A heuristic for learning decision trees and pruning them into classification rules [*]

José Ranilla, Oscar Luaces and
Antonio Bahamonde
Artificial Intelligence Center
University of Oviedo at Gijón
E-33271 Gijón, Spain
E-mail: {ranilla,oluaces,antonio}@aic.uniovi.es

Abstract
Let us consider a set of training examples described by continuous or symbolic attributes with categorical classes. In this paper we present a measure of the potential quality of a region of the attribute space to be represented as a rule condition to classify unseen cases. The aim is to take into account the distribution of the classes of the examples. The resulting measure, called *impurity level*, is inspired by a similar measure used in the instance-based algorithm IB3 for selecting suitable paradigmatic exemplars that will classify, in a nearest-neighbor context, future cases. The features of the impurity level are illustrated using a version of Quinlan's well-known C4.5 where the information-based heuristics are replaced by our measure. The experiments carried out to test the proposals indicate a very high accuracy reached with sets of classification rules as small as those found by RIPPER.

Keywords: Machine learning, classification rules, pruning, decision trees, impurity level.

1. Introduction

Let us start with a collection of training examples described by a finite set of attributes whose values can be either numeric or symbolic. One such attribute is the class; here we expect classes to be symbolic labels. In this environment, a machine learning algorithm must induce a function able to predict the class of unseen cases. To build this function we assume that training examples are a representative sample of the future cases that must be correctly classified. Additionally, we assume a kind of continuity in class distributions; that is, in the surroundings of a set of near examples of the same class, we hope to find cases of that class, this is called the similarity hypothesis [30]. In this paper, we present a measure called *impurity level* that it based on this similarity hypothesis.

Given a concentration of training examples, our measure considers the impurity of their class mixing, in addition to the rarity of the accumulation in the training set. Thus, the areas concentrating most of the examples of an infrequent class will have a very low impurity level, and will be considered as a good candidate to somehow be part of the classification function to be returned by a machine learning algorithm. Another measure that also takes into account the same circumstances as our *impurity level* was proposed by Todorovski et al. in [34], it is called *weighted relative accuracy*, WRAcc.

To implement the idea of our measure, we need to define effectively the concept of example concentration. Usually, we first delimit a territory in the attribute space and then we gather the training examples falling within this space. Thus, in an instance-based scenario, the region that is nearer to one classification example than the others is a straightforward realization of the domain of that example. In fact, the impurity level is based on Aha et al.'s method to select acceptable classification instances in IB3 [1,2]. On the other hand, the conditions of a classification rule, define the set of examples fulfilling them.

We have chosen the rule approach to illustrate the qualities of our measure. We thus implemented a version of Quinlan's C4.5 [25] using the impurity level for tree growing and rule pruning. The resulting system is called IL-rules and, like C4.5-rules, it acts in two steps. First, to induce decision trees, we only replaced the information gain ratio of C4.5 by the impurity level. Then, to obtain a set of rules, the pruning mechanism imple-

[*]This research was funded in part under MCyT and Feder grants PB98-1556 and TIC2001-3579.

mented, according to the impurity level philosophy, selects the most promising conditions of each branch of the decision tree. IL-rules finally returns a subset of these optimized branches to be part of a classification function that applies the nearest rule to each new case. In this way, we have a machine learning algorithm belonging to the family of those that try to combine the advantages of rule sets and instance-based systems. An algorithm belonging to the family of Salzberg's EACH [32], revised by Wettschereck and Dietterich [35], Domingos' RISE [12], Ranilla's FAN [29,27], Luaces' INNER [20,22,21], or Del Coz's BETS [9,10].

The result of our reconstruction of Quinlan's learner is that we have interchangeable pieces available to assemble different C4.5-like machine learning systems for building and then pruning decision trees to obtain rule sets. So, we can build decision trees by means of gain ratio or impurity level, and then we can use a pruning process in order to obtain a set of rules. If we use the final stage of C4.5, we obtain rule sets whose application follows the standard procedure. Alternatively, rules will be used by means of a minimum-distance criterion whenever we use the impurity level as the driving force of the pruning algorithm.

The experiments reported at the end of the paper show that our IL-rules produces small rule sets with a high classification accuracy. To compare our scores we used C4.5-rules since our system is based on it. Additionally, given the features of the knowledge induced by our system, we also provide a comparison with Cohen's RIPPER [8], an improved version of Fürnkranz and Widmer's IREP/IREP* [14]. Finally, we compare IL-rules with a version built with the measure by Todorovski et al., WRAcc-rules.

2. A brief introduction to C4.5

In this section we summarize the basic elements of Quinlan's C4.5 algorithm. First, we briefly describe the method used to induce decision trees; our system employs it except for the use of the impurity level instead of the information gain ratio. Then, we describe the rule generation process, which is completely replaced in our algorithm by a mechanism based on the impurity level, which prunes the original rules obtained from the decision tree.

2.1. The process of building a decision tree

The method implemented in C4.5 to build decision trees constitutes the paradigm of the *divide-and-conquer* approach in the machine learning literature. It is based on a previous work of Quinlan called ID3 [24]. Both share the same central idea to build decision trees with the *Concept Learning Systems* (CLS) of Hunt [16]. Hunt's method to construct decision trees from a set of examples E is as follows:

- If E is composed of examples of one single class C, then the tree will be a leaf labeled by class C.
- If E is empty, the tree will be a leaf, but its label must be determined using heuristic approaches. C4.5 uses the most frequent class of the examples of the parent node.
- If E contains examples belonging to different classes then the tree consists of a node labeled with a *test* based on single attribute values and with as many branches as the number of possible results that can be obtained from that test. The original set E is thus divided into disjoint subsets, one for every possible result, which will be associated to the corresponding branch.

The building process continues by applying this method recursively to each subset.

Once a decision tree has been built, the classification of an example consists in applying the test of the root node to the example and, depending on the result, descending through the corresponding branch to the next node, repeating the process until one leaf is reached; the label of the leaf is the predicted class for the example.

The quality of a decision tree can be considered from two points of view: *classification accuracy* and *simplicity.* The former is a desirable property for obvious reasons, while the latter is closely related to the comprehensibility of the resulting tree, and is frequently a faithful partner of accuracy. But unfortunately, an exhaustive exploration to find the smallest tree consistent with the training data is an acknowledged NP-complete problem [17]. Consequently, most building methods are greedy algorithms in which once a test has been heuristically selected, it is not revised later.

C4.5 uses the *gain ratio* to select the attribute whose value will be tested to decide how the original training set will be partitioned. The gain ratio is a refinement of the *information gain*, used in C4.5's precursor ID3.

Several different approaches to the test selection problem have been studied. There is a section in Breslow's paper [6] especially devoted to reviewing some alternative test selection criteria. Among others, we may mention the Gini index [5], the function $G(p) = 2\sqrt{p(1-p)}$ proposed in [11], possibility-based measures [4], etc. However, none of these clearly outperforms the gain ratio used in C4.5.

The information gain, and consequently the gain ratio, is based on Shannon's information theory. In the following, we reproduce the formulas needed to compute these measures to ease the introduction of our measure later, in Section 3. Let us suppose that a training set E is composed of examples belonging to one of the C_j possible classes, where $j = 1 \dots k$. If $\text{freq}(C_j, E)$ stands for the frequency of class C_j in the set of examples E, then Equation (1) measures the average amount of information needed to identify the class of an example in E; this is also known as the *entropy* of E.

$$\text{info}(E) = -\sum_{j=1}^{k} \frac{\text{freq}(C_j, E)}{|E|} \cdot \log_2\left(\frac{\text{freq}(C_j, E)}{|E|}\right) \quad (1)$$

The training set E can be partitioned on the basis of any of the attributes involved in the description of the examples. If an attribute X with n different results E_i for the test is selected, the expected information of E is calculated as the weighted sum of every E_i entropy.

$$\text{info}_X(E) = \sum_{i=1}^{n} \frac{|E_i|}{|E|} \cdot \text{info}(E_i) \quad (2)$$

Hence, the information gained by partitioning E with attribute X is obtained by

$$\text{gain}(X) = \text{info}(E) - \text{info}_X(E) \quad (3)$$

Every time ID3 has to label a node, it selects the test with maximum gain, or in other words, it selects the attribute that divides the set of examples into partitions with the minimum entropy average.

This criterion guides ID3 to obtain quite good partitions, but it favors the selection of attributes with a lot of possible values. Frequently, this bias implies poor generalization, so C4.5 tries to correct this deficiency by calculating the so-called gain ratio, whose expression is

$$\text{gain-ratio}(X) = \frac{\text{gain}(X)}{\text{split-info}(X)} \quad (4)$$

where *split-info* is the *information* provided by a probability distribution $\left\{\frac{|E_i|}{|E|} : i = 1..n\right\}$, where E_i are the subsets of training examples for which the X value is the i-th possible value of this attribute. Its expression is, therefore

$$\text{split-info}(X) = -\sum_{i=1}^{n} \frac{|E_i|}{|E|} \cdot \log_2\left(\frac{|E_i|}{|E|}\right) \quad (5)$$

2.2. Rule generation

Decision trees, even when pruned, can be quite complex to provide understanding about a problem. The underlying difficulty is inherent to the structure used to represent the classification knowledge and it is aggravated when the size of the tree increases. The most evident way to deal with this problem is by changing the representation; so C4.5 offers the possibility of transforming its decision trees into *classification rule sets* (also known as *decision lists*) by means of an additional procedure; the resulting learning algorithm will be referred to as *C4.5-rules* in the remainder of this paper.

Transforming a decision tree into a rule set is as simple as generating a rule for each path from the root node to every leaf. The antecedents of each rule will be the tests of the nodes in the corresponding path and the consequent will be the label of the leaf.

The result of this trivial conversion is usually a rule set where some of the rules are redundant or have unnecessary, superfluous antecedents, so a more elaborated transformation mechanism is needed. C4.5-rules proceeds by deleting antecedents of the rules when their elimination does not worsen the *estimated accuracy* of the rule. This estimation is calculated by forcing some statistical concepts a little: we can consider the scores obtained by the rule on the training set as a sample of the behaviour on a population of examples that will be classified by the rule. Although the exact error probability cannot be determined, we can bind it for a given confidence level, *CF*, by means of its probability distribution confidence limits. The upper limit, $U_{CF}(f, n)$, is then used as a pessimistic estimation of the probability of error of this rule.

Thus, a pruned rule R' (R without antecedent X) will be preferable to its original version R if $U_{CF}(f_R, n_R) \geq U_{CF}(f_{R'}, n_{R'})$, where

- f_R and $f_{R'}$ are the number of training examples incorrectly covered by R and R', respectively.
- n_R and $n_{R'}$ are the total number of examples covered by R and R', respectively.

This generalization process is repeated for each path in the unsimplified decision tree, provoking, as a side effect, the deletion of some rules due to duplications or unacceptable error rates.

The rules so obtained are not necessarily mutually exclusive since there will be examples satisfying the antecedents of more than one rule, so a *conflict resolution* procedure is needed. C4.5-rules orders the rules so that the first rule covering a case will be the one that will classify it. This ordering is obtained by considering one set of rules per class; the number of rules in each set is reduced using a pruning method based on the *Minimum Description Length* (MDL) principle [31] and then subsets so obtained are ordered depending on the number of incorrectly covered examples.

The MDL-based procedure selects, for each class, a subset of rules with the lowest *cost of codification.* The cost of codification of a subset S is computed as:

$$C(S) = C_X(S) + 0.5 \cdot C_T(S) \tag{6}$$

where $C_T(S)$ is the cost of codification of the theory represented by the subset of rules and $C_X(S)$ is the cost of codification of the exceptions to that theory. The cost of codification of the theory is calculated as

$$C_T(S) = \sum_{i=1}^{|S|} C(R_i) - \log_2(|S|!) \tag{7}$$

which represents the sum of the number of bits needed to codify each rule R_i minus the number of bits required to indicate a particular ordering, $\log_2(|S|!)$, since rule ordering is not important for each particular subset, given that all of them predict the same class.

Exceptions are encoded indicating which cases are incorrectly covered by the rules in S, *false positives*, and those that would be correctly classified if they were covered, *false negatives.* Thus,

$$C_X(S) = \log_2 \binom{r}{fp} + \log_2 \binom{|E| - r}{fn} \tag{8}$$

where *fp* and *fn* are the false positives and false negatives, respectively; r is the number of cases covered by the rules in S; and $|E|$ is the number of cases in the training set E.

Once a subset of rules has been established for each class, C4.5-rules orders the subsets by selecting that with the lowest number of false positives as the first one; then, the false positives of the remaining subsets are recalculated to select the next one, and so on.

Finally, a *default rule* is added to deal with cases not covered by any other rule. The class of the default rule will be the one with the highest number of uncovered cases, or that with the highest absolute frequency in case of ties.

3. The impurity level measure

As pointed out in the introduction, the impurity level is based on Aha et al.'s mechanism in IB3 [1,2] for selecting a set of representative instances from a set of training examples. This measure was formerly [28] devised as a mechanism to estimate the quality of classification rules and was first implemented in FAN, and used later in other systems like INNER and BETS with noteworthy success.

Throughout the paper we consider classification rules represented by

$$R : C \leftarrow Ant_1 \wedge Ant_2 \wedge \ldots \wedge Ant_n \tag{9}$$

where the antecedents of the rule are conditions involving single attribute values; the intended meaning is that whenever all the conditions are fulfilled by a case, then the predicted class is the label C of the conclusion of the rule.

The quality of a rule R (Equation (9)) predicting class C can be estimated in terms of *impurity* by means of the success probability of the rule, $p(R)$, computed as the percentage of correctly covered examples, that is

$$p(R) = \frac{s_R}{n_R} \tag{10}$$

where s_R is the number of examples correctly covered by R and n_R is the total number of examples covered by the rule. However, to take into account the number of instances involved in these computations, we will consider the confidence interval of the success probability of rules, $\mathrm{CI}(R) = [\mathrm{CI}_l(R), \mathrm{CI}_h(R)]$. This interval is calculated by

$$\mathrm{CI}(R) = \frac{p(R) + \frac{z^2}{2n_R} \pm z\sqrt{\frac{p(R)(1-p(R))}{n_R} + \frac{z^2}{4n_R^2}}}{1 + \frac{z^2}{n_R}} \tag{11}$$

where z is a constant obtained from a normal distribution table which depends on the confidence level used (by default the confidence level used is 95%, hence z is 1.96).

Once the confidence interval is calculated, the peculiarities of the predicted class C must be considered in order to allow fair comparisons between rules concluding different classes. For this purpose we compute the confidence interval of the success probability when unconditionally predicting C. This is equivalent to computing the confidence interval of a rule Θ_C concluding C without any antecedent, sometimes called the *default rule* of class C.

$$\Theta_C : \qquad C \leftarrow \tag{12}$$

Now we are ready to define the impurity level of R as the overlapping percentage of two confidence intervals, $\mathrm{CI}(R)$ and $\mathrm{CI}(\Theta_C)$. In symbols,

$$\mathrm{IL}(R) = \frac{\mathrm{CI}_h(\Theta_C) - \mathrm{CI}_l(R)}{\mathrm{CI}_h(R) - \mathrm{CI}_l(R)} \times 100 \tag{13}$$

Roughly speaking, the impurity level of a rule attempts to resolve to what extent it is worth evaluating rule antecedents to predict a rule conclusion class instead of simply always predicting that class. The lower the impurity level, the more valuable the rule. Therefore, while we try to figure out which rule set to return from the training examples, rules with a lower impurity level will be preferred to rules with a higher impurity level.

The idea of estimating a rule's accuracy using a relative measure with respect to the accuracy of a default rule is also present in [34]. In this work Todorovski et al. investigate the use of the *weighted relative accuracy* (WRAcc) as a quality estimator of a rule; for a given a rule this measure is defined as

$$\mathrm{WRAcc}(R) = \frac{n_R}{|E|}(p(R) - p(\Theta_C)) \tag{14}$$

where $|E|$ is the number of examples in the training set. In this equation, $p(R) - p(\Theta_C)$ measures the accuracy gain of rule R with respect to a rule (see (12)) unconditionally predicting C. This gain is weighted by the proportion of training examples covered by the rule; the intention is to avoid highly specific rules. In Section 6 we briefly report some result obtained using this measure instead of our impurity level.

4. Using IL in decision tree construction

In this section we describe an algorithm similar to C4.5. Our modifications to Quinlan's algorithm are primarily based on the use of the impurity level, both for selecting tests in the tree-building stage and for rule generation. We show how the underlying principles used to calculate the impurity level of a rule can be easily adapted to evaluate potential splits of the data.

The impurity level of a rule can be easily generalized to select tests to build decision trees. IL-rules uses the exact same approach described above to build a decision tree, but using the impurity level instead of the gain ratio. The idea is to indicate how *impure* the subsets obtained by partitioning a set of examples on each possible attribute are, allowing us to select the test (attribute) that produces the partition with the lowest impurity. Therefore, we define the impurity level of a subset of training examples, E_i, as the impurity level of a rule whose conditions are fulfilled by the examples in E_i and whose conclusion is the majority class in E_i.

Analogous to info_X calculated by Equation (2), the impurity level due to attribute X is computed as a weighted sum of the impurity level of the subsets it produces

$$\mathrm{IL}_X(E) = \sum_{i=1}^{n} \frac{|E_i|}{|E|} \cdot \mathrm{IL}(E_i) \tag{15}$$

To illustrate the use of the impurity level we are going to detail the construction of a decision tree with the data set shown in Table 1. This data set, taken from [24], is composed of examples described by three attributes: Height, Hair and Eyes; and the classes to be predicted, which can be $\oplus$ or $\ominus$.

Initially, we can take three different tests to label the root node, so we have to compute the impurity level for each one. In the case of attribute Height, which can take two different values, the original set, let us name it E, can be split into two subsets, E_{short} containing *short* individuals and E_{tall} containing *tall* individuals.

Subset E_{short} has $n = 3$ examples, 2 $\ominus$ and 1 $\oplus$; so the majority class is $\ominus$ with probability $p = \frac{2}{3}$. The confidence interval $\mathrm{CI}(E_{\mathrm{short}})$ calculated by Equation (11) using the default value, 1.96, for z is $[0.208, 0.939]$ approximately.

Now we have to calculate the confidence interval of the success probability when predicting $\ominus$ but

Table 1

A simple data set built up by examples belonging to two different classes and described by three symbolic attributes.

Height	Hair	Eyes	CLASS
short	blonde	blue	$\oplus$
short	dark	blue	$\ominus$
tall	dark	brown	$\ominus$
tall	blonde	brown	$\ominus$
tall	dark	blue	$\ominus$
short	blonde	brown	$\ominus$
tall	red	blue	$\oplus$
tall	blonde	blue	$\oplus$

considering E, and this probability is $p = \frac{5}{8}$, given that there are 5 $\ominus$ in the $n = 8$ examples of E. With these values, the confidence interval $\mathrm{CI}(E)$ is $[0.306, 0.863]$.

The impurity level of E_{short} is then calculated as

$$\mathrm{IL}(E_{\text{short}}) = \frac{\mathrm{CI}_h(E) - \mathrm{CI}_l(E_{\text{short}})}{\mathrm{CI}_h(E_{\text{short}}) – \mathrm{CI}_l(E_{\text{short}})} \cdot 100 = \\ = \frac{0.863 - 0.208}{0.939 - 0.208} \cdot 100 = 89.603 \quad (16)$$

This calculation is repeated for E_{tall}, where there are $n = 5$ examples, 3 $\ominus$ and 2 $\oplus$. So once more the majority class is $\ominus$ with probability $p = \frac{3}{5}$, thus giving $\mathrm{CI}(E_{\text{tall}}) = [0.231, 0.882]$. The confidence interval when predicting $\ominus$ in E was previously calculated, so the impurity level of E_{tall} is

$$\mathrm{IL}(E_{\text{tall}}) = \frac{\mathrm{CI}_h(E) - \mathrm{CI}_l(E_{\text{tall}})}{\mathrm{CI}_h(E_{\text{tall}}) - \mathrm{CI}_l(E_{\text{tall}})} \cdot 100 = \\ = \frac{0.863 - 0.231}{0.882 - 0.208} \cdot 100 = 97.081 \quad (17)$$

We can now calculate the impurity level corresponding to the attribute Height using Equation (15), obtaining the following

$$\mathrm{IL}_{\text{Height}}(E) = \frac{|E_{\text{short}}|}{|E|} \cdot \mathrm{IL}(E_{\text{short}}) + \\ + \frac{|E_{\text{tall}}|}{|E|} \cdot \mathrm{IL}(E_{\text{tall}}) = \\ = \frac{3}{8} \cdot 89.603 + \frac{5}{8} \cdot 97.081 = 94.277 \quad (18)$$

These calculations must also be done for attributes Hair and Eyes; their impurity levels are $\mathrm{IL}_{\text{Hair}}(E) \approx 75$ and $\mathrm{IL}_{\text{Eyes}}(E) \approx 73$. Therefore, a test on attribute Eyes will be selected to label the root node. Thus, we obtain a preliminary tree with a root node and two descendants, given that Eyes can be blue or brown. The node with brown-eyed examples is a leaf, since all of them belong to class $\ominus$. However, the building process must continue from the node with blue-eyed examples as the training set (i.e. considering $E = E_{\text{blue}}$) and two possible tests, the remaining attributes: Height and Hair. In this case the selected test is Hair and thus the final decision tree built by means of the impurity level is shown in Figure 1.

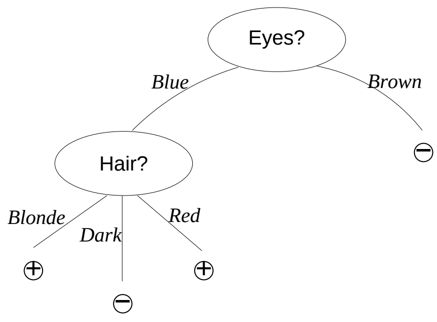

Figure 1. Using the impurity level, we obtain this decision tree for the data set in Table 1.

4.1. *Missing values and continuous attributes: their peculiarities*

The described process for building decision trees is directly applicable to problems in which the examples are composed of discrete attributes with a finite (usually small) number of possible different values, where the value of every attribute is always known. Sometimes, however, we have to manage datasets with examples where the value of some attribute is unknown, so a special handling is needed to deal with these *missing values*. C4.5's approach consists in computing the gain ratio for every attribute, considering only examples with known values and a fictitious subset containing the examples with the missing value, if there are any.

Additionally, a weighted scheme is used, associating with every example a probability of belonging to a subset. Thus, an example with an unknown value in the attribute used as a test is considered to belong to every subset with the probability of such subset, i.e. the number of examples of the node divided by the number of examples in the parent node.

Another difficulty arises when the examples contain continuous attributes, since they can take (potentially) infinitely many different values. The applicable test for these kind of attributes consists in selecting a threshold t to divide the data set E in two parts: examples whose value for the attribute are lower than or equal to the threshold, $E_{\leq t}$, and those with values higher than the threshold, $E_{>t}$.

Given a continuous attribute X taking n different values in the data set E, there are $n-1$ non trivial thresholds; if X values are ordered as $\{v_1, v_2, \ldots, v_n\}$, every potential threshold can be calculated as $t_i = \frac{v_i+v_{i+1}}{2}$ with $i = 1, \ldots, n-1$. However, instead of t_i, C4.5 uses the highest v_i lower than the calculated t_i to ensure that the selected threshold occurs in the data set, so the possible thresholds are $\{v_1, \ldots, v_{n-1}\}$.

The threshold that maximizes the splitting criterion will be selected. In C4.5 release 7, this criterion was the gain ratio, but release 8 includes a modification [26] where the gain ratio is penalized by subtracting $\frac{\log_2(n-1)}{|E|}$. This modification clearly improves the handling of continuous attributes, correcting the bias in the choice of a test towards continuous attributes with many different values.

The impurity level version, IL-rules, applies the same discretization method. Like C4.5, our system also applies a penalty, trying not to favor the selection of attributes with many possible thresholds. However, the penalty is applied in a slightly different way due to the semantics of the impurity level measure; in IL-rules, the impurity level is increased by a fraction of itself. This fraction is determined by $\frac{\log_2(n-1)}{|E|}$. Thus, our system will select a threshold t which minimizes the *penalized impurity level*, $\mathrm{IL}^p_{(X,t)}$, whose expression is

$$\mathrm{IL}^p_{(X,t)}(E) = \mathrm{IL}_{(X,t)}(E) \cdot \left(1 + \frac{\log_2(n-1)}{|E|}\right) \quad (19)$$

where $\mathrm{IL}_{(X,t)}(E)$ is the impurity level of the splitting of E into $E_{\leq t}$ and $E_{>t}$. So, $\mathrm{IL}_{(X,t)}(E)$ is calculated by

$$\mathrm{IL}_{(X,t)} = \frac{|E_{\leq t}|}{|E|} \cdot \mathrm{IL}(E_{\leq t}) + \frac{|E_{>t}|}{|E|} \cdot \mathrm{IL}(E_{>t}) \quad (20)$$

5. Rule generation in IL-rules

C4.5-rules has its counterpart in our impurity level-based algorithm. The procedure to obtain rules from trees implemented in IL-rules is completely different from that of C4.5-rules except for the first, trivial stage, that is to say, the generation of one rule for every path from the root node to each leaf. Once the tree has been rewritten as a rule set, IL-rules applies a generic method to improve the quality of these preliminary rules. This method is likewise based on the impurity level, and constitutes part of the systems Fan [29,27] and Inner [20,21].

The method to improve rule sets has some similarities with the approach followed by Cohen's Grow [7] in the sense that both algorithms start building a theory that is augmented with generalizations of the rules induced. These generalizations are constructed by an antecedent pruning mechanism. Then, a rule selection procedure reduces the final rule set. The starting theory for IL-rules is the set of rules obtained directly from the decision tree; this set of rules is augmented with partial descriptions of each rule by the *rule qualification* process and the resulting set of rules is pruned by the *rule selection* process. Also, we can find some similarities with specific parts of Ripper, an improved version of Fürnkranz and Widmer's Irep which, in turn, was developed as an alternative approach to Grow. In the next sections we are going to briefly describe our rule qualification and selection processes, pointing out those similarities.

5.1. Rule qualification: pruning antecedents

From the trivial conversion of a decision tree to a set of *preliminary rules*, the qualification algorithm (see Algorithm 1) returns optimized, less impure, versions of each original rule.

The impurity level plays an important role in the qualification process. In order to avoid an exhaustive exploration of possibilities, the algorithm heuristically guides the search for a rule whose antecedents will be a subset of those in the original, unpruned rule. Thus, we start ordering the antecedents of the rule to be qualified by their impurity level. Let us suppose that the ordered antecedents of R are:

$$R : C \leftarrow Ant_1 \wedge Ant_2 \wedge \ldots \wedge Ant_n. \quad (21)$$

This means that:

$$\mathrm{IL}(C \leftarrow Ant_1) \leq \ldots \leq \mathrm{IL}(C \leftarrow Ant_n). \quad (22)$$

Hence, the first step of the qualification process starts with an initial empty set of descriptions and progressively adds partial descriptions of the orig-

```
Function QUALIFY ($Rule$, $Examples$) : SetOfRules
$RuleList = \emptyset$, $class$ = CLASSOF ($Rule$)
/* We order the antecedents of $Rule$ attending to their Impurity Level */
$SortedListOfAntecedents$ = SORTBYIL (BODYOF ($Rule$), $Examples$)
$Body$ = TRUE
repeat/* First step */
   $Body = Body \wedge$ POP ($SortedListOfAntecedents$)
   $p$ = SUCCESSPROB ('$class \leftarrow Body$', $Examples$)
   if $p \geq$ THRESHOLD then
      $RuleList$ = INSERTBYIL ('$class \leftarrow Body$', $RuleList$, $Examples$)
   end if
until $p = 1 \vee SortedListOfAntecedents = \emptyset$
$R_{\text{best}}$ = FIRST ($RuleList$)
   /* Recall that $RuleList$ was ordered by the Impurity Level of rules */
let $Body = A_1 \wedge A_2 \wedge \ldots \wedge A_k \wedge A_{k+1}$ be $R_{\text{best}}$'s body
$p_{\text{best}}$=SUCCESSPROB ($R_{\text{best}}$, $Examples$)
if $k \geq 2$ then/* Second step */
   for $i = k$ downto 1 do
      $TentativeBody$ = ALLBUT ($A_i$, $Body$)
      if SUCCESSPROB ('$class \leftarrow TentativeBody$', $Examples$) $> p_{\text{best}}$ then
         $RuleList$ = INSERTBYIL ('$class \leftarrow TentativeBody$',$RuleList$,$Examples$)
         $Body = TentativeBody$
      end if
   end for
end if
$R_{\text{best}}$ = FIRST ($RuleList$) /* Notice that $RuleList$ could have changed */
$IL_{\text{best}}$=IMPURITYLEVEL ($IL_{\text{best}}$, $Examples$)
return {$R$: $R \in RuleList \wedge$ IMPURITYLEVEL ($R$,$Examples$) $\leq IL_{\text{best}}+10$ }
```

Algorithm 1. Qualification pseudo-code. This algorithm implements the pruning of the rule antecedents according to the impurity level, returning a set of possible optimized versions.

inal rule being qualified. These descriptions are of the form:

$$R_i : C \leftarrow \text{Ant}_1 \wedge \ldots \wedge \text{Ant}_i \tag{23}$$

where $i = 1 \ldots n$. Only those descriptions with a success probability higher than an *acceptance threshold* are saved. Furthermore, if a partial description with no errors is found, no more descriptions are added. The way these descriptions are constructed looks like RIPPER's rule growing process, which starts from an empty conjunction of antecedents and repeatedly adds the condition that maximizes an information gain measure. The difference, however, is that at each iteration RIPPER replaces the previous description of the rule with a new version having one more condition, while IL-rules saves every partial description until the iterative process finishes, which yields, in general, a set of rules instead of just one rule.

Once we have a set of partial descriptions, the second step consists in selecting the best rule, namely R_{best}, to add more partial descriptions by greedily deleting one antecedent after the other, starting at the penultimate. At every iteration the new partial description is constructed by eliminating the corresponding antecedent in the previously accepted partial description; only those descriptions with a success probability higher than that of R_{best} will be accepted. This part of the qualification process is also similar to the rule pruning used in RIPPER, but again we add several pruned versions of R_{best} to the rule set, while RIPPER replaces the grown rule with its pruned version.

Additionally, two noticeable differences between IL-rules and Cohen's algorithm are:

- RIPPER splits the training data set into two partitions, one used for rule growing and the other for rule pruning.
- RIPPER implements a separate-and-conquer approach, where only those training examples not covered by previously induced rules are used to induce a new rule.

Finally, we return a subset of descriptions filtered from all the possibilities considered so far. The qualified subset is composed of the rule

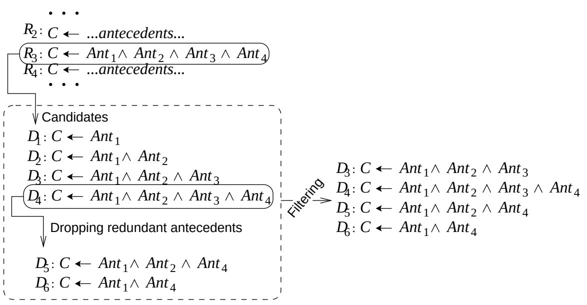
R2: C ← ...antecedents...
R3: C ← Ant1∧ Ant2 ∧ Ant3 ∧ Ant4
R4: C ← ...antecedents...
Candidates
D1: C ← Ant1
D2: C ← Ant1∧ Ant2
D3: C ← Ant1∧ Ant2 ∧ Ant3
D4: C ← Ant1∧ Ant2 ∧ Ant3 ∧ Ant4
Dropping redundant antecedents
D5: C ← Ant1∧ Ant2 ∧ Ant4
D6: C ← Ant1∧ Ant4
Filtering
D3: C ← Ant1∧ Ant2 ∧ Ant3
D4: C ← Ant1∧ Ant2 ∧ Ant3 ∧ Ant4
D5: C ← Ant1∧ Ant2 ∧ Ant4
D6: C ← Ant1∧ Ant4

Figure 2. Qualification process. Suppose R_3 has its antecedents ordered and is going to be qualified. The first step generates four possible descriptions: the candidates. If all of them surpass the acceptance threshold and D_4, which coincides with R_3, is the best, then it is selected to generate new descriptions by dropping its antecedents from the penultimate to the first, giving D_5 and D_6. If both have a success probability higher than that of D_4 then they are added to the previous set, giving $\{D_1$, D_2, D_3, D_4, D_5, $D_6\}$. Only the best rule and those with similar impurity levels will be returned. In this example, these will be $\{D_3$, D_4, D_5, $D_6\}$.

with the lowest impurity level together with those of similar impurity level (see last step in Algorithm 1). Figure 2 depicts a diagram of a hypothetical example of the qualification process.

5.2. *Rule selection: deleting rules*

Rule selection is intended to prune the rule set obtained by the qualification. Taking into account the fact that rule generation is applied to unpruned trees and that the qualification process generally produces more than one description for each preliminary rule, a pruning process is needed to obtain more compact rule sets. Such sets are more comprehensible and they usually provide more accurate generalizations.

Algorithm 2 depicts the selection algorithm. The first stage is especially devised to deal with overfitting, since it is aimed at detecting and deleting those rules classifying too specific peculiarities of data caused by noisy examples disturbing the induction process. This task is accomplished by DeleteNoisyRules, shown in Algorithm 3. Basically the procedure deletes rules whose success probability is lower than a *noise threshold*, computed for every problem as follows. We select those rules whose impurity level is lower (i.e. better) than the impurity level of the worst of the best rules of each class (see Figure 3); then we compute

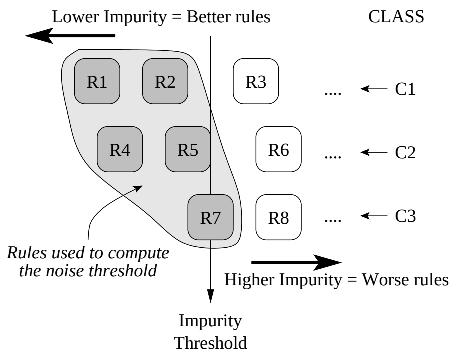
Lower Impurity = Better rules
CLASS
R1
R2
R3
.... ← C1
R4
R5
R6
.... ← C2
R7
R8
.... ← C3
Rules used to compute the noise threshold
Higher Impurity = Worse rules
Impurity Threshold

Figure 3. The worst of the best rules of each class in this example is R_7, so the noise threshold will be computed from the error probability of $\{R_1, R_2, R_4, R_5, R_7\}$.

the error probability of this rule set, which is the sum of incorrectly covered examples divided by the number of applicable examples. This error probability has a confidence interval whose midpoint constitutes the noise threshold.

Once we have deleted the noisy rules, we try to FindBestRuleSet (see Algorithm 4). This part of the selection algorithm consists of two phases. The first phase looks for the best impurity level threshold, which enables a reduction in size of the rule set, yielding the same or higher accuracy. For every possible threshold starting at the highest impurity level in the rule set, our process evaluates the subset containing only those rules with a lower or equal impurity level, and at least one for every class. If there are K different impurity levels, we select the best of the $K - 1$ possible subsets in terms of accuracy.

This is the first time in our rule generation process where rules compete to classify examples, and that they are applied on the basis of a minimum distance criterion. The next section is devoted to describe the distance function used in our algorithm.

The second phase when searching for the best rule set tries to detect and eliminate useless rules still undeleted in prior steps. The process goes through the ordered rule set and for each rule, starting with the worst one, it computes the accuracy of the resulting rule set should that rule be deleted. Only when the accuracy does not decrease is the rule permanently removed; the accuracy of the resulting rule set is then considered the new threshold for subsequent eliminations.

There is a final stage in the rule selection process related to the comprehensibility of the result-

```
Function Select ($RuleSet$, $Examples$) : SetOfRules
$RuleSet$ = DeleteNoisyRules ($RuleSet$)
$RuleSet$ = FindBestRuleSet ($RuleSet$, $Examples$)
if all attributes are symbolic then
    $RuleSet$ = AddDefaultRule ($RuleSet$, $Examples$)
end if
return ($RuleSet$)
```

Algorithm 2. General description of selection function. This returns the final rule set from the rules provided by qualification.

```
Function DeleteNoisyRules ($RuleSet$, $Examples$) : SetOfRules
/* As a side effect of the qualification process, all rules have some numbers attached:
      applicable training examples, those at distance 0;
      failed, applicable with different class;
      successful, applicable with the same class */
$WorseBestImpurityLevel$ =
    Max (Min (ImpurityLevel ($R$,$Examples$): $R$ rule of class $C$): $C \in$ Set of classes)
    /* Recall that better impurity levels correspond to lower values */
let $FailedSum$ and $ApplicableSum$ be the sums of failed and applicable ex-
    amples of rules with Impurity Level lower (better) than $WorseBestImpu$-
    $rityLevel$
$Threshold$ = Middle point of
    ConfidenceInterval ($FailedSum/ApplicableSum$, $ApplicableSum$, $z$)
    /* ConfidenceInterval ($p, n, z$) computes Equation (11) */
return {$Rule$: $Rule \in RuleSet \wedge$ SuccessProb ($Rule$, $Examples$) < $Threshold$}
```

Algorithm 3. To filter rules that presumably provide from noisy data we use this function.

ing rule set. Whenever all the attributes describing the training examples have symbolic values, IL-rule selection is allowed to include a default rule that will be applied to a given case when no other rule is at distance zero. To achieve this, our default rules have no conditions at all, they consist of a class label like in Formula (12). In data sets with some continuous attributes, the default rule would destroy the benefits of the application by means of a minimal distance criterion, so IL-rules never includes default rules in these cases.

5.3. Measuring distances

In order to choose the closest rule to an example, our distance function must be able to measure distances from rules to examples. From a geometrical point of view, a rule r can be seen as a hyperrectangle in the attribute space representing r's antecedents, and we define the distance from r to an example e as a function of the distances between their projections. For this purpose we use a HEOM-like [36] metric defined as:

$$\text{distance}(r, e) = \sqrt{\sum_{a=1}^{m} \text{difference}_a^2(r_a, e_a)} \quad (24)$$

where m is the number of attributes describing the examples, r_a is the condition on attribute a in rule r and e_a is the actual value of attribute a in example e. For every attribute a, in turn, differences are calculated using the normalized Euclidean distance if a takes continuous values, or using the *overlap* function (likewise HEOM) if a is a symbolic attribute.

$$\text{difference}_a(r_a, e_a) = = \begin{cases} \text{overlap}(r_a, e_a), & \text{if } a \text{ is symbolic} \\ \text{n\_eucl}(r_a, e_a), & \text{if } a \text{ is numeric} \end{cases} \quad (25)$$

The overlap metric [36] yields a difference of 1 when the symbolic value of the attribute is different than the value mentioned in the condition of the rule, and 0 otherwise, that is,

$$\text{overlap}(r_a, e_a) = \begin{cases} 0, \text{ if } e_a \text{ fulfills } r_a \\ 1, \text{ otherwise} \end{cases} \quad (26)$$

For numerical attributes, rules will have conditions like $(a > v_1)$, $(a \leq v_2)$ or $(a > v_1 \wedge a \leq v_2)$,

```
Function FINDBESTRULESET (RuleSet, Examples) : SetOfRules
/* We first gather the list of impurity levels of all available rules. The list will be
    sorted from worse (higher) to better (smaller) values */
SortedListOfIL = SORT (IMPURITYLEVEL (Rule, Examples): Rule $\subset$ RuleSet)
BestRuleSet = RuleSet
BestSuccesses = EVALUATE (BestRuleSet, Examples)
/* Now we test if any other rule set (rules whose impurity levels are better than a
    given threshold) can improve the classification accuracy */
for each 2 consecutive values $x_0$ and $x_1$ of SortedListOfIL do
    TentativeThreshold = $(x_0 + x_1)/2$
    TentativeRuleSet =
            RULESBETTERTHAN (TentativeThreshold, RuleSet, Examples)
    ClassificationSuccesses = EVALUATE (TentativeRuleSet, Examples)
    if ClassificationSuccesses $\geq$ BestSuccesses then
        BestSuccesses = ClassificationSuccesses
        BestRuleSet = TentativeRuleSet
    end if
end for
/* Second phase: assume BestRuleSet is ordered from worse to better rules */
for each Rule $\in$ BestRuleSet do
    TentativeRuleSet = ALLBUT (Rule, BestRuleSet)
    ClassificationSuccesses = EVALUATE (TentativeRuleSet, Examples)
    if ClassificationSuccesses $\geq$ BestSuccesses then
        BestSuccesses = ClassificationSuccesses
        BestRuleSet = TentativeRuleSet
    end if
end for
return (BestRuleSet)

Function RULESBETTERTHAN (Threshold, RuleSet, Examples) : SetOfRules
/* We assume that all classes have at least one rule in RuleSet */
List = (Rule $\in$ RuleSet: IMPURITYLEVEL (Rule, Examples) < Threshold)
if some class has no rules in List then
    List = List $\cup$ Best rule of the class in RuleSet
end if
List = DELETEREDUNDANTRULES (List)
return (List)
```

Algorithm 4. This is the main function in the selection of a final rule set; it exploits the impurity level features as well as a distance-based evaluation method.

due to the kind of tests used in the nodes of decision trees. These conditions can be expressed using intervals; thus, $a \in (v_1, \infty)$, $a \in (-\infty, v_2]$ and $a \in (v_1, v_2]$ represent the aforesaid conditions, respectively. With such representation, we define

$$\text{n\_eucl}(r_a, e_a) = \begin{cases} 0, & \text{if } e_a \text{ fulfills } r_a \\ \frac{|e_a - v_a|}{4\sigma_a}, & \text{otherwise} \end{cases} \quad (27)$$

where v_a, for a given e_a, is the nearest boundary of the interval used in r_a. Differences are normalized by means of a commonly used large value: four times the standard deviation, σ_a, of the observed attribute values [36].

To completely specify the distance function we must define how to deal with missing values. If r_a is missing then no particular value of a is required to apply r, i.e. the value of a makes no difference. A missing e_a means that the value of a is unknown in example e. In both cases our difference function will return a value of 0 to make the value of a have no influence in the distance computation. RISE [12] deals with missing numerical values in the exact same way.

6. Experimental results

In this section, we experimentally justify the goals of the impurity level as a heuristic for learning concise and accurate classification rules. To this end, we report the scores obtained by IL-rules, the rule generation procedure based on the impurity level. We will compare these results with those obtained with Quinlan's C4.5-rules (release 8) and with Cohen's RIPPER in terms of *accuracy* and *size* of the induced knowledge. To close the section, we discuss the experimental scores of WRAcc-rules, a system built like our IL-rules, but using the weighted relative accuracy presented in Section 3.

The comparison with C4.5-rules is needed since our system shares an important part of the induction algorithm with it. Additionally, we included RIPPER in the comparisons reported here due to its high accuracy and effectiveness for pruning rules.

To carry out the experiments, we choose a well-known set of 16 problems used by Holte in [15]. These problems were downloaded from the UCI Machine Learning Repository [3].

Following the recommendations in [19], we used a 10-fold stratified cross-validation repeated 5 times. The experiments were carried out using some tools of the $\mathcal{MLC}$++ [18] in order to assure that the algorithms were run on identical training and test sets. Table 2 shows the average classification errors, and the number of rules and antecedents.

When we compare our approach with C4.5-rules, we observe a slight improvement in average accuracy in the collection of datasets used for experiments; in fact it is a difference statistically significant, as reported in Table 4A, although the correlation between the errors of both algorithms is 0.99, and there are only 9 differences favorable to IL-rules, and 7 favorable to C4.5-rules. However, when we compare the size of rule sets, the differences are clearly significant (see Tables 4B, and 4C); IL-rules produces about half the number of rules with half the number of antecedents than C4.5-rules. On the other hand, IL-rules is slower than C4.5-rules; on average our system employs 1.55 second per dataset, while C4.5-rules only requires 0.52, nevertheless, the difference is only statistically significant at a confidence level of 92.67%.

6.1. Hybrid systems: the influence of the tree generation process

The aim of the next step in our experiments is to gain insight into the reasons that allows IL-rules obtain these results with respect to C4.5rules. So, let us recall that IL-rules generates rules from trees built using the impurity level for test selection. Bearing this in mind, a question arises: do the differences appear at the stage of tree generation or later?

To answer this question, we have analyzed the results achieved just before the decision trees are transformed into rules, realizing that the treebuilding process based on the impurity level usually induces bigger trees than the process based on the information gain ratio, although the difference in accuracy (in favor of IL) is not statistically significant. Hence, given that the induced trees are different, a deeper investigation needs to be done in order to clarify how much of the success of our rule generation process depends on the trees. For this reason, we have experimented with two *hybrid* systems consisting of:

– A tree-building process based on the information gain ratio followed by an impurity level based pruning to generate rule sets. We call this GR_tree+IL_pruning.
– A tree-building process based on the impurity level together with an MDL rule generation, IL_tree+MDL.

Table 3 shows the results of these hybrid systems; the experiment was likewise a ten-fold 5-times stratified cross- validation carried out under the same conditions of randomness as the previous series comparing the "pure" systems.

If we now analyze the results of the hybrid and pure systems with their stages of tree generation and rule pruning, we realize that neither the exclusive use of the impurity level in the decision tree building nor the exclusive use of our pruning mechanism yield significant improvement in accuracy. However, the best average accuracy is attained when both ingredients are combined in IL-rules. Table 4A corroborates this observation, showing that the differences in accuracy are only significant among IL-rules and any of the other systems in the comparison.

On the other hand, when we compare the size of the knowledge learned (it does not matter whether this is the number of rules or the number of antecedents, see Tables 4B and 4C), the systems with an impurity level pruning clearly obtain smaller sizes. This is especially interesting in the case of IL-rules since it is also the most accurate in the comparison.

Table 2

Experimental results obtained with a 10-fold 5-times stratified cross-validation. For each dataset and algorithm, we report the average values of: classification errors, number of rules induced, and number of antecedents of these rules.

	IL-rules			C4.5-rules			Ripper		
Dataset	Error	# Rules	# Ant.	Error	# Rules	# Ant.	Error	# Rules	# Ant.
BC	27.63	4.60	7.74	30.84	8.20	17.08	28.54	2.82	5.46
CH	2.10	9.80	36.42	0.97	26.78	99.98	1.36	17.36	69.24
G2	18.31	6.20	11.42	21.87	8.10	21.04	20.06	4.34	9.04
GL	30.57	8.68	26.36	32.22	14.10	50.78	34.21	9.48	26.28
HD	17.73	7.14	17.50	20.98	13.26	35.76	20.36	4.34	10.96
HE	19.33	5.58	15.34	20.43	7.92	20.46	19.78	2.96	5.68
HO	15.82	3.82	10.08	17.45	6.00	11.42	15.11	3.82	8.94
HY	0.99	4.58	10.64	0.78	6.34	13.10	0.81	2.18	3.76
IR	5.07	3.28	4.20	4.40	4.02	6.08	5.20	3.08	5.02
LA	17.80	3.64	7.24	17.00	3.98	5.78	14.87	3.78	6.34
LY	22.85	7.78	14.94	23.25	10.58	23.56	23.21	6.54	14.70
MU	1.52	4.70	3.86	0.03	17.66	26.38	0.00	8.84	20.00
SE	2.25	4.54	13.98	2.35	12.68	41.72	2.33	4.50	15.06
SO	0.00	4.00	4.00	2.90	4.00	5.90	2.10	4.08	6.18
VO	4.78	2.32	4.26	4.37	6.10	13.76	4.27	4.68	4.80
V1	10.71	4.34	11.38	10.16	11.20	29.30	10.89	2.74	14.82
Av.	12.34	5.31	12.46	13.13	10.06	26.38	12.69	5.35	14.14

Table 3

Hybrid systems comparison. 10-fold 5-times stratified cross-validation carried out on identical training and test sets pairs than the experiments described at Table 2.

	GR_tree+IL_pruning			IL_tree+MDL		
Dataset	Error	# Rules	# Ant.	Error	# Rules	# Ant.
BC	27.35	3.94	6.64	30.11	6.72	12.92
CH	1.88	9.82	36.40	1.17	25.08	98.42
G2	21.12	6.94	15.64	18.68	9.64	25.98
GL	32.04	11.90	37.52	29.47	11.84	40.90
HD	18.09	6.92	16.02	21.89	11.42	30.02
HE	19.84	5.46	14.54	18.46	5.66	12.88
HO	18.05	2.14	2.96	15.38	6.80	13.76
HY	0.83	4.62	12.18	0.97	6.68	14.22
IR	4.80	3.18	4.16	4.67	3.98	5.94
LA	16.93	2.62	5.00	17.20	3.52	4.04
LY	24.89	7.62	14.84	24.97	9.98	21.16
MU	1.11	5.00	5.94	0.01	16.94	26.28
SE	2.26	5.14	16.54	2.31	13.70	42.92
SO	2.90	4.00	5.90	0.00	5.00	7.00
VO	4.87	2.46	4.46	4.96	6.12	14.16
V1	10.56	3.96	9.59	10.95	9.90	25.52
Av.	12.97	5.36	13.02	12.58	9.56	24.76

Table 4

Tables A, B and C show the average error, number of rules and antecedents under column *Avg.* The rest of these tables should be interpreted as follows: a number means that the system in the column has a smaller error, number of rules or antecedents, respectively, than the system in the corresponding row; the difference is statistically significant and the number is the confidence level, using a one-tail paired t-test. A label *n.s.* means that the values compared are not significantly different.

	Avg.	IL-rules	IL-tree+MDL	Ripper	GR-tree+IL-rules
IL-rules	*12.34*	......	......	......	......
IL-tree+MDL	*12.58*	n.s.	......	......	......
Ripper	*12.69*	n.s.	n.s.	......	......
GR-tree+IL-rules	*12.97*	97.02%	n.s.	n.s.	......
C4.5-rules	*13.13*	95.65%	n.s.	n.s.	n.s.

A) Accuracy comparison.

	Avg.	IL-rules	Ripper	GR-tree+IL-rules	IL-tree+MDL
IL-rules	*5.31*	......	......	......	......
Ripper	*5.35*	n.s.	......	......	......
GR-tree+IL-rules	*5.36*	n.s.	n.s.	......	......
IL-tree+MDL	*9.56*	99.93%	100.00%	99.92%	......
C4.5-rules	*10.06*	99.95%	100.00%	99.95%	n.s.

B) Number of rules comparison.

	Avg.	IL-rules	Ripper	GR-tree+IL-rules	IL-tree+MDL
IL-rules	*12.46*	......	......	......	......
GR-tree+IL-rules	*13.02*	n.s.	......	......	......
Ripper	*14.14*	n.s.	n.s.	......	......
IL-tree+MDL	*24.76*	99.64%	99.58%	99.57%	......
C4.5-rules	*26.38*	99.83%	99.80%	99.82%	n.s.

C) Number of antecedents comparison.

6.2. *Comparison with* Ripper

The scores of IL-rules and Ripper, in accuracy and size of the induced knowledge, are similar (see Tables 2 and 4). The most obvious difference between Ripper and IL-rules is that our system uses a partial matching mechanism or distance-based evaluation method. The results obtained by IL-rules and by C4.5-rules, and specially those obtained in the hybrid systems comparison, suggest that this kind of rule evaluation permits our pruning mechanism to induce fewer rules, since we do not need to cover all the attribute space. Let us recall that one of the stages of our *rule selection* process drops some rules when their covered examples can be correctly classified by other rules in the surroundings. To analyze this, we have computed how many examples are not covered by any rule in the experiments with Holte's datasets. Table 5 reports the average percentage of test examples classified by rules at distance greater than zero, i.e. uncovered examples. In general, our system induces rule sets that do not cover the whole attribute space for almost any dataset, except for the MU and SO.

However, when comparing with Ripper, we observe that Cohen's algorithm induces nearly the same number of rules (in average) as IL-rules. Therefore, the use of partial matching is not the only way to obtain compact rule sets. The differences between partial and full matching arise in datasets whose examples are described mostly by continuous attributes. Additionally, the conditions of Ripper rules can be interpreted as regions parallel to the axes. Then, both systems have completely different geometrical tools to separate in classes the regions of attribute space. Thus, to investigate the consequences of these facts in terms of accuracy, we have carried out some additional

Table 5
Percentage of examples not covered by rules; i.e. examples that must be classified by rules at distances greater than zero. The average is 8.23%

Dataset	BC	CH	G2	GL	HD	HE	HO	HY
Uncov.(%)	11.33	0.74	0.87	18.82	13.44	10.99	21.41	1.55
Dataset	IR	LA	LY	MU	SE	SO	VO	V1
Uncov.(%)	2.93	9.53	13.83	0.00	2.37	0.00	7.43	16.50

experiments with Ripper and IL-rules, selecting those problems in the Holte's collection with continuous attributes (G2, GL and IR), together with other datasets downloaded from the UCI repository. The results of this comparison are shown in Table 6, where we have observed that Ripper induces more compact rule sets at the price of accuracy. The results are statistically significant, in number of rules (in favor of Ripper) and in accuracy (in favor of IL-rules), with confidence levels in a one-tail paired t test of 98.26% and 98.85%, respectively.

Other dimensions to compare the behaviour of learning systems are the presence of irrelevant attributes and noise. Therefore, we have made additional experimentation to study the behaviour of IL-rules and Ripper in this environment. For this purpose we have used a well-known artificial domain generator, which simulates the status of a seven LED (Light-Emitting Diodes) display when representing each of the ten decimal digits. This generator is able to perturb the correct value of each attribute with a given probability, as well as to add irrelevant features to predict the class. We have used this generator to carry out two different experiments:

– *Noisy attributes*: we generated 30 datasets with 1000 examples each one. For every dataset, we progressively increased the noise probability from 0% to 30% for each of the 7 attributes.
– *Irrelevant attributes*: for a given noise probability (10%), we generated 30 datasets increasing the number of irrelevant attributes; thus, the last dataset had 37 attributes: 7 relevant (although noisy), and 30 irrelevant.

To isolate the effect of irrelevant attributes in the second experiment we have modified the code of the led generator to use two different random generators, one for the seven relevant attributes and another for the rest. This modification guarantees that the values of the relevant attributes are exactly the same in the 30 datasets. For this purpose we have used the random number generator called Mersenne Twister [23].

The results of these experiments (five tenfold stratified cross-validations with every dataset) are graphically depicted in Figure 4, and suggest that IL-rules is slightly more robust than Ripper in the presence of noise and irrelevant attributes.

The reduced average number of rules and antecedents induced by Ripper in the first (noise) experiment are due to the low number of them induced above the threshold of 23% of noise, where Ripper induces sometimes even fewer rules than the number of classes. All differences in accuracy are statistically significant ($> 99.99\%$) if favor of IL-rules, as well as the differences in the number of rules and antecedents for the second experiment (irrelevant attributes).

6.3. WRAcc-rules

Additional work has been carried out trying to find out to what extent the impurity level is a key in the overall performance of our proposed pruning method. For this purpose we built an algorithm on top of the same template used for IL-rules, but replacing the impurity level with the weighted relative accuracy mentioned in Section 3, we call it WRAcc-rules. The average classification error in Holte's datasets was 14.34%, far from the 12.34% achieved with IL-rules. On the other hand, we obtained the lower average number of rules (3.08) and antecedents (4.76) with this measure, which is probably due to its bias to penalize too specific rules. These results corroborate those obtained by Todorovski et al. in [34], which conclude that WRAcc reduces considerably the size of the rule sets at the expense of accuracy.

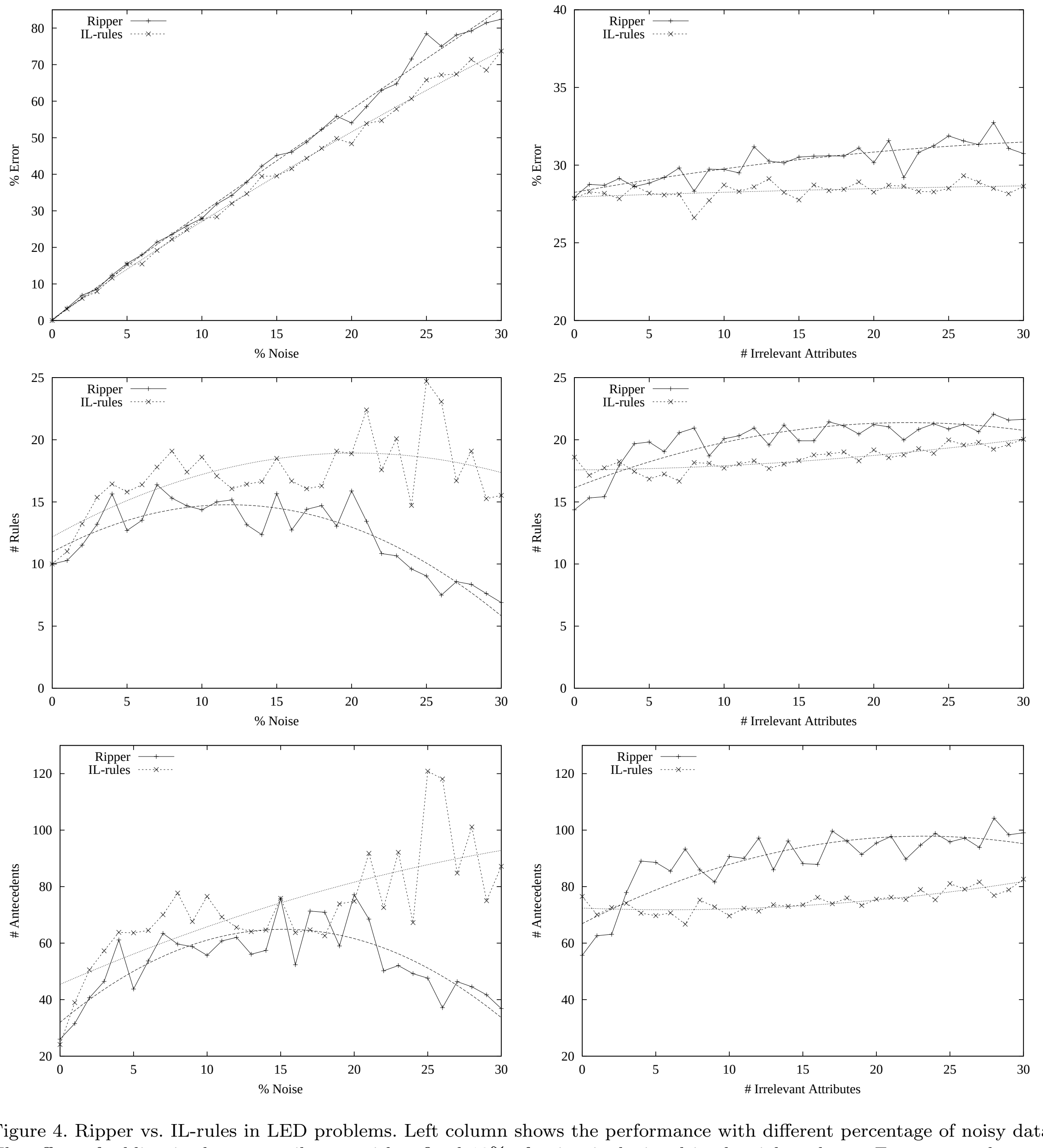

Figure 4. Ripper vs. IL-rules in LED problems. Left column shows the performance with different percentage of noisy data. The effect of adding irrelevant attributes, with a fixed 10% of noise, is depicted in the right column. From top to bottom, these graphics show the evolution of classification errors, number of rules, and number of antecedents, all with their trend lines.

Table 6
Cross validation scores reached by Ripper and IL-rules in a collection of datasets described only by continuous attributes.

	IL-rules			Ripper		
Dataset	Error	# Rules	# Ant.	Error	# Rules	# Ant.
BREAST (Winsconsin)	3.83	4.98	10.60	4.41	6.00	15.10
G2	18.31	6.20	11.42	20.06	4.34	9.04
GL	30.57	8.68	26.36	34.21	9.48	26.28
HEART	18.44	8.18	21.42	19.41	4.18	10.14
IONOSFERE	10.60	5.98	13.26	10.88	5.22	10.66
IR	5.07	3.28	4.20	5.20	3.08	5.02
PIMA	25.94	8.74	19.64	24.72	3.64	9.76
VEHICLE	29.50	19.84	68.44	32.66	14.50	48.62
WAVEFORM-21	20.08	36.70	173.02	20.50	27.70	156.04
WAVEFORM-40	19.79	31.06	143.94	20.88	30.00	158.86
WINE	5.29	4.38	9.04	6.75	4.38	9.10
Av.	17.04	12.55	45.58	18.15	10.23	41.69

7. Conclusions

We have presented a heuristic measure that tries to capture the classification quality of a rule when it is applied by means of a minimum-distance criterion. We call it *impurity level*, and it was used in [9,10,20,21,27,28,29] to build different machine learning algorithms with the common property of producing a very small set of accurate rules. In this paper we have studied the possibility of using the *impurity level* to drive both a decision tree generation algorithm, and a pruning method to induce classification rule sets. For this purpose, we built a learning algorithm, IL-rules, entirely based on our measure using the structure of Quinlan's C4.5-rules.

The experiments reported in the previous section endorse the ability of our pruning method to obtain more concise rule sets than C4.5-rules. Furthermore, these rule sets are also slightly more accurate in the collection of Holte's datasets used for experimentation.

In fact, the scores of IL-rules are quite similar to those reached by Cohen's Ripper. The differences between our algorithm and Ripper can be explained by the use of partial (IL-rules) and full matching (Ripper). We have illustrated this point experimentally, comparing the accuracy and size of rule sets induced from datasets whose examples are described by continuous attributes. Additionally, we studied the performance of both systems with respect to the presence of noisy data and irrelevant attributes.

8. Acknowledgments

This work was partially supported with funds from the Spanish Ministerio de Ciencia y Tecnología and Feder under grants PB98-1556 and TIC2001-3579. We also want to thank: the anonymous reviewers and the action editor, Gerhard Widmer, for their invaluable comments to improve this work; David Aha for some suggestions during the writing of this paper; and William Cohen for sharing the source code of Ripper.

References

[1] D. Aha. *A Study of Instance-based Algorithms for Supervised Learning Tasks: Mathematical, Empirical, and Psychological Evaluations.* PhD thesis, University of California at Irvine, 1990.

[2] D. Aha, D. W. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[3] C. L. Blake and C. J. Merz. UCI repository of machine learning databases. [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. University of California, Irvine, Dept. of Information and Computer Sciences, 1998.

[4] C. Borgelt, J. Gebhardt, and R. Kruse. Concepts for probabilistic and possibilistic induction of decision trees on real world data. In *Proc. of the EUFIT '96*, volume 3, pages 1556–1560, 1996.

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth International Group, 1984. Belmont, CA.

[6] L. A. Breslow and D. W. Aha. Simplifying decision trees: a survey. *Knowledge Engineering Review*, 12(1):1–40, 1997.

[7] W. W. Cohen. Efficient pruning methods for separate-and-conquer rule learning systems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 988–994, 1993.

[8] W. W. Cohen. Fast effective rule induction. In *International Conference on Machine Learning*, pages 115–123, 1995.

[9] J. J. del Coz, O. Luaces, J. R. Quevedo, J. Alonso, J. Ranilla, and A. Bahamonde. Self-organizing cases to find paradigms. In *Proceedings of the International Work-Conference on Artificial and Natural Neural Networks, IWANN '99*, volume 1606 of *LNCS*, pages 527–536. Springer-Verlag, 1999.

[10] J. J. del Coz. BETS*: Sistema de aprendizaje basado en la selección de ejemplos paradigmáticos*. PhD thesis, Dept. of Computer Science, University of Oviedo at Gijón, 2000.

[11] T. Dietterich, M. Kearns, and Y. Mansour. Applying the weak learning framework to understand and improve C4.5. In *Proc. 13th International Conference on Machine Learning*, pages 96–104. Morgan Kaufmann, 1996.

[12] P. Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24:141–168, 1996.

[13] J. Fürnkranz. Pruning algorithms for rule learning. *Machine Learning*, 27(2):139–171, 1997.

[14] J. Fürnkranz and G. Widmer. Incremental reduced error pruning. In *International Conference on Machine Learning*, pages 70–77, 1994.

[15] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[16] E. B. Hunt, J. Marin, and P. T. Stone. *Experiments in Induction.* Academic Press, New York, NY, 1966.

[17] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[18] R. Kohavi, G. John, R. Long, and D. Manley. MLC++: A machine learning library in C++. In *Proceedings of the Sixth International Conference on Tools with Artificial Intelligence*, pages 740–743. IEEE Computer Society Press, 1994.

[19] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145, 1995.

[20] O. Luaces. *Un Sistema de Aprendizaje de Reglas Explícitas mediante la Generalización de Instancias.* PhD thesis, Department of Computer Science, University of Oviedo at Gijón, 1999.

[21] O. Luaces and A. Bahamonde. Inflating examples to obtain rules. Technical report, Artificial Intelligence Center, University of Oviedo at Gijón, 2000.

[22] O. Luaces, J. J. del Coz, J. R. Quevedo, J. Alonso, J. Ranilla, and A. Bahamonde. Autonomous clustering for machine learning. In *Proceedings of the International Work-Conference on Artificial and Natural Neural Networks, IWANN '99*, volume 1606 of *Lecture Notes in Computer Sciences*, pages 497–506, Alicante, Spain, 1999. Springer-Verlag.

[23] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[24] J. R. Quinlan. Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 463–482. Tioga, Palo Alto, 1983.

[25] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, San Mateo, California, 1993.

[26] J. R. Quinlan. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[27] J. Ranilla and A. Bahamonde. FAN: Finding Accurate iNductions. *International Journal of Human Computer Studies*, 56(4):445–474, June 2002.

[28] J. Ranilla, R. Mones, and A. Bahamonde. El nivel de impureza de una regla de clasificación aprendida a partir de ejemplos. *Novática*, 131:37–43, 1998.

[29] J. Ranilla. *ABANICO: Aprendizaje Basado en Agrupación Numérica en Intervalos Continuos.* PhD thesis, Department of Computer Science, University of Oviedo at Gijón, 1998.

[30] L. Rendell. A general framework for induction and a study of selective induction. *Machine Learning*, 1(2):177–226, 1986.

[31] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431, 1983.

[32] S. Salzberg. *Learning with Nested Generalized Exemplars.* Kluwer Academic Publishers, Boston, MA, 1990.

[33] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423 (Part I) and 623–656 (Part II), 1948.

[34] L. Todorovski, P. Flach, and N. Lavrač. Predictive performance of weighted relative accuracy. In Djamel A. Zighed, Jan Komorowski, and Jan Zytkow, editors, *4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2000)*, pages 255–264. Springer-Verlag, 2000.

[35] D. Wettschereck and T. G. Dietterich. An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, 19:5–27, 1995.

[36] D. R. Wilson and T. R. Martínez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6(1):1–34, 1997.