

Genetic-Based Synthetic Data Sets for the Analysis of Classifiers Behavior

Núria Macià, Albert Orriols-Puig, and Ester Bernadó-Mansilla
Grup de Recerca en Sistemes Intel·ligents
Enginyeria i Arquitectura La Salle - Universitat Ramon Llull
Quatre Camins, 2 08022, Barcelona (Spain)
{nmacia,aorriols,esterb}@salle.url.edu

Abstract

In this paper, we highlight the use of synthetic data sets to analyze learners behavior under bounded complexity. We propose a method to generate synthetic data sets with a specific complexity, based on the length of the class boundary. We design a genetic algorithm as a search technique and find it useful to obtain class labels according to the desired complexity. The results show the suitability of the genetic algorithm as a framework to provide artificial benchmark problems that can be further enriched with the use of multi-objective and niching strategies.

1. Introduction

Usually, learner performance is evaluated in terms of accuracy, interpretability, and efficiency among others. The first one of these measures is the most significant to determine the learner quality and is usually estimated by testing the learners over different data sets. In the last few years, the UCI Repository [2], which offers a large set of real-world problems, has become the most used. Nevertheless, previous studies have shown that classifier accuracy depends both on the constraints of the method and the intrinsic complexities of the data [3]. Thus, the estimation of the algorithm accuracy is obviously biased by the complexity of the data set itself. The limitation of this approach is that we never know whether the maximum achievable accuracy rate is reached or there is still place for performance improvements. The comparison of algorithm performance is also misled if the practitioner does not know the characteristics of the problem under study. Trying to identify these characteristics is not trivial, since there are many difficulties mixed up in real-world problems, such as inconsistency, uncertainty, missing values, and sampling sparsity.

One accurate approach to study classifier behavior and extract meaningful insights is to use artificial data sets, carefully designed to test the algorithm under known scenarios.

With the use of artificial data sets, we can define problems with different levels of complexity including individual or compound difficulties and find types of problems that fall in the domain of competence of a particular learner as well as identify adversarial problems to which the learner performance can still be improved. This can provide a framework for new-generation benchmark problems to test the different algorithms performance.

The distribution of classes on the feature space has been identified as one of the critical dimensions of data complexity [4]. Previous studies have defined a set of measures that try to quantify such class geometry complexities [11]. In particular, the length of class boundary measured over the data set has been shown very correlated with some classifiers error [5]. For this reason, we believe that the generation of data sets using this complexity indicator can provide us with useful benchmark problems. In a preliminary study [13], we presented a method for the generation of synthetic data sets based on this measure. The method consisted in labeling the instances of the data set according to the specified complexity. However, since many possible labelings were available for a given data set, an exhaustive search was not considered. Instead, we used an heuristic search that could guarantee that neither the demanded complexity was achieved nor a collection of diverse solutions was obtained. In order to overcome the identified problems, in this paper we propose the use of *genetic algorithms* (GAs) [12] to guide the search and generate synthetic data sets based on the boundary length. This technique gives much enhanced flexibility to search in the space of possible solutions. Although this paper is restricted to the design of a simple GA, the results hold promise for enhancing the framework including niching algorithms and multi-objective strategies to obtain diverse solutions with several mixed complexities.

The remainder of the paper is organized as follows. Section 2 briefly reviews the analysis of data complexity and focuses on one of complexity measures that estimates the data class separability. Then, we introduce how to generate synthetic data sets using this complexity measure. Section

3 describes in detail the design of the GA and its implementation. In section 4, we test the suitability of this approach to obtain artificial data sets and we analyze to what extent these data sets are useful to study the learners performance. Section 5 discusses the limitations of this work and suggests some future research lines. And finally, Section 6 presents the conclusions.

2. Data Complexity

Since some authors observed that the learners performance is strongly data dependent, the analysis of data complexity has taken more relevance to understand this relationship, know how to overcome this dependence, and improve the learner accuracy. Recently, several authors have proposed different measures to characterize the difficulty of problems. We highlight the work of Ho and Basu [11] in which they gathered and proposed a set of descriptors that characterized different aspects of complexity: (i) the discriminative power of attributes, (ii) the separability of classes, (iii) the geometry of manifolds expanded by each of the classes, and (iv) the sparsity. Measures of separability of classes were the most significant ones to characterize learners performance [5] and seemed helpful for classifier selection. In particular, the *length of the class boundary* achieved high correlations with several algorithms performance.

The length of the class boundary is a measure of class separability that indicates the density of points located near to the class boundary. To compute its value for a given data set, a *minimum spanning tree* (MST) is built with all the points contained in the data set where the distance of the edges corresponds to the Euclidean distance between the connected points. Then, the number of edges that join pairs of points belonging to opposite classes is counted. This value is divided by the total number of connections, and the resulting ratio is considered as an estimate of boundary length. Highly interleaved or randomly labeled data usually provide large boundary lengths. In contrast, in data sets where classes are well separated, the length of the class boundary tends to be small.

As we mentioned previously, we hypothesize that complexity measures can also be useful to create data sets with known properties. Based on this hypothesis, we present a technique to generate synthetic data sets with specific complexities. Firstly, we have to determine the following parameters: the number of features m , the number of instances n , and the desired complexity b , which corresponds to the length of class boundary. After setting these parameters, we generate n points with the values of the attributes distributed uniformly in the m -dimensional space. Then, we construct the MST and label the class of each point to achieve the specified boundary b . If we assume two-class

problems, we observe that, for a given MST and a boundary b , there are $2^{\binom{n-1}{(n-1)-p}}$ different labelings, where p is the number of edges that connect points of different classes, i.e., $p = b \cdot (n - 1)$ and $b \in [0, 1]$. By varying the values of m , n , and b , we can build a collection of data sets with different levels of difficulty with respect to the class separability. However, as the number of instances increases, the number of possible labelings grows exponentially, and there are more intermediate complexity levels. For instance, for a data set with 5 instances, there are 32 labelings and 4 complexity levels $\{0, 1/4, 2/4, 3/4, 1\}$, i.e. $n - 1$. In contrast, a data set composed of 1001 instances has 2^{1001} labelings scattered in 1000 different complexity levels. For this reason, we propose the use of a GA that searches for the class labeling according to the desired complexity. The next section provides the details.

3. Generating Boundedly-Difficult Data Sets with Genetic Algorithms

In the previous section, we highlighted the importance of generating *boundedly-difficult problems* to test the behavior of learning techniques and compare their performance with the one presented by other learning systems. This led to the definition of an optimization problem in which data sets with a specific boundary complexity are searched in the problem—i.e., data set—space. The size of the solution space of this optimization problem increases exponentially with the number of input instances, making traditional exhaustive search mechanisms not suitable for generating medium- or large-size data sets with a given boundary complexity [13]. Here, we design a generational GA [8] with a fixed-size population to tackle the problem of generating data sets with a boundary complexity b_{obj} defined by the user.

The GA already departs from a set of n instances, where each instance is defined by m continuous attributes. These m attributes are randomly initialized with a value ranging in the interval $[0,1]$. Note that the instances are not labeled. The goal of the GA is to label these instances according to the desired complexity b_{obj} . Therefore, the search space consists of all the possible combinations of classes; that is, new instances are not generated during the learning process of the GA.

As follows, we explain the knowledge representation used to codify the data set solutions and the design of the different genetic operators.

3.1. Knowledge Representation

The GA evolves a population of N individuals, in which each individual is a candidate solution to the optimization problem. An individual is represented with a k -ary string

of length n , in which the bit i stores the class of the i th instance of the data set (k is a configuration parameter that indicates the maximum number of classes in the generated data sets). Therefore, the individual gives a label to each one of the examples in the data set, and the GA searches for the best combination of labels that yields a given complexity. Then, the phenotype of a genotypical representation is obtained by labeling each instance of the training data set.

3.2. Design of the Genetic Operators

In our implementation, we used a generational GA with a population of fixed size N that works as follows. At the beginning, the population is randomly initialized and the resulting individuals are evaluated. Then, the GA bases the search on the interaction of three primary genetic operators that are iteratively applied: *selection*, *crossover*, and *mutation*. Simply stated, selection chooses K parents, allocating more offspring to better individuals. Then, pairs of these parents are selected without replacement, and they undergo crossover and mutation with probabilities χ and μ respectively. If neither operator is applied, the offsprings are an exact copy of the parents. This results in another population of candidate solutions. Then, these solutions are evaluated and the whole new population replaces the older one.

More specifically, we used the following genetic operators: (i) s -wise tournament selection, where tournaments of s randomly chosen parents are held, and the best parent is selected for reproduction; (ii) two-point crossover, which, provided two parents, randomly generates two cut points and uses them to shuffle the information of both parents; and (iii) bit-wise mutation, which flips the value of the bit selected for mutation, i.e., when generating multi-class data sets, bit-wise mutation randomly generates one of the possible classes and assigns the new value to the mutated bit. To evaluate each new individual, we used the MST generated with the meta-level information. The process to obtain the fitness of the individual is as follows. We label the instances of the data set according to the information of the given individual and count the ratio of the number of edges of the MST that connect instances of different classes to the total number of edges. This results in the boundary complexity b_i of the candidate solution i . Then, the fitness of this individual is computed as $fitness_i = |b_{obj} - b_i|$, where b_{obj} is the user-defined boundary complexity. Notice that this strategy prevents the system from having to build a MST for each rule evaluation, which would be too much time-consuming, specially when optimizing large data sets.

4. Experiments and Results

The experiments are divided into two parts. Firstly, we generated synthetic data sets using the GA to optimize the

length of the class boundary and analyzed the suitability of the GA for this application. Secondly, we tested different classifier systems on our synthetic test bed to analyze their performance with respect to this complexity measure.

With regard to the synthetic data sets generation, a difficulty found is how to adjust the GA's parameters such as population size, maximum number of generations, and crossover and mutation rates. We performed several runs of the GA with different population sizes to identify which configurations converged to a solution. Taking into account these previous adjustments, we scaled the population size according to the number of instances of the data set, and the number of generations according to the complexity level. In the test application, we determined a crossover rate of 0.8 and the mutation rate was defined as $1/n$ where n is the individual size. We decided to analyze binary class discriminant problems whose instances are described by continuous values because the length of the class boundary was initially designed for this type of attributes. Nonetheless, our approach is valid for any type of attributes as it suffices to adapt the distance function of MST's algorithm to be able to manage categorical attributes. On the other hand, although we focused the study on binary class problems, the k -ary representation of the GA allows us to easily extend the study to multiple classes. Yet, it is still an open issue to determine whether the analysis of data should be performed directly over multiple classes or on the contrary over two classes by transforming data sets that do not meet this property. Moreover, although class distribution affects classifier accuracy, it is not introduced as a restriction in the generation. Anyway, *a posteriori* analysis shows that 91% of data sets have a class balance of 40-60%.

For each complexity level from 0.2 to 0.8, we generated data sets composed of 101 and 501 instances both with 2 attributes, and we ran these experiments 50 and 10 times respectively. We evolved populations of 500 and 1000 individuals during 1000 generations. Table 1 shows some statistics that describe the obtained populations. We represent the generation in which the GA reaches the first solution, the number of solutions contained in the population after iterating the predefined number of generations, and the real number of solutions with a different structure. Surprisingly, data sets with intermediate complexity are found in the early generations or generation 0 (see for example, boundary lengths of 0.4 and 0.6 with 101 instances). This is due to the random initialization of individuals, which tends to provide medium complexity around 0.4 and 0.6. Although the GA could be stopped in the early generations, we preferred to continue evolving the population with the aim of enlarging the set with structurally different solutions.

Besides, Table 1 shows that the number of different solutions is smaller than the number of solutions. This is due to two possible reasons. The first one is that the fi-

Table 1. Statistics of the populations obtained by the genetic algorithm.

# Instances	Population size	Runs		0.2	0.4	0.6	0.8
101	500	50	First solution in generation	22.82	0	0	22.24
			Solutions (%)	10.43	18.73	18.89	10.67
			Different solutions (%)	9.04	16.93	17.29	9.26
501	1000	10	First solution in generation	85.5	7.7	8.2	85.4
			Solutions (%)	9.41	15.05	14.24	10.82
			Different solutions (%)	8.46	13.90	13.06	9.73

nal population of the GA contained several individuals representing the same solution. This is usual in GAs; even though there is a diverse set of solutions equivalently good, the bias of the selection mechanisms leads one of the solutions to take over part of the population. If all solutions are to be obtained, then explicit niching algorithms should be included. The second reason of having fewer different solutions than individuals is that different individual codifications can represent the same solution. That is, there are two possible codifications per solution. For example, individuals {0010, 1101} are equivalent.

Another property of this optimization problem is that it is a combinatorial problem with a symmetry centered in the medium complexity. It means that the opposite extremes of boundary levels have the same number of different labelings. For this reason, we observe that the percentage of solutions between the complexities 0.2 and 0.8, and between 0.4 and 0.6 are very similar. Furthermore, as we move away from the medium complexity, there are less labelings in each degree of complexity. Hence, the percentage of achieved solutions for complexities 0.2 and 0.8 is lower than for complexities 0.4 and 0.6.

In the second part of the experimentation, we ran three classifier systems over a large test bed. To avoid correlations due to similarities between the learning process of the classifier and the complexity measure, we selected learners from different paradigms such as the C4.5 induction tree [15], the Naïve Bayes probabilistic classifier [7], and the SMO support vector machine [14]. We estimated the accuracy of classifiers using a 10-fold cross-validation procedure [6]. The algorithms were run with the software Weka [18] with their default configuration.

We synthetically generated a collection composed of more than 15000 data sets. However, this amount is not distributed evenly among the different complexity levels. So, in order to obtain comparable results, we must perform the analysis on the same number of data sets, which is restricted to the complexity level that gets the minimum number of data sets. In the case of 101 instances, this value was 9.044%, so we randomly selected 8% of the data sets for each complexity level, which corresponded to 2000 data sets. We proceeded in this way for 501 instances, where the

lower threshold was 8%, resulting in 800 data sets. Figure 1 depicts the classification accuracy obtained by the classifiers. The x-axis represents the length of the class boundary and the y-axis represents the accuracy rate. The three upper plots refer to data sets containing 101 instances and the lower ones to data sets containing 501 instances. For each complexity level b , a box plot shows the range of values of the accuracy rate obtained by the classifier.

Note that although the three classifiers belong to different learning schemes, they tend to behave similarly. The higher the complexity level, the lower the accuracy rate. This indicates that the length of the class boundary is a dimension of complexity that negatively influences the classifier performance. This metric is a significant estimate to predict the classifier accuracy. Nevertheless, we appreciate a considerable variability in the obtained accuracy rate. Therefore, we should consider that the length of the class boundary, even though it is able to explain significantly the classifier error, is not enough to characterize the difficulty of the problem. As a further work, we should combine this metric with other meaningful dimensions of complexity to see whether the classifier behavior can be better explained. Since this measure provides an estimation of the class separability, it would be interesting to use measures that evaluate other aspects such as the discriminative power of features or the geometry of manifolds. Also, it would be useful to identify the minimum set of independent measures that are able to describe the whole complexity space of real-world problems.

5. Discussion

The experimental results presented in the previous section showed that the designed approach holds promise, being able to generate data sets with different complexities which could not be obtained with other search mechanisms [13]. Along with the many strengths of the method, we also want to draw some observations that will guide our future work. These observations are mainly related to: (i) the efficiency and the scalability of our proposal; (ii) the capacity of our approach to generate data sets that satisfy multi-

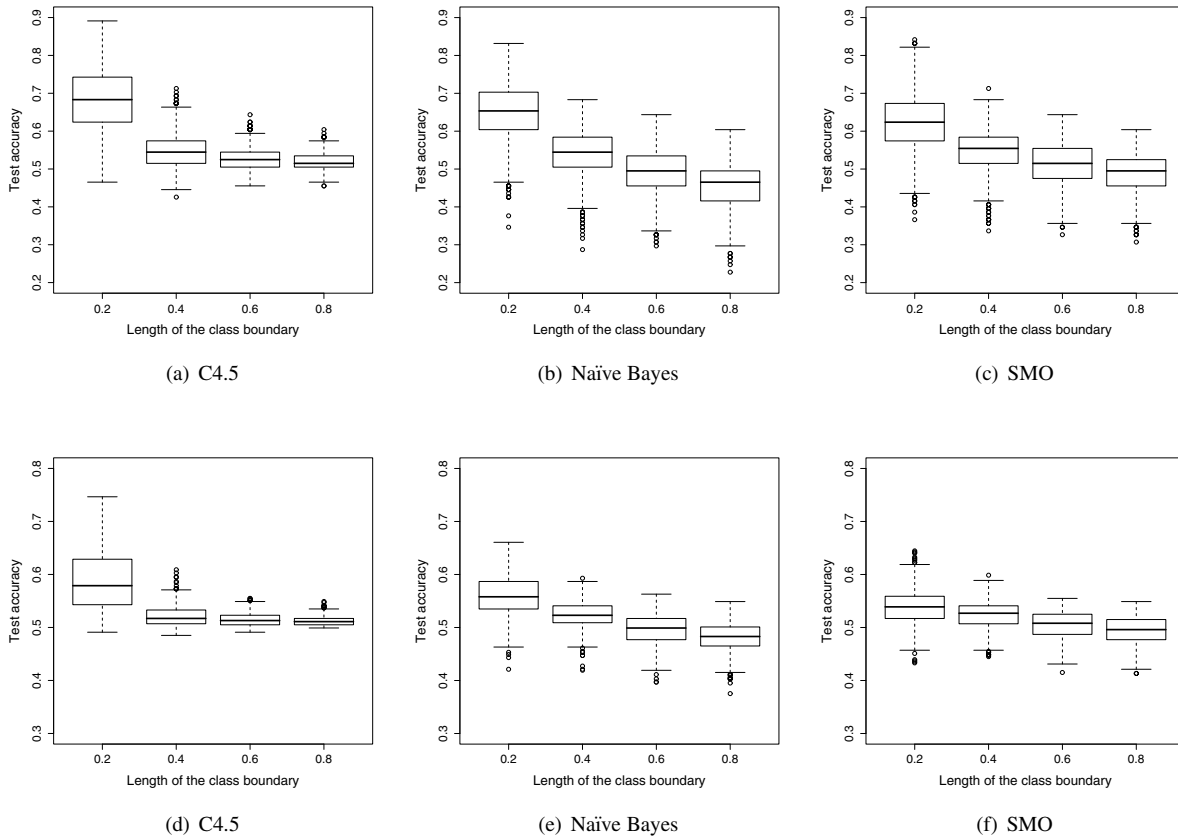


Figure 1. Accuracy of classifiers with respect to genetic-based artificial data sets with increasing levels of *boundary length*. Plots (a), (b), and (c) correspond to data sets with 101 instances and plots (d), (e), and (f) belong to data sets with 501 instances.

ple criteria; and (iii) the structure of the resulting data sets and similarity of this structure with real-world classification problems.

The first important aspect worth mentioning is how our approach scales as the size of the data sets increases. The method proposed in this paper relies on a simple GA, using an elemental implementation that does not expressly optimize the cost of the algorithm. The main purpose of the paper was to define an accurate knowledge representation that enabled the creation of simple, accurate genetic operators, and show that this simple GA was able to generate data sets with a given boundary difficulty. Having empirically demonstrated the performance of the algorithm in this respect, we now can move our implementation to a more efficient set of GA implementations, which are usually referred to as *competent GAs* [9]. Moreover, efficiency enhancement techniques [17] and evaluation relaxation algorithms [16] could be used to speed-up our proposal. Considering all

these approaches, competent GAs have been shown to be able to solve huge optimization problems that contain up to a billion variables [10], which promotes their application to generate large data sets.

A second important aspect relates to the variability that has been observed in the accuracy of the learners for different data sets with the same boundary complexity. As already announced in [11, 5], this indicates that the descriptive power of the boundary complexity may not be enough to describe the complexity of a classification problem. In [11], the authors proposed to define a minimum set of measures that permitted to define the global complexity of a method. Herein, we could easily move our problem to a multi-objective optimization problem, since GA, differently from many other optimization techniques, is a natural support for solving optimization problems with several objectives [1].

Finally, the third aspect is associated with the structure

of the resulting solutions and its similarity with real-world classification problems. Ideally, we would expect to run a single iteration of a GA and obtain a set of different solutions that have the boundary complexity specified by the user; moreover, the structure of these solutions would resemble the structure of real-world classification problems. Evolving a population with diverse structural solutions can be easily achieved by informing the crossover and the mutation operators—or even creating new operators—to prevent the system from generating similar solutions. We pursue to analyze the relationship between synthetic data sets and real-world problems to know whether these two types of problems are comparable through the length of boundary. If this is indeed so, and the classifier behavior is similar in both synthetic data sets and real-world problems with the same boundary complexity, we will be capable of evaluating the quality of the learners with synthetic data sets and providing a set of benchmark problems. Forcing the solutions to resemble real-world problems is a complex task, since it involves defining the typical problem structures of real-world problems. Nonetheless, this objective could be partially achieved by generating the initial data set according to fixed distributions, and fixing the class of some of the instances. As further work, we will take this approach to generate data sets with a partially defined structure and that have a certain value for one or several complexity metrics.

6. Conclusions

Synthetic data sets are needed to provide a controlled scenario where we study learners behavior on problems with certain complexities. Guiding the generation of data using complexity measures is a way to obtain problems with particular properties to evaluate the learners' limitations. In this work, we proposed the generation of synthetic data sets based on data complexity. Principally, we focused on the length of the class boundary and found that it is a dominant factor to assess the complexity of the data set. The GA allowed us to deal with a preliminary approach and to achieve data sets with any demanded boundary length. However, future work pretends to benefit from the GA's ability as a multi-objective technique and points out the generation of synthetic data set using different complexity descriptors at the same time.

Acknowledgements

The authors would like to thank *Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, the Ministerio de Educación y Ciencia* for its support under project TIN2005-08386-C05-04, *Generalitat de Catalunya* for its support under grants 2005FI-00252 and 2005SGR-00302, and the *Govern d'Andorra* for its research grant.

References

- [1] A. Abraham, L. Jain, and R. Goldberg, editors. *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*. Springer, 2006.
- [2] A. Asuncion and D. Newman. UCI machine learning repository. University of California, Irvine, 2007. <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
- [3] M. Basu and T. K. Ho. *Data Complexity in Pattern Recognition*. Springer-Verlag, 2006.
- [4] E. Bernadó-Mansilla and T. K. Ho. On classifier domains of competence. In *the 17th Int. Conf. on Pattern Recognition*, volume 1, pages 136–139. IEEE-CS, 2004.
- [5] E. Bernadó-Mansilla and T. K. Ho. Domain of competence of xcs classifier system in complexity measurement space. *IEEE-TEC*, 9(1):82–104, 2005.
- [6] T. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.
- [7] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.
- [8] D. E. Goldberg. *Genetic algorithms in search, optimization & machine learning*. Addison Wesley, 1989.
- [9] D. E. Goldberg. *The design of innovation: Lessons from and for competent genetic algorithms*. Kluwer Academic Publishers, 2002.
- [10] D. E. Goldberg, K. Sastry, and X. Llorà. Toward routine billion-variable optimization using genetic algorithms: Short communication. *Complexity*, 12(3):27–29, 2007.
- [11] T. K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 24(3):289–300, 2002.
- [12] J. H. Holland. *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975.
- [13] N. Macià, E. Bernadó-Mansilla, and A. Orriols-Puig. Preliminary approach on synthetic data sets generation based on class separability measure. In *the 19th Int. Conf. on Pattern Recognition (In press)*, 2008.
- [14] J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods: Support Vector Learning*, pages 185–208. MIT Press, 1998.
- [15] J. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers, 1993.
- [16] K. Sastry, C. F. Lima, and D. E. Goldberg. Evaluation relaxation using substructural information and linear estimation. In *Proc. of the Genetic and Evolutionary Computation Conference*, pages 419–426, 2006.
- [17] K. Sastry, M. Pelikan, and D. E. Goldberg. Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. In *Congress on Evolutionary Computation*, volume 1, pages 720–727, 2004.
- [18] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.