TWO-WAY INDUCTION

PEDRO DOMINGOS
Department of Information and Computer Science
University of California, Irvine
Irvine, California 92717, U.S.A.

Received 14 April 1995
Revised 10 January 1996

ABSTRACT

General-to-specific learners like ID3 and CN2 perform well when the target concept descriptions are general, but often have difficulties when they are specific or mixed. This problem can be alleviated by combining them with a specific-to-general learning component, resulting in a two-way induction system. In this paper one design for such a component is proposed, as well as two methods for combining the two components. Experiments on artificial domains show the combined learner to match or outperform "pure" versions of C4.5 and CN2 across the entire generality spectrum, with the advantage increasing for greater concept specificity. Experiments on 24 real-world domains from the UCI repository confirm the utility of two-way induction: the combined learner achieves higher accuracy than C4.5 in 17 domains (at the 5% significance level in 12), and similar results are obtained with CN2. Closer observation of the system's behavior leads to a better understanding of its ability to correct overly-general rules with specific ones, and shows that there is still room for improvement.

Keywords: Machine learning, rule induction, knowledge acquisition, search.

1 Introduction and Motivation

Most widely-used decision-tree and rule learners perform general-to-specific induction: they start with an empty concept description, and gradually add restrictions to it until there is not enough evidence to continue, or perfect discrimination is achieved. ID3/C4.5 [14] and CN2 [4] are examples of systems that function in this way. They have some significant advantages: because only statistical measures are used to evaluate induction steps, as opposed to individual examples being considered, good noise immunity can be achieved. Also, when the induction process stops early, as it often does, learning can be fast, and the resulting descriptions concise.

However, as in any search process, finding the target tends to be harder when it lies farther from the search's starting point, in this case the null concept description. The opportunities for the search to take the wrong path accumulate, and the probability of an incorrect end result increases. As a result, if all or part of the concept description is fairly specific, i.e., if it includes a large fraction of the attributes used to describe examples, systems like C4.5 and CN2 may be unable to find it; they are prone to either stopping the search too soon, leading to an overly general description, or to inducing erroneous rules/branches that may or may not be pruned. This performance degradation as concept specificity increases is indeed verified in artificial domains in a later section of this paper.

Specific-to-general learners, on the other hand, should find it easier to induce fairly specific descriptions. They have other disadvantages, however: search has to start from specific examples, making it more sensitive to noise, and early decisions, based on little evidence, can be wrong and difficult to correct later on. As a result, even though some specific-to-general learners exist (e.g., EACH [16]), they have not come into widespread use.

More generally, when the target description is a mix of general rules and more specific "exception" areas—a pattern which appears to occur frequently in real applications—we can expect both approaches to have problems: general-to-specific learners may not recognize the exception areas, and specific-to-general ones may induce only imperfect, corrupted versions of the general rules. A natural solution would then be to combine the two search directions in a single system, allowing both the induction of rules starting from the null description and from specific examples, and employing some conflict resolution strategy in parts of the instance space where rules' predictions disagree. This paper describes and evaluates one realization of this idea.

C4.5 and CN2 were used in turn as the general-to-specific learner. The next section describes the specific-to-general component used, and the following one describes two methods for combining the two components. The results of experiments on natural and artificial domains are then reported and interpreted. Finally related work is discussed, and some conclusions and directions for future work are presented.

2 Specific-to-General Induction

Specific-to-general induction is performed by the RISE algorithm, of which the learning and classification procedures will be considered in turn. More details can be found in [7, 6].

2.1 Representation and search

Each example is a vector of attribute-value pairs, together with a specification of the class to which it belongs; attributes can be either nominal (symbolic) or

Table 1: The RISE algorithm.

Input: ES is the training set.

Procedure RISE (ES)

Let RS be ES.
Compute $Acc(RS)$.
Repeat
  For each rule R in RS,
    Find the nearest example E to R not already covered by it
      (and of the same class).
    Let R' = Most_Specific_Generalization(R, E).
    Let RS' = RS with R replaced by R'.
    If $Acc(RS') \geq Acc(RS)$
    Then Replace RS by RS',
      If R' is identical to another rule in RS,
      Then delete R' from RS.
Until no increase in $Acc(RS)$ is obtained.
Return RS.

---

numeric. Each rule consists of a conjunction of antecedents and a predicted class. Each antecedent is a condition on a single attribute, and there is at most one antecedent per attribute. Conditions on nominal attributes are equality tests of the form $a_i = v_j$, where a_i is the attribute and v_j is one of its legal values. Conditions on numeric attributes take the form of allowable intervals for the attributes, i.e., $a_i \in [v_{j1}, v_{j2}]$, where v_{j1} and v_{j2} are two legal values for a_i. Exemplars (i.e., examples used as prototypes for classification) are viewed as maximally specific rules, with conditions on all attributes and degenerate (point) intervals for numeric attributes. A rule is said to *cover* an example if the example satisfies all of the rule's conditions; a rule is said to *win* an example if it is the nearest rule to the example according to the distance metric that will be described below.

The RISE algorithm is summarized in Table 1. RISE searches for "good" rules in a specific-to-general fashion, starting with a rule set that is the training set of examples itself. RISE looks at each rule in turn, finds the nearest example of the same class that it does not already cover (i.e., that is at a distance greater than zero from it), and attempts to minimally generalize the rule to cover it. The generalization procedure is outlined in Table 2. If the change's effect on global accuracy is positive, it is retained; otherwise it is discarded. Generalizations are also accepted if they appear to have no effect on accuracy; this reflects a simplicity bias. This procedure is repeated until, for each rule, attempted generalization fails.

A potential difficulty is that measuring the accuracy of a rule set on the training

Table 2: Generalization of a rule to cover an example.

Inputs: $R = (a_1, a_2, \ldots, a_A, c_R)$ is a rule,
$E = (e_1, e_2, \ldots, e_A, c_E)$ is an example.
a_i is either True, $x_i = r_i$, or $r_{i,lower} \leq x_i \leq r_{i,upper}$.

Function Most_Specific_Generalization (R, E)

For each attribute i,
If a_i = True then Do nothing.
Else if i is symbolic and $e_i \neq r_i$ then a_i = True.
Else if $e_i > r_{i,upper}$ then $r_{i,upper} = e_i$.
Else if $e_i < r_{i,lower}$ then $r_{i,lower} = e_i$.

---

set requires matching all rules with all training examples, and this would entail a high computational cost if it was repeatedly done as outlined. Fortunately, at each step only the *change* in accuracy needs to be computed. Each example memorizes the distance to its nearest rule and its assigned class. When a rule is generalized, all that is necessary is then to match that single rule against all examples, and check if it wins any that it did not before, and what its effect on these is. Previously misclassified examples that are now correctly classified add to the accuracy, and previously correctly classified examples that are now misclassified subtract from it. If the former are more numerous than the latter, the change in accuracy is positive, and the generalization is accepted. With this optimization, RISE's worst-case time complexity has been shown to be quadratic in the number of examples and the number of attributes, which is comparable to that of general-to-specific rule induction algorithms [7].

2.2 Classification

At performance time, classification of each test example is performed by finding the nearest rule to it, and assigning the example to the rule's class. The distance measure used is a combination of Euclidean distance for numeric attributes, and a simplified version of Stanfill and Waltz's value difference metric for symbolic attributes [18].

Let $E = (e_1, e_2, \ldots, e_A, c_E)$ be an example with value e_i for the ith attribute and class c_E. Let $R = (a_1, a_2, \ldots, a_A, c_R)$ be a rule with class c_R and condition a_i on the ith attribute, where a_i = True if there is no condition on i, otherwise a_i is $x_i = r_i$ if i is symbolic and a_i is $r_{i,lower} \leq x_i \leq r_{i,upper}$ if i is numeric. The distance $\Delta(R, E)$ between R and E is then defined as:

$$\Delta(R, E) = \sum_{i=1}^{A} \delta^2(i) \tag{1}$$

where the component distance $\delta(i)$ for the ith attribute is:

$$\delta(i) = \begin{cases} 0 & \text{if } a_i = \text{True} \\ SVDM(r_i, e_i) & \text{if } i \text{ is symbolic} \wedge a_i \neq \text{True} \\ \delta_{num}(i) & \text{if } i \text{ is numeric} \wedge a_i \neq \text{True} \end{cases} \tag{2}$$

$SVDM(r_i, e_i)$ is the simplified value difference metric, defined as:

$$SVDM(x_i, x_j) = \sum_{h=1}^{C} |P(c_h|x_i) - P(c_h|x_j)| \tag{3}$$

where x_i and x_j are any legal values of the attribute, C is the number of classes, c_h is the hth class, and $P(c_h|x_i)$ denotes the probability of c_h conditioned on x_i. The essential idea behind VDM-type metrics is that two values should be considered similar if they make similar class predictions, and dissimilar if their predictions diverge. This has been found to give good results in several domains [5]. Notice that, in particular, $SVDM(x_i, x_j)$ is always 0 if $i = j$.

The component distance for numeric attributes is defined as:

$$\delta_{num}(i) = \begin{cases} 0 & \text{if } r_{i,lower} \leq e_i \leq r_{i,upper} \\ \frac{e_i - r_{i,upper}}{x_{max} - x_{min}} & \text{if } e_i > r_{i,upper} \\ \frac{r_{i,lower} - e_i}{x_{max} - x_{min}} & \text{if } e_i < r_{i,lower} \end{cases} \tag{4}$$

x_{max} and x_{min} being respectively the maximum and minimum observed values for the attribute.

The distance from a missing numeric value to any other is defined as 0. If a symbolic attribute's value is missing, it is assigned the special value "?". This is treated as a legitimate symbolic value, and its distance to all other values of the attribute is computed and used. When coupled with SVDM, this is a sensible policy: a missing value is taken to be roughly equivalent to a given possible value if it behaves similarly to it, and inversely if it doesn't.

When two or more rules are equally close to a test example, the rule that was most accurate on the training set wins. So as to not unduly favor more specific rules, the Laplace-corrected accuracy is used [12]:

$$LAcc(R) = \frac{N_{corr}(R) + 1}{N_{won}(R) + C} \tag{5}$$

where R is any rule, C is the number of classes, $N_{won}(R)$ is the total number or examples won by R, $N_{corr}(R)$ is the number of examples among those that R correctly classifies, and C is the number of classes. The effect of the Laplace correction is to make the estimate of a rule's accuracy converge to the "random

guess" value of $1/C$ as the number of examples won by the rule decreases. Thus rules with high apparent accuracy are favored only if they also have high statistical support, i.e., if that apparent accuracy is not simply the result of a small sample.

3 Two-Way Induction

Applying the RISE algorithm to a training set produces a set of rules which will henceforth be called the S rules (for "specific"). Applying the general-to-specific learner produces a set of G rules. If CN2 is used, the rules it outputs are used directly. If C4.5 is used, C4.5RULES is first applied to convert the decision tree it produces into a set of rules [14].

Two algorithms for combining the S and G rules were designed, TWI-1 and TWI-2. In TWI-1, the sets of S and G rules are merged to form a single set, deleting any duplicates. The Laplace accuracy of each rule on the examples that it *covers* is then measured, and the classification procedure is applied to each training example in turn; each rule memorizes the number of examples it won, and how many of them it classified correctly. At the end, the Laplace accuracy of each rule on the examples that it *won* is computed, and this is the value retained. The earlier estimates were necessary to break ties during the classification cycle that produced the final ones. At classification time, the procedure described in the previous section is applied without any distinction between S and G rules. The nearest rule to the test example wins; if two or more rules are equally near, the one with the highest Laplace accuracy prevails.

In the TWI-2 algorithm, the sets of S and G rules are kept separate, and the Laplace accuracy of each rule on the training examples that it covers is measured. At classification time, the two sets of rules are first applied separately. A winner among the S rules is found by the procedure previously outlined. To select the G winner, the G rules are matched against the test example. If only one G rule covers it, that rule is chosen as the G winner. If more than one rule covers the example, the one with the highest Laplace accuracy wins. If no G rule covers the example, the default rule is chosen as the G winner. Of the two finalists (S and G), the one with the highest accuracy then wins and classifies the example.

4 Empirical Study

An empirical study was conducted to evaluate the contribution of the specific-to-general learner, and the performance of the two-way induction system relative to its individual components. A recent version of CN2 was used [3]. The default settings for C4.5, C4.5RULES and CN2 were kept. Note that all results for C4.5RULES (abbreviated C4.5R) and CN2 below are those obtained using their own classification procedures, not the one above for G rules. Experiments in artificial and real-world domains are described in turn.

4.1 Artificial domains

The goal of the study in artificial domains was to verify in ideal, controlled conditions if the hypotheses put forth in the introduction to this paper hold. Following a philosophy justified elsewhere [17, 1], classes of domains rather that individual concept definitions were considered. The study was carried out using C4.5R, RISE and TWI-2.

The independent variable of interest is the specificity of the target concept description. A good operational measure of it is the average length of the rules comprising the description: rules with more conditions imply a more specific concept. The dependent variables are the out-of-sample accuracies of the three algorithms. Concepts defined as Boolean functions in disjunctive normal form were used as targets. The datasets were composed of 100 examples described by 16 attributes. The average number of literals C in each disjunct comprising the concept was varied from 1 to 16. The number of disjuncts was set to $\text{Min}\{2^{C-1}, 25\}$. This attempts to keep the fraction of the instance space covered by the concept roughly constant, up to the point where it would require more rules than could possibly be learned. Equal numbers of positive and negative examples were included in the dataset, and positive examples were divided evenly among disjuncts. In each run a different target concept was used, generating the disjuncts at random, with length given by a binomial distribution with mean C and variance $C(1-\frac{C}{16})$; this is obtained by including each feature in the disjunct with probability $\frac{C}{16}$. Twenty runs were conducted, with two-thirds of the data used for training and the remainder for testing.

The results are shown graphically in Fig. 1. The most salient aspect is the large difference in difficulty between short and long rules for all learners. Concepts with very few (approx. three or less) conditions per rule are so simple that both types of learner are able to learn them easily; the expected advantage of the general-to-specific approach is not realized. In separate experiments, corrupting the data with 10% and 20% noise degraded the performance of all algorithms equally, again giving no advantage to C4.5R. At the other end, however, specific-to-general learning has a clear advantage for concepts with 12 or more conditions per rule; all differences here are significant at the 5% level using a one-tailed paired t test. (Error bars are not shown to avoid further cluttering the graph.) The combined approach is able to effectively leverage both components, even when one is substantially less accurate than the other; TWI performs better than the best of C4.5 and RISE in 10 of the 16 data points, including better than RISE at the 5% error level for $C = 13$ and $C = 14$.

The slight upward trend in C4.5R's curve for $C > 10$ was investigated by repeating the experiments with 32 attributes, 400 examples, a maximum of 50 rules and $C = 1, \ldots, 32$. C4.5R's lag increases, but the upward trend is maintained; on inspection of the rules C4.5R produces, this is revealed to be due to the fact that, as the concept rules become more and more specific, it becomes possible to induce short rules for its negation. The hardest concepts, for which both the concept and

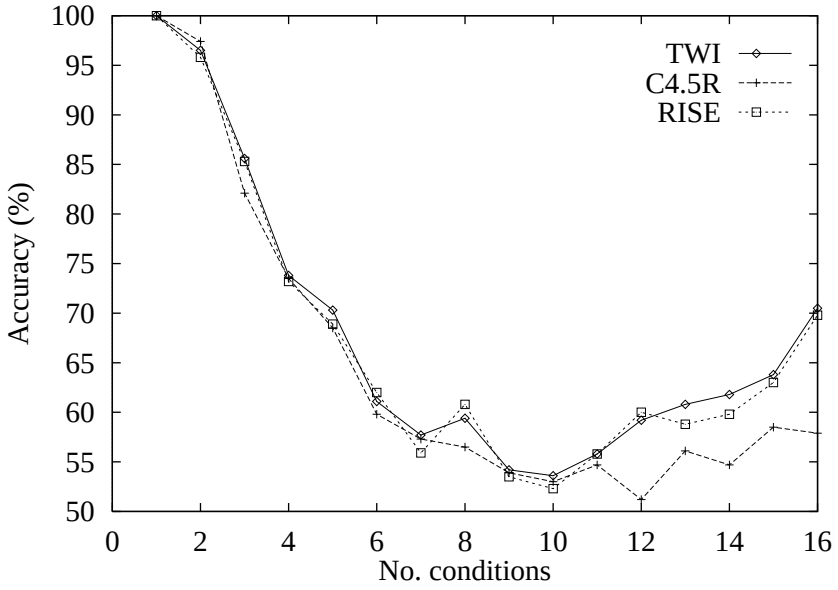

Figure 1: Accuracy as a function of concept specificity.

its negation have necessarily long rules, are for intermediate values of C.

4.2 Natural domains

Whether the expected advantages of adding a specific-to-general component to a general-to-specific learner are realized in real-world applications was investigated by carrying out experiments on 24 domains from the UCI repository [11]. Again, 20 runs were conducted for each dataset, with two-thirds of the data used for training. The default classifier (assigning examples to the most frequent class) was included as a baseline. The results are shown in Table 3 for C4.5 used as the G component, and in Table 4 for CN2. The average accuracy and sample standard deviation are tabulated for each algorithm in each domain. The superscripts denote significance levels for the difference in accuracy between TWI-2 and the corresponding algorithm, using a one-tailed paired t test: 1 denotes 0.5%, 2 is 1%, 3 is 2.5%, 4 is 5%, 5 is 10%, and 6 is above 10%. For example, in the horse colic domain in Table 3, TWI-2 obtains an accuracy of 83.8% with a standard deviation of 3.5%, is better than C4.5 at the 1% level, and better than RISE at the 0.5% level.

Table 5 summarizes these results. The first line shows the number of domains in which TWI-2 achieved higher accuracy than the corresponding system, vs. the number in which the reverse happened. The second line considers only those do-

Table 3: Experimental results when G is C4.5.

Domain	TWI-2	TWI-1	C4.5	RISE	Default
Audiology	78.9±4.3	66.3±6.6[1]	70.6±5.7[1]	77.3±4.9[1]	21.3±2.6[1]
Breast	69.4±4.5	67.5±5.5[4]	67.8±6.6[5]	68.2±4.2[3]	67.6±7.6[6]
Credit	83.5±1.7	84.2±2.9[6]	84.8±2.5[3]	83.1±2.0[4]	57.4±3.8[1]
Echocard.	68.1±4.9	67.3±8.2[6]	65.9±7.6[6]	67.4±4.9[5]	67.8±6.6[6]
Glass	70.2±6.5	64.1±10.1[2]	64.9±9.2[1]	69.7±6.1[6]	31.7±5.5[1]
Heart	79.8±3.7	76.6±3.1[1]	77.8±4.3[4]	79.6±3.9[6]	55.0±3.4[1]
Hepatitis	78.1±5.4	78.8±3.7[6]	78.6±5.3[6]	76.9±5.3[3]	78.1±3.1[6]
Horse colic	83.8±3.5	81.2±6.3[5]	81.2±4.4[2]	81.9±3.2[1]	63.6±3.9[1]
Iris	93.0±2.7	93.6±2.7[6]	93.2±2.5[6]	92.9±2.8[6]	26.5±5.2[1]
LED	69.5±4.0	69.1±3.8[6]	69.0±3.8[6]	67.9±3.6[3]	9.9±3.0[1]
Labor	83.7±9.9	81.6±10.7[6]	86.3±8.6[6]	89.2±10.6[3]	65.0±9.5[1]
Liver	64.0±5.6	65.8±4.6[6]	64.4±3.9[6]	63.4±5.4[4]	58.1±3.4[1]
Lung	48.6±15.1	40.5±14.0[4]	40.5±14.0[4]	50.5±15.2[5]	26.8±12.3[1]
Lymphogr.	78.4±6.2	76.9±3.9[6]	75.6±4.9[4]	80.2±6.8[1]	57.3±5.4[1]
Diabetes	70.8±3.0	73.0±3.1[2]	74.3±3.0[1]	70.4±3.0[4]	66.0±2.3[1]
Post-oper.	67.8±6.0	71.0±5.2[2]	68.2±6.9[6]	62.3±9.1[1]	71.2±5.2[2]
Pr. tumor	41.4±4.6	35.4±5.4[1]	37.5±5.7[1]	39.8±5.3[1]	24.6±3.2[1]
Promoters	87.7±5.8	80.6±10.1[1]	80.4±10.0[1]	87.7±5.5[6]	43.1±4.2[1]
Solar flare	71.9±3.1	71.2±4.1[6]	71.1±4.1[6]	70.8±2.7[2]	25.2±4.4[1]
Sonar	74.7±12.1	64.3±9.4[1]	65.4±7.1[1]	76.0±11.4[3]	50.8±7.6[1]
Soybean	83.1±6.8	75.1±5.8[1]	78.9±5.9[1]	84.8±6.5[1]	9.1±2.1[1]
Voting	95.9±1.6	95.7±1.7[6]	95.8±1.3[6]	95.5±1.5[5]	60.5±3.1[1]
Wine	96.3±2.3	91.3±5.6[1]	91.8±5.6[1]	96.9±1.8[5]	36.4±9.9[1]
Zoology	93.5±3.8	90.0±5.2[1]	89.6±4.7[1]	93.2±3.7[6]	39.4±6.4[1]

mains in which the observed difference is significant at the 5% level, and shows that most of the previous "wins" were indeed significant. For example, when G was C4.5, TWI-2 did better than C4.5 in 17 domains overall, and worse in 7; with 5% significance it did better in 12 and worse in 2. All other comparisons are similarly favorable to TWI-2. The third line shows the results of applying a sign test to the values of line one. This consists of considering the number of wins as a binomial variable, and asking how unlikely the value obtained is if TWI-2 is assumed to be no more accurate than the corresponding algorithm. For example, 17 wins in 24 trials has a probability of occurrence of only 3%. This test shows that all the numbers of wins obtained are significant at the 5% level, resulting in high confidence that TWI-2 is a more accurate learner than either component alone, if the set of domains used is assumed to be representative of real-world tasks.

4.3 Discussion

The utility of two-way induction is clearly shown by the results above, but it is important to understand why this advantage is observed. Inspection of the S and G

Table 4: Experimental results when G is CN2.

Domain	TWI-2	TWI-1	CN2	RISE	Default
Audiology	77.3±4.4	63.9±7.4[1]	71.0±5.1[1]	77.3±4.9[6]	21.3±2.6[1]
Breast	68.4±6.3	67.9±6.4[6]	67.9±7.1[6]	68.2±4.2[6]	67.6±7.6[6]
Credit	84.4±2.2	82.3±2.2[1]	82.0±2.2[1]	83.1±2.0[1]	57.4±3.8[1]
Echocard.	68.0±5.3	66.0±5.9[3]	68.2±7.2[6]	67.4±4.9[6]	67.8±6.6[6]
Glass	69.9±6.8	58.9±6.3[1]	63.8±5.5[1]	69.7±6.1[6]	31.7±5.5[1]
Heart	81.2±4.3	79.5±2.9[3]	79.7±2.9[4]	79.6±3.9[2]	55.0±3.4[1]
Hepatitis	79.6±5.3	80.5±4.5[6]	80.3±4.2[6]	76.9±5.3[1]	78.1±3.1[5]
Horse colic	83.1±3.6	83.0±3.6[6]	82.5±4.2[6]	81.9±3.2[1]	63.6±3.9[1]
Iris	93.4±2.7	93.2±2.8[6]	93.3±3.6[6]	92.9±2.8[4]	26.5±5.2[1]
LED	69.5±4.0	68.9±4.5[6]	69.5±3.8[6]	67.9±3.6[1]	9.9±3.0[1]
Labor	87.4±10.6	82.9±8.9[3]	82.1±6.9[4]	89.2±10.6[6]	65.0±9.5[1]
Liver	65.9±5.2	66.4±3.8[6]	65.0±3.8[6]	63.4±5.4[1]	58.1±3.4[1]
Lung	42.3±14.5	39.1±14.8[6]	38.6±13.5[6]	50.5±15.2[1]	26.8±12.3[1]
Lymphogr.	80.9±6.3	79.4±5.0[5]	78.8±4.9[4]	80.2±6.8[6]	57.3±5.4[1]
Diabetes	72.3±2.7	73.7±2.9[2]	73.8±2.7[1]	70.4±3.0[1]	66.0±2.3[1]
Post-oper.	68.3±4.9	65.0±6.9[3]	60.8±8.2[1]	62.7±9.2[1]	71.2±5.2[1]
Pr. tumor	41.4±5.2	38.9±5.1[2]	39.8±5.2[4]	39.8±5.3[1]	24.6±3.2[1]
Promoters	85.6±6.2	78.1±9.1[1]	75.9±8.8[1]	87.7±5.5[5]	43.1±4.2[1]
Solar flare	71.4±2.6	69.6±3.6[2]	70.4±3.0[5]	70.8±2.6[6]	25.2±4.4[1]
Sonar	73.8±10.6	63.2±9.0[1]	66.2±7.5[1]	76.0±11.4[3]	50.8±7.6[1]
Soybean	83.5±6.7	77.8±6.7[1]	77.4±7.2[1]	84.8±6.5[1]	9.1±2.1[1]
Voting	95.0±1.8	95.5±1.5[4]	95.8±1.6[1]	95.5±1.5[4]	60.5±3.1[1]
Wine	95.3±2.6	91.0±4.5[1]	90.8±4.7[1]	96.9±1.8[1]	36.4±9.9[1]
Zoology	93.7±3.3	90.9±5.1[1]	90.6±5.0[1]	93.2±3.7[5]	39.4±6.4[1]

rules shows that the S rules are indeed substantially more specific than the G ones. G rules typically contain a small number of conditions (approx. 1 to 5), whereas S rules often contain conditions on half of all the attributes. Further, tracing of TWI-2 reveals that by far the majority of S wins occur when the test example is also covered by a G rule, but G rules are still correct on most of the examples they cover. S rules thus encapsulate small exception areas to the broad generalizations represented by G rules, as intended.

TWI-2 performed better than TWI-1, and this was to be expected. TWI-1's naive method of combining S and G rules (merge all) has several disadvantages. The G rules were not designed to be applied in a best-match manner. Being very general, they easily win examples outside them over more appropriate S rules. The S rules, conversely, were designed for best-match classification, but in the large sections of the instance space covered by G rules, they can only win examples that they actually cover.

TWI-2's accuracy is bounded from below by the fraction of cases on which the S and G components agree and are correct, and from above by 100% minus the

Table 5: Summary of TWI-2's results vs. other algorithms.

Measure	G = C4.5			G = CN2		
	TWI-1	C4.5	RISE	TWI-1	CN2	RISE
No. wins	18-6	17-7	17-6	20-4	19-4	16-7
No. signif. wins	11-2	12-2	11-4	14-2	13-2	10-5
Sign test	1.0	3.0	2.0	0.1	0.2	5.0

fraction of cases on which they agree and are both incorrect. Further, when there are more than two classes, S and G components may differ and both be wrong. Extracting the cases where TWI-2's decision makes a difference, we see that it makes the correct decision approximately two-thirds of the time, on average. This is well above chance, but still leaves substantial room for improvement, at least in theory.

The study above was mainly concerned with accuracy. Two other variables of interest are speed and comprehensibility of the results. In practice, RISE is slower than C4.5 and CN2, but this is only noticeable in larger datasets (more than 500 examples). Also, because the S rules are longer, they are harder for a human to understand; but since in TWI-2 the distinction between S and G rules is maintained, the G rules can be seen as an approximate and accessible model of the domain, while the S rules represent second-order refinements and exceptions that should be taken into account to achieve higher accuracy.

5 Related Work

Mitchell's [10] version space approach is an early example of two-way induction. TWI is a heuristic algorithm, in contrast to Mitchell's exhaustive candidate elimination procedure, but has the advantage that its worst-case space and time complexities are respectively linear and low-order polynomial, instead of exponential. Also, unlike version spaces, TWI is able to deal with disjunctive concepts, noise, and missing and continuous attributes.

The research described in this paper falls in the general area of empirical multi-strategy learning, which attempts to produce more accurate learners by combining two or more individual algorithms [9]. The MCS system combines decision trees with the nearest-neighbor algorithm and linear machines [2]. FCLS combines rules with specific examples in a best-match framework [20]. Quinlan has combined decision trees and other methods with nearest-neighbor in regression tasks [15]. The post-pruning step used in many rule and decision-tree learners can be seen as a form of specific-to-general induction, and Quinlan [13] and others have shown it to be effective. Specific-to-general induction is performed by systems like NGE [16] and KBNGE [19], but neither combines it with general-to-specific learning. Golding and Rosenbloom's Anapron system for name pronunciation [8] performs search in neither direction, but is similar in spirit to TWI: a set of expert-supplied rules

functions as the G component, and a set of implicit rules (the "arules") contained in a case base functions as the S component.

6 Conclusions and Future Work

This paper has shown that the accuracy of general-to-specific learners like ID3/C4.5 and CN2 can be substantially improved by adding a specific-to-general component to them. As the observations made indicate, there is still room for improvement, and work on more sophisticated methods of combining the two components is underway. Inputting unpruned rules or trees and combining the post-pruning stage with the specific-to-general induction process is also a potentially productive avenue, and preliminary work has been done. Another direction for future research is to use an expert-supplied domain theory as the G component, leading to a combined analytical-empirical learner.

Acknowledgments

This work was partly supported by JNICT/ Programa Ciência and Fulbright scholarships. The author is grateful to Dennis Kibler and Mike Pazzani, to the creators of the C4.5 and CN2 systems, and to all the people who provided the datasets used in the empirical study. Please see the documentation in the UCI Repository for detailed information.

References

[1] D. W. Aha, *Generalizing from case studies: A case study,* Proc. 9th International Workshop on Machine Learning, Aberdeen, Scotland (1992) 1–10.

[2] C. E. Brodley, *Addressing the selective superiority problem: Automatic algorithm/model class selection,* Proc. 10th International Conference on Machine Learning, Amherst, MA (1993) 17–24.

[3] P. Clark and R. Boswell, *Rule induction with CN2: Some recent improvements,* Proc. 6th European Working Session on Learning, Porto, Portugal (1991) 151–163.

[4] P. Clark and T. Niblett, *The CN2 induction algorithm,* Machine Learning **3** (1989) 261–283.

[5] S. Cost and S. Salzberg, *A weighted nearest neighbor algorithm for learning with symbolic features,* Machine Learning **10** (1993) 57–78.

[6] P. Domingos, *The RISE 2.0 system: A case study in multistrategy learning,* Technical Report 95-2, Department of Information and Computer Science, University of California at Irvine (1995).

[7] P. Domingos, *Rule induction and instance-based learning: A unified approach,* Proc. 14th International Joint Conference on Artificial Intelligence, Montreal, Canada (1995) 1226–1232.

[8] A. R. Golding and P. S. Rosenbloom, *Improving rule-based systems through case-based reasoning,* Proc. 9th National Conference on Artificial Intelligence, Menlo Park, CA (1991) 22–27.

[9] R. Michalski and G. Tecuci, eds., *Machine Learning: A Multistrategy Approach,* Morgan Kaufmann, San Mateo, CA (1994).

[10] T. M. Mitchell, *Generalization as search,* Artificial Intelligence **18** (1982) 203–226.

[11] P. M. Murphy and D. W. Aha, *UCI repository of machine learning databases,* machine-readable data repository, Department of Information and Computer Science, University of California at Irvine, 1995.

[12] T. Niblett, *Constructing decision trees in noisy domains,* Proc. 2nd European Working Session on Learning, Bled, Yugoslavia (1987) 67–78.

[13] J. R. Quinlan, *Simplifying decision trees,* International Journal of Man-Machine Studies **27** (1987) 221–234.

[14] J. R. Quinlan, *C4.5: Programs for Machine Learning,* Morgan Kaufmann, San Mateo, CA (1993).

[15] J. R. Quinlan. *Combining instance-based and model-based learning,* Proc. 10th International Conference on Machine Learning, Amherst, MA (1993) 236–243.

[16] S. Salzberg, *A nearest hyperrectangle learning method,* Machine Learning **6** (1991) 251–276.

[17] C. Schaffer, *Analysis of artificial data sets,* Proc. 2nd International Symposium on Artificial Intelligence (1989).

[18] C. Stanfill and D. Waltz, *Toward memory-based reasoning,* Communications of the ACM **29** (1986) 1213–1228.

[19] D. Wettschereck, *A hybrid nearest-neighbor and nearest-hyperrectangle algorithm,* Proc. 9th European Conference on Machine Learning, Catania, Italy (1994) 323–335.

[20] J. Zhang, *A method that combines inductive learning with exemplar-based learning,* Proc. 2nd IEEE International Conference on Tools for Artificial Intelligence, San Jose, CA (1990) 31–37.