PERGAMON Computers and Mathematics with Applications 45 (2003) 737–748

An International Journal
computers & mathematics
with applications

www.elsevier.com/locate/camwa

Learning Classification Rules from Data

A. AN
Department of Computer Science
York University
Toronto, Ontario M3J 1P3 Canada
aan@cs.yorku.ca

(Received October 2001; accepted January 2002)

Abstract—We present ELEM2, a machine learning system that induces classification rules from a set of data based on a heuristic search over a hypothesis space. ELEM2 is distinguished from other rule induction systems in three aspects. First, it uses a new heuristic function to guide the heuristic search. The function reflects the degree of relevance of an attribute-value pair to a target concept and leads to selection of the most relevant pairs for formulating rules. Second, ELEM2 handles inconsistent training examples by defining an unlearnable region of a concept based on the probability distribution of that concept in the training data. The unlearnable region is used as a stopping criterion for the concept learning process, which resolves conflicts without removing inconsistent examples. Third, ELEM2 employs a new rule quality measure in its post-pruning process to prevent rules from overfitting the data. The rule quality formula measures the extent to which a rule can discriminate between the positive and negative examples of a class. We describe features of ELEM2, its rule induction algorithm and its classification procedure. We report experimental results that compare ELEM2 with C4.5 and CN2 on a number of datasets.

Keywords—Machine learning, Rule induction, Classification, Data mining, Artificial intelligence.

1. INTRODUCTION

Machine learning is concerned with the question of how to construct a computer program that automatically learns new facts and theories from data. Rule induction is a special kind of machine learning technique that reasons from specific cases to general principles that are expressible as if-then rules. A number of rule induction systems, such as C4.5 [1] and CN2 [2], have been constructed and applied to discover knowledge from data in different applications, yet many suffer from poor performance in prediction accuracy in many practical domains. While it seems unlikely to have an algorithm to perform best in all the domains of interest, it may well be possible to produce learners that perform better on a wide variety of real-world domains.

To achieve this objective, we proposed ELEM2, a rule induction method that employs new strategies to enhance the induction and classification processes. In designing ELEM2, we adopt

This research was supported by grants from the Natural Sciences and Engineering Research Council (NSERC) and the Institute for Robotics and Intelligent Systems (IRIS). I would also like to thank N. Cercone of University of Waterloo for his suggestions on this work.

0898-1221/03/$ - see front matter Typeset by AMS-TEX
PII: S0898-1221(03)00034-8

the *separate and conquer* learning strategy used by *sequential covering* algorithms, such as CN2 [2] and PRISM [3]. These algorithms induce one rule at a time by selecting attribute-value pairs, remove the data the new rule covers, and iterate this process. However, ELEM2 differs from other sequential covering algorithms in several aspects.

First, to select an attribute-value pair, ELEM2 employs a new heuristic function for evaluating attribute-value pairs. Various evaluation criteria have been used in different learning algorithms. For example, ID3 [4] employs an entropy-based information gain to find the most relevant attribute to grow decision trees. PRISM [3] uses another form of information gain which can be characterized in terms of apparent classificatory accuracies on the training set to measure the relevance of attribute-value pairs with respect to a target concept. LEM2 [5] basically considers as the most relevant selector the attribute-value pair that has the largest coverage over the positive examples. We argue that both coverage and information gain should be considered when measuring the relevance of selectors. Selectors that cover a large number of positive examples may also cover negative examples well. On the other hand, when only considering information gain, no matter whether the consideration is in terms of entropy or apparent classificatory accuracies, such consideration can lead to generation of rules covering few examples [1]. Such rules tend to have higher predictive error rates [6]. We propose an alternative evaluation function. The new function is defined in terms of both a classification gain and the coverage of an attribute-value pair over a set of the training data and also reflects the degree of relevance of the attribute-value pair to a target concept with regard to the training data.

Second, ELEM2 addresses the issue of handling inconsistent examples. Inconsistent examples in the training set usually confuse a learning system when the system tries to identify common properties of a set of objects. One way to handle this problem is to remove all or part of the inconsistent examples to reduce confusion. This may not be a good idea, especially in very noisy environments in which most of the examples may need to be eliminated. Also, inconsistent examples may provide useful information for probability analysis during induction. To handle this problem, ELEM2 defines an unlearnable region for each concept based on the probability distribution of the concept over the training data. The unlearnable region of a concept is used as a stopping criterion for the concept learning process: if the positive examples that are not yet covered by the already generated concept descriptions fall into the unlearnable region of the concept, the process for learning that concept stops.

Third, ELEM2 employs a new rule quality measure in its process of post-pruning rules. Post-pruning is a technique that prevents a generated rule from *overfitting* the data. There have been several post-pruning methods in the literature. For example, C4.5 [1] uses an error-based technique called *pessimistic* pruning for pruning decision trees, which estimates the predictive error rate of concept descriptions by adjusting the apparent error rate on the training set. Another example is AQ15 [7], which associates conjunctions in a generated rule with weights and the ones with the least weights were removed to avoid overfitting the data. The weight is defined as the number of training examples explained *uniquely* by the conjunction. ELEM2 takes another route in doing post-pruning. It defines a rule quality measure based on the relative distribution of a rule with respect to the positive and negative examples the rule covers. The rule quality measure is used in the pruning process to determine which part(s) of a rule can be removed.

The paper is organized as follows. In Section 2, we present the strategies that ELEM2 uses to select attribute-value pairs. In particular, we describe how ELEM2 groups attribute-value pairs to formulate a search space and how the evaluation function is defined. In Section 3, an unlearnable region of a concept is defined which is used by ELEM2 for handling inconsistency in the training data. In Section 4, we discuss the rule quality measure used by ELEM2 for post-pruning generated rules. The ELEM2 induction algorithm and its classification strategy are presented in Sections 5 and 6. In Section 7, an example is given to illustrate ELEM2's rule induction and classification processes. In Section 8, we report some experimental results for evaluating ELEM2. Finally, we conclude the paper in Section 8.

2. ATTRIBUTE-VALUE PAIR SELECTION

An *attribute-value pair* is a relation between the attribute and its values, which is represented in the form $\langle a \, \mathrm{rel} \, v \rangle$, where a is an attribute, rel denotes a relational operator (e.g., $=, \neq, \leq, >, \ldots,$ or $\in$), and v is a specific value or a set of values of the attribute.[1] Let t denote an attribute-value pair. We use a_t, rel_t, and v_t to denote the attribute, the relational operator, and the value or the set of values in t, respectively. We define that the *complement* of an attribute-value pair t, denoted as $\neg t$, is the attribute-value pair $\langle a_t \, \neg\mathrm{rel} \, v_t \rangle$, where $\neg$rel is the negation of rel. For example, the complement of $\langle a = v \rangle$ is $\langle a \neq v \rangle$.

ELEM2 induces rules for a target concept by selecting relevant attribute-value pairs from a space of attribute-value pairs. This section addresses two issues related to the attribute-value pair selection in ELEM2. First, we discuss how ELEM2 formulates the attribute-value pair space. Then we introduce the evaluation function that ELEM2 uses to select an attribute-value pair from the formulated space.

2.1. Grouping Attribute-Value Pairs

In formulating the space of attribute-value pairs, many induction algorithms consider only single-valued attribute-value pairs. This consideration may cause the learning algorithm to generate more rules that cover small portions of examples. To overcome this problem, ELEM2 works with attribute-value pairs whose value may be a disjunction or a range of values. We refer to the combination of values as grouping attribute values. Since there are a large number of possible combinations, ELEM2 considers only reasonable groupings that can be easily made use of or refined by the learning algorithm. This strategy avoids producing an exponentially large search space. In grouping values, we use different strategies for different kinds of attributes.

2.1.1. Grouping nominal attribute values

For a nominal attribute, the initial search space of attribute-value pairs is made of attribute-value pairs with single values or set values that exclude only one single value. That is, for each value v_i in the domain of a nominal attribute a, two pairs, $\langle a = v_i \rangle$ and $\langle a \neq v_i \rangle$, are included in the initial search space. If a single-valued pair is selected, ELEM2 considers dynamic grouping of values by recursively grouping one more value at a time. Details about how to dynamically group these single-valued attribute-value pairs will be described in the ELEM2 induction algorithm section.

2.1.2. Grouping continuous attribute values

In ELEM2, continuous attributes are discretized by using user-supplied discretization formulas or by applying an automatic discretization method [8]. Suppose $\{x_1, x_2, \ldots, x_n\}$ is the set of cut-points for a discretized continuous attribute a, where $x_i < x_{i+1}$ for $i = 1, \ldots, n-1$. Grouping of values of this kind of attribute is carried out before the induction process commences. The grouping method is based on n binary splits, where n is the number of cut-points. For each cut-point x_i $(i = 1, \ldots, n)$, two groups: $\langle a \leq x_i \rangle$ and $\langle a > x_i \rangle$, are generated. In this way, a total of $2n$ pairs can be obtained for an attribute with n cut-points. Therefore, in our method, only $2n$ attribute-value pairs need to be examined during induction. Our evaluation function for selecting attribute-value pairs, described below, further allows us to exam only half of these attribute-value pairs because the significance value of $\langle a \leq x_i \rangle$ is the additive inverse of that for $\langle a > x_i \rangle$.

[1] When rel is a set operator, such as $\in$ and $\notin$, v is a set of values. Otherwise, v is a single value.

2.2. Evaluating Attribute-Value Pairs

ELEM2 generates decision rules for a target concept by performing a general-to-specific search in a hypothesis space. At each step of specialization, a heuristic function is used to evaluate attribute-value pairs. The function assigns a significance value to each considered pair in order for the most significant attribute-value pair to be selected. The significance function is defined according to the relevance of an attribute-value pair to the target concept. An attribute-value pair t is *relevant* to a concept c with respect to a set, S, of examples if

$$P(t) > 0 \qquad \text{and} \qquad P(c \mid t) \neq P(c),$$

where $P(t)$ denotes the probability that an example in S satisfies the relation expressed by t, $P(c)$ denotes the probability of the examples occurring in S that belong to concept c, and $P(c \mid t)$ is the probability that an example in S belongs to c given that the example satisfies t. Under this definition, t is relevant to the concept c if it can change the probability of c, or in other words, if c is conditionally dependent on t.

In a set of training samples, there may exist more than one attribute-value pair that is relevant to a concept. Some pairs may be strongly relevant, while others may not be so relevant. To measure the degree of relevance, we use an evaluation function to assign a significance value to each attribute-value pair. The function is defined as

$$\mathrm{SIG}_c(t) = P(t)(P(c \mid t) - P(c)). \tag{1}$$

According to this definition, if $P(c \mid t) = P(c)$, i.e., t is not relevant to the concept c, then the degree of relevance of t to c is equal to 0; if $P(c \mid t) \neq P(c)$, i.e., t is relevant to c, then the degree of relevance is proportional to both the difference between $P(c \mid t)$ and $P(c)$ and the coverage of t over the training set currently being considered. The range of this function is $(-1, 1)$. If the value stays positive, then the higher the value, the more relevant the attribute-value pair t with respect to the target concept c; if the value is negative, the lower the value, the more relevant the attribute-value pair $\neg t$ (i.e., the complement of pair t) with respect to c. We use $P(t)$ as a coefficient of the function since we believe that, say, a 95% accurate rule which covers 1000 training cases is better than a 100% accurate rule that covers only one case. This helps avoid the overfitting problem.

The significance function has a nice property, expressed as follows. Given a concept c and an attribute-value pair t, it can be proved that [9]

$$\mathrm{SIG}_c(t) = -\mathrm{SIG}_c(\neg t). \tag{2}$$

This means that the SIG values for a selector and its complement are additively inverse. This observation allows us to narrow the search space of selectors by half since the value for one of them can be obtained from the value for the other. Therefore, using this evaluation function is more efficient in practice than using other functions that do not have this feature.

3. HANDLING INCONSISTENCY

In real-world applications, the set of training data may be inconsistent due to incomplete or noisy information. Two examples are inconsistent if they have identical attribute values for the condition attributes, but are labelled as belonging to different concepts. Inconsistent data in the training set may confuse a learning algorithm and result in a failure in deriving decision rules. ELEM2 handles the problem of inconsistency by computing an unlearnable region for each concept, inspired by [10]. Let $R = \{X_1, X_2, \ldots, X_n\}$, where X_i $(1 \leq i \leq n)$ is a set of examples that are identical in terms of condition attribute values and there are a total of n sets of this kind in the training set. We can predict that any example that matches the condition part of the

examples in X_i belongs to the concept c with the probability $P(c \mid X_i)$, which is the probability that an example belongs to c given that the example is in X_i. The *classification gain* of X_i with respect to a concept c is defined as [10]

$$\mathrm{CG}_c(X_i) = P(c \mid X_i) - P(c),$$

which measures how much is gained by classifying a new example into c based on the information about the probabilities of the set X_i and the concept c. The *negative region* of a concept c is defined as

$$\mathrm{NEG}(c) = \bigcup_{P(c|X_i) \leq P(c)} X_i,$$

which means, if $\mathrm{CG}_c(X_i) \leq 0$, then X_i belongs to the negative region of c. The *unlearnable region* of a concept c, denoted as $\mathrm{URL}(c)$, is defined as the set of positive examples of c that exist in $\mathrm{NEG}(c)$.

During ELEM2's rule induction, if the positive members of the currently considered set of training examples belong to the unlearnable region of the target concept, the induction process for this concept stops. This prevents ELEM2 from learning from the inconsistent examples that do not provide positive classification gain.

4. POST-PRUNING INDUCED RULES

Noise or coincidental regularities in training data can lead to the learning algorithm to produce long and distorted concept descriptions, which cover a small number of anomalous examples in the training set. These descriptions are commonly known as *small disjuncts* [6]. Generating these small disjuncts not only increases the induction time and the complexity of the concept description but also decreases predictive performance of the learned knowledge on unknown objects since the rules applied to the noisy examples may misclassify correct examples. Post-pruning is a technique that rule induction algorithms use to handle the small disjunct problem. *Post-pruning* allows the induction process to run to completion (i.e., form a concept description completely consistent with the training data or as nearly consistent as possible if the complete consistency is impossible) and then 'post-prunes' the over-fitted concept description by removing the components deemed unreliable. A criterion is needed in post-pruning to check whether a component in a concept description should be removed.

In ELEM2, a rule quality measure is used as a criterion for post-pruning. The measure is inspired by a query term weighting formula used in probabilistic information retrieval [11]. The formula measures the extent to which a query term can discriminate between relevant and irrelevant documents. To use the formula in rule induction, we make an analogy and consider a rule as a query term in the information retrieval setting and positive or negative examples as relevant or irrelevant documents, respectively. Therefore, we can use the same formula to measure the extent to which a rule (r) can discriminate between the positive and negative examples of a concept (c) as follows:

$$Q(r) = \log \frac{P(r \mid c)(1 - P(r \mid \neg c))}{P(r \mid \neg c)(1 - P(r \mid c))}.$$

Using simple proportion estimations of the two probabilities, the formula becomes

$$Q(r) = \log \frac{m(N - n - M + m)}{(n - m)(M - m)},$$

where N is the total number of examples in the training data, M is the number of positive examples of concept c in the training data, n is the number of examples covered by rule r, and m is the number of positive examples covered by rule r. Practically, to prevent zero division and avoid infinite rule quality values, the formula is further adjusted to

$$Q(r) = \log \frac{(m + 0.5)(N - n - M + m + 0.5)}{(n - m + 0.5)(M - m + 0.5)}.$$

The process of post-pruning a rule is to recursively conduct the following.

1. Compute a rule quality value for the rule;
2. Check the pairs in the reverse order in which they were selected to see if a pair can be removed without causing the rule quality to decrease. If yes, remove it.

This process is repeated until no pair can be removed.

5. THE ELEM2 RULE INDUCTION ALGORITHM

ELEM2 learns a set of rules for one concept at a time. For each concept, it learns a disjunctive set of conjunctive rules and uses a sequential covering strategy to learn this set of rules, i.e., it induces one rule at a time, removes the data covered by the rule and then iterates the process. The learning algorithm is described as follows. For each concept c, we have the following.

1. Compute the unlearnable region of c: URL(c).
2. Let CS be the current training set.
3. Calculate the significance value, $\text{SIG}_c(av_i)$, of each attribute-value pair av_i in the attribute-value pair space with respect to CS.
4. Select the pair av for which $\text{SIG}_c(av)$ is a maximum. If more than one pair has the maximum SIG_c value, select among them the pair that covers the most training examples in CS. If another tie occurs, select the pair whose attribute has the highest priority.[2] If a further tie occurs, i.e., the attributes in question have the same priority, select the pair that is examined first.
5. If the attribute a in the selected pair av is a nominal attribute and av is single-valued, dynamic grouping is performed as follows:
   (a) let TCS be the set of examples in CS that are not covered by av;
   (b) compute the SIG values of all other pairs of attribute a with respect to TCS;
   (c) group with av the pairs whose SIG value is greater than or equal to $\text{SIG}_c(av)$.
6. Remove from CS the examples that are not covered by av.
7. Repeat Steps 3–6 until CS contains only examples of the concept c or the positive examples it contains belong to URL(c). The induced rule r is a conjunction of all the attribute-value pairs selected.
8. Post-prune the induced rule r.
9. Remove all the examples covered by this rule from the current training set.
10. Repeat Steps 2–9 until all the examples of c have been removed or the remaining examples of c belong to URL(c).

When the rules for one concept have been induced, the training set is restored to its initial state and the algorithm is applied again to induce a set of rules describing the next concept.

6. CLASSIFICATION USING RULES

Three cases are possible for matching an example with a set of rules: there may be only one match (i.e., the example matches only one rule), more than one match (i.e., the example matches more than one rule), or no match (i.e., the example does not match any rules). We refer to these three cases as single-match, multiple-match, and no-match. The single-match is not a problem since the example can be classified into the concept indicated by the matched rule. In the multiple-match case, if the matched rules indicate the same concept, then the example is classified into this concept. If the matched rules do not agree on the concepts, then ELEM2 computes a decision score for each concept that the matched rules indicate. The decision score

[2] In the ELEM2 implementation, we allow the user to specify priorities of the attributes. An attribute that has a higher priority is considered to be more important or more relevant to the learning task.

of a concept c is defined as

$$\mathrm{DS}(c) = \sum_{i=1}^{n} Q(r_i),$$

where r_i is a matched rule that indicates c, n is the number of this kind of rules, and $Q(r_i)$ is the quality of rule r_i. ELEM2 classifies the example into the concept with the highest decision score.

In the case of no-match, partial matching is considered where some attribute-value pairs of a rule may match the values of corresponding attributes in the new example. A partial matching score between an example e and a rule r with n attribute-value pairs, m of which match the corresponding attributes of e, is defined as follows:

$$\mathrm{PMS}(r) = \frac{m}{n} \times Q(r).$$

Based on the partial matching scores of the partially-matched rules, ELEM2 assigns a decision score to each concept indicated by these rules. The decision score of a concept c is defined as follows:

$$\mathrm{DS}(c) = \sum_{i} \mathrm{PMS}(r_i),$$

where i ranges from 1 to the number of partially matched rules indicating concept c. In decision making, the new example is classified into the concept with the highest value of the decision score.

7. AN EXAMPLE

Consider the data set shown in Table 1. The data set describes a set of patients in terms of four attributes: *headache, pain, temperature,* and *flu*. Suppose that attribute flu is the decision attribute and others are condition attributes, which means that the values for flu represent the concepts that we are concerned about and we would like to induce rules from this data set so that the rules can be used to predict whether a patient has flu based on the values of the patient for the three condition attributes.

Table 1. An example data set.

Patients	Headache	Pain	Temperature	Flu
x_1	yes	yes	normal	no
x_2	yes	yes	high	yes
x_3	yes	yes	very high	yes
x_4	no	no	high	no
x_5	no	yes	normal	no
x_6	no	no	high	no
x_7	no	yes	very high	yes
x_8	no	no	high	yes
x_9	no	yes	very high	no
x_{10}	no	no	high	no

7.1. Inducing Rules

To learn rules for concept ⟨flu = yes⟩, ELEM2 conducts the following.

1. Compute the unlearnable region of the concept by
   - dividing the training examples into the following subsets according to their condition attribute values:
     $X_1 = \{x_1\}$;
     $X_2 = \{x_2\}$;

$X_3 = \{x_3\}$;
$X_4 = \{x_4, x_6, x_8, x_{10}\}$;
$X_5 = \{x_5\}$;
$X_6 = \{x_7, x_9\}$;

- computing the classification gain of each subset with respect to the concept[3]:
  $\mathrm{CG}_{\langle \text{flu=yes} \rangle}(X_1) = P(\langle \text{flu} = \text{yes} \rangle | X_1) - P(\langle \text{flu} = \text{yes} \rangle) = -2/5$;
  $\mathrm{CG}_{\langle \text{flu=yes} \rangle}(X_2) = P(\langle \text{flu} = \text{yes} \rangle | X_2) - P(\langle \text{flu} = \text{yes} \rangle) = 3/5$;
  $\mathrm{CG}_{\langle \text{flu=yes} \rangle}(X_3) = P(\langle \text{flu} = \text{yes} \rangle | X_3) - P(\langle \text{flu} = \text{yes} \rangle) = 3/5$;
  $\mathrm{CG}_{\langle \text{flu=yes} \rangle}(X_4) = P(\langle \text{flu} = \text{yes} \rangle | X_4) - P(\langle \text{flu} = \text{yes} \rangle) = -3/20$;
  $\mathrm{CG}_{\langle \text{flu=yes} \rangle}(X_5) = P(\langle \text{flu} = \text{yes} \rangle | X_5) - P(\langle \text{flu} = \text{yes} \rangle) = -2/5$;
  $\mathrm{CG}_{\langle \text{flu=yes} \rangle}(X_6) = P(\langle \text{flu} = \text{yes} \rangle | X_6) - P(\langle \text{flu} = \text{yes} \rangle) = 1/10$;
- computing the negative region of the concept, which is the union of the subsets whose classification gain is negative:

$$\mathrm{NEG}(\langle \text{flu} = \text{yes} \rangle) = X_1 \cup X_4 \cup X_5 = \{x_1, x_4, x_5, x_6, x_8, x_{10}\};$$

- computing the unlearnable region of the concept by selecting the positive examples of the concept in the negative region:

$$\mathrm{URL}(\langle \text{flu} = \text{yes} \rangle) = \{x_8\}.$$

2. Initialize CS to be the current training set: $\{x_1, x_2, \ldots, x_{10}\}$.
3. Calculate the significance value of each attribute-value pair with respect to the concept and CS. The results are shown in Table 2. In the calculation, probabilities are estimated using frequencies.

Table 2. SIG values of attribute-value pairs in the first iteration.

Attribute-Value Pair	SIG	Complement of the Pair	SIG of the Complement
⟨headache = yes⟩	0.08	⟨headache = no⟩	−0.08
⟨pain = yes⟩	0.06	⟨pain = no⟩	−0.06
⟨temperature = normal⟩	−0.08	⟨temperature ≠ normal⟩	0.08
⟨temperature = high⟩	0	⟨temperature ≠ high⟩	0
⟨temperature = very high⟩	0.08	⟨temperature ≠ very high⟩	−0.08

4. Select pair ⟨temperature ≠ normal⟩ from the table because it has the highest SIG value (0.08) and it covers the most training examples in CS among the pairs whose SIG value is 0.08. Since ⟨temperature ≠ normal⟩ is not single-valued, the step of dynamic grouping is skipped.
5. Update CS by removing the examples that are not covered by ⟨temperature ≠ normal⟩. The new CS is $\{x_2, x_3, x_4, x_6, x_7, x_8, x_9, x_{10}\}$.
6. Because CS contains negative examples and its positive examples ($\{x_2, x_3, x_7, x_8\}$) do not all fall in URL(⟨flu = yes⟩), which is $\{x8\}$, go back to Step 3 to calculate the SIG values of the remaining attribute-value pairs based on the new CS. The results are shown in Table 3.

Table 3. SIG values of attribute-value pairs in the second iteration.

Attribute-Value Pair	SIG	Complement of the Pair	SIG of the Complement
⟨headache = yes⟩	0.125	⟨headache = no⟩	-0.125
⟨pain = yes⟩	0.125	⟨pain = no⟩	-0.125
⟨temperature = high⟩	−0.0625	⟨temperature = very high⟩	0.0625

[3]In the computation, probabilities are estimated using frequencies based on the training data.

In the table, ⟨temperature = very high⟩ is considered to be the complement of ⟨temperature = high⟩ because ⟨temperature ≠ normal⟩ has been chosen previously for the rule being generated and all examples in CS satisfy the condition of ⟨temperature ≠ normal⟩.

7. Two pairs (⟨headache = yes⟩ and ⟨pain = yes⟩) in Table 3 have the highest SIG value. Select ⟨pain = yes⟩ because it has better coverage.
8. Since the attribute in the selected pair is nominal and the pair is single-valued, dynamic grouping of values can be conducted[4] by the following.
   - Set TCS to be the set of examples in CS that are not covered by ⟨pain = yes⟩, which is $\{x_4, x_6, x_8, x_{10}\}$.
   - Calculate the SIG value of ⟨pain = no⟩ based on TCS. The value is 0.
   - Since the value is not as high as the SIG value for ⟨pain = yes⟩, no value is grouped.
9. Update CS by removing the examples that are not covered by the selected pair ⟨pain = yes⟩. The new CS is $\{x_2, x_3, x_7, x_9\}$.
10. Because the new CS contains a negative example (x_9) and its positive examples do not fall in URL(⟨flu = yes⟩), go back to Step 3 to select another attribute-value pair based on the new CS, which leads to selection of ⟨headache = yes⟩.
11. Update CS by removing the examples that are not covered by the selected pair ⟨headache = yes⟩. The new CS is $\{x_2, x_3\}$.
12. Because the new CS contains only positive examples of the concept, the specialization process for generating the first rule stops. The generated rule, denoted as r_1, is "if ⟨temperature ≠ normal⟩ and ⟨pain = yes⟩ and ⟨headache = yes⟩, then ⟨flu = yes⟩".
13. Post-prune rule r_1 as follows.
    - Computing the rule quality value for r_1:

$$Q(r_1) = \log_{10} \frac{(2 + 0.5) \times (10 - 2 - 4 + 2 + 0.5)}{(2 - 2 + 0.5) \times (4 - 2 + 0.5)} = 1.114.$$

    - Considering removing attribute-value pair ⟨headache = yes⟩ from the condition part of r_1, which results in rule r_{1a}. Calculate the rule quality value of r_{1a} as follows:

$$Q(r_{1a}) = \log_{10} \frac{(3 + 0.5) \times (10 - 4 - 4 + 3 + 0.5)}{(4 - 3 + 0.5) \times (4 - 3 + 0.5)} = 0.932.$$

      Since $Q(r_{1a})$ is less than $Q(r_1)$, pair ⟨headache = yes⟩ should not be removed.
    - Considering removing attribute-value pair ⟨pain = yes⟩ from r_1, which results in rule r_{1b}. The rule quality value of r_{1b} is calculated as

$$Q(r_{1b}) = \log_{10} \frac{(2 + 0.5) \times (10 - 2 - 4 + 2 + 0.5)}{(2 - 2 + 0.5) \times (4 - 2 + 0.5)} = 1.114.$$

      Since $Q(r_{1b})$ is not less than $Q(r_1)$, pair ⟨pain = yes⟩ is removed. The resulting rule, i.e., r_{1b}, is "if ⟨temperature ≠ normal⟩ and ⟨headache = yes⟩, then ⟨flu = yes⟩".
    - The pruning process is recursively applied to r_{1b}, but no pair can be removed. Therefore, r_{1b} is regarded as the first generated rule. We rename it as r_1 for later use.
14. Set CS to be the training examples that are not covered by the already generated rule, r_1, which is $\{x_1, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$.
15. Since CS contains positive examples of the concept and not every positive example in CS belongs to URL(⟨flu = yes⟩), repeat the induction process starting from Step 3 to generate other rules for the concept until CS contains no positive examples or the positive examples it contains belong to URL(⟨flu = yes⟩).

[4]Because attribute pain has only two values, only one value remains after ⟨pain = yes⟩ is selected. Therefore, dynamic grouping is not actually conducted in the execution of ELEM2 because grouping with all remaining values does not make sense. We list the dynamic grouping steps in this example only for the illustration purpose.

Table 4. The set of induced rules.

Rules	Condition	Decision	Rule Quality
r_1	⟨temperature ≠ normal⟩ and ⟨headache = yes⟩	⟨flu = yes⟩	1.114
r_2	⟨temperature ≠ normal⟩	⟨flu = yes⟩	0.699
r_3	⟨temperature = normal⟩	⟨flu = no⟩	0.699
r_4	⟨headache = no⟩ and ⟨temperature ≠ very high⟩	⟨flu = no⟩	0.623

To learn rules for concept ⟨flu = no⟩, the process is repeated. The final set of rules for both concepts is shown in Table 4.

7.2. Classification

Suppose there is a new patient with symptoms described in Table 5. We would like to determine whether the patient has flu by using the induced rules to classify the patient.[5] To classify this new example, we match the example with the induced rules. Two rules, r_2 and r_4, are found to be matched with the example. Because the matched rules classify the example into two different classes, we compute a decision score for each of the two classes as follows:

$$\text{DS}(\langle\text{flu} = \text{yes}\rangle) = Q(r_2) = 0.699,$$
$$\text{DS}(\langle\text{flu} = \text{no}\rangle) = Q(r_4) = 0.623.$$

Since DS(⟨flu = yes⟩) > DS(⟨flu = no⟩), the new example is classified into class ⟨flu = yes⟩.

Table 5. A new example.

Patients	Headache	Pain	Temperature	Flu
x	no	yes	high	?

8. EMPIRICAL EVALUATION

To evaluate the system, we have conducted experiments with ELEM2 on a number of data sets taken from the UCI repository [12]. Our objective is to check the usefulness of the rule sets generated by ELEM2 in terms of their predictive accuracy. In our evaluation, we compare ELEM2 to C4.5 and CN2[6].

We first conducted experiments on the MONK's problems. The MONK's problems are three artificially constructed problems. Detailed descriptions of these problems can be found in [12]. Problem 1 (M_1) is in standard disjunctive normal form and is supposed to be easily learnable by a symbolic learning algorithm such as C4.5, CN2, and ELEM2. Conversely, problem 2 (M_2) is similar to parity problems. It combines different attributes in a way which makes it complicated to describe in DNF or CNF using the given attributes only. Problem 3 (M_3) is also in DNF. However, the training data of Problem 3 contains 5% misclassification noise and serves to evaluate the algorithms under the presence of noise.

On each of the MONK's problems, we presented C4.5, CN2, or ELEM2 with a training set and examined the predictive accuracy of the induced rules on a test set. Table 6 reports the

[5]The data set used in this example was artificially generated for illustration purposes. Hence, the rules induced from the data set are of no medical value and are not supposed to be used in practice.

[6]We chose C4.5 and CN2 because they are two standard machine learning algorithms and their codes are available to us. C4.5 has been widely used to compare with newly-proposed algorithms. We chose CN2 also because CN2 is similar to ELEM2 in the aspect that they both use the "sequential covering" learning strategy. C4.5 can generate both decision trees and decision rules. We chose to use generation of decision rules. When running C4.5, we used option '$-s$' to allow grouping of attribute values. Other options in C4.5 were kept as default settings. For CN2, we used the default settings which generate unordered sets of rules and use Laplacian error estimate as the search heuristic. For ELEM2, we used Version 3.4.

testing accuracy of each algorithm on each MONK's problem. In all cases, ELEM2 produces more accurate rules than CN2. For the simple problem M_1, both C4.5 and ELEM2 gives 100% accurate predictions. For the difficult problem M_2, all three algorithms do not produce accurate classification rules since the concept description languages used by the three algorithms do not fit the problem well. Nevertheless, ELEM2 has the best classification accuracy among the three algorithms. Problem M_3 is not difficult for the three learners, but it involves noisy data. From the table, we can see that ELEM2 better handles the noise in this problem than C4.5 and CN2.

Table 6. Comparison in predictive accuracy.

Algorithms	M_1	M_2	M_3
C4.5	100%	64.8%	94.4%
CN2	98.6%	75.7%	90.7%
ELEM2	100%	81.7%	96.8%

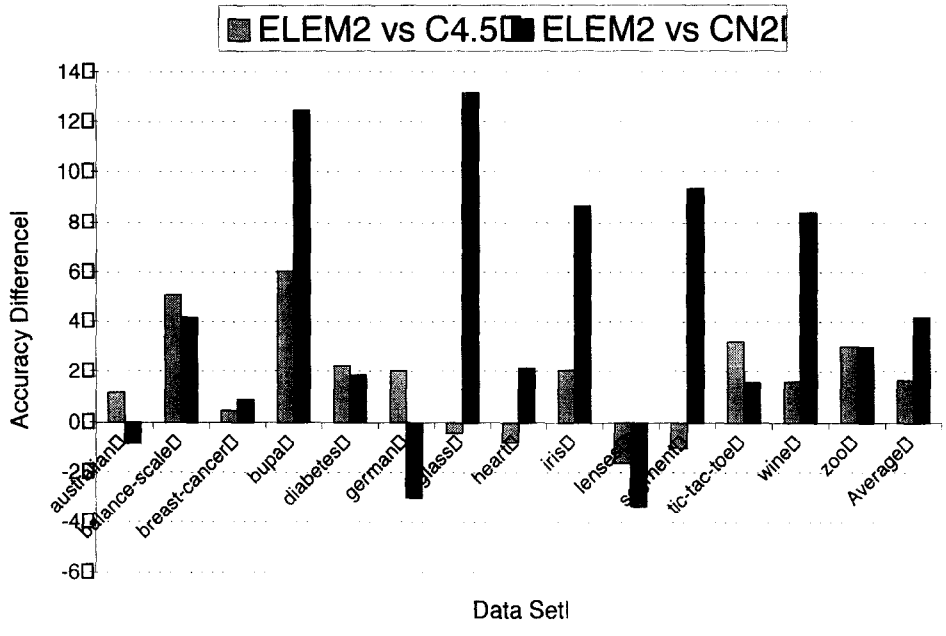

Figure 1. Performance comparison of ELEM2 with C4.5 and CN2.

To further compare ELEM2 with C4.5 and CN2, we conducted experiments with these algorithms on 14 real-world data sets from the UCI repository [12]. Figure 1 shows the accuracy difference between ELEM2 and C4.5 and the accuracy difference between ELEM2 and CN2. The accuracy of an algorithm on a data set is obtained by using ten-fold cross validation. A positive bar in the figure indicates that ELEM2 performs better than C4.5 or CN2 on the corresponding data set. The results show that ELEM2 outperforms C4.5 on 10 of the 14 data sets and that it outperforms CN2 on 11 of the 14 data sets. The last column represents the average of the accuracy differences on the 14 data sets, which indicates that ELEM2 is generally able to learn more accurate representations of the hidden patterns in the data than C4.5 and CN2.

9. CONCLUSIONS

We have presented ELEM2, a method for inducing classification rules from a set of examples. ELEM2 employs a number of new strategies to improve the predictive performance of generated rules. We proposed a significance function for evaluating attribute-value pairs based on the degree of their relevance to a target concept. A new method for handling inconsistent training examples

by determining the unlearnable region of each concept is also presented. We also propose a rule quality measure for use in the post-pruning process. Our experimental results show that ELEM2 outperforms C4.5 and CN2 in terms of predictive accuracies on most of the tested problems.

REFERENCES

1. J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA, (1993).
2. P. Clark and T. Niblett, The CN2 induction algorithm, *Machine Learning* **3**, 261–283, (1989).
3. J. Cendrowska, PRISM: An algorithm for inducing modular rules, In *Knowledge Acquisition for Knowledge-Based Systems*, (Edited by B. Gaines and J. Boose), Academic Press, (1988).
4. J.R. Quinlan, Learning efficient classification procedures and their application to chess end games, In *Machine Learning: An Artificial Intelligence Approach, Volume 1*, (Edited by R.S. Michalski, J.G. Carbonell and T.M. Mitchell), (1983).
5. J.W. Grzymala-Busse, LERS—A system for learning from examples based on rough sets, In *Intelligent Decision Support: Handbook of Applications and Advances of Rough Sets Theory*, (Edited by R. Slowinski), pp. 3–18, Kluwer Academic, (1992).
6. R. Holte, L. Acker and B. Porter, Concept learning and the problem of small disjuncts, In *Proceedings of the* 11^{th} *International Joint Conference on Artificial Intelligence*, Detroit, MI, (1989).
7. R.S. Michalski, I. Mozetic, J. Hong and N. Lavrac, The multi-purpose incremental learning system AQ15 and its testing application to three medical domains, *Proceedings of AAAI 1986*, 1041–1045, (1986).
8. A. An and N. Cercone, Discretization of continuous attributes for learning classification rules, In *Proceedings of the Third Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-99)*, Beijing, China, (1999).
9. A. An, *Analysis Methodologies for Integrated and Enhanced Problem Solving*, Ph.D. Thesis, Dept. of Computer Science, University of Regina, Regina, Canada, (1997).
10. H.J. Hamilton, N. Shan and N. Cercone, RIAC: A rule induction algorithm based on approximate classification, Technical Report CS-96-06, University of Regina, (1996).
11. S.E. Robertson and K. Sparck Jones, Relevance weighting of search terms, *Journal of the American Society for Information Science* **27**, 129–146, (1976).
12. P.M. Murphy and D.W. Aha, *UCI Repository of Machine Learning Databases*, URL: `http://www.ics.uci.edu/ mlearn/MLRepository.html`. For information contact `ml-repository@ics.uci.edu`., (1994).