A Pathology of Bottom-Up Hill-Climbing in Inductive Rule Learning

Johannes Fürnkranz

Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Vienna, Austria
E-mail: juffi@oefai.at

Abstract. In this paper, we close the gap between the simple and straight-forward implementations of top-down hill-climbing that can be found in the literature, and the rather complex strategies for greedy bottom-up generalization. Our main result is that the simple bottom-up counterpart to the top-down hill-climbing algorithm is unable to learn in domains with dispersed examples. In particular, we show that guided greedy generalization is impossible if the seed example differs in more than one attribute value from its nearest neighbor. We also perform an empirical study of the commonness of this problem is in popular benchmark datasets, and present average-case and worst-case results for the probability of drawing a pathological seed example in binary domains.

1 Introduction

Despite its popularity, it is quite well-known that top-down hill-climbing is a problematic rule learning strategy for cases where the target concept contains a set of conditions which are highly relevant as a conjunction but where each of them appears to be irrelevant in isolation (as, e.g., in the parity problem). For such cases, it is often suggested to use a bottom-up strategy, i.e., to generalize a single example by greedily deleting conditions from the corresponding most specific rule. Despite this fact, no description of this simple and efficient form of bottom-up learning can be found in the literature.

Some time ago, we intended to close this gap and started to implement a bottom-up hill-climbing algorithm, only to find out that a straight-forward prototype implementation failed to learn a simple parity concept with a few irrelevant attributes. Thus the intended improvement over top-down hill-climbing did not materialize. As the idea for such a bottom-up procedure is fairly obvious, we consider it quite likely that other researchers have tried similar approaches and encountered a similar failure.

This paper tries to explain this negative experience by pointing out a pathology in bottom-up hill-climbing that inhibits learning in certain domains. We start with a brief survey of hill-climbing in rule learning, primarily focusing on the top-down algorithm that forms the core of many heuristic rule learning algorithms (Section 2). In Section 3, we discuss some weaknesses of top-down approaches which are commonly used for motivating the use of bottom-up search strategies,

```
procedure SeparateAndConquer(Examples)

$Theory = \emptyset$
while Positive(Examples) $\neq \emptyset$
    $Rule$ = FindBestRule(Examples)
    $Theory = Theory \cup Rule$
    $Examples = Examples \setminus$ Cover(Rule,Examples)
return(Theory)
```

Fig. 1. The top-level loop of separate-and-conquer rule-learning algorithms.

before we present the simple bottom-up counter-part to these top-down algorithms and discuss its relation to other bottom-up approaches in the literature. Section 4 presents the main result of the paper and discusses its commonness based on average-case and worst-case analyses of datasets, as well as its limitations. Finally, we also sketch that our result may point to a way for improving bottom-up algorithms by replacing the random selection of seed examples with a bias that prefers examples in comparably dense regions of the example space.

2 Top-down Hill-Climbing

Rule learning typically follows the so-called *separate-and-conquer* or *covering* strategy, which learns rules successively. The examples covered by previously learned rules are removed from the training set (*separated*) before subsequent rules are learned (before the rest of the training set is *conquered*). The basic algorithm is shown in Fig. 1.

This learning strategy has its roots in the early days of machine learning in the AQ family of rule learning algorithms (Michalski, 1969; Kaufman and Michalski, 2000). It is fairly popular in *propositional rule learning* (cf. CN2 (Clark and Niblett, 1989; Clark and Boswell, 1991) or Ripper (Cohen, 1995)) as well as in *inductive logic programming (ILP)* (cf. Foil (Quinlan, 1990) and its successors (Džeroski and Bratko, 1992; Fürnkranz, 1994; De Raedt and Van Laer, 1995; Quinlan and Cameron-Jones, 1995) or Progol (Muggleton, 1995)). A detailed survey of this large family of algorithms can be found in (Fürnkranz, 1999).

Various approaches that adhere to this framework differ in the way single rules are learned, i.e., in the implementation of the FindBestRule procedure of Fig. 1. The vast majority of algorithms uses a heuristic top-down hill-climbing strategy, i.e., they learn a rule by successively specializing the current best rule. The prevalent specialization operator is to greedily add the most promising condition to the body of the rule. A simple version of this algorithm is shown in Fig. 2.

The algorithm starts with the most general rule (the rule with the body TRUE which covers all training examples), from which it computes all refinements that can be obtained by adding single conditions to the body of the rule.

```
procedure TopDown(Examples)

BestRule = {true}
BestEval = EvaluateRule(BestRule,Examples)
loop
    Refinements = AddCondition(BestRule)
    MaxEval = 0
    foreach Refinement ∈ Refinements
        Evaluation = EvaluateRule(Refinement,Examples)
        if Evaluation > MaxEval
            MaxRule = Refinement
            MaxEval = Evaluation
    if MaxEval ≥ BestEval
        BestRule = MaxRule
        BestEval = MaxEval
    else
        return(BestRule)
```

Fig. 2. A top-down hill-climbing implementation of FindBestRule

These refinements specialize the rule, i.e. the rule will in general cover less examples than it did without the additional condition. If the excluded examples are mainly negative, the proportion of positive examples among the covered examples will increase, which will be reflected by an increase of the heuristic value of the rule (for which various heuristics can be found in the literature; Fürnkranz 1999; Lavrač et al. 1999). If the best refinement has a higher evaluation than its predecessor, it replaces it, and the search continues with the refinements of the new best rule. Otherwise, the previous best rule is a local optimum and the search stops. In the worst case, this will happen when no further refinement of the best rule is possible.

More advanced versions of this algorithm differ from this simple version mostly by adding additional mechanisms for preventing overfitting of the training data, for example by decoupling the stopping of the specialization process from the used quality measure or by generalizing overly specialized rules in a separate *pruning* phase (Fürnkranz, 1997).

3 Bottom-Up Hill-Climbing

Despite the fact that most algorithms search the rule space in a top-down fashion, we have not seen any convincing argument against bottom-up induction, i.e., against algorithms that successively generalize from a most specific rule (typically a single training example) instead of successively specializing the most general rule. Quite in contrary, there is evidence of situations where bottom-up induction may be the preferable approach (cf. Section 3.1). Nevertheless, no counterpart to the algorithm shown in Fig. 2 can be found in the literature. In the remainder of this section, we will discuss such an algorithm in detail (Section 3.2), and also briefly survey related work in bottom-up hill-climbing in rule learning (Section 3.3).

3.1 Motivation: Pathologies of Top-Down Hill-Climbing

Top-down hill-climbing algorithms like the one discussed in the previous sections are quite popular and reasonably successful. However, a fundamental shortcoming of this approach is that it only evaluates the value of single conditions in isolation (or in the context of all previously added conditions of a rule). The problem with this is that there are cases where a conjunction of conditions may be strongly relevant, while each of its member terms in isoluation may appear to be irrelevant.

The best-known examples for this problem are XOR or parity problems. A *parity problem* consists of a few relevant and several irrelevant binary attributes. The class label of each example is 1 if there is an even number of ones among the values of the relevant attributes, and 0 otherwise. For a top-down learner, the individual attributes are indiscriminable because each condition that tests the value of one of the binary attributes will cover approximately half positive and half negative examples. Hence, there is no information that can be used for guiding the search for an appropriate refinement.

Similarly, many top-down ILP algorithms such as Foil (Quinlan, 1990) have the problem that they cannot learn relations which introduce new variables, because such relations often do not contain any information that helps to discriminate between positive and negative examples. A typical example would be the literal `parent(X,Z)`, which is necessary for learning the concept `grandparent(X,Y)`. As every person `X` has two parents `Z` (irrespective of whether `Y` is her grandparent or not), this literal does not help to discriminate between positive and negative examples. However, its addition is necessary for learning the target concept. Many systems are able to handle a subset of these problems by adding so-called *determinate literals* (literals that have at most one valid binding and therefore do not blow up the number of possible derivations for the clause; Quinlan 1991) to the clause even if they do not have any value, but the principal problem remains.

Finally, Chevaleyre and Zucker (2001) show a convincing argument that top-down hill-climbing may have severe problems in multi-instance learning problems, i.e., in problems where examples are not described with single feature vectors but with *bags* of feature vectors. The problem occurs when each possible value of an attribute is assumed by at least one instance of each bag. In this case, every condition that tests on values of this attribute covers all examples, positive or negative. Thus, a target concept that consists of a conjunction of values of two (or more) of such attributes cannot be learned because each individual attribute will appear to be completely irrelevant. A detailed example can be found ib (Chevaleyre and Zucker, 2001). As multi-instance learning is a special case of ILP (De Raedt, 1998), this problem is also relevant to ILP in

```
procedure BottomUp(Examples)

BestRule = SelectRandomExample(Positive(Examples))
BestEval = EvaluateRule(BestRule,Examples)
loop
    Refinements = DeleteCondition(BestRule)
    MaxEval = 0
    foreach Refinement ∈ Refinements
        Evaluation = EvaluateRule(Refinement,Examples)
        if Evaluation > MaxEval
            MaxRule = Refinement
            MaxEval = Evaluation
    if MaxEval > BestEval
        BestRule = MaxRule
        BestEval = MaxEval
    else
        return(BestRule)
```

Fig. 3. A bottom-up hill-climbing implementation of FindBestRule

general. It may occur whenever multiple variable bindings are possible for an unbound existentially quantified variable in the body of a rule.[1]

In all of the above cases, the conjunction of several conditions is relevant, while some or all of the conditions in isolation may appear to be irrelevant, and will therefore not be selected by a top-down hill-climbing strategy. On the other hand, a bottom-up search strategy could naturally deal with such cases because it starts from most specific rules (basically single examples) and successively generalizes these. As all conjunctions of the target rule must also be present in the examples covered by the rule, bottom-up strategies are therefore not prone to this type of problem. Consequently, it is frequently suggested to use a bottom-up search strategy to cope with such pathologies of top-down search.

3.2 The Basic Algorithm

The bottom-up algorithm of Fig. 3 differs only in three points from the top-down algorithm of Fig. 2. First, it uses a different procedure for computing the refinements of a rule: while TopDown successively *specializes* its best rule by adding conditions, BottomUp successively *generalizes* its best rule by deleting conditions. Naturally, it also has to use a different initialization from which it can start its search, namely it initializes its best rule to a random positive example. The third difference is that TopDown stops its search when the best refinement of a rule does not improve upon its predecessor, while BottomUp generalizes as

[1] The problem may also be viewed as a generalization of the problem discussed in the previous paragraph.

long as the best refinement is *no worse* than its predecessor. Thus, both methods implement a preference bias for shorter rules.

The above algorithm may be generalized by extending DELETECONDITION to consider not only the deletion of single conditions, but also all possible pairs, triples, etc. We refer to this generalization as **n-step look-ahead hill-climbing**, where n denotes the maximum number of conditions that are considered for deletion in one refinement step.

3.3 Alternative Approaches to Bottom-Up Rule Learning

The simple and straight-forward hill-climbing algorithm of Fig. 3 has a fundamental problem, which we will discuss in section 4. Nevertheless, a variety of successful bottom-up approaches can be found in the rule learning literature. These will be briefly discussed in the remainder of this section.

AQ-based approaches: AQ (Michalski, 1969) and its successors (e.g., Michalski, 1980; Michalski et al., 1986; Bergadano et al., 1992; Kaufman and Michalski, 2000) learn bottom-up in the sense that they search the space of generalizations of a single randomly selected seed example. However, this space of generalizations is searched top-down, i.e., the seed example is only used for constraining the search space, but not as the starting point of the search. In ILP, an equivalent strategy is used in Progol, where the seed example is minimally generalized with respect to the available background knowledge and a given maximum inference depth, and the resulting *bottom clause* is then again used for constraining the search space for a subsequent top-down best-first search for the best generalization (Muggleton, 1995).

***lgg*-based Approaches:** Most algorithms that search in a bottom-up fashion do not generalize from single examples, but generalize from pairs of examples. The prototype of this type of algorithm is the ILP system Golem (Muggleton and Feng, 1990), which constructs minimal generalizations of pairs of examples, so-called (relative) *least general generalizations (lggs)*. Similar ideas have subsequently also been used in propositional rule learning, for example in DLG (Webb, 1992; Webb and Agar, 1992) and RISE (Domingos, 1996).

The main problem with this approach is its quadratic computational complexity resulting from the fact that in each generalization step, the current rule has to be generalized with each training example and the resulting rules have to be tested on all training examples. Golem addresses this problem by restricting its attention to a fixed number of generalizations constructed from a randomly drawn subsample of all example pairs. DLG uses a greedy version of the algorithm that incrementally generalizes the first example with all examples in succession, replacing the current rule with the generalization when the heuristic evaluation improves. Thus it finds a final rule in a single pass through the examples (with quadratic effort) but the obtained generalization depends on the (randomized) order of the training examples.

Table 1. A pathological example where bottom-up hill-climbing cannot find the target concept $A_1 = 1 \rightarrow +$.

A_1	A_2	A_3	Class
1	1	1	$+$
1	0	0	$+$
0	1	0	$-$
0	0	1	$-$

Pruning Algorithms: The only case we know of where bottom-up hill-climbing of the type shown in Fig. 3 is used in rule learning is for pruning, i.e., for simplifying previously learned rules by successively generalizing them. For example, the incremental reduced error pruning strategy (Fürnkranz and Widmer, 1994), which forms the basis of Ripper (Cohen, 1995), generalizes a rule by greedily deleting single conditions immediately after it has been learned. The problem with bottom-up hill-climbing, which we will discuss in the following section, does not apply to this scenario for reasons that will be discussed further below.

4 A Pathology of Bottom-Up Hill-Climbing

Consider the sample database shown in Table 1. The target concept $A_1 = 1 \rightarrow +$ is quite obvious and can be found without problems by top-down hill-climbing algorithms. However, one can easily verify that it can only be found by chance by a bottom-up hill-climbing algorithm of the type shown in Fig. 3 (in the remainder of this section, we will only be concerned with this type). Assume, e.g., that the first example is picked as the seed example. The corresponding rule would be $A_1 = 1 \wedge A_2 = 1 \wedge A_3 = 1 \rightarrow +$.

If the learner now tries to generalize this rule by deleting the first condition, the resulting rule $A_2 = 1 \wedge A_3 = 1 \rightarrow +$ does not cover any additional examples. This is good because we obviously do not want to drop the condition $A_1 = 1$. However, the situation is analogous for the other two conditions: no matter what condition is deleted from the rule, the resulting generalization does not cover any additional examples, positive or negative. Also note that this situation does not depend on the choice of the seed example: the same problem occurs for picking any of the other three examples as a seed example.

The basic question at this point is, whether this is simply a carefully constructed example with only minor relevance for practical applications, or whether this a prototypical case that illustrates a common problem. In the following, we will argue for the latter.

4.1 The Problem

The following simple observation shows that the above-mentioned pathology occurs whenever the seed example differs from every other example in at least two attributes.

Theorem 1. *Bottom-up hill-climbing with n-step look-ahead is unable to discriminate between generalizations of an example if the example differs from its nearest neighbour in at least $n + 1$ attribute values.*

Proof. Let E be the training examples, each of them described with m symbolic attributes $A_1, ..., A_m$, $s = (s_1, \ldots, s_m) \in E$ a randomly chosen seed example, and H be the Hamming distance between two examples, i.e,

$$H(e, f) = \sum_{i=1\ldots m} \delta(e_i, f_i) \text{ for } e, f \in E \text{ where } \delta(a, b) = \begin{cases} 0 & \text{if } a = b \\ 1 & \text{if } a \neq b \end{cases}$$

Let us first consider a one-step look-ahead, i.e., $n = 1$: Then all candidate generalizations of the conjunction $A_1 = s_1 \wedge \ldots \wedge A_m = s_m$ are obtained by deleting single conditions $A_i = s_i, 1 \leq i \leq m$. Obviously, all of them constrain the examples they cover to be identical to s except for the value of attribute A_i, which could take an arbitrary value. Consequently, no other example $e \neq s$ can be covered by a candidate rule because all covered examples e may differ from s in at most $n = 1$ attribute values (i.e., $H(s, e) \leq 1$), while we assumed that each example $e \neq s$ differs from s in at least $n + 1 = 2$ attributes values, i.e., $\min_{e \in E, e \neq s} H(s, e) \geq 2$.

The generalization of this argument to general $n > 1$ is quite straight-forward. In this case, the candidate generalizations of the conjunction are obtained by deleting (up to) n conditions from the rule. Thus, no generalization of the seed example can cover additional examples iff the seed differs from all examples in at least $n + 1$ attribute values. □

In particular, this theorem states that simple bottom-up hill-climbing will not work well whenever the chances are high that a seed example is picked that has a distance of ≥ 2 to its nearest neighbor. In the following section, we will discuss how common this situation is.

4.2 Commonness of the Problem

In order to estimate how common this problem is in real-world applications we analyzed 13 benchmark datasets with respect to the distribution of distances between nearest neighbor pairs. The datasets are all datasets with less than 10,000 examples and only symbolic attributes among the datasets that are used within the METAL project.[2] Most of them originate from the UCI repository (Blake and Merz, 1998). For each dataset, we computed the distance of each example to its nearest neighbor. Neighbors with distance 0 (i.e., duplicate examples) were ignored for this computation because they are already covered by the most specific rule (the rule that conditions on all values of the example) and therefore also by all its generalizations. The last column of Table 2 shows the proportion of such examples.[3]

[2] We ignored trivial variants of datasets (e.g., *vote*/*vote1*) as well as artificial domains for which the dataset consists of a complete enumeration of the sample space (e.g., *monk*, *parity*). More on the latter can be found further below.

[3] This is out of the scope of this paper, but we were surprised to find that some domains contain quite a significant portion of duplicate examples. For some com-

Table 2. Partial histograms of minimum example distances in 13 symbolic domains with less than 10,000 examples (training and test sets are pooled together). The fourth column (in bold font) shows the proportion of examples that have a nearest neighbor with distance 1. The two following columns show the proportions of distances 2 and > 2. The penultimate column shows the maximum distance between an example and its nearest neighbor in this dataset. The final column shows the proportion of examples that have a duplicate in the dataset (examples with distance $= 0$ are ignored for computing the nearest neighbor).

			$\min_e H(s,e)$			$\max_s$	
dataset	#exs	#atts	$= 1$	$= 2$	> 2	$\min_e H$	duplicates
titanic	2201	3	**1.000**	0.000	0.000	1	1.000
car	1728	6	**1.000**	0.000	0.000	1	0.000
led7	3200	7	**1.000**	0.000	0.000	1	0.996
mushrooms	8124	22	**1.000**	0.000	0.000	1	0.000
solar flares(x,c,m)	1389	10	**0.986**	0.014	0.000	2	0.866
kr vs. kp	3196	36	**0.958**	0.036	0.006	5	0.000
vote	300	16	**0.553**	0.217	0.230	6	0.223
soybean	683	35	**0.483**	0.253	0.264	8	0.139
acetylation	1511	50	**0.142**	0.068	0.790	42	0.097
splice	3190	60	**0.020**	0.020	0.961	47	0.097
led24	3200	24	**0.018**	0.146	0.836	6	0.001
dna-splice	3186	180	**0.008**	0.015	0.977	53	0.097
tic-tac-toe	958	9	**0.000**	1.000	0.000	2	0.000

The resulting probability estimates are shown (in bold face) in the fourth column of Table 2: these probabilities denote the chance of picking a seed example for which bottom-up hill-climbing (with 1-step look-ahead) can perform no meaningful generalization because no candidate generalization will cover an additional example. In some domains (e.g., *mushrooms* or *titanic*, which has only 3 attributes, but 2201 examples) this does not seem to be a problem, while it seems to become increasingly more problematic with a growing number of attributes. A particularly interesting case is the *tic-tac-toe* domain (bottom line of Table 2), whose special properties prevent the use of bottom-up hill-climbing entirely. This database encodes all won positions of a tic-tac-toe game, with the same player on the move. It is easy to see that, if both players make a move, the board position will change on two squares, and that therefore pairs of positions with the same player to move will always differ in at least two attributes.

Also of interest is the discrepancy between the *led7* and the *led24* domains, which differ only in the fact that *led24* contains an additional 17 irrelevant attributes. This completely reverses the picture: While *led7* is conveniently over-

monly used benchmark sets (e.g., *vote* or *soybean*) this begs the question whether in these domains hold-out (or cross-validation) error estimates are optimistically biased because of their lack of independence of the training and test sets.

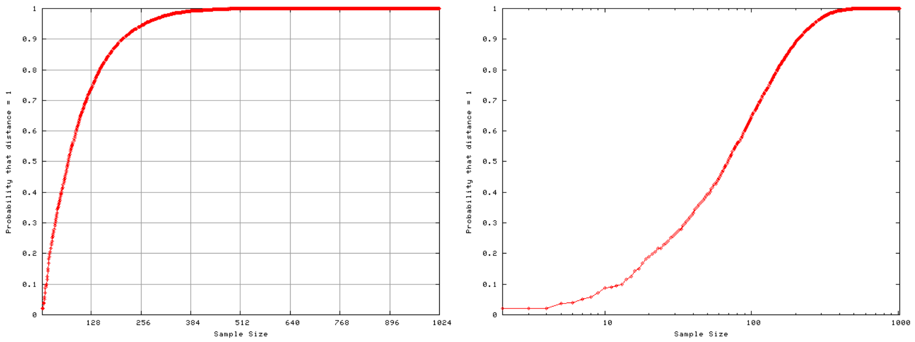

Fig. 4. Probability for drawing an example that has a neighbor at distance 1 over different training set sizes for a domain with 10 binary attributes (estimated from 100 different sequences of nested subsets; regular plot and plot with logarithmic scale for sample sizes).

sampled (there are only $2^7 = 128$ unique examples in this domain[4]), it is practically impossible to learn *led24* with bottom-up hill-climbing.

This is interesting because with the same argument it can be shown that it is practically impossible to learn a parity problem of similar proportions, i.e., it defeats one of the main motivations for considering bottom-up hill-climbing. Fig. 4 illustrates this situation for a domain with 10 Boolean attributes, such as *led10*. The graphs show the probability distribution for picking an example that has a nearest neighbor at distance 1 over the size of subset (shown on a logarithmic scale in the graph to the right). The function is estimated by counting the proportion of such examples in 100 sequences of nested subsets from 2 examples up to all 1024 examples of the domain. If approximately half of the examples are present, one can be quite certain that bottom-up hill-climbing will work. Thereafter, the probability starts to decrease slowly for lower example sizes, and the decline picks up pace at about 250 examples.

Fig. 5 shows a similar plot for subsets of size up to 4800 for a binary domain with 24 attributes (like *led24*). While 4800 appears to be a reasonable training set size for a domain with 24 attributes (there are 200 examples per attribute), this is still a very low number in comparison to the $2^{24} = 16,777,216$ examples of the full domain. The curve shown in Fig. 5 is still in the quasi-linear ascent at the beginning (compare with Fig. 4), and the probability estimates are still below 0.01, i.e. there is not even a 1% chance that bottom-up hill-climbing will pick a suitable example. This explains the difference between the *led7* and *led24* results in Table 2, and shows that bottom-up hill-climbing is very sensitive to the addition of irrelevant attributes.

[4] It should be noted that there is a designated train/test-split for this domain where only a fraction of the 3200 examples are actually used for training.

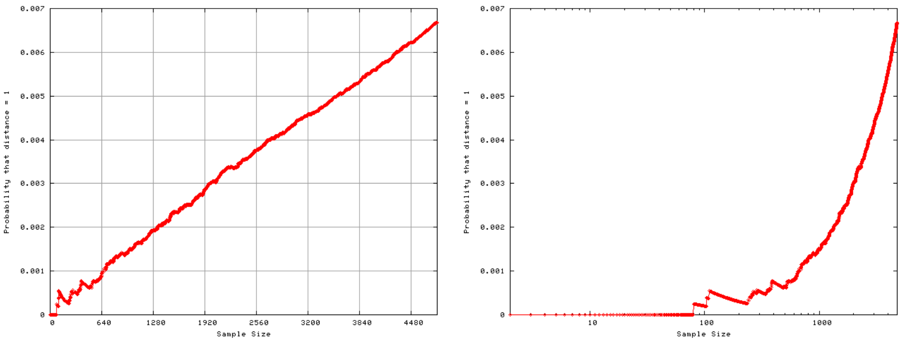

Fig. 5. Probability estimates for drawing an example that has a neighbor at distance 1 over different training set sizes for a domain with 24 binary attributes (estimated from 100 different sequences of nested subsets with size ≤ 4800; regular plot and plot with logarithmic scale for sample sizes).

While above analysis was concerned with the average case, it is fairly straightforward to derive a tight worst-case bound for the largest dataset that can be constructed so that each pair of examples has a distance ≥ 2:

Theorem 2. *The largest dataset for a binary domain with n attributes for which all pairs of examples differ in at least two attributes contains 2^{n-1} examples.*

Proof. There are 2^{n-1} examples with $n-1$ attributes. We extend each example with an n-th attribute as a parity value (i.e., we add 0 for an even number of 1's in the first $n-1$ attributes and 1 for an odd number). Clearly, examples that had a distance of 1 with $n-1$ attributes, now have a distance of 2 because they will in addition differ in the parity attribute. Thus, we have training set with 2^{n-1} examples and pairwise examples distances ≥ 2.

There can also be no larger set because each additional example must share the first $n-1$ attributes with one of the examples in the original dataset. Thus it can only differ in the last attribute, and the distance to its nearest neighbor is ≤ 1. □.

4.3 Discussion

A few comments are at hand to put the above results into the right perspective. First, it should be noted that our results only deal with symbolic and possibly with integer domains. Attributes with continuous value ranges typically have to be generalized to an interval, which may increase chances of covering additional examples. However, this may also result in an undesirable bias that favors generalization of continuous attributes over generalization of symbolic attributes, in case the examples are fairly scattered in the projection to the symbolic subspace.

We also tacitly assumed that bottom-up hill-climbing *can* make a useful generalization if there is at least one other examples that differs in only one value. Obviously, this needs not be the case, because the distribution of positive and negative class labels can be more or less arbitrary at such small samples.

Furthermore, the above-mentioned limitations only apply to bottom-up hill-climbing that starts from the most specific rule, i.e., the equivalent of a single example. Bottom-up hill-climbing of the type shown in Fig. 3 *is* successfully used for pruning, i.e., for simplifying previously learned rules (cf. Section 3.3). It can work there because generalization starts at a point where the last specialization step (i.e., the last addition of a condition to the rule) has decreased the coverage of the rule on the training set. Thus, even if no generalization of the learned rule increases its coverage on the validation set (where the pruned rule is evaluated), the information about the order in which the conditions have been added in the specialization phase may be used for making a reasonable tie-breaking choice. Typically, the conditions added first will be more relevant and more reliable than conditions that have been added towards the end.

It is also unclear whether these results can be carried over to other uses of hill-climbing in machine learning. An obvious candidate would be backwards wrapper feature subset selection, which starts with a full set of features and successively considers individual features for elimination (Kohavi and John, 1997). However, the situation here is somewhat different. Although it can be expected that many features look alike (at least if highly selective inducers like decision trees or rule learners are used), there will be some differences that can be used for guiding the search (e.g., one can expect that the use of a different root attribute in a decision tree, which is guaranteed to happen in this process, will lead to different performance estimates).

There are several extensions to basic hill-climbing that one could expect to help with our type of problem. We already discussed the case of *deeper look-aheads* (i.e., to generalize several steps at the same time) and showed that increasing the look-ahead by one step increases the allowable minimum example distance by one. Thus, the examples may be less dense in the domain. However, Table 2 shows that a 2-step look-ahead could in most cases (a notable exception is *tic-tac-toe*) only marginally improve the situation (the probabilities of failure are the sum of the columns '$= 1$' and '$= 2$ (or one minus the values of the column '> 2'). These small improvements have to be paid with a quadratic increase of candidate generalizations that must be considered. Moreover, the penultimate column of Table 2 shows that there are several domains that contain examples whose nearest neighbor differs in five or more attributes, in the DNA-related domains even more than 40 attributes. The exponential growth in the search space makes such deep look-aheads prohibitive.

Another technique for fighting the myopic behavior of hill-climbing is the use of *beam search*. At each generalization step, beam search not only remembers the option that appears to be best (as in hill-climbing), but instead remembers the best b options (where b is the size of the beam). Again, it cannot solve the principal problem that all candidate generalizations will look alike, but by remembering more than one candidate, it increases the chance that a correct generalization is among those remembered. Of course, if the beam is made large enough to remember all possible candidates, we end up with exhaustive search.

It should also be noted that our analysis is only concerned with the counterpart of the popular top-down algorithm. More elaborate generalization operators, such as the use of *lgg*s can be employed successfully, as many implementations demonstrate. However, these operators are also quite costly. *lgg*s, for example, compute a generalization for each pair of examples instead of each single example, and, for this reason, are often combined with subsampling (cf. Section 3.3).

If one is willing to deal with such quadratic worst-case costs, our results may also lead to a constructive way for selecting good seed examples for bottom-up hill-climbing, namely examples that have a number of neighbors that are fairly close. A similar technique is used in RISE (Domingos, 1996) for picking the best pair of examples for an *lgg*-based generalization step. The advantage would be that pairwise example differences are only needed for picking the seed example (instead of at each generalization step, as would be the case for *lgg*s). Also, the algorithm may not need to compute all pairwise differences because it may stop as soon as it has discovered a candidate seed example that lies in a reasonably dense neighborhood. Whether such an approach could lead to a competitive bottom-up hill-climber remains subject to future work.

5 Conclusions

In this paper, we closed a gap in the literature on hill-climbing approaches in rule learning. While the simple top-down hill-climbing approach prevails in rule learning, and a few more complex bottom-up algorithms can be found as well, we do not know of any paper that discusses the simple bottom-up dual of the popular top-down approach. Our negative result shows that such an algorithm will fail to generalize in cases where the selected seed example is fairly distant from all its neighbors. For propositional domains, we show that the likeliness of such a pathological case increases with the number of attributes, and that several thousand examples are not sufficient to facilitate greedy bottom-up generalization for some twenty Boolean attributes.

Our results do not touch upon bottom-up approaches previously studied in the literature, but these have to buy the additional power of their generalization operator with a considerably larger number of investigated hypotheses (most approaches rely on generalization from pairs of examples instead of single examples). However, if one does not eschew quadratic run-time complexities, our results may also point to a way of improving future bottom-up systems by replacing the random selection of seed examples with a guided search for examples in denser neighborhoods of the example space.

Acknowledgements

I would like to thank the participants (and the organizers!) of the Leuven-Freiburg workshop 2002 for valuable comments and discussions.

The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Education, Science and Culture. This work was supported by an APART stipend of the *Austrian Academy of Sciences* and by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant no. P12645-INF.

References

F. Bergadano, S. Matwin, R. S. Michalski, and J. Zhang. Learning two-tiered descriptions of flexible concepts: The POSEIDON system. *Machine Learning*, 8:5–43, 1992.

C. L. Blake and C. J. Merz. UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 1998. Department of Information and Computer Science, University of California at Irvine, Irvine CA.

Y. Chevaleyre and J.-D. Zucker. A framework for learning rules from multiple instance data. In L. D. Raedt and P. Flach, editors, *Proceedings of the 12th European Conference on Machine Learning (ECML-01)*, pages 49–60, Freiburg, Germany, 2001. Springer-Verlag.

P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Proceedings of the 5th European Working Session on Learning (EWSL-91)*, pages 151–163, Porto, Portugal, 1991. Springer-Verlag.

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

W. W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Proceedings of the 12th International Conference on Machine Learning (ML-95)*, pages 115–123, Lake Tahoe, CA, 1995. Morgan Kaufmann.

L. De Raedt. Attribute value learning versus inductive logic programming: The missing links (extended abstract). In D. Page, editor, *Proceedings of the 8th International Conference on Inductive Logic Programming (ILP-98)*, pages 1–8. Springer-Verlag, 1998.

L. De Raedt and W. Van Laer. Inductive constraint logic. In *Proceedings of the 5th Workshop on Algorithmic Learning Theory (ALT-95)*. Springer-Verlag, 1995.

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

S. Džeroski and I. Bratko. Handling noise in Inductive Logic Programming. In S. H. Muggleton and K. Furukawa, editors, *Proceedings of the 2nd International Workshop on Inductive Logic Programming (ILP-92)*, number TM-1182 in ICOT Technical Memorandum, pages 109–125, Tokyo, Japan, 1992. Institue for New Generation Computer Technology.

J. Fürnkranz. FOSSIL: A robust relational learner. In F. Bergadano and L. De Raedt, editors, *Proceedings of the 7th European Conference on Machine Learning (ECML-94)*, volume 784 of *Lecture Notes in Artificial Intelligence*, pages 122–137, Catania, Italy, 1994. Springer-Verlag.

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

J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, February 1999.

J. Fürnkranz and G. Widmer. Incremental Reduced Error Pruning. In W. Cohen and H. Hirsh, editors, *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 70–77, New Brunswick, NJ, 1994. Morgan Kaufmann.

K. A. Kaufman and R. S. Michalski. An adjustable rule learner for pattern discovery using the AQ methodology. *Journal of Intelligent Information Systems*, 14:199–216, 2000.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1–2):273–324, 1997. Special Issue on *Relevance*.

N. Lavrač, P. Flach, and B. Zupan. Rule evaluation measures: A unifying view. In S. Džeroski and P. Flach, editors, *Ninth International Workshop on Inductive Logic Programming (ILP'99)*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 174–185. Springer-Verlag, June 1999.

R. S. Michalski. On the quasi-minimal solution of the covering problem. In *Proceedings of the 5th International Symposium on Information Processing (FCIP-69)*, volume A3 (Switching Circuits), pages 125–128, Bled, Yugoslavia, 1969.

R. S. Michalski. Pattern recognition and rule-guided inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:349–361, 1980.

R. S. Michalski, I. Mozetič, J. Hong, and N. Lavrač. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI-86)*, pages 1041–1045, Philadelphia, PA, 1986.

S. H. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13(3,4):245–286, 1995. Special Issue on Inductive Logic Programming.

S. H. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the 1st Conference on Algorithmic Learning Theory*, pages 1–14, Tokyo, Japan, 1990.

J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5: 239–266, 1990.

J. R. Quinlan. Determinate literals in inductive logic programming. In *Proceedings of the 8th International Workshop on Machine Learning (ML-91)*, pages 442–446, 1991.

J. R. Quinlan and R. M. Cameron-Jones. Induction of logic programs: FOIL and related systems. *New Generation Computing*, 13(3,4):287–312, 1995. Special Issue on Inductive Logic Programming.

G. I. Webb. Learning disjunctive class descriptions by least generalisation. Technical Report TR C92/9, Deakin University, School of Computing & Mathematics, Geelong, Australia, September 1992.

G. I. Webb and J. W. M. Agar. Inducing diagnostic rules for glomerular disease with the DLG machine learning algorithm. *Artificial Intelligence in Medicine*, 4:419–430, 1992.