ELEM2: A Learning System for More Accurate Classifications

Aijun An and Nick Cercone

Department of Computer Science, University of Waterloo
Waterloo, Ontario N2L 3G1, Canada

Abstract. We present ELEM2, a new method for inducing classification rules from a set of examples. The method employs several new strategies in the induction and classification processes to improve the predictive performance of induced rules. In particular, a new heuristic function for evaluating attribute-value pairs is proposed. The function is defined to reflect 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. Another feature of ELEM2 is that it handles inconsistent training data by defining an unlearnable region of a concept based on the probability distribution of that concept in the training data. To further deal with imperfect data, ELEM2 makes use of the post-pruning technique to remove unreliable portions of a generated rule. A new rule quality measure is proposed for the purpose of post-pruning. The measure is defined according to the relative distribution of a rule with respect to positive and negative examples. To show whether ELEM2 achieves its objective, we report experimental results which compare ELEM2 with C4.5 and CN2 on a number of datasets.

1 Introduction

Induction is a process that reasons from specific cases to general principles. Rule induction covers a special, and prevalent, case of induction, in which the results of induction are expressible as condition-action rules. A number of rule induction systems, such as C4.5 [15], AQ15 [12] and CN2 [4], have been constructed and applied to discover knowledge from collected data in different applications, yet many suffer from poor performance in prediction accuracy in many practical domains. It has been shown repeatedly that each method works best in some, but not all [2, 7]. 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.

Our objective is to work towards this direction by proposing ELEM2, a new rule induction method that employs new strategies to enhance the induction and classification processes. Similar to some other learning algorithms (such as CN2 [4] and PRISM [3]), ELEM2 generates rules from a set of training examples by searching for a hypothesis in a general-to-specific manner in which an attribute-value pair is selected at each specification step. However, ELEM2 differs from other 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 [14] 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 [8] 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 [15]. Such rules tend to have higher predictive error rates [10]. We propose an alternative evaluation function. The new function is defined in terms of both an 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.

Secondly, 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.

Thirdly, ELEM2 employs a new rule quality measure for the purpose of handling imperfect data by post-pruning generated rules. Post-pruning is a technique that prevents a generated rule from *overfitting* the data.[1] There have been several post-pruning methods in the literature. For example, C4.5 [15] 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 [12], 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. A

[1] We say that a rule *overfits* the training examples if it performs well over the training examples but less well over the entire distribution of instances (i.e., including instances beyond the training set).

problem with AQ15 is that it is hard to specify when to stop the pruning. 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 formula is chosen from four alternatives that represent different kinds of distributions. We choose from these four alternatives the one with the most information and the best experimental results.

The rest of the paper is organized as follows. In Section 2, we present the strategies that ELEM2 uses for selecting 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. Section 7 reports some experimental results for evaluating ELEM2. We conclude the paper with a summary of ELEM2 and some suggestions for future work.

2 Selection of Attribute-Value Pairs

An *attribute-value pair* is a relation between the attribute and its values, which is represented in the form $\langle a\ rel\ v\rangle$, where a is an attribute, rel denotes a relational operator (e.g. $=, \neq, \leq, >, \cdots$, or $\in$) and v is a specific value or a set of values of the attribute. 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 in t, respectively. We define that the *complement* of an attribute-value pair t is the attribute-value pair $\langle a_t, \neg rel, v_t\rangle$, where $\neg$ 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.

Grouping discrete attribute values There are two kinds of discrete attributes: *nominal attributes* and *ordered discrete attributes*. The values of a nominal attribute do not show any inherent order among themselves, while an ordered discrete attribute has an ordered set of values. For example, an attribute *colour* with a set of values being $\{red, blue, yellow, orange\}$ is a normal discrete attribute, while an attribute *temperature* with a value set being $\{low, medium, high\}$ is an ordered discrete attribute.

For a nominal attribute, ELEM2 uses a dynamic grouping strategy, i.e., the value groups are determined after an attribute-value pair of this attribute is chosen as a candidate selector during the induction process. The initial search space of attribute-value pairs before induction is made of attribute-value pairs with single values. Details about how to dynamically group these single-valued attribute-value pairs will be described in the ELEM2 rule induction algorithm in Section 5.

For an ordered discrete attribute, grouping of its values is carried out before the induction process commences. The grouping method is based on $n-1$ binary splits, where n is the number of all possible values of the attribute. Let $\{s_1, s_2, ..., s_n\}$ be the set of values of an ordered discrete attribute a, where $s_i < s_{i+1}$ for $i = 1, ..., n-1$. For each value $s_i (i = 1, ..., n-1)$, two groups: $\{s_1, ..., s_i\}$ and $\{s_{i+1}, ..., s_n\}$, are generated, i.e., two attribute-value pairs: $\langle a \in \{s_1, ..., s_i\}\rangle$ and $\langle a \in \{s_{i+1}, ..., s_n\}\rangle$, are obtained. In this way, a total of $2(n-1)$ pairs can be obtained for an attribute with n possible values. We put half of these pairs into the search space for the selection process. For each value $s_i (i = 1, ..., n-1)$, only the first pair, $\langle a \in \{s_1, ..., s_i\}\rangle$, is included in the search space. We drop off the second pair because it is the complement of the first pair and its degree of relevance to a target concept, measured by our evaluation function described in the next section, is the additive inverse of the degree of relevance for the first pair. We can use the evaluation information about one pair to judge the other one. Therefore, in our method, only $n-1$ attribute-value pairs need to be examined during induction instead of $2^n - 2$ possibilities.

Grouping continuous attribute values In ELEM2, continuous attributes are discretized by using user-supplied discretization formulas or by applying one of the automatic discretization methods[6]. Suppose $\{x_1, x_2, ..., x_n\}$ is the set of cut-points for a discretized continuous attribute a. Our grouping method for this kind of attributes is similar to the one for an ordered discrete attribute. For each cut-point x_i, $(i = 1, ..., n)$, two attribute-value pairs: $\langle a \leq x_i\rangle$ and $\langle a > x_i\rangle$, are generated. The search space consists of only the first pairs in these binary splits. Therefore, the search space has a total of n attribute-value pairs for a continuous attribute with n cut-points.

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 av is *relevant* to a concept c with respect to a set, S, of examples if

$$P(av) > 0 \text{ and } P(c|av) \neq P(c),$$

where $P(av)$ denotes the probability that an example in S satisfies the relation expressed by av, $P(c)$ denotes the probability of the examples occurring in S that belong to concept c, and $P(c|av)$ is the probability that an example in S belongs to c given that the example satisfies av.[2] Under this definition, av is relevant to the concept c if it can change the probability of c, or in other words, if c is conditionally dependent on av.

In a set of training samples, there may exist more than one attribute-value pair that are relevant to a concept. Some pairs may be strongly relevant, while others may not be 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

$$SIG_c(av) = P(av)(P(c|av) - P(c)). \tag{1}$$

According to this definition, if $P(c|av) = P(c)$, i.e., av is not relevant to the concept c, then the degree of relevance of av to c is equal to 0; if $P(c|av) \neq P(c)$, i.e., av is relevant to c, then the degree of relevance is proportional to both the difference between $P(c|av)$ and $P(c)$ and the coverage of av 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 av with respect to the target concept c; if the value is negative, the lower the value, the more relevant the attribute-value pair $\neg av$ (i.e., the complement of pair av) with respect to c. We use $P(av)$ 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, it can be proved that [1]

$$SIG_c(av_i) = -SIG_c(\neg av_i). \tag{2}$$

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

[2] The probabilities here are determined by analysing a set S of training examples. Therefore, they can be considered as posterior probabilities.

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 [9]. Let $R = \{X_1, X_2, \cdots, X_n\}$, 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 $P(c|X_i)$, 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 [9]:

$$CG_c(X_i) = P(c|X_i) - P(c),$$

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

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

which means, if $CG_c(X_i) < 0$, then X_i belongs to the negative region of c. The *unlearnable region* of a concept c, denoted as $ULR(c)$, is defined as the set of positive examples of c that exist in $NEG(c)$.

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 is stopped. This prevents ELEM2 from learning from the inconsistent examples that do not provide positive classification gain.

4 Post-Pruning Induced Rules

Systems interacting with real-world data must address the issues raised by imperfect training data. The training data are imperfect when there is noise in the data or when the number of training examples is too small to produce a representative sample of the true target function. The noise or coincidental regularities within the training data can lead to serious problems for the learning task. One problem is that learning algorithms can be misled by the imperfect data to produce long and distorted concept descriptions. By trying to fit every example into the concept descriptions, faulty and noisy examples are included, leading to cluttered and complex concept descriptions which cover a small number of anomalous examples in the training set. These descriptions are commonly known as *small disjuncts* [10]. 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.

	Positive	Negative	Total
Covered by r	m	$n-m$	n
Not covered by r	$M-m$	$N-M-n+m$	$N-n$
Total	M	$N-M$	N

Table 1. Example Distribution for Rule r

In ELEM2, a rule quality measure is used as a criterion for post-pruning. The measure is defined according to the relative distribution of a rule with respect to the positive and negative examples it covers. We consider several alternatives when defining the rule quality measure. Given a set S of training data and a rule r learned from S, let N denote the number of examples in S, M be the number of positive examples in S, n denote the number of examples covered by r, and m be the number of positive examples covered by r. Consider the contingency table of example distribution for r (See Table 1). We assume that rule quality formulae have to in some way reflect the relative distribution of rules with respect to positive and negative examples. Specifically, four formulae can be derived from the table as follows:

$$Q_1 = \frac{(\frac{m}{M})}{(\frac{n}{N})} \tag{3}$$

$$Q_2 = \frac{(\frac{m}{M})}{(\frac{n-m}{N-M})} \tag{4}$$

$$Q_3 = \frac{(\frac{m}{M-m})}{(\frac{n}{N-n})} \tag{5}$$

$$Q_4 = \frac{(\frac{m}{M-m})}{(\frac{n-m}{N-M-n+m})} \tag{6}$$

Informally, for the given rule r, Q_1 represents the ratio of the proportion of positive examples which r covers to the proportion of the entire training set S which r covers, while Q_2 represents the ratio of the proportion of positive examples to that of negative examples. Q_3 represents the ratio between the "positive odds" for the rule (*i.e.* the ratio between the number of positive examples which r covers and the number which it does not cover) and the "total odds" for r, while Q_4 represents the ratio between the rule's positive odds and its "negative odds". Thus, Formulae Q_1 and Q_2 are related by using proportions, while Q_3 and Q_4 use odds; but Q_1 and Q_3 respectively are related by comparing the positive example distribution of a rule to its entire collection distribution, while Q_2 and Q_4 are related by comparing positive and negative distributions.

Analyze the formulae in another way. Given a training set and a concept in it, N and M are independent with the rule generated from the set. Thus, Q_1 is actually determined by the apparent accuracy of the rule, which is usually estimated as $\frac{m}{n}$. Since the apparent accuracy does not usually reflect the predictive accuracy, Q_1 is not a good choice for measuring rules' quality because our objective is to learn rules that classify *new* examples well. Q_2 has the flavour of measuring both coverage and accuracy. It can be specified in terms of probabilities as follows:

$$Q_2 = \frac{P(satisfy(r,e)|positive(e))}{P(satisfy(r,e)|negative(e))}$$

where P denotes probability, $satisfy(r,e)$ means r is satisfied by example e, $positive(e)$ means that e is a positive example, and $negative(e)$ means e is negative. Similarly, Q_4 can be represented as:

$$Q_4 = \frac{P(satisfy(r,e)|positive(e))P(\neg satisfy(r,e)|negative(e))}{P(satisfy(r,e)|negative(e))P(\neg satisfy(r,e)|positive(e))}$$

Although $P(\neg satisfy(r,e)|negative(e))$ and $P(\neg satisfy(r,e)|positive(e))$ depend on $P(satisfy(r,e)|negative(e))$ and $P(satisfy(r,e)|positive(e))$ respectively, the distinction between the two formulae concerns *explicit* recognition of rule unsatisfaction in calculation of probabilities. We argue that Q_4 is better than Q_2 in the sense that Q_4 explicitly contains more information than Q_2. A similar conclusion can be drawn between Q_1 and Q_3 that Q_3 explicitly contains more information than Q_1.

The difference between Q_3 and Q_4 is that Q_3 compares the positive distribution of a rule to its entire set distribution, while Q_4 compares positive and negative distributions. To see which one is a better measure for rule quality, we conducted experiments. In the experiments, we run ELEM2 with measure Q_3 and ELEM2 with Q_4 for ten-fold evaluation on 14 randomly selected datasets. The results (presented in [1]) show that the program with Q_4 gives better predictive accuracy means and accuracy deviations than the one with Q_3 on most of the tested datasets Therefore, we choose Q_4, which represents the ratio between the rule's positive odds and its negative odds (see Formula 6 and Table 1), as a criterion for measuring rule qualities and this criterion is used by ELEM2 to decide when to stop post-pruning. The post-pruning process is as follows:

1. Sort the selected pairs in a rule in the reverse order to that in which they were selected;
2. Check the pairs in this order to see if they can be removed without causing the rule quality to decrease. If yes, remove them.

5 The ELEM2 Induction Algorithm

ELEM2 induces rules using the *separate and conquer* strategy, 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 briefly described as follows. If the training set contains examples of more than one concept, then for each concept c:

1. Compute the unlearnable region of the concept: $ULR(c)$;
2. Let CS be the current training set;
3. Calculate the significance value, $SIG(av_i)$, of each attribute-value pair av_i in the attribute-value pair space with respect to CS;
4. Select the pair av for which $|SIG(av)|$ is a maximum;
5. If the attribute a in the selected pair av is a nominal attribute and av's SIG value is positive, 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 the selected pair av the pairs with SIG value greater than or equal to $|SIG(av)|$;
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 $ULR(c)$. The induced rule r is a conjunction of all the attribute-value pairs selected;
8. Post-prune the induced rule r using the rule quality measure $Q_4(r)$;
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 $ULR(c)$.

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 Induced Rules

In general, rules induced from a set of data are used to classify new objects into an appropriate concept. The central task of a classification algorithm is to determine if an example satisfies a rule. This is also referred to as the example matching a rule. 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 rules), 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 the system activates a conflict resolution scheme for the best decision. The conflict resolution scheme computes a decision score for each concept that the matched rules indicate. The decision score of a concept c is defined as:

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

where r_i is a matched rule that indicates c, n is the number of this kind of rules, and $RB(r_i)$ is the reliability of rule r_i, which is defined as

$$RB(r_i) = log(Q_4(r_i))$$

After computing the decision scores for all the concepts indicated by the matched rules, 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:

$$PMS(r) = \frac{\sum_{k=1}^{m} RB(mav_k)}{\sum_{j=1}^{n} RB(av_j)} \times RB(r),$$

where av_j $(j = 1, \cdots, n)$ denotes a pair in r, mav_k $(k = 1, \cdots, m)$ denotes a matched pair in r, $RB(r)$ is the reliability of rule r, $RB(mav_k)$ and $RB(av_j)$ stand for the reliabilities of pairs mav_k and av_j respectively. The reliability of an attribute-value pair av is defined similar to the definition of the reliability of a rule as:

$$RB(av) = log\frac{P(av|pos)P(\neg av|neg)}{P(av|neg)P(\neg av|pos)},$$

where $P(av|pos)$ denotes the probability that an example satisfies av conditioned on the example is positive, $P(av|neg)$ denotes the corresponding probability for negative examples, $P(\neg av|pos)$ is the probability that an example does not satisfy av conditioned on the example is positive, and $P(\neg av|neg)$ is the corresponding probability given that the example is negative. 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:

$$DS(c) = \sum_i PMS(r_i),$$

where $i = 0$ 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 Empirical Evaluation

ELEM2 has been implemented in C under Unix environments. To evaluate the system, we have conducted experiments with ELEM2 on a number of actual data sets taken from the UCI repository [13]. Our objective is to check the usefulness of the rule sets generated by ELEM2 in terms of their predictive accuracy. We report the experimental results and compare them with the results from C4.5 and CN2.

7.1 Evaluation Methods

One method for evaluating a learning system is to artificially construct a training and a test dataset so that the characteristics of the training data, such as the complexity of concepts and the noise level of the training data, are available for analyzing the learner's capability. Three artificially designed domains, the MONK's problems, were obtained from the UCI repository to evaluate ELEM2 for this purpose. Each MONK's problem contains a training and a test dataset. The classification accuracy over the testing set is measured to show the learner's predictive performance.

Another evaluation method is x-fold cross validation. At the expense of computational resources, this method gives a more reliable estimate of the accuracy of a learning algorithm than a single run on a held-out test set. x-fold cross validation involves randomly partitioning the database into x disjoint data sets, then providing the learning algorithm with $x - 1$ of them as training data and using the remaining one as test cases. This process is repeated x times using different possible test sets. Each time a classification accuracy is obtained. The mean of the accuracies from the x runs and the standard deviation of the accuracy can then be calculated to measure testing performance. Ten-fold cross validation is used in our experiments to compare ELEM2 with two other algorithms on actual data sets.

7.2 Evaluation of ELEM2 on MONK's Domain

MONK's Problems The MONK's problems are three artificially constructed problems. They are derived from an artificial robot domain, in which robots (examples) are described by six nominal attributes [16]. The sizes of the value sets of the six attributes are 3, 3, 2, 3, 4 and 2, respectively as shown below:

$$\begin{aligned}
head\_shape &\in \{round, square, octagon\}\\
body\_shape &\in \{round, square, octagon\}\\
is\_smiling &\in \{yes, no\}\\
holding &\in \{sword, balloon, flag\}\\
jacket\_color &\in \{red, yellow, green, blue\}\\
has\_tie &\in \{yes, no\}
\end{aligned}$$

Consequently, the example space contains 432 ($3 \times 3 \times 2 \times 3 \times 4 \times 2$) possible examples. The three MONK's problems, referred to as $M1, M2$, and $M3$, are all binary classification tasks defined over the same space. They differ in the type of concept to be learned and in the amount of noise in the training examples. Each problem is given by a logical description of a concept. Robots belong either to this concept or not, but instead of providing a complete concept description to the learning problem, only a subset of all 432 possible robots with its classification is given. The learning task involves generalizing over these examples and, if the particular learning technique at hand allows this, to derive a simple class description. The three MONK's problems are specially designed as follows:

- Problem M_1: $(head\_shape = body\_shape)$ OR $(jacket\_color = red)$
  From 432 possible examples (referred to as MonkTest 1, 216 positive and 216 negative), 124 (62 positive and 62 negative) were randomly selected for the training set (referred to as MonkTrain 1). There were no misclassifications in the training set.
- Problem M_2: *exactly two of the six attributes have their first value*
  From 432 possible examples (referred to as MonkTest 2, 142 positive and 290 negative), 169 (64 positive and 105 negative) were randomly selected as training examples (referred to as MonkTrain 2). Again, there was no noise in Training Set 2.
- Problem M_3: $(jacket\_color = green)$ AND $(holding = sword)$ OR $(jacket\_color$ *is NOT blue*$)$ AND $(body\_shape$ *is NOT octagon*$)$ From 432 possible examples (referred to as MonkTest 3, 228 positive and 204 negative), 122 (60 positive and 62 negative) were selected randomly, and among them there were 5% misclassifications, i.e. noise in the training set (referred to as MonkTrain 3).

Problem 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 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 is again in DNF and serves to evaluate the algorithms under the presence of noise.

Performance Comparison with C4.5 and CN2 A performance comparison of ELEM2 with C4.5 and CN2 has been conducted on the MONK's problems.

All three systems provide the facilities for inducing rules from a training set and evaluating the induced rules on a test set.[3] In the experiment on each problem M_i, we presented each learning system with the training set MonkTrain i, recorded the number of rules generated from each algorithm, and examined the performance of the induced rules on the test set MonkTest i.

Algorithms	MonkTest 1	MonkTest 2	MonkTest 3
C4.5	100%	64.8%	94.4%
CN2	98.6%	75.7%	90.7%
ELEM2	100%	78.7%	96.3%

Table 2. Comparison in Predictive Accuracy

A comparison in terms of the percentage of the test examples correctly classified is illustrated in Table 2. 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 problem M_2, which is difficult to describe in CNF or DNF using the given attributes only, 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.

7.3 Comparison of ELEM2 with C4.5 and CN2 on Actual Data Sets

To further compare ELEM2 with C4.5 and CN2, we have conducted experiments with these algorithms on 14 real-world data sets from the UCI repository [13]. Description of these datasets in terms of their number of concepts, number of condition attributes, number of examples and application domain is given in Table 3. Table 4 shows the results of ten-fold evaluation of C4.5, CN2 and ELEM2 on the 14 data sets. For each data set and each algorithm, we report the average of accuracies from the ten runs and their standard deviation.[4] When running C4.5, we used the option '-s' to allow grouping of attribute values. Other

[3] C4.5 can generate both decision trees and decision rules. We chose to use generation of decision rules. The decision rules are generated from unpruned decision tree(s) and are then generalized using pessimistic pruning technique. CN2 can generate both ordered and unordered sets of rules. We used the default settings which generate unordered sets of rules and use Laplacian error estimate as the search heuristic.

[4] The standard deviation is calculated using the following formula:

$$stdev = \sqrt{\frac{n\sum x_i^2 - (\sum x_i)^2}{n(n-1)}}$$

Datasets	No. of Concepts	No. of Cond. Attr.	No. of Examples	Domain
australia	2	14	690	Credit card application approval
balance-scale	3	4	625	Balance scale classification
breast-cancer	2	9	683	Medical diagnosis
bupa	2	6	345	Liver disorder database
diabetes	2	8	768	Medical diagnosis
german	2	20	1000	Credit database to classify people as good or bad credit risks
glass	6	9	214	Glass identification for criminological investigation
heart	2	13	270	Heart disease diagnosis
iris	3	4	150	Iris plant classification
lenses	3	4	24	Database for fitting contact lenses
segment	7	18	2310	image segmentation database
tic-tac-toe	2	9	958	Tic-Tac-Toe Endgame database
wine	3	13	178	Wine recognition data
zoo	7	16	101	Animal classification

Table 3. Description of Datasets.

options in C4.5 were kept as default settings. For CN2, all runs used default parameters so that Laplacian estimate was employed as the search heuristic and unordered rules were generated, which means that the improved version of CN2 was used.

The best result for each problem is highlighted in boldface in the table. Among the 14 problems, ELEM2 gives the best results in terms of predictive accuracy for 10, C4.5 for 3 and CN2 for 1. In terms of standard deviation, on 7 out of 12 data sets, ELEM2 gives the smallest number; C4.5 does it on 5 data sets; and CN2 on 2. At the bottom of the table, the average of the accuracy means or deviations of each algorithm over the 14 datasets indicate that ELEM2 is generally able to learn more accurate and more stable representations of the hidden patterns in the data than C4.5 and CN2.

8 Conclusions

We have presented ELEM2, a new method for inducing classification rules from a set of examples. The method 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

where x_i $(i = 1, 2, \cdots, n)$ is the accuracy from the ith run on a data set. Here, for ten-fold cross validation, $n = 10$.

	Accuracy Mean			Accuracy Standard Deviation		
Datasets	C4.5	CN2	ELEM2	C4.5	CN2	ELEM2
australian	82.9%	84.92%	**86.52%**	5.45%	6.19%	**3.55%**
balance-scale	78.10%	79.05%	**81.14%**	5.63%	**3.23%**	5.29%
breast-cancer	95.6%	95.17%	**96.19%**	**1.56%**	2.77%	2.51%
bupa	68.4%	62.00%	**69.24%**	**4.00%**	9.53%	7.88%
diabetes	70.8%	71.21%	**74.10%**	5.53%	6.12%	**4.82%**
german	69.3%	**74.30%**	74.00%	**3.40%**	4.90%	4.62%
glass	68.7%	55.17%	**72.88%**	13.96%	12.11%	**10.11%**
heart	**80.8%**	77.85%	79.63%	**6.93%**	10.51%	7.46%
iris	**95.3%**	88.68%	94.67%	6.31%	8.34%	**5.26%**
lenses	73.3%	75.01%	**76.67%**	34.43%	**22.56%**	26.29%
segment	**96.6%**	86.24%	95.93%	**0.84%**	2.99%	2.14%
tic-tac-toe	96.8%	98.44%	**99.17%**	1.79%	1.51%	**1.28%**
wine	92.8%	86.00%	**97.78%**	7.44%	5.91%	**3.88%**
zoo	92.1%	92.09%	**98.00%**	6.30%	7.87%	**4.22%**
AVERAGE	82.96%	80.43%	**85.42%**	7.39%	7.04%	**6.38%**

Table 4. Performance Comparison of ELEM2 with C4.5 and CN2.

determining the unlearnable region of each concept is also presented. To further handle imperfect training data, post-pruning generated rules is used to prevent rules from overfitting the training data. A new rule quality measure based on example distribution is proposed as a criterion for stopping the post-pruning process. We have conducted empirical evaluation of ELEM2 on a number of designed and real-world databases. The results show that ELEM2 outperforms C4.5 and CN2 in terms of predictive accuracies on most of the tested problems. In future work, we will investigate how much each of the new ideas employed in ELEM2 contributes to the performance improvement. We also plan to clarify the bias of ELEM2. ELEM2 reduces the error rates on many data sets, but not all. More studies need to be done to find out the relations between the algorithm and the nature of problems.

9 Acknowledgements

This research was conducted when the first author was a PhD student at the University of Regina under the supervision of Professors Nick Cercone and Christine Chan. We would like to acknowledge the financial support received from Natural Science and Engineering Research Council of Canada (NSERC) and Telecommunications Research Laboratories (TRLabs). We would also like to thank Mr. Ning Shan for his suggestions on early versions of this work.

References

1. An, A. 1997. *Analysis Methodologies for Integrated and Enhanced Problem Solving.* Ph.D. Thesis, Dept. of Computer Science, University of Regina, Regina, Canada.
2. Brodley, C.E. 1993. "Addressing the Selective Superiority Problem: Automatic Algorithm/model Class Selection." *Proceedings of the 10th Machine Learning Conference.* pp.17-24.
3. Cendrowska, J. 1988. "PRISM: An Algorithm for Inducing Modular Rules". In Gaines, B. and Boose, J. (eds.): *Knowledge Acquisition for Knowledge-Based Systems.* Academic Press.
4. Clark, P. and Niblett, T. 1989. "The CN2 Induction Algorithm". *Machine Learning*, 3, pp.261-283.
5. Cooper, W.S. 1973. "On Selecting a Measure of Retrieval Effectiveness." *Journal of the American Society for Information Science.* Vol.24.
6. Creecy, R.H., Masand, B.M., Smith, S.J. and Waltz, D.L. 1992. "Trading MIPS and Memory for Knowledge Engineering". *Communications of the ACM*, 35, pp.48-64.
7. Domingos, P. 1995. "Rule Induction and Instance-Based Learning: A Unified Approach." *IJCAI-95.* Montreal, Canada. pp.1226-1232.
8. Grzymala-Busse, J.W. 1992. "LERS-A System for Learning From Examples Based on Rough Sets", in Slowinski, R.(ed.): *Intelligent Decision Support: Handbook of Applications and Advances of Rough Sets Theory*, Kluwer Academic Publishers, pp.3-18.
9. Hamilton, H.J., Shan, N. and Cercone, N. 1996. "RIAC: A Rule Induction Algorithm Based on Approximate Classification". *Technical Report CS-96-06*, University of Regina.
10. Holte, R., Acker, L. and Porter, B. 1989. "Concept Learning and the Problem of Small Disjuncts". *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, Michigan.
11. Kerber, R. 1992. "ChiMerge: Discretization of Numeric Attributes", *Proceedings of the 10th National Conference on Artificial Intelligence AAAI-92*, San Jose, CA.
12. Michalski, R.S., Mozetic, I., Hong, J. and Lavrac, N. 1986. "The Multi-Purpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains". *Proceedings of AAAI 1986.* pp.1041-1045.
13. Murphy, P.M. and Aha, D.W. 1994. *UCI Repository of Machine Learning Databases.* URL: http://www.ics.uci.edu/ mlearn/MLRepository.html. For information contact ml-repository@ics.uci.edu.
14. Quinlan, J.R. 1983. "Learning efficient classification procedures and their application to chess end games". In Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (eds.): *Machine Learning: An Artificial Intelligence Approach.* Vol.1.
15. Quinlan, J.R. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers. San Mateo, CA.
16. Wnek, J., Sarma, J., Wahab, A. and Michalski, R. 1990. "Comparison Learning Paradigms via Diagrammatic Visualization: A Case Study in Single Concept Learning Using Symbolic, Neural Net and Genetic Algorithm Methods". *Technical Report*, Computer Science Department, George Mason University.

This article was processed using the LaTeX macro package with LLNCS style