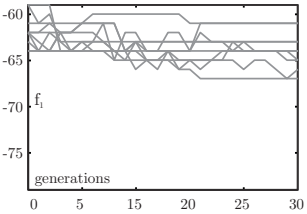

Fig. 6.i: Dataset $\mathbf{A}_3$, EDDIE-101.

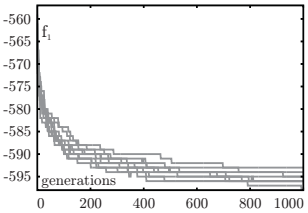

Fig. 6.j: Dataset $\mathbf{A}_4$, rule-based, rank 1.

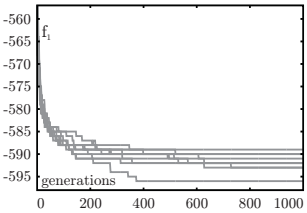

Fig. 6.k: Dataset $\mathbf{A}_4$, tree-based, rank 3.

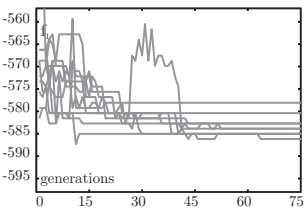

Fig. 6.l: Dataset $\mathbf{A}_4$, EDDIE-101.

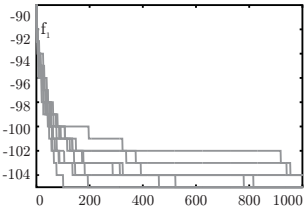

Fig. 6.m: Dataset A_5, rule-based.

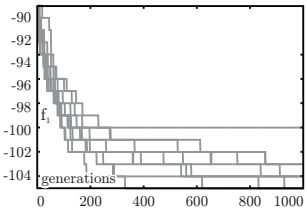

Fig. 6.n: Dataset A_5, tree-based, rank 3.

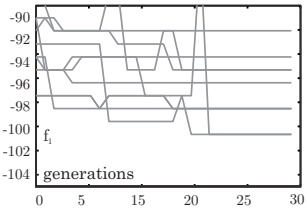

Fig. 6.o: Dataset A_5, EDDIE-101.

Fig. 6: Convergence of the evolutionary data mining approaches.

References

1. Alba Torres E, Tomassini M (2002) Parallelism and Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation 6(5):443–462
2. Anderson E (1935) The Irises of the Gaspé Peninsula. Bulletin of the American Iris Society 59:2–5
3. Au WH, Chan KCC, Yao X (2003) A Novel Evolutionary Data Mining Algorithm with Applications to Churn Prediction. IEEE Transactions on Evolutionary Computation 7(6):532–545
4. Avnimelech R, Intrator N (1999) Boosting Regression Estimators. Neural Computation 11(2):499–520
5. Bacardit J, Butz MV (2007) Data Mining in Learning Classifier Systems: Comparing XCS with GAssist. In: Revised Selected Papers of the International Workshops on Learning Classifier Systems, Springer, Lecture Notes in Artificial Intelligence, vol 4399, pp 282–290
6. Bäck T (1996) Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press
7. Bako L (2010) Real-time Classification of Datasets with Hardware Embedded Neuromorphic Neural Networks. Briefings in Bioinformatics 11(3):348–363
8. Balian R (2004) Entropy, A Protean Concept. In: Poincaré Seminar 2003, Birkhäuser Verlag, Progress in Mathematical Physics, vol 38, pp 119–144
9. Bernadó E, Llorà X, Garrell i Guiu JM (2001) XCS and GALE: A Comparative Study of Two Learning Classifier Systems with Six Other Learning Algorithms on Classification Tasks. In: Advances in Learning Classifier Systems, Revised Papers of IWLCS'01, Springer, Lecture Notes in Artificial Intelligence, vol 2321, pp 115–132
10. Breiman L (1996) Bagging Predictors. Machine Learning 24(2):123–140
11. Breiman L (2001) Random Forests. Machine Learning 45(1):5–32
12. Bull L, Bernadó-Mansilla E, Holmes J (eds) (2008) Learning Classifier Systems in Data Mining, Studies in Computational Intelligence, vol 125. Springer
13. Cantú-Paz E, Kamath C (2000) Using Evolutionary Algorithms to Induce Oblique Decision Trees. In: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann Publishers, pp 1053–1060
14. Chiong R (ed) (2009) Nature-Inspired Algorithms for Optimisation, Studies in Computational Intelligence, vol 193. Springer
15. Chiong R, Neri F, McKay RI (2009) Nature that Breeds Solutions. In: Nature-Inspired Informatics for Intelligent Applications and Knowledge Discovery: Implications in Business, Science and Engineering, Information Science Reference, chap 1, pp 1–24
16. Corcoran AL, Sen S (1994) Using Real-Valued Genetic Algorithms to Evolve Rule Sets for Classification. In: Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE Computer Society, vol 1, pp 120–124
17. De Jong KA, Spears WM (1991) Learning Concept Classification Rules using Genetic Algorithms. In: Mylopoulos J, Reiter R (eds) Proceedings of the 12th International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers, vol 2, pp 651–656
18. Fernández A, García S, Luengo J, Bernadó-Mansilla E, Herrera F (2010) Genetics-Based Machine Learning for Rule Induction: State of the Art, Taxonomy, and Comparative Study. IEEE Transactions on Evolutionary Computation

10.1109/TEVC.2009.2039140
19. Fidelis M, Lopes HS, Freitas AA (2000) Discovering Comprehensible Classification Rules with a Genetic Algorithm. In: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE Computer Society, vol 1, pp 805–810
20. Forina M, Lanteri S, Armanino C, et al (1988) PARVUS – An Extendible Package for Data Exploration, Classification and Correlation. Tech. rep., Institute of Pharmaceutical and Food Analysis and Technologies: Genoa, Italy
21. Forsyth R (1981) BEAGLE – A Darwinian Approach to Pattern Recognition. Kybernetes 10(3):159–166
22. Frank E, Hall MA, Holmes G, Kirkby R, Pfahringer B, Witten IH, Trigg L (2005) WEKA – A Machine Learning Workbench for Data Mining. In: The Data Mining and Knowledge Discovery Handbook, Springer, chap 62, pp 1305–1314
23. Frawley WJ, Piatetsky-Shapiro G, Matheus CJ (1992) Knowledge Discovery in Databases: An Overview. AI Magazine 13(3):213–228
24. Freitas AA (1997) A Genetic Programming Framework for Two Data Mining Tasks: Classification and Generalized Rule Induction. In: Proceedings of the Second Annual Conference on Genetic Programming, Morgan Kaufmann Publishers, pp 96–101
25. Freitas AA (2002) Data Mining and Knowledge Discovery with Evolutionary Algorithms. Natural Computing Series, Springer
26. García-Almanza AL, Tsang EPK (2006) The Repository Method for Chance Discovery in Financial Forecasting. In: Proceedings of the 10th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, Part III, Springer, Lecture Notes in Artificial Intelligence, vol 4253, pp 30–37
27. García-Almanza AL, Tsang EPK, Galván-López E (2008) Evolving Decision Rules to Discover Patterns in Financial Data Sets. In: Computational Methods in Financial Engineering – Essays in Honour of Manfred Gilli, Springer-Verlag, chap II-5, pp 239–255
28. Gehrke J, Ramakrishnan R, Ganti V (1998) RainForest – A Framework for Fast Decision Tree Construction of Large Datasets. In: Proceedings of 24rd International Conference on Very Large Data Bases, Morgan Kaufmann Publishers, pp 416–427
29. Ghosh A, Jain LC (eds) (2005) Evolutionary Computation in Data Mining, Studies in Fuzziness and Soft Computing, vol 163. Springer
30. Gong G, Cestnik B (1988) Hepatitis Data Set. UCI Machine Learning Repository, University of California
31. Grzymala-Busse JW (1997) A New Version of the Rule Induction System LERS. Fundamenta Informaticae – Annales Societatis Mathematicae Polonae, Series IV 31(1):27–39
32. Harding JA, Shahbaz M, Srinivas, Kusiak A (2006) Data Mining in Manufacturing: A Review. Journal of Manufacturing Science and Engineering 128(4):969–977
33. Holland JH (1986) Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. In: Machine Learning: An Artificial Intelligence Approach, vol II, William Kaufmann, pp 593–623
34. Holmes G, Donkin A, Witten IH (1994) WEKA: A Machine Learning Workbench. In: Proceedings of the Second Australia and New Zealand Conference on Intelligent Information Systems, IEEE Computer Society Press, pp 357–361
35. Hsu PL, Lai R, Chiu CC (2003) The Hybrid of Association Rule Algorithms and Genetic Algorithms for Tree Induction: An Example of Predicting the Student Course Performance. Expert Systems with Applications – An International Journal

25(1):51–62
36. Jabeen H, Baig AR (2010) Review of Classification using Genetic Programming. International Journal of Engineering Science and Technology 2(2):94–103
37. Kharbat F, Bull L, Odeh M (2007) Mining Breast Cancer Data with XCS. In: Proceedings of 9th Genetic and Evolutionary Computation Conference, ACM Press, pp 2066–2073
38. Koza JR (1990) Concept Formation and Decision Tree Induction using the Genetic Programming Paradigm. In: Proceedings of the 1st Workshop on Parallel Problem Solving from Nature, Springer, Lecture Notes in Computer Science, vol 496, pp 124–128
39. Koza JR (1992) Genetic Programming: On the Programming of Computers by Means of Natural Selection. Bradford Books, MIT Press
40. Li J (2001) FGP: A Genetic Programming Based Tool for Financial Forecasting. PhD thesis, University of Essex
41. Li J, Li X, Yao X (2005) Cost-Sensitive Classification with Genetic Programming. In: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE Computer Society, pp 2114–2121
42. Liu TY, Yang Y, Wan H, Zeng HJ, Chen Z, Ma WY (2005) Support Vector Machines Classification with a Very Large-Scale Taxonomy. ACM SIGKDD Explorations Newsletter 7(1):36–43
43. Martin WN, Lienig J, Cohoon JP (1997) Island (Migration) Models: Evolutionary Algorithms Based on Punctuated Equilibria. In: Handbook of Evolutionary Computation, Computational Intelligence Library, Oxford University Press, chap C6.3, pp 448–463
44. Mehta M, Agrawal R, Rissanen J (1996) SLIQ: A Fast Scalable Classifier for Data Mining. In: Advances in Database Technology – 5th International Conference on Extending Database Technology, Springer, Lecture Notes in Computer Science, vol 1057, pp 18–32
45. Muni DP, Pal NR, Das J (2004) A Novel Approach to Design Classifiers using Genetic Programming. IEEE Transactions on Evolutionary Computation 8(2):183–196
46. Orriols-Puig A, Bernadó-Mansilla E (2009) Evolutionary Rule-Based Systems for Imbalanced Data Sets. Soft Computing 13(3):213–225
47. Orriols-Puig A, Casillas J, Bernadó-Mansilla E (2008) Genetic-Based Machine Learning Systems are Competitive for Pattern Recognition. Evolutionary Intelligence 1(3):209–232
48. Orriols-Puig A, Casillas J, Bernadó-Mansilla E (2009) Fuzzy-UCS: a Michigan-style Learning Fuzzy-Classifier System for Supervised Learning. IEEE Transactions on Evolutionary Computation 13(2):260–283
49. Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann Series in Machine Learning, Morgan Kaufmann Publishers
50. Rastogi R, Shim K (1998) PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning. In: Proceedings of 24rd International Conference on Very Large Data Bases, Morgan Kaufmann Publishers, pp 404–415
51. Schapire RE (1990) The Strength of Weak Learnability. Machine Learning 5:197–227
52. Shafer JC, Agrawal R, Mehta M (1996) SPRINT: A Scalable Parallel Classifier for Data Mining. In: Proceedings of 22th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers, pp 544–555

53. Siegel S, Castellan Jr NJ (1956) Nonparametric Statistics for The Behavioral Sciences. Humanities/Social Sciences/Languages, McGraw-Hill
54. Sir Fisher RA (1936) The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics 7:179–188
55. Smith SF (1980) A Learning System based on Genetic Adaptive Algorithms. PhD thesis, University of Pittsburgh
56. Spears WM, De Jong KA (1990) Using Genetic Algorithms for Supervised Concept Learning. In: Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence, EEE Computer Society Press, pp 335–341
57. Stefanowski J, Slowinski K (1996) Rough Sets as a Tool for Studying Attribute Dependencies in the Urinary Stones Treatment Data Set. In: Rough Sets and Data Mining: Analysis of Imprecise Data, Kluwer Academic Publishers, pp 177–196
58. Tanwani AK, Farooq M (2009) Performance Evaluation of Evolutionary Algorithms in Classification of Biomedical Datasets. In: Proceedings of the 11th Annual Conference – Companion on Genetic and Evolutionary Computation Conference, ACM, pp 2617–2624
59. Tapia JJ, Morett E, Vallejo EE (2009) A Clustering Genetic Algorithm for Genomic Data Mining. In: Foundations of Computational Intelligence – Volume 4: Bio-Inspired Data Mining, Studies in Computational Intelligence, vol 204, Springer, pp 249–275
60. Tsang EPK, Butler JM, Li J (1998) EDDIE Beats the Bookies. International Journal of Software, Practice and Experience 28(10):1033–1043
61. Tsang EPK, Li J, Markose SM, Er H, Salhi A, Iori G (2000) EDDIE in Financial Decision Making. Journal of Management and Economics 4(4)
62. Tsang EPK, Yung P, Li J (2004) EDDIE-Automation – A Decision Support Tool for Financial Forecasting. Decision Support Systems 37(4):559–565
63. van Veldhuizen DA, Merkle LD (1999) Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD thesis, Air University, Air Force Institute of Technology: Wright-Patterson Air Force Base, OH, USA
64. Wang P, Weise T, Chiong R (2011) Novel evolutionary algorithms for supervised classification problems: An experimental study. Evolutionary Intelligence 4(1):3–16, DOI 10.1007/s12065-010-0047-7
65. Weise T (2009) Evolving Distributed Algorithms with Genetic Programming. PhD thesis, University of Kassel, Distributed Systems Group: Kassel, Germany
66. Weise T (2009) Global Optimization Algorithms – Theory and Application. URL `http://www.it-weise.de/`
67. Weise T, Geihs K (2006) DGPF – An Adaptable Framework for Distributed Multi-Objective Search Algorithms Applied to the Genetic Programming of Sensor Networks. In: Proceedings of the Second International Conference on Bioinspired Optimization Methods and their Applications, Jožef Stefan Institute, Informacijska Družba, pp 157–166
68. Wilcoxon F (1945) Individual Comparisons by Ranking Methods. Biometrics Bulletin 1(6):80–83
69. Wolberg WH, Mangasarian O (1989) Breast Cancer Wisconsin (Original) Data Set. UCI Machine Learning Repository, University of California
70. Zhang GP (2000) Neural Networks for Classification: A Survey. IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews 30(4):451–462

71. Zitzler E, Deb K, Thiele L (2000) Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2):173–195


```
@article{WWC2011NEAFSCPAES,
  author    = {Pu Wang and Thomas Weise and Raymond Chiong},
  title     = {Novel Evolutionary Algorithms for Supervised
               Classification Problems: An Experimental Study},
  journal   = {Evolutionary Intelligence},
  pages     = {3--16},
  month     = mar,
  volume    = {4},
  number    = {1},
  year      = {2011},
  publisher = {Springer-Verlag GmbH},
  address   = {Berlin, Germany},
  doi       = {10.1007/s12065-010-0047-7},
}
```