A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery

Alex A. Freitas

Postgraduate Program in Computer Science, Pontificia Universidade Catolica do Parana
Rua Imaculada Conceicao, 1155. Curitiba - PR. 80215-901. Brazil.
E-mail: alex@ppgia.pucpr.br Web page: http://www.ppgia.pucpr.br/~alex

Abstract: This chapter discusses the use of evolutionary algorithms, particularly genetic algorithms and genetic programming, in data mining and knowledge discovery. We focus on the data mining task of classification. In addition, we discuss some preprocessing and postprocessing steps of the knowledge discovery process, focusing on attribute selection and pruning of an ensemble of classifiers. We show how the requirements of data mining and knowledge discovery influence the design of evolutionary algorithms. In particular, we discuss how individual representation, genetic operators and fitness functions have to be adapted for extracting high-level knowledge from data.

1. Introduction

The amount of data stored in databases continues to grow fast. Intuitively, this large amount of stored data contains valuable hidden knowledge, which could be used to improve the decision-making process of an organization. For instance, data about previous sales might contain interesting relationships between products and customers. The discovery of such relationships can be very useful to increase the sales of a company. However, the number of human data analysts grows at a much smaller rate than the amount of stored data. Thus, there is a clear need for (semi-)automatic methods for extracting knowledge from data.

This need has led to the emergence of a field called data mining and knowledge discovery [66]. This is an interdisciplinary field, using methods of several research areas (specially machine learning and statistics) to extract high-level knowledge from real-world data sets. Data mining is the core step of a broader process, called knowledge discovery in databases, or knowledge discovery, for short. This process includes the application of several preprocessing methods aimed at facilitating the application of the data mining algorithm and postprocessing methods aimed at refining and improving the discovered knowledge.

This chapter discusses the use of evolutionary algorithms (EAs), particularly genetic algorithms (GAs) [29], [47] and genetic programming (GP) [41], [6], in

data mining and knowledge discovery. We focus on the data mining task of classification, which is the task addressed by most EAs that extract high-level knowledge from data. In addition, we discuss the use of EAs for performing some preprocessing and postprocessing steps of the knowledge discovery process, focusing on attribute selection and pruning of an ensemble of classifiers.

We show how the requirements of data mining and knowledge discovery influence the design of EAs. In particular, we discuss how individual representation, genetic operators and fitness functions have to be adapted for extracting high-level knowledge from data.

This chapter is organized as follows. Section 2 presents an overview of data mining and knowledge discovery. Section 3 discusses several aspects of the design of GAs for rule discovery. Section 4 discusses GAs for performing some preprocessing and postprocessing steps of the knowledge discovery process. Section 5 addresses the use of GP in rule discovery. Section 6 addresses the use of GP in the preprocessing phase of the knowledge discovery process. Finally, section 7 presents a discussion that concludes the chapter.

2. An Overview of Data Mining and Knowledge Discovery

This section is divided into three parts. Subsection 2.1 discusses the desirable properties of discovered knowledge. Subsection 2.2 reviews the main data mining tasks. Subsection 2.3 presents an overview of the knowledge discovery process.

2.1 The Desirable Properties of Discovered Knowledge

In essence, data mining consists of the (semi-)automatic extraction of knowledge from data. This statement raises the question of what kind of knowledge we should try to discover. Although this is a subjective issue, we can mention three general properties that the discovered knowledge should satisfy; namely, it should be accurate, comprehensible, and interesting. Let us briefly discuss each of these properties in turn. (See also section 3.3.)

As will be seen in the next subsection, in data mining we are often interested in discovering knowledge which has a certain predictive power. The basic idea is to predict the value that some attribute(s) will take on in “the future”, based on previously observed data. In this context, we want the discovered knowledge to have a high predictive accuracy rate.

We also want the discovered knowledge to be comprehensible for the user. This is necessary whenever the discovered knowledge is to be used for supporting a decision to be made by a human being. If the discovered “knowledge” is just a black box, which makes predictions without explaining them, the user may not trust it [48]. Knowledge comprehensibility can be achieved by using high-level knowledge representations. A popular one, in the context of data mining, is a set of IF-THEN (prediction) rules, where each rule is of the form:

IF <some_conditions_are_satisfied>
THEN <predict_some_value_for_an_attribute>

The third property, knowledge interestingness, is the most difficult one to define and quantify, since it is, to a large extent, subjective. However, there are some aspects of knowledge interestingness that can be defined in objective terms. The topic of rule interestingness, including a comparison between the subjective and the objective approaches for measuring rule interestingness, will be discussed in section 2.3.2.

2.2 Data Mining Tasks

In this section we briefly review some of the main data mining tasks. Each task can be thought of as a particular kind of problem to be solved by a data mining algorithm. Other data mining tasks are briefly discussed in [18], [25].

2.2.1 Classification. This is probably the most studied data mining task. It has been studied for many decades by the machine learning and statistics communities (among others). In this task the goal is to predict the value (the class) of a user-specified goal attribute based on the values of other attributes, called the predicting attributes. For instance, the goal attribute might be the *Credit* of a bank customer, taking on the values (classes) "good" or "bad", while the predicting attributes might be the customer's *Age*, *Salary*, *Current_account_balance*, whether or not the customer has an *Unpaid Loan*, etc.

Classification rules can be considered a particular kind of prediction rules where the rule antecedent ("IF part") contains a combination - typically, a conjunction - of conditions on predicting attribute values, and the rule consequent ("THEN part") contains a predicted value for the goal attribute. Examples of classification rules are:

IF (*Unpaid_Loan?* = "no") and (*Current_account_balance* > $3,000)
THEN (*Credit* = "good")
IF (*Unpaid_Loan?* = "yes") THEN (*Credit* = "bad")

In the classification task the data being mined is divided into two mutually exclusive and exhaustive data sets, the training set and the test set. The data mining algorithm has to discover rules by accessing the training set only. In order to do this, the algorithm has access to the values of both the predicting attributes and the goal attribute of each example (record) in the training set.

Once the training process is finished and the algorithm has found a set of classification rules, the predictive performance of these rules is evaluated on the *test* set, which was not seen during training. This is a crucial point.

Actually, it is trivial to get 100% of predictive accuracy in the training set by completely sacrificing the predictive performance on the test set, which would be useless. To see this, suppose that for a training set with n examples the data mining algorithm "discovers" n rules, i.e. one rule for each training example, such

that, for each "discovered" rule: (a) the rule antecedent contains conditions with exactly the same attribute-value pairs as the corresponding training example; (b) the class predicted by the rule consequent is the same as the actual class of the corresponding training example. In this case the "discovered" rules would trivially achieve a 100% of predictive accuracy on the training set, but would be useless for predicting the class of examples unseen during training. In other words, there would be no generalization, and the "discovered" rules would be capturing only idiosyncrasies of the training set, or just "memorizing" the training data. In the parlance of machine learning and data mining, the rules would be overfitting the training data.

For a comprehensive discussion about how to measure the predictive accuracy of classification rules, the reader is referred to [34], [67].

In the next three subsections we briefly review the data mining tasks of dependence modeling, clustering and discovery of association rules. Our main goal is to compare these tasks against the task of classification, since space limitations do not allow us to discuss these tasks in more detail.

2.2.2 Dependence Modeling. This task can be regarded as a generalization of the classification task. In the former we want to predict the value of several attributes - rather than a single goal attribute, as in classification. We focus again on the discovery of prediction (IF-THEN) rules, since this is a high-level knowledge representation.

In its most general form, any attribute can occur both in the antecedent ("IF part") of a rule and in the consequent ("THEN part") of another rule - but not in both the antecedent and the consequent of the same rule. For instance, we might discover the following two rules:

IF (*Current_account_balance* > $3,000) AND (*Salary* = "high")
THEN (*Credit* = "good")
IF (*Credit* = "good") AND (*Age* > 21) THEN (*Grant_Loan?* = "yes")

In some cases we want to restrict the use of certain attributes to a given part (antecedent or consequent) of a rule. For instance, we might specify that the attribute *Credit* can occur only in the consequent of a rule, or that the attribute *Age* can occur only in the antecedent of a rule.

For the purposes of this chapter we assume that in this task, similarly to the classification task, the data being mined is partitioned into training and test sets. Once again, we use the training set to discover rules and the test set to evaluate the predictive performance of the discovered rules.

2.2.3 Clustering. As mentioned above, in the classification task the class of a training example is given as input to the data mining algorithm, characterizing a form of supervised learning. In contrast, in the clustering task the data mining algorithm must, in some sense, "discover" classes by itself, by partitioning the examples into clusters, which is a form of unsupervised learning [19], [20].

Examples that are similar to each other (i.e. examples with similar attribute values) tend to be assigned to the same cluster, whereas examples different from each other tend to be assigned to distinct clusters. Note that, once the clusters are found, each cluster can be considered as a "class", so that now we can run a classification algorithm on the clustered data, by using the cluster name as a class label. GAs for clustering are discussed e.g. in [50], [17], [33].

2.2.4 Discovery of Association Rules. In the standard form of this task (ignoring variations proposed in the literature) each data instance (or "record") consists of a set of binary attributes called items. Each instance usually corresponds to a customer transaction, where a given item has a true or false value depending on whether or not the corresponding customer bought that item in that transaction. An association rule is a relationship of the form IF X THEN Y, where X and Y are sets of items and $X \cap Y = \emptyset$ [1], [2]. An example is the association rule:

IF *fried_potatoes* THEN *soft_drink*, *ketchup* .

Although both classification and association rules have an IF-THEN structure, there are important differences between them. We briefly mention here two of these differences. First, association rules can have more than one item in the rule consequent, whereas classification rules always have one attribute (the goal one) in the consequent. Second, unlike the association task, the classification task is asymmetric with respect to the predicting attributes and the goal attribute. Predicting attributes can occur only in the rule antecedent, whereas the goal attribute occurs only in the rule consequent. A more detailed discussion about the differences between classification and association rules can be found in [24].

2.3 The Knowledge Discovery Process

The application of a data mining algorithm to a data set can be considered the core step of a broader process, often called the knowledge discovery process [18]. In addition to the data mining step itself, this process also includes several other steps. For the sake of simplicity, these additional steps can be roughly categorized into data preprocessing and discovered-knowledge postprocessing.

We use the term data preprocessing in a general sense, including the following steps (among others) [55]:

(a) Data Integration - This is necessary if the data to be mined comes from several different sources, such as several departments of an organization. This step involves, for instance, removing inconsistencies in attribute names or attribute value names between data sets of different sources.

(b) Data Cleaning - It is important to make sure that the data to be mined is as accurate as possible. This step may involve detecting and correcting errors in the data, filling in missing values, etc. Data cleaning has a strong overlap with data integration, if this latter is also performed. It is often desirable to involve the user in data cleaning and data integration, so that (s)he can bring her/his background

knowledge into these tasks. Some data cleaning methods for data mining are discussed in [32], [59].

(c) Discretization - This step consists of transforming a continuous attribute into a categorical (or nominal) attribute, taking on only a few discrete values - e.g., the real-valued attribute *Salary* can be discretized to take on only three values, say "low", "medium", and "high". This step is particularly required when the data mining algorithm cannot cope with continuous attributes. In addition, discretization often improves the comprehensibility of the discovered knowledge [11], [52].

(d) Attribute Selection - This step consists of selecting a subset of attributes relevant for classification, among all original attributes. It will be discussed in subsection 2.3.1.

Discovered-knowledge postprocessing usually aims at improving the comprehensibility and/or the interestingness of the knowledge to be shown to the user. This step may involve, for instance, the selection of the most interesting rules, among the discovered rule set. This step will be discussed in subsection 2.3.2.

Note that the knowledge discovery process is inherently iterative, as illustrated in Fig. 1. As can be seen in this figure, the output of a step can be not only sent to the next step in the process, but also sent – as a feedback - to a previous step.

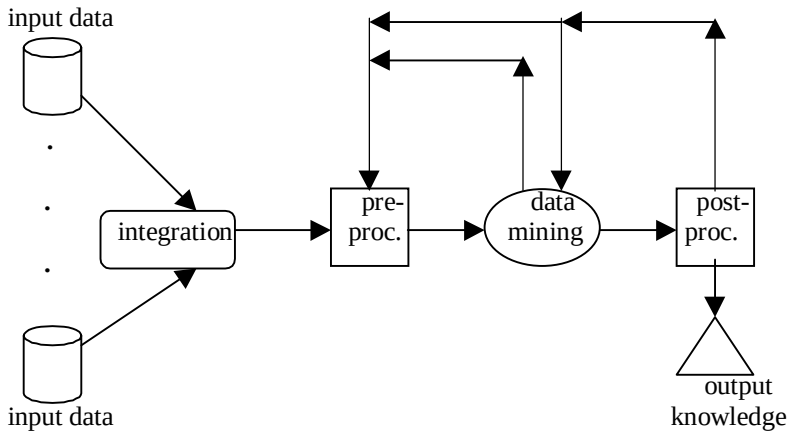

Fig. 1. An overview of the knowledge discovery process

2.3.1 Attribute Selection. This consists of selecting, among all the attributes of the data set, a subset of attributes relevant for the target data mining task. Note that a number of data mining algorithms, particularly rule induction ones, already perform a kind of attribute selection when they discover a rule containing just a few attributes, rather than all attributes. However, in this section we are interested in attribute selection as a preprocessing step for the data mining algorithm.

Hence, we first select an attribute subset and then give only the selected attributes for the data mining algorithm.

The motivation for this kind of preprocessing is the fact that irrelevant attributes can somehow "confuse" the data mining algorithm, leading to the discovery of inaccurate or useless knowledge [38]. Considering an extreme example, suppose we try to predict whether the credit of a customer is good or bad, and suppose that the data set includes the attribute *Customer_Name*. A data mining algorithm might discover too specific rules of the form: IF (*Customer_Name* = "a_specific_name") THEN (*Credit* = "good"). This kind of rule has no predictive power. Most likely, it covers a single customer and cannot be generalized to other customers. Technically speaking, it is *overfitting* the data. To avoid this problem, the attribute *Customer_Name* (and other attributes having a unique value for each training example) should be removed in a preprocessing step.

Attribute selection methods can be divided into filter and wrapper approaches. In the filter approach the attribute selection method is independent of the data mining algorithm to be applied to the selected attributes.

By contrast, in the wrapper approach the attribute selection method uses the result of the data mining algorithm to determine how good a given attribute subset is. In essence, the attribute selection method iteratively generates attribute subsets (candidate solutions) and evaluates their qualities, until a termination criterion is satisfied. The attribute-subset generation procedure can be virtually any search method. The major characteristic of the wrapper approach is that the quality of an attribute subset is directly measured by the performance of the data mining algorithm applied to that attribute subset.

The wrapper approach tends to be more effective than the filter one, since the selected attributes are "optimized" for the data mining algorithm. However, the wrapper approach tends to be much slower than the filter approach, since in the former a full data mining algorithm is applied to each attribute subset considered by the search. In addition, if we want to apply several data mining algorithms to the data, the wrapper approach becomes even more computationally expensive, since we need to run the wrapper procedure once for each data mining algorithm.

2.3.2 Discovered-Knowledge Postprocessing. It is often the case that the knowledge discovered by a data mining algorithm needs to undergo some kind of postprocessing. Since in this chapter we focus on discovered knowledge expressed as IF-THEN prediction rules, we are mainly interested in the postprocessing of a discovered rule set.

There are two main motivations for such postprocessing. First, when the discovered rule set is large, we often want to simplify it - i.e., to remove some rules and/or rule conditions - in order to improve knowledge comprehensibility for the user.

Second, we often want to extract a subset of interesting rules, among all discovered ones. The reason is that although many data mining algorithms were

designed to discover accurate, comprehensible rules, most of these algorithms were not designed to discover *interesting* rules, which is a rather more difficult and ambitious goal, as mentioned in section 2.1.

Methods for selection of interesting rules can be roughly divided into subjective and objective methods. Subjective methods are user-driven and domain-dependent. For instance, the user may specify rule templates, indicating which combination of attributes must occur in the rule for it to be considered interesting - this approach has been used mainly in the context of association rules [40]. As another example of a subjective method, the user can give the system a general, high-level description of his/her previous knowledge about the domain, so that the system can select only the discovered rules which represent previously-unknown knowledge for the user [44].

By contrast, objective methods are data-driven and domain-independent. Some of these methods are based on the idea of comparing a discovered rule against other rules, rather than against the user's beliefs. In this case the basic idea is that the interestingness of a rule depends not only on the quality of the rule itself, but also on its similarity to other rules. Some objective measures of rule interestingness are discussed in [26], [22], [23].

We believe that ideally a combination of subjective and objective approaches should be used to try to solve the very hard problem of returning interesting knowledge to the user.

3. Genetic Algorithms (GAs) for Rule Discovery

In general the main motivation for using GAs in the discovery of high-level prediction rules is that they perform a global search and cope better with attribute interaction than the greedy rule induction algorithms often used in data mining [14].

In this section we discuss several aspects of GAs for rule discovery. This section is divided into three parts. Subsection 3.1 discusses how one can design an individual to represent prediction (IF-THEN) rules. Subsection 3.2 discusses how genetic operators can be adapted to handle individuals representing rules. Section 3.3 discusses some issues involved in the design of fitness functions for rule discovery.

3.1 Individual Representation

3.1.1 Michigan versus Pittsburgh Approach. Genetic algorithms (GAs) for rule discovery can be divided into two broad approaches, based on how rules are encoded in the population of individuals ("chromosomes"). In the Michigan approach each individual encodes a *single* prediction rule, whereas in the Pittsburgh approach each individual encodes *a set of* prediction rules.

It should be noted that some authors use the term "Michigan approach" in a

narrow sense, to refer only to classifier systems [35], where rule interaction is taken into account by a specific kind of credit assignment method. However, we use the term "Michigan approach" in a broader sense, to denote any approach where each GA individual encodes a single prediction rule.

The choice between these two approaches strongly depends on which kind of rule we want to discover. This is related to which kind of data mining task we are addressing. Suppose the task is classification. Then we usually evaluate the quality of the rule set as a whole, rather than the quality of a single rule. In other words, the interaction among the rules is important. In this case, the Pittsburgh approach seems more natural.

On the other hand, the Michigan approach might be more natural in other kinds of data mining tasks. An example is a task where the goal is to find a small set of high-quality prediction rules, and each rule is often evaluated independently of other rules [49]. Another example is the task of detecting rare events [65].

Turning back to classification, which is the focus of this chapter, in a nutshell the pros and cons of each approach are as follows. The Pittsburgh approach directly takes into account rule interaction when computing the fitness function of an individual. However, this approach leads to syntactically-longer individuals, which tends to make fitness computation more computationally expensive. In addition, it may require some modifications to standard genetic operators to cope with relatively complex individuals. Examples of GAs for classification which follow the Pittsburgh approach are GABIL [13], GIL [37], and HDPDCS [51].

By contrast, in the Michigan approach the individuals are simpler and syntactically shorter. This tends to reduce the time taken to compute the fitness function and to simplify the design of genetic operators. However, this advantage comes with a cost. First of all, since the fitness function evaluates the quality of each rule separately, now it is not easy to compute the quality of the rule set as a whole - i.e. taking rule interactions into account. Another problem is that, since we want to discover a set of rules, rather than a single rule, we cannot allow the GA population to converge to a single individual - which is what usually happens in standard GAs. This introduces the need for some kind of niching method [45], which obviously is not necessary in the case of the Pittsburgh approach. We can avoid the need for niching in the Michigan approach by running the GA several times, each time discovering a different rule. The drawback of this approach is that it tends to be computationally expensive. Examples of GAs for classification which follow the Michigan approach are COGIN [30] and REGAL [28].

So far we have seen that an individual of a GA can represent a single rule or several rules, but we have not said yet how the rule(s) is(are) encoded in the genome of the individual. We now turn to this issue. To follow our discussion, assume that a rule has the form "IF $cond_1$ AND ... AND $cond_n$ THEN $class = c_i$", where $cond_1$ ... $cond_n$ are attribute-value conditions (e.g. *Sex* = "M") and c_i is the class predicted by the rule. We divide our discussion into two parts, the representation of the rule antecedent (the "IF" part of the rule) and the representation of the rule consequent (the "THEN" part of the rule). These two

issues are discussed in the next two subsections. In these subsections we will assume that the GA follows the Michigan approach, to simplify our discussion. However, most of the ideas in these two subsections can be adapted to the Pittsburgh approach as well.

3.1.2 Representing the Rule Antecedent (a Conjunction of Conditions). A simple approach to encode rule conditions into an individual is to use a binary encoding. Suppose that a given attribute can take on k discrete values. Then we can encode a condition on the value of this attribute by using k bits. The i-th value (i=1,...,k) of the attribute domain is part of the rule condition if and only if the i-th bit is "on" [13].

For instance, suppose that a given individual represents a rule antecedent with a single attribute-value condition, where the attribute is *Marital_Status* and its values can be "single", "married", "divorced" and "widow". Then a condition involving this attribute would be encoded in the genome by four bits. If these bits take on, say, the values "0 1 1 0" then they would be representing the following rule antecedent:

IF (*Marital_Status* = "married" OR "divorced")

Hence, this encoding scheme allows the representation of conditions with internal disjunctions, i.e. with the logical OR operator within a condition.

Obviously, this encoding scheme can be easily extended to represent rule antecedents with several conditions (linked by a logical AND) by including in the genome an appropriate number of bits to represent each attribute-value condition.

Note that if all the k bits of a given rule condition are "on", this means that the corresponding attribute is effectively being ignored by the rule antecedent, since any value of the attribute satisfies the corresponding rule condition. In practice, it is desirable to favor rules where some conditions are "turned off" - i.e. have all their bits set to "1" – in order to reduce the size of the rule antecedent. (Recall that we want comprehensible rules and, in general, the shorter the rule is the more comprehensible it is.) To achieve this, one can automatically set all bits of a condition to "1" whenever more than half of those bits are currently set to "1". Another technique to achieve the same effect will be discussed at the end of this subsection.

The above discussion assumed that the attributes were categorical, also called nominal or discrete. In the case of continuous attributes the binary encoding mechanism gets slightly more complex. A common approach is to use bits to represent the value of a continuous attribute in binary notation. For instance, the binary string "0 0 0 0 1 1 0 1" represents the value 13 of a given integer-valued attribute.

Instead of using a binary representation for the genome of an individual, this genome can be expressed in a higher-level representation which directly encodes the rule conditions. One of the advantages of this representation is that it leads to a more uniform treatment of categorical and continuous attributes, in comparison

with the binary representation.

In any case, in rule discovery we usually need to use variable-length individuals, since, in principle, we do not know a priori how many conditions will be necessary to produce a good rule. Therefore, we might have to modify crossover to be able to cope with variable-length individuals in such a way that only valid individuals are produced by this operator.

For instance, suppose that we use a high-level representation for two individuals to be mated, as follows (there is an implicit logical AND connecting the rule conditions within each individual):

(*Age* > 25) (*Marital_Status* = "Married")
(*Has_a_job* = "yes") (Age < 21)

As a result of a crossover operation, one of the children might be an invalid individual (i.e. a rule with contradicting conditions), such as the following rule antecedent:

IF (*Age* > 25) AND (*Age* < 21).

To avoid this, we can modify the individual representation to encode attributes in the same order that they occur in the data set, including in the representation "empty conditions" as necessary. Continuing the above example, and assuming that the data set being mined has only the attributes *Age*, *Marital_Status*, and *Has_a_job*, in this order, the two above individuals would be encoded as follows:

(*Age* > 25) (*Marital_Status* = "married") ("empty conditon")
(*Age* < 21) ("empty condition") (*Has_a_job* = "yes")

Now each attribute occupies the same position in the two individuals, i.e. attributes are aligned [21]. Hence, crossover will produce only valid individuals.

This example raises the question of how to determine, for each gene, whether it represents a normally-expressed condition or an empty condition. A simple technique for solving this problem is as follows. Suppose the data being mined contains *m* attributes. Then each individual contains *m* genes, each of them divided into two parts. The first one specifies the rule condition itself (e.g. *Age* > 25), whereas the second one is a single bit. If this bit is "on" ("off") the condition is included in (excluded from) the rule antecedent represented by the individual. In other words, the "empty conditions" in the above example are represented by turning off this bit. Since we want rule antecedents with a variable number of conditions, this bit is usually subject to the action of genetic operators [43].

3.1.3 Representing the Rule Consequent (Predicted Class). Broadly speaking, there are at least three ways of representing the predicted class (the "THEN" part of the rule) in an evolutionary algorithm. The first possibility is to encode it in the genome of an individual [13], [30] – possibly making it subject to evolution.

The second possibility is to associate all individuals of the population with the same predicted class, which is never modified during the running of the algorithm. Hence, if we want to discover a set of classification rules predicting *k*

different classes, we would need to run the evolutionary algorithm at least k times, so that in the i-th run, i=1,..,k, the algorithm discovers only rules predicting the i-th class [37], [43].

The third possibility is to choose the predicted class most suitable for a rule, in a kind of deterministic way, as soon as the corresponding rule antecedent is formed. The chosen predicted class can be the class that has more representatives in the set of examples satisfying the rule antecedent [28] or the class that maximizes the individual's fitness [49].

The above first and third possibilities have the advantage of allowing that different individuals of the population represent rules predicting different classes. This avoids the need to perform multiple runs of the evolutionary algorithm to discover rules predicting different classes, which is the case in the above second possibility. Overall, the third possibility seems more sound than the first one.

3.2 Genetic Operators for Rule Discovery

There has been several proposals of genetic operators designed particularly for rule discovery. Although these genetic operators have been used mainly in the classification task, in general they can be also used in other tasks that involve rule discovery, such as dependence modeling. We review some of these operators in the following subsections.

3.2.1 Selection. REGAL [27] follows the Michigan approach, where each individual represents a single rule. Since the goal of the algorithm is to discover a set of (rather than just one) classification rules, it is necessary to avoid the convergence of the population to a single individual (rule).

REGAL does that by using a selection procedure called universal suffrage. In essence, individuals to be mated are "elected" by training examples. An example "votes" for one of rules that cover it, in a probabilistic way. More precisely, the probability of voting for a given rule (individual) is proportional to the fitness of that rule. Only rules covering the same examples compete with each other. Hence, this procedure effectively implements a form of niching, encouraging the evolution of several different rules, each of them covering a different part of the data space.

3.2.2 Generalizing/Specializing Crossover. The basic idea of this special kind of crossover is to generalize or specialize a given rule, depending on whether it is currently overfitting or underfitting the data, respectively [27], [3]. Overfitting was briefly discussed in sections 2.2.1 and 2.3.1. Underfitting is the dual situation, in which a rule is covering too many training examples, and so should be specialized. A more comprehensive discussion about overfitting and underfitting in rule induction (independent of evolutionary algorithms) can be found e.g. in [57].

To simplify our discussion, assume that the evolutionary algorithm follows

the Michigan approach - where each individual represents a single rule - using a binary encoding (as discussed in subsection 3.1.2). Then the generalizing / specializing crossover operators can be implemented as the logical OR and the logical AND, respectively. This is illustrated in Fig. 2, where the above-mentioned bitwise logical functions are used to compute the values of the bits between the two crossover points denoted by the "|" symbol.

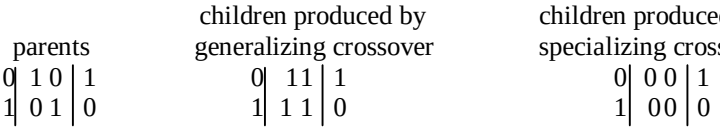

Fig. 2. Example of generalizing / specializing crossover

3.2.3 Generalizing/Specializing-Condition Operator. In the previous subsection we saw how the crossover operator can be modified to generalize/ specialize a rule. However, the generalization/specialization of a rule can also be done in a way independent of crossover. Suppose, e.g., that a given individual represents a rule antecedent with two attribute-value conditions, as follows - again, there is an implicit logical AND connecting the two conditions in (1):

(*Age* > 25) (*Marital_Status* = "single"). (1)

We can generalize, say, the first condition of (1) by using a kind of mutation operator that subtracts a small, randomly-generated value from 25. This might transform the rule antecedent (1) into, say, the following one:

(*Age* > 21) (*Marital_Status* = "single"). (2)

Rule antecedent (2) tends to cover more examples than (1), which is the kind of result that we wish in the case of a generalization operator. Another way to generalize rule antecedent (1) is simply to delete one of its conditions. This is usually called the drop condition operator in the literature.

Conversely, we could specialize the first condition of rule antecedent (1) by using a kind of mutation operator that adds a small, randomly-generated value to 25. Another way to specialize (1) is, of course, to add another condition to that rule antecedent.

3.3. Fitness Functions for Rule Discovery

Recall that, as discussed in section 2.1, ideally the discovered rules should: (a) have a high predictive accuracy; (b) be comprehensible; and (c) be interesting. In this subsection we discuss how these rule quality criteria can be incorporated in a fitness function. To simplify our discussion, throughout this subsection we will again assume that the GA follows the Michigan approach - i.e. an individual represents a single rule. However, the basic ideas discussed below can be easily adapted to GAs following the Pittsburgh approach, where an individual represents

a rule set.

Let a rule be of the form: IF A THEN C, where A is the antecedent (a conjunction of conditions) and C is the consequent (predicted class), as discussed earlier. A very simple way to measure the predictive accuracy of a rule is to compute the so-called confidence factor (CF) of the rule, defined as:

$$CF = |A \& C| / |A|,$$

where |A| is the number of examples satisfying all the conditions in the antecedent A and |A & C| is the number of examples that both satisfy the antecedent A and have the class predicted by the consequent C. For instance, if a rule covers 10 examples (i.e. |A| = 10), out of which 8 have the class predicted by the rule (i.e. |A&C| = 8) then the CF of the rule is CF = 80%.

Unfortunately, such a simple predictive accuracy measure favors rules overfitting the data. For instance, if |A| = |A & C| = 1 then the CF of the rule is 100%. However, such a rule is most likely representing an idiosyncrasy of a particular training example, and probably will have a poor predictive accuracy on the test set. A solution for this problem is described next.

The predictive performance of a rule can be summarized by a 2 x 2 matrix, sometimes called a confusion matrix, as illustrated in Fig. 3. To interpret this figure, recall that A denotes a rule antecedent and C denotes the class predicted by the rule. The class predicted for an example is C if and only if the example satisfies the rule antecedent. The labels in each quadrant of the matrix have the following meaning:

TP = True Positives = Number of examples satisfying A and C
FP = False Positives = Number of examples satisfying A but not C
FN = False Negatives = Number of examples not satisfying A but satisfying C
TN = True Negatives = Number of examples not satisfying A nor C

Clearly, the higher the values of TP and TN, and the lower the values of FP and FN, the better the rule.

		actual class	
		C	not C
predicted class	C	TP	FP
	not C	FN	TN

Fig. 3. Confusion matrix for a classification rule

Note that the above-mentioned CF measure is defined, in terms of the notation of Fig. 3, by: CF = TP / (TP + FP). We can now measure the predictive accuracy of a rule by taking into account not only its CF but also a measure of how "complete" the rule is, i.e. what is the proportion of examples having the predicted class C that is actually covered by the rule antecedent. The rule completeness measure, denoted Comp, is computed by the formula:

Comp = TP / (TP + FN). In order to combine the CF and Comp measures we can define a fitness function such as:

$$\text{Fitness} = \text{CF} \times \text{Comp}.$$

Although this fitness function does a good job in evaluating predictive performance, it has nothing to say about the comprehensibility of the rule. We can extend this fitness function (or any other focusing only on the predictive accuracy of the rule) with a rule comprehensibility measure in several ways. A simple approach is to define a fitness function such as

$$\text{Fitness} = w_1 \times (\text{CF} \times \text{Comp}.) + w_2 \times \text{Simp},$$

where Simp is a measure of rule simplicity (normalized to take on values in the range 0..1) and w_1 and w_2 are user-defined weights. The Simp measure can be defined in many different ways, depending on the application domain and on the user. In general, its value is inversely proportional to the number of conditions in the rule antecedent – i.e., the shorter the rule, the simpler it is.

Several fitness functions that take into account both the predictive accuracy and the comprehensibility of a rule are described in the literature - see e.g. [37], [28], [51], [21].

Noda and his colleagues [49] have proposed a fitness function which takes into account not only the predictive accuracy but also a measure of the degree of interestingness of a rule. Their GA follows the Michigan approach and was developed for the task of dependence modeling. Their fitness function is essentially a weighted sum of two terms, where one term measures the predictive accuracy of the rule and the other term measures the degree of interestingness (or surprisingness) of the rule. The weights assigned to each term are specified by the user. Another fitness function involving a measure of rule interestingness, more precisely a variation of the well-known J-measure, is discussed in [4].

In the above projects the rule interestingness measure is objective. An intriguing research direction would be to design a fitness function based on a subjective rule interestingness measure. In particular, one possibility would be to design a kind of interactive fitness function, where the fitness of an individual depends on the user's evaluation. A similar approach has been reported in an image-enhancement application [53], where the user drives GP by deciding which individual should be the winner in tournament selection; and in an attribute-selection task [60], where a user drives a GA by interactively and subjectively selecting good prediction rules.

4. Genetic Algorithms (GAs) for the Knowledge Discovery Process

This section is divided into two parts. Subsection 4.1 discusses GAs for data preprocessing, particularly attribute selection; whereas subsection 4.2 discusses a

GA for discovered-knowledge postprocessing, particularly "pruning" an ensemble of classifiers.

4.1 Genetic Algorithms (GAs) for Data Preprocessing

As discussed in section 2.3.1, one of the key problems in preparing a data set for mining is the attribute selection problem. In the context of the classification task, this problem consists of selecting, among all available attributes, a subset of attributes relevant for predicting the value of the goal attribute.

The use of GAs for attribute selection seems natural. The main reason is that the major source of difficulty in attribute selection is attribute interaction, and one of the strengths of GAs is that they usually cope well with attribute interactions.

In addition, the problem definition lends itself to a very simple, natural genetic encoding, where each individual represents a candidate attribute subset (a candidate solution, in this problem). More precisely, we can represent a candidate attribute subset as a string with m binary genes, where m is the number of attributes and each gene can take on the values 1 or 0, indicating whether or not the corresponding attribute is in the candidate attribute subset. For instance, assuming a 5-attribute data set, the individual "0 1 1 0 0" corresponds to a candidate solution where only the second and third attributes are selected to be given to the classification algorithm.

Then, a simple GA, using conventional crossover and mutation operators, can be used to evolve the population of candidate solutions towards a good attribute subset. The "trick" is to use a fitness function that is a direct measure of the performance achieved by the classification algorithm accessing only the attributes selected by the corresponding individual. With respect to the categorization of wrapper and filter approaches for attribute selection discussed in section 2.3.1, this approach is clearly an instance of the wrapper approach.

Note that in the above simple encoding scheme an attribute is either selected or not, but there is no information about the relative relevance of each attribute. It is possible to use an alternative encoding scheme where highly relevant attributes will tend to be replicated in the genome. This replication will tend to reduce the probability that a highly relevant attribute be removed from the individual due, for instance, to a harmful mutation.

Such an alternative encoding scheme was proposed by Cherkauer & Shavlik [12]. In their scheme, each gene of an individual contains either an attribute A_i, i=1,...,m, or no attribute, denoted by 0. The length of the individual is fixed, but it is not necessarily equal to m, the number of attributes. An attribute is selected if it occurs at least once in the individual. For instance, assuming a 10-attribute data set and a 5-gene string, the individual "0 A_8 0 A_8 A_4" represents a candidate solution where only attributes A_8 and A_4 are selected to be given to the classification algorithm.

Note that this example suggests an intriguing possibility. Suppose we are interested not only in selecting a subset of attributes, but also in determining how

relevant each of the selected attributes are. In the above example perhaps we could consider that A_8 is presumably more relevant than A_4, since the former occurs twice in the genome of the individual, whereas the latter occurs just once.

In any case it is possible to use a GA to optimize attribute weights directly (assigning to an attribute a weight that is proportional to its relevance), rather than to simply select attributes. This approach has been used particularly for optimizing attribute weights for nearest neighbor algorithms [39], [54].

Comprehensive comparisons between GA and other attribute-selection algorithms, across a number of data sets, are reported in [69] and [42]. In these projects GA was used as a wrapper to select attributes for a constructive neural network and a nearest neighbor algorithm, respectively. Overall, the results show that GA is quite competitive with other respectable attribute-selection algorithms. In particular, the results reported in [42] indicate that in large-scale attribute-selection problems, where the number of attributes is greater than 100, GA becomes the only practical way to get reasonable attribute subsets.

In [60] it is reported that the use of an interactive GA for attribute selection led to the discovery of rules that are easy-to-understand and simple enough to make practical decisions on a marketing application involving oral care products. A further discussion on the use of genetic algorithms in attribute selection can be found in [5], [63], [31], [46].

4.2 Genetic Algorithms (GAs) for Discovered-Knowledge Postprocessing

GAs can be used in a postprocessing step applied to the knowledge discovered by a data mining algorithm. As an example, suppose that the data mining step of the knowledge discovery process has produced an ensemble of classifiers (e.g. rule sets), rather than a single classifier (e.g. a single rule set). Actually, generating an ensemble of classifiers is a relatively recent trend in machine learning when our primary goal is to maximize predictive accuracy, since it has been shown that in several cases an ensemble of classifiers has a better predictive accuracy than a single classifier [58], [15]. When an ensemble of classifiers is produced, it is common to assign a weight to each classifier in the ensemble. Hence, when classifying a new test example, the class assigned to that example is determined by taking a kind of weighted vote of the classes predicted by the individual classifiers in the ensemble.

However, there is a risk of generating too many classifiers which end up overfitting the training data. Therefore, it is desirable to have a procedure to "prune" the ensemble of classifiers, which is conceptually similar to prune a rule set or a decision tree.

To address this problem, Thompson [61], [62] has proposed a GA to optimize the weights of the classifiers in the ensemble. The proposed GA uses a real-valued individual encoding. Each individual has n real-valued genes, where n is the number of classifiers in the ensemble. Each gene represents the voting weight of its corresponding classifier. The fitness function consists of measuring the predictive accuracy of the ensemble with the weights proposed by the individual.

This predictive accuracy is measured on a separate data subset, called the "pruning" set (or hold-out set). This is a part of the original training set reserved only for fitness-evaluation purposes, whereas the remaining part of the original training set is used only for generating the ensemble of classifiers.

Note that the number of classifiers in the ensemble can be effectively reduced if the voting weight of some classifier(s) is(are) set to 0. Actually, one of the mutation methods with which the author has experimented consists of simply setting a gene (voting weight) to 0.

5. Genetic Programming (GP) for Rule Discovery

GP can be considered as a more open-ended search paradigm, in comparison with GA [41], [6]. The search performed by GP can be very useful in classification and other prediction tasks, since the system can produce many different combinations of attributes - using the several different functions available in the function set - which would not be considered by a conventional GA. Hence, even if the original attributes do not have much predictive power by themselves, the system can effectively create "derived attributes" with greater predictive power, by applying the function set to the original attributes. The potential of GP to create these derived attributes will be discussed in more detail in section 6.

Before we move on to that section, we discuss next two issues in the use of GP for rule discovery, namely individual representation (subsection 5.1) and discovery of comprehensible rules (subsection 5.2).

5.1 Individual Representation

The application of standard GP to the classification task is relatively straightforward, as long as all the attributes are numeric. In this case we can include in the function set several kinds of mathematical function appropriate to the application domain and include in the terminal set the predicting attributes - and possibly a random-constant generator. Once we apply the functions in the internal nodes of a GP individual to the values of the attributes in the leaf nodes of that individual, the system computes a numerical value that is output at the root node of the tree. Assuming a two-class problem, if this output is greater than a given threshold the system predicts a given class, otherwise the system predicts the other class.

In this section, however, we are interested in using GP for discovering high-level, comprehensible prediction (IF-THEN) rules, rather than just producing a numerical signal in the root node. The first obstacle to be overcome is the closure property of GP. This property means that the output of a node in a GP tree can be used as the input to any parent node in the tree. Note that this property is satisfied in the above case of standard GP applied to numeric data, since, in principle, the number returned by a mathematical function can be used as the input to another mathematical function. (In practice, some mathematical functions have to be

slightly modified to satisfy the closure property.)

When mining a data set containing a mixture of continuous and categorical (or nominal) attribute values this property is not satisfied by standard GP. Different attributes are associated with different operators/functions. E.g. the condition "*Age* < 18" is valid, but the condition "*Sex* < female" is not.

Several solutions have been proposed to cope with the closure property of GP, when addressing the classification task. One approach is based on the use of constrained-syntax GP. The key idea is that, for each function available in the function set, the user specifies the type of its arguments and the type of its result [7]. Crossover and mutation are then modified to create only valid trees, by respecting the user-defined restrictions on tree syntax.

An example is shown in Table1, where, for each row, the second and third columns specify the data type of the input and of the output of the function specified in the first column. Once this kind of specification is available to the GP, the system can generate individuals such as the one shown in Fig. 4. This figure assumes that Atr1 is a categorical (nominal) attribute, whereas Atr3, Atr5, Atr6 are real-valued attributes.

Table 1: Example of data type definitions for input and output of functions.

Functions	data type of input arguments	data type of output
+, -, *, /	(real, real)	real
≤, >	(real, real)	boolean
=	(nominal, nominal)	boolean
AND, OR	(boolean, boolean)	boolean

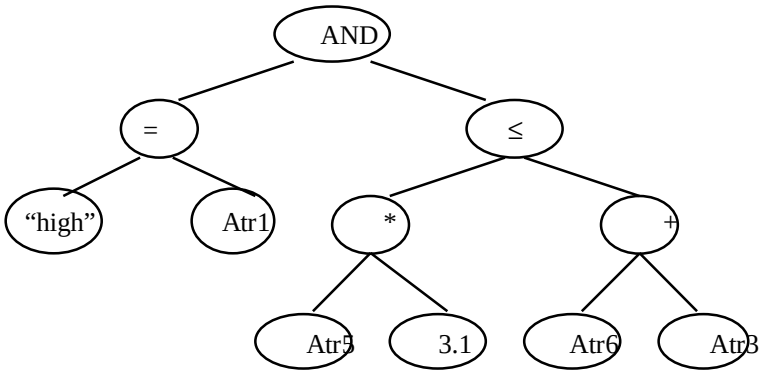

Fig. 4. Example of a GP individual (representing a rule antecedent) meeting the data type restrictions specified in Table 1

Another approach to constrained-syntax GP consists of having a user-defined, domain-dependent grammar specify the syntax of valid rules. The grammar can

also be used to incorporate domain-specific knowledge into the GP system. This approach, discussed in detail in [68], has led to the discovery of interesting knowledge (in the opinion of medical experts) in real-world fracture and scoliosis databases.

A different approach for coping with the problem of closure in GP for rule discovery consists of somehow modifying the data being mined - rather than modifying the standard GP algorithm. An example of this approach consists of booleanizing all the attributes being mined and then using logical operators (AND, OR, etc.) in the function set. Hence, the output of any node in the tree will be a boolean value, which can be used as input to any logical operator in the corresponding parent node. Systems based on this approach are described in [16], [8].

5.2 Discovering Comprehensible Rules with Genetic Programming (GP)

In principle a GP system for rule discovery could use fitness functions similar to the ones used by GAs for rule discovery. There are, however, some important differences. In particular, since the size of GP trees can grow a lot, in general it is more necessary to incorporate some measure of rule comprehensibility in the fitness function of a GP for rule discovery than in the fitness function of a GA for rule discovery. Actually, using GP to discover comprehensible classification rules can be considered a research issue. Some recent proposals to cope with this issue are briefly reviewed in the following.

First of all, it should be noted that, although knowledge comprehensibility is a kind of subjective concept, the data mining literature often uses an objective measure of rule comprehensibility: in general, the shorter (the fewer the number of conditions in) the rule, the more comprehensible it is. The same principle applies to rule sets. In general, the fewer the number of rules in the rule set, the more comprehensible it is. Our discussion in the following paragraphs assumes this kind of objective measure of rule comprehensibility. In other words, we are interested in GP systems that return a small number of short rules - as long as the discovered rule set still has a high predictive accuracy, of course.

The simplest approach to favor the discovery of short rules is to include a penalty for long rules in the fitness function. An example of the use of this approach can be found in [9], where a part of the fitness function is a direct measure of rule simplicity, given by the formula:

$$\text{Simplicity} = (MaxNodes - 0.5\ NumNodes - 0.5) / (MaxNodes - 1),$$

where *MaxNodes* is the maximum allowed number of nodes of a tree (individual) and *NumNodes* is the current number of nodes of the tree. This formula produces its maximum value of 1.0 when a rule is so simple that it contains just one term, and it produces its minimum value of 0.5 when the number of tree nodes equals the allowed maximum. This approach has led to the discovery of short, simple rules in a real-world application involving chest-pain diagnosis.

A more elaborated approach for discovering comprehensible rules is, for

instance, the hybrid GA/GP system to evolve decision trees proposed by [56]. The system is a hybrid GA/GP in the sense that it uses a GA to evolve a population of "programs" (decision trees). The system also uses a fitness function that considers both a tree's predictive accuracy and its size, in order to achieve the goal of minimizing tree size without unduly reducing predictive accuracy.

6. Genetic Programming (GP) for Data Preprocessing

In section 4.1 we saw that a simple GA can be naturally applied to an important data preprocessing task, namely the selection of relevant attributes for data mining. Sometimes, however, the preprocessing task may be more naturally addressed by a more open-ended evolutionary algorithm such as GP.

A good example is the preprocessing task of attribute construction - also called constructive induction. In this task the goal is to automatically construct new attributes, applying some operations to the original attributes, such that the new attributes make the data mining problem easier.

To illustrate the importance of this preprocessing task, consider the commonplace case where a classification algorithm can discover only propositional ("0-th order") logic rules. Suppose that we want to apply this algorithm to a data set where each record contains several attributes that can be used to predict whether the shares of a company will go up or down in the financial market. Now suppose that the set of predicting attributes includes the attributes *Income* and *Expenditure* of a company. Our propositional-logic classification algorithm would not be able to discover rules of the form:

IF (*Income* > *Expenditure*) AND . . . THEN (*Shares* = *up*)
IF (*Income* < *Expenditure*) AND . . . THEN (*Shares* = *down*)

because these rules involve a first-order logic condition, namely a comparison between two attributes – rather than between an attribute and its value.

A suitably-designed attribute construction algorithm could automatically construct a binary attribute such as "*(Income > Expenditure)?*", taking on the values "yes" or "no". The new attribute would then be given to the classification algorithm, in order to improve the quality of the rules to be discovered by the latter.

The major problem in attribute construction is that the search space tends to be huge. Although in the above very simple example it was easy to see that a good attribute could be constructed by using the relational operator ">", in practice there are a large number of candidate operations to be applied to the original attributes. In addition, the construction of a good attribute often requires that the attribute construction algorithm generates and evaluates combinations of several original attributes, rather than just two attributes as in the above example. GP can be used to search this huge search space.

An example of the use of GP in attribute construction is found in [36]. In this

project a GP-based system is compared with two other attribute construction methods, namely LFC and GALA. The comparison is made across 12 data sets. Overall, the predictive accuracy of the GP-based system was considerably better than LFC's one and somewhat better than GALA's one. A hybrid GA/GP system, performing attribute selection and attribute construction at the same time, is discussed in [64]. This approach has substantially reduced error rate in a face-recognition problem.

7. Discussion and Research Directions

We have begun our discussion of data mining and knowledge discovery by identifying, in section 2.1, three desirable properties of discovered knowledge. These properties are predictive accuracy, comprehensibility and interestingness. We believe a promising research direction is to design evolutionary algorithms which aim at discovering truly interesting rules. Clearly, this is much easier said than done. The major problem is that rule interestingness is a complex concept, involving both objective and subjective aspects. Almost all the fitness functions currently used in evolutionary algorithms for data mining focus on the objective aspect of rule quality, and in most cases only predictive accuracy and rule comprehensibility are taken into account. However, these two factors alone do not guarantee rule interestingness, since a highly-accurate, comprehensible rule can still be uninteresting, if it corresponds to a piece of knowledge previously known by the user.

Concerning data mining tasks, which correspond to kinds of problems to be solved by data mining algorithms, in this chapter we have focused on the classification task only (see section 2.2.1), due to space limitations. However, many of the ideas and concepts discussed here are relevant to other data mining tasks involving prediction, such as the dependence modeling task briefly discussed in section 2.2.2.

We have discussed several approaches to encode prediction (IF-THEN) rules into the genome of individuals, as well as several genetic operators designed specifically for data mining purposes. A typical example is the use of generalizing/specializing crossover discussed in section 3.2.2. Another example is the information-theoretic rule pruning operator proposed in [10]. We believe that the development of new data mining-oriented operators is important to improve the performance of evolutionary algorithms in data mining and knowledge discovery. Using this kind of operator makes evolutionary algorithms endowed with some "knowledge" about what kind of genome-modification operation makes sense in data mining problems. The same argument holds for other ways of tailoring evolutionary algorithms for data mining, such as developing data mining-oriented individual representations.

We have also discussed the use of evolutionary algorithms in the preprocessing and postprocessing phases of the knowledge discovery process.

Although there has been significant research on the use of GAs for attribute selection, the use of evolutionary algorithms in other preprocessing tasks and in postprocessing tasks seems to be less explored. In particular, we believe that a promising research direction is to use evolutionary algorithms for attribute construction (or constructive induction). Open-ended evolutionary algorithms, such as GP, can be suitable for this difficult, important data mining problem.

Acknowledgments

I thank Heitor S. Lopes for useful comments on the first draft of this chapter. My research on data mining with evolutionary algorithms is partially supported by grant 300153/98-8 from CNPq (the Brazilian government's National Council of Scientific and Technological Development).

References

[1] Agrawal R, Imielinski T and Swami A. Mining association rules between sets of items in large databases. *Proc. 1993 Int. Conf. Management of Data (SIGMOD-93)*, 207-216. May 1993.

[2] Agrawal R, Mannila H, Srikant R, Toivonen H and Verkamo AI. Fast discovery of association rules. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P and Uthurusamy R. (Eds.) *Advances in Knowledge Discovery and Data Mining*, 307-328. AAAI/MIT Press. 1996.

[3] Anglano C, Giordana A, Lo Bello G, Saitta L. Coevolutionary, distributed search for inducing concept descriptions. *Lecture Notes in Artificial Intelligence 1398. ECML-98: Proc. 10th Europ. Conf. Machine Learning*, 422-333. Springer-Verlag, 1998.

[4] Araujo DLA, Lopes HS and Freitas AA. A parallel genetic algorithm for rule discovery in large databases. *Proc. 1999 IEEE Systems, Man and Cybernetics Conf.*, v. 3, 940-945. Tokyo, 1999.

[5] Bala J, De Jong K, Huang J, Vafaie H, and Wechsler H. Using learning to facilitate the evolution of features for recognizing visual concepts. *Evolutionary Computation 4(3) - Special Issue on Evolution, Learning, and Instinct: 100 years of the Baldwin Effect*. 1997.

[6] Banzhaf W, Nordin P, Keller RE, Francone FD *Genetic Programming ~ an Introduction: On the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann, 1998.

[7] Bhattacharyya S, Pictet O, Zumbach G. Representational semantics for genetic programming based learning in high-frequency financial data. *Genetic Programming 1998: Proc. 3rd Annual Conf.*, 11-16. Morgan Kaufmann, 1998.

[8] Bojarczuk CC, Lopes HS, and Freitas AA. Discovering comprehensible classification rules using genetic programming: a case study in a medical domain. *Proc. Genetic and Evolutionary Computation Conference (GECCO-99)*, 953-958. Orlando, FL, USA, July/1999.

[9] Bojarczuk CC, Lopes HS, Freitas AA. Genetic programming for knowledge discovery in chest pain diagnosis. *IEEE Engineering in Medicine and Biology Magazine – special issue on data mining and knowledge discovery, 19(4)*, 38-44, July/Aug. 2000.

[10] Carvalho DR and Freitas AA. A hybrid decision tree/genetic algorithm for coping with the problem of small disjuncts in data mining. *Proc. Genetic and Evolutionary Computation Conf (GECCO-2000)*, 1061-1068. Las Vegas, NV, USA. July 2000.

[11] Catlett J. On changing continuous attributes into ordered discrete attributes. *Proc. European Working Session on Learning (EWSL-91). Lecture Notes in Artificial Intelligence 482*, 164-178. Springer-Verlag, 1991.

[12] Cherkauer KJ & Shavlik JW. Growing simpler decision trees to facilitate knowledge discovery. *Proc. 2nd Int. Conf. Knowledge Discovery & Data Mining (KDD-96)*, 315-318. AAAI Press, 1996.

[13] De Jong KA, Spears WM and Gordon DF. Using genetic algorithms for concept learning. *Machine Learning* 13, 161-188, 1993.

[14] Dhar V, Chou D, Provost F. Discovering interesting patterns for investment decision making with GLOWER – a Genetic Learner Overlaid with Entropy Reduction. *To appear in Data Mining and Knowledge Discovery journal.* 2000.

[15] Domingos P. Knowledge acquisition from examples via multiple models. *Machine Learning: Proc. 14th Int. Conf. (ICML-97)*, 98-106. Morgan Kaufmann, 1997.

[16] Eggermont J, Eiben AE, and van Hemert JI. A comparison of genetic programming variants for data classification. *Proc. Intelligent Data Analysis (IDA-99)*. 1999.

[17] Falkenauer E. *Genetic Algorithms and Grouping Problems.* John Wiley & Sons, 1998.

[18] Fayyad UM, Piatetsky-Shapiro G and Smyth P. From data mining to knowledge discovery: an overview. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P and Uthurusamy R. *Advances in Knowledge Discovery & Data Mining*, 1-34. AAAI/MIT, 1996.

[19] Fisher DH. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 1987, 139-172.

[20] Fisher D and Hapanyengwi G. Database management and analysis tools of machine induction. *Journal of Intelligent Information Systems*, 2(1), Mar. 1993, 5-38.

[21] Flockhart IW and Radcliffe NJ. GA-MINER: parallel data mining with hierarchical genetic algorithms - final report. *EPCC-AIKMS-GA-MINER-Report 1.0*. University of Edinburgh, UK, 1995.

[22] Freitas AA. On objective measures of rule surprisingness. *Lecture Notes in Artificial Intelligence 1510: Principles of Data Mining and Knowledge Discovery (Proc. 2nd European Symp., PKDD´98, Nantes, France)*, 1-9. Springer-Verlag, 1998.

[23] Freitas AA. On Rule Interestingness Measures. *Knowledge-Based Systems 12(5-6)*, 309-315. Oct. 1999.

[24] Freitas, AA. Understanding the crucial differences between classification and discovery of association rules - a position paper. *To appear in ACM*

SIGKDD Explorations, 2(1), 2000.
[25] Freitas AA and Lavington SH. *Mining Very Large Databases with Parallel Processing*. Kluwer, 1998.
[26] Gebhardt F. Choosing among competing generalizations. *Knowledge Acquisition* 3, 1991, 361-380.
[27] Giordana A, Saitta L, Zini F. Learning disjunctive concepts by means of genetic algorithms. *Proc. 10th Int. Conf. Machine Learning (ML-94)*, 96-104. Morgan Kaufmann, 1994.
[28] Giordana A and Neri F. Search-intensive concept induction. *Evolutionary Computation 3(4)*: 375-416, Winter 1995.
[29] Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.
[30] Greene DP & Smith SF. Competition-based induction of decision models from examples. *Machine Learning 13*, 229-257. 1993.
[31] Guerra-Salcedo C & Whitley D. Feature selection mechanisms for ensemble creation: a genetic search perspective. In: Freitas AA (Ed.) *Data Mining with Evolutionary Algorithms: Research Directions – Papers from the AAAI Workshop*, 13-17. Technical Report WS-99-06. AAAI Press, 1999.
[32] Guyon I, Matic N and Vapnik V. Discovering informative patterns and data cleaning. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P and Uthurusamy R. (Eds.) *Advances in Knowledge Discovery and Data Mining*, 181-203. AAAI/MIT Press. 1996.
[33] Hall LO, Ozyurt IB and Bezdek JC. Clustering with a genetically optimized approach. *IEEE Trans. Evolutionary Computation 3(2)*, 103-112. July 1999.
[34] Hand DJ. *Construction and Assessment of Classification Rules.* John Wiley & Sons, 1997.
[35] Holland JH. Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In: Mitchell T et al. (Eds.) *Machine Learning, Vol. 2*, 593-623. Morgan Kaufmann, 1986.
[36] Hu Y-J. A genetic programming approach to constructive induction. *Genetic Programming 1998: Proc. 3rd Annual Conf.*, 146-151. Morgan Kaufmann, 1998.
[37] Janikow CZ. A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning* 13, 189-228. 1993.
[38] John GH, Kohavi R and Pfleger K. Irrelevant features and the subset selection problem. *Proc. 11th Int. Conf. Machine Learning*, 121-129. 1994.
[39] Kelly Jr. JD and Davis L. A hybrid genetic algorithm for classification. *Proc. 12th Int. Joint Conf. on AI*, 645-650. 1991.
[40] Klemettinen M, Mannila H, Ronkainen P, Toivonen H and Verkamo AI. Finding interesting rules from large sets of discovered association rules. *Proc. 3rd Int. Conf. on Information and Knowledge Management.* Gaithersburg, Maryland. Nov./Dec. 1994.
[41] Koza JR. *Genetic Programming: on the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.
[42] Kudo M & Skalansky J. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition 33(1)*, 25-41, Jan. 2000.
[43] Kwedlo W and Kretowski M. Discovery of decision rules from databases: an evolutionary approach. *Proc. 2nd European Symp. on Principles of Data*

Mining and Knowledge Discovery (PKDD-98). Lecture Notes in Artificial Intelligence 1510, 371-378. Springer-Verlag, 1998.

[44] Liu B, Hsu W. and Chen S. Using general impressions to analyze discovered classification rules. *Proc. 3rd Int. Conf. Knowledge Discovery & Data Mining*, 31-36. AAAI Press, 1997.

[45] Mahfoud SW. *Niching Methods for Genetic Algorithms.* Ph.D. Thesis. Univ. of Illinois at Urbana-Champaign. IlliGAL Report No. 95001. May 1995.

[46] Martin-Bautista MJ and Vila MA. A survey of genetic feature selection in mining issues. *Proc. Congress on Evolutionary Computation (CEC-99)*, 1314-1321. Washington D.C., July 1999.

[47] Michalewicz Z. *Genetic Algorithms + Data Structures = Evolution Programs.* 3rd Ed. Springer-Verlag, 1996.

[48] D. Michie, D.J. Spiegelhalter and C.C. Taylor. *Machine Learning, Neural and Statistical Classification*. New York: Ellis Horwood, 1994.

[49] Noda E, Freitas AA and Lopes HS. Discovering interesting prediction rules with a genetic algorithm. *Proc. Conference on Evolutionary Computation – 1999 (CEC-99)*, 1322-1329. Washington D.C., USA, July/1999.

[50] Park Y and Song M. A genetic algorithm for clustering problems. *Genetic Programming 1998: Proc. 3rd Annual Conf.*, 568-575. Morgan Kaufmann, 1998.

[51] Pei M, Goodman ED, Punch WF. Pattern discovery from data using genetic algorithms. *Proc. 1st Pacific-Asia Conf. Knowledge Discovery & Data Mining (PAKDD-97)*. Feb. 1997.

[52] Pfahringer B. Supervised and unsupervised discretization of continuous features. *Proc. 12th Int. Conf. Machine Learning*, 456-463. 1995.

[53] Poli R and Cagnoni S. Genetic programming with user-driven selection: experiments on the evolution of algorithms for image enhancement. *Genetic Programming 1997: Proc. 2nd Annual Conf.*, 269-277. Morgan Kaufmann, 1997.

[54] Punch WF, Goodman ED, Pei M, Chia-Sun L, Hovland P, Enbody R. Further research on feature selection and classification using genetic algorithms. *Proc. 5th Int. Conf. Genetic Algorithms (ICGA-93)*, 557-564. Morgan Kaufmann, 1993.

[55] Pyle D. *Data Preparation for Data Mining*. Morgan Kaufmann, 1999.

[56] Ryan MD & Rayward-Smith VJ. The evolution of decision trees. *Genetic Programming 1998: Proc. 3rd Annual Conf.*, 350-358. Morgan Kaufmann, 1998.

[57] Schaffer C. Overfitting avoidance as bias. *Machine Learning* 10, 153-178. 1993.

[58] Schapire RE, Freund Y, Bartlett P, Lee WS. Boosting the margin: a new explanation for the effectiveness of voting methods. *Machine Learning: Proc. 14th Int. Conf. (ICML-97)*, 322-330. Morgan Kaufmann, 1997.

[59] Simoudis E, Livezey B and Kerber R. Integrating inductive and deductive reasoning for data mining. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P and Uthurusamy R. (Eds.) *Advances in Knowledge Discovery and Data Mining*, 353-373. AAAI/MIT Press. 1996.

[60] Terano T & Ishino Y. Interactive genetic algorithm based feature selection and its application to marketing data analysis. In: Liu H & Motoda H (Eds.)

Feature Extraction, Construction and Selection: a data mining perspective, 393-406. Kluwer, 1998.

[61] Thompson S. Pruning boosted classifiers with a real valued genetic algorithm. *Research & Develop. in Expert Systems XV - Proc. ES'98*, 133-146. Springer-Verlag, 1998.

[62] Thompson S. Genetic algorithms as postprocessors for data mining. In: Freitas AA (Ed.) *Data Mining with Evolutionary Algorithms: Research Directions – Papers from the AAAI Workshop*, 18-22. Technical Report WS-99-06. AAAI Press, 1999.

[63] Vafaie H and De Jong K. Robust feature selection algorithms. *Proc. 1993 IEEE Int. Conf on Tools with AI*, 356-363. Boston, Mass., USA. Nov. 1993.

[64] Vafaie H and De Jong K. Evolutionary feature space transformation. In: Liu H & Motoda H (Eds.) *Feature Extraction, Construction and Selection: a data mining perspective*, 307-323. Kluwer, 1998.

[65] Weiss GM & Hirsh H. Learning to predict rare events in event sequences. *Proc. 4th Int. Conf. Knowledge Discovery and Data Mining*, 359-363. AAAI Press, 1998.

[66] Weiss SM and Indurkhya N. *Predictive Data Mining: a practical guide*. Morgan Kaufmann, 1998.

[67] Weiss SM and Kulikowski CA. *Computer Systems that Learn*. Morgan Kaufmann, 1991.

[68] Wong ML & Leung KS. *Data Mining Using Grammar-Based Genetic Programming and Applications*. Kluwer, 2000.

[69] Yang J and Honavar V. Feature subset selection using a genetic algorithm. In: Liu H & Motoda H (Eds.) *Feature Extraction, Construction and Selection: a data mining perspective*, 117-136. Kluwer, 1998.