How Useful Is Relevance?

Rich Caruana
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
caruana@cs.cmu.edu

Dayne Freitag
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
dayne@cs.cmu.edu

Abstract

Eliminating irrelevant attributes prior to induction boosts the performance of many learning algorithms. Relevance, however, is no guarantee of usefulness to a particular learner. We test two methods of finding relevant attributes, FOCUS and RELIEF, to see how the attributes they select perform with ID3/C4.5 on two learning problems from a calendar scheduling domain. A more direct attribute selection procedure, hillclimbing in attribute space, finds superior attribute sets.

1 INTRODUCTION

An attribute that is irrelevant is not useful for induction. But not all attributes that *are* relevant are necessarily useful for induction either. Several methods for estimating attribute relevance have been devised. Are the attributes selected by these methods good attributes to use for learning?

Let $\mathcal{A}$ be the set of attributes potentially available to a learner L trying to learn problem P given typical training sets. Imagine that the learner is given only the attributes in $A \subset \mathcal{A}$. We call some attribute, a, $a \notin A$ but $a \in \mathcal{A}$, *USEFUL* to P for L using A on if it improves L's performance on P on average over typical training sets with attributes A. For example, an attribute would be useful to P for ID3/C4.5 if, along with the other attributes already in the training set, it improves the accuracy of the ID3/C4.5 induced tree on P. We call a set of attributes A *MAXIMALLY USEFUL* if there does not exist some other set $A' \subset \mathcal{A}$ for which the performance of L on P with typical training sets is better.

We call an attribute *RELEVANT* to P if there exists any learner L' using any set of attributes A' and any distribution of typical training sets for which it is useful. Note that usefulness is measured with respect to a particular learner using a particular set of attributes given a particular distribution of training patterns, whereas relevance just depends on the existence of such a learner, attribute set, and pattern distribution. All useful attributes are relevant, but not all relevant attributes are necessarily useful; useful attributes are often a small subset of relevant attributes.

Proving that an attribute is *irrelevant* is difficult: it requires showing that no P, A, and distribution of training sets exists for which the attribute is useful. To show that an attribute is *relevant*, however, one need only demonstrate that some learner benefits from it. Demonstrating that an attribute is *useful* to P for L using A with typical training sets is usually not too hard: measuring the performance of the learner with and without the attribute suffices. Unfortunately, what we really want is the maximally useful *set* of attributes, not properties of single attributes. But, because attributes often interact, finding this set directly is usually prohibitive. Can relevance be used as an efficient proxy for usefulness?

We have run experiments on two calendar scheduling problems using FOCUS and RELIEF to select attributes for ID3/C4.5. Depending on the ID3/C4.5 options used, the performance using the attributes selected by FOCUS and RELIEF ranges from poor to good. But, for each set of ID3/C4.5 options, hillclimbing is able to find attribute sets that perform better. If the attribute set to use depends on the learning procedure, it is unlikely that algorithm independent measures such as attribute relevance often will yield optimal performance.

2 THE CALENDAR APPRENTICE

The Calendar Apprentice (CAP) (4)(9)(13) is an apprentice system that learns to schedule meetings by watching users' daily scheduling. CAP uses features such as the meeting attendees, the type of meeting, and the current state of the user's calendar to predict the meeting's `location`, `day-of-week`, `start-time`, and `duration`. CAP uses ID3/C4.5 (15) run on an accumulating database of previously scheduled meetings to predict these values for new meetings.

Scheduling is a rich and dynamic domain: new attendees appear frequently, scheduling during the summer is different from scheduling during the fall or at holidays, user habits change frequently, and there is a rich hierarchy of meeting types (e.g., advisor-advisee meeting or faculty meeting), a variety of temporal regularities (e.g., twice-a-week to once-in-a-lifetime occurrences), and a broad range of attendee statuses (e.g., graduate student or university president). Because of this, there are many attributes potentially available from which to learn. Of these, CAP currently considers 32 attributes for predicting `day-of-week` and `start-time`. If, however, all 32 of these attributes are given to ID3/C4.5 for learning, generalization is often poor on these tasks.

3 SELECTING ATTRIBUTES

We test three approaches to attribute selection. Two of these use relevance to estimate attribute usefulness. The other uses the learning algorithm itself to estimate the usefulness of attribute sets. Our training data is recent scheduling events. Attribute selection finds the attribute set that would have performed best historically, which CAP then uses for future learning. In domains lacking this temporal character, a single set of data is used for both feature selection and the final induction. Cross-validation is one good way to do this(14)(8)(2).

FOCUS

FOCUS (1) searches for the minimum-size attribute subset that maintains consistency on the training data. There are two difficulties applying FOCUS in CAP. First, the CAP database is not consistent even when all attributes are present. FOCUS will fail to find a consistent attribute set if run on a large enough sample. We generalize FOCUS as follows: if the original sample contains K inconsistencies with all attributes, our FOCUS returns the smallest subset of attributes that yields exactly K inconsistencies.

The second difficulty is that FOCUS exhaustively searches all possible subsets of a given size before going on to the next size. We've enhanced FOCUS to improve its average case performance. We run two FOCUS-like algorithms at the same time. One searches the smallest subsets first, exactly as in (1). The other uses depth-first search starting with the largest subset. If a small subset yields consistency, the first FOCUS will find it efficiently, but if a large subset is necessary, the second FOCUS will find it efficiently. (The worst case occurs when both methods find the same subset at the same time, but this only doubles the total cost.) Using the two also has the advantage that at any point the first FOCUS provides a lower bound on the number of attributes required while the second FOCUS provides an upper bound.

RELIEF

RELIEF (6) is an attribute relevance estimation algorithm inspired by instance-based learning techniques. RELIEF associates with each attribute a weight indicating the relative relevance of that attribute to making class distinctions. It is attractive for three reasons. First, it is computationally efficient. Second, it can detect feature relevance when features interact. Third, the relevance weights give a degree of relevance allowing attributes to be ranked by relevance.

The main difficulty in applying RELIEF to CAP is that RELIEF is designed for boolean classification tasks; the CAP problems have more than two classes. It is not obvious how best to modify RELIEF to handle non-boolean classification. Kononenko (7) tests two extensions to RELIEF, RELIEF-E and RELIEF-F, to solve this problem and reports that RELIEF-F outperforms RELIEF-E on test problems. We tried both. RELIEF-F made it difficult to determine where to draw the line between what attributes to include and exclude. We tried several thresholds and report here on the best found for RELIEF-F.

ATTRIBUTE HILLCLIMBING

In contrast to FOCUS and RELIEF, hillclimbing uses the learner to search for useful attributes. John et al. (8) refer to this as the *wrapper* model for feature selection. The attribute selection procedure is *wrapped* around the induction algorithm. The training data is broken into an induction set that is given to the learner (here ID3/C4.5) and a hillclimbing test set that is used to measure the generalization performance of what is learned. Events before an arbitrarily chosen date become the induction set, those after become the hillclimbing test set. Hillclimbing uses the performance on the test set to evaluate neighboring points.

Hillclimbing Methods We test five hillclimbing methods. Four of these are similar to variable selection methods used commonly in statistics (5). They proceed by either adding or deleting an attribute to the current set at each step in the search. They differ from each other only in the starting point and in what kinds of moves (addition and/or deletion) are permitted. All four greedily make the move which yields the best performance. Forward selection (FS), starts with the empty set and adds attributes until all have been added. Backward elimination (BE), begins with all attributes and removes them one by one until all have been removed. The other two perform bidirectional search and can either add or remove an attribute at any point. They differ only in their starting point. Forward *stepwise* selection (FSS) begins with the empty set, while backward *stepwise* elimination starts with the full set. ("Selection" and "elimination" are misnomers in FSS and BSE; both can add or delete attributes at any step of their search other than the first.)

We test one more form of hillclimbing, BSE-SLASH, which is based on the observation that when there are many attributes, some learning procedures frequently do not use most of them. BSE–SLASH starts with the full attribute set (like BSE), but, after taking a step, eliminates (*slashes*) any attributes not used in what was learned. BSE–SLASH does this at every step during search. This allows BSE–SLASH to jump to regions of the search space where all the attributes in the active subset are playing a role in what is learned.

The three bidirectional hillclimbing methods, FSS, BSE, and BSE–SLASH, step to the best neighboring point even if this is poorer than the current point. This allows them to escape local maxima but makes looping a problem. We use a simple cycle detection scheme that forces the search to consider only unvisited points in the space. This means the bidirectional hillclimbers will not usually terminate on large spaces. We stop them after some arbitrary amount of computation, at which point they report the best performing attribute subset found so far.

Caching to Speed Search When the training sample is small, decision trees grown from large sets of attributes often do not use all the attributes. Where this is the case, if decision trees repeatedly will be grown from similar sets of attributes, it is possible to cache results to reduce the cost of growing subsequent trees. For example, if ID3 is given attributes {A,B,C,D,E} but does not use attributes D and E in the decision tree it generates, then it is not necessary later to grow decision trees given attributes {A,B,C}, {A,B,C,D}, or {A,B,C,E}—ID3, a deterministic algorithm, will ignore attributes D and E if either is given with {A,B,C} and will generate identical trees. If we cache decision trees, recording both the attributes given to ID3 and the attributes used in the learned decision tree, rapid lookup can be used before new decision trees are grown to see if the outcome is determinable from prior experience. See (3) for more details of this caching scheme.

Tables 1 and 2 show the kind of speedup possible. With caching, and using C4.5, the bidirectional hillclimbers take roughly a day to consider 100,000–1,000,000 trees (but only growing about 5,000–20,000 trees).[1] Cache lookup time is negligible even after caching thousands of decision trees despite the fact that the cache code is Lisp and the tree induction code is C. Without caching it would take months to evaluate a million trees. Note that while most progress occurs early in the bidirectional searches, some improvements do occur late in the searches so it is desirable to extend the searches as far as possible.

Search Method	Trees Considered	Trees Grown	Caching Speedup
FS	528	513	1.03
BE	528	133	3.97
FSS	100,000	3686	27.13
BSE	100,000	2,545	39.29
BSE	600,000	4,541	132.13
BSE–SLASH	100,000	12,435	8.04

Table 1: Speedup Due to Caching for Day of Week

Search Method	Trees Considered	Trees Grown	Caching Speedup
FS	528	528	1.00
BE	528	162	3.26
FSS	100,000	14,536	6.88
BSE	100,000	5,147	19.43
BSE	600,000	20,122	29.82
BSE–SLASH	100,000	15,418	6.49

Table 2: Speedup Due to Caching for Start Time

4 EMPIRICAL ANALYSIS

Figure 1 presents the performance as a function of computational cost of each of the five algorithms on the two test problems using information gain. The generalization performance at each point is the performance of the attribute set that would be returned by the search if it were halted at that point and the best attribute set found so far reported. This is the performance measured on the test set used for hillclimbing, thus it is an optimistic measure of the generalization performance one would expect using that attribute set for unseen instances. See Table 4 for the performance on unseen instances.

The graphs in Figure 1 measure cost in terms of the number of times an instance's attribute value is queried. This is a more accurate estimate of cost than counting the number of decision trees grown because it is usually more costly to grow decision trees if there are many attributes than if there are only a few. Note that the graphs show the complete searches of the non-bidirectional methods (FS and BE), but only the very early part of the bidirectional searches.

The graphs suggest five things.

1. There is no clear winner. BSE yields the best performance on Day of Week, but BSE-SLASH does best on Start Time.
2. There are clear losers—at least on these problems; the three bidirectional methods outperform both FS and BE in most regions of the graphs. Even if computational cost is an issue, there appears to be little advantage to using the naturally terminating methods FS and BE.
3. The method that is leading changes frequently, and apparently unpredictably, as the searches progress; it would be risky to assume that because one method has an early lead that it will perform best in the long run. The safest bet may be to run all three hillclimbing strategies interleaved. (Note that different methods can share one cache.)
4. The bidirectional methods allow a tradeoff between the quality of the result and computational cost: they are easy to stop and restart and, given more cycles, may continue to find better answers.
5. Perhaps most importantly, all the methods find attribute subsets that yield substantial increases in

[1] On a Sparc ELC with 32 attributes and induction sets containing 100 instances.

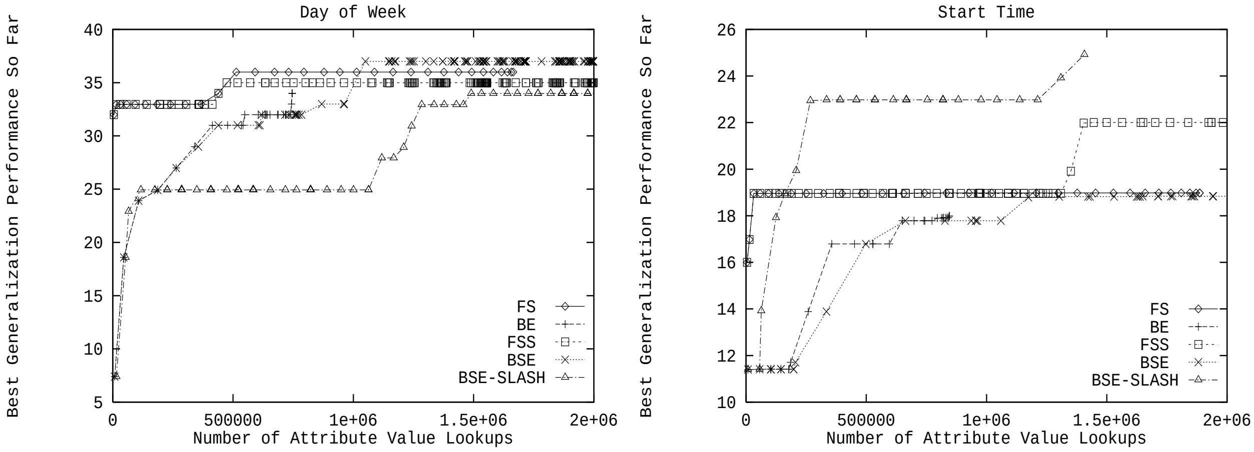

Figure 1: Generalization Performance of Attribute Sets vs. Computational Cost

generalization performance. Clearly some form of attribute selection is useful.

The data in Figure 1 are collected with a locally developed version of ID3/C4.5 that uses information gain. Inspection of the attribute sets explored by these methods shows that all the selection procedures quickly eliminate the high arity attributes information gain is biased to prefer. What advantage, if any, would attribute selection confer to a "smarter" induction algorithm that did not have this bias? To answer this, we ran similar experiments using the default settings of Quinlan's C4.5 program, most notably gain ratio.

Table 3 compares the generalization performance of the best performing attribute sets found with the five greedy hillclimbing strategies with the performance of FOCUS and RELIEF using ID3 with information gain and C4.5 with gain ratio. FOCUS and RELIEF were allowed to use the entire training set for relevance determination; unlike the hillclimbing methods, they do not require that a subset of the training data be reserved as a hillclimbing test set. The entries in the table are the performances when tested on *new,* i.e., previously unseen, instances.

As expected, gain ratio was better than information gain at ignoring high arity attributes. But the performance with all attributes using gain ratio was not as good as the performance of information gain combined with attribute selection. Thus attribute selection mitigated the attribute arity problem of information gain and also conferred additional benefit. Surprisingly, the performance after attribute selection with information gain and with gain ratio was similar. For these test problems at least, it appears that most of the advantage of using gain ratio could be acquired by using information gain and carefully selecting attributes. This suggests that a tradeoff exists between effort spent crafting the bias of the learning procedure and effort spent selecting attributes. Some of the poor

Search Method	**Day of Week**		**Start Time**	
	IG	GR	IG	GR
ALL ATTRIBUTES	4	23	8	4
FS	23	–	14	–
BE	32	–	14	–
FSS	27	29	7	8
BSE	34	34	11	11
BSE–SLASH	14	32	11	13
FOCUS	4	25	7	4
RELIEF-E	10	–	7	–
RELIEF-F	7	24	3	4

Table 3: Percent Accuracy on Unseen Instances using Information Gain (IG) and Gain Ratio (GR)

biases of the learning procedure can, in some domains, be successfully mitigated by careful attribute selection.

The data in Table 3 suggest that the greedy methods find attribute sets that perform better than FOCUS and RELIEF on these problems. The poor performance of FOCUS and RELIEF could be due to the inappropriateness to these tasks of their bias for small attribute sets.[2] Table 4 compares the number of attributes returned by the different methods. All the methods return comparably sized attribute sets, except for forward selection, which seems to need more. Thus the problem FOCUS and RELIEF have in this domain is not due to the inappropriateness of the bias for few features—there are small attribute sets that generalize well on these tasks.

The problem, then, must be the emphasis of FOCUS and RELIEF on *consistency.* FOCUS searches for the smallest attribute set that yields consistency

[2]The only size bias of the hillclimbing methods is to prefer the smallest attribute set when attribute sets of different size have equal performance.

Search Method	Day of Week		Start Time	
	IG	GR	IG	GR
ALL ATTRIBUTES	32	32	32	32
FS	12	–	14	–
BE	4	–	4	–
FSS	5	4	5	5
BSE	7	5	6	6
BSE–SLASH	5	5	6	7
FOCUS	9	9	6	6
RELIEF-E	8	–	7	–
RELIEF-F	6	6	9	9

Table 4: Number of Attributes Used by Each Method

on the training set. RELIEF, which searches for relevant attributes, defines relevance in terms of consistency. It searches for the subset of attributes that are most strongly positively correlated with consistency and most strongly negatively correlated with inconsistency in neighborhoods of the training set. This emphasis on consistency causes both FOCUS and RELIEF to select some attributes that, while relevant, are not useful for generalization, and to ignore other attributes that, while less relevant than those selected, would be more useful to learning.

Hillclimbing seems to be capable of reasonable attribute selection. Given that the efficiency of hillclimbing can be increased for some learners, it is not clear that one needs to resort to proxy measures like relevance to perform attribute selection.

5 Acknowledgements

Thanks to David Zabowski, Tom Mitchell, Andrew Moore, and Ron Kohavi for many useful comments and suggestions.

Research sponsored by the Wright Laboratory, Aeronautical Systems Center and the Advanced Research Projects Agency under grant F33615-93-1-1330, and by a National Science Foundation Graduate Research Fellowship.

References

[1] H. Almuallim and T.G. Dietterich, "Learning With Many Irrelevant Features," *AAAI-91 proceedings, 9th National Conference on Artificial Intelligence,* 1991.

[2] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees,*, Wadsworth International Group, 1984.

[3] R. Caruana and D. Freitag, "Greedy Attribute Selection," *Proceedings of the Eleventh International Conference on Machine Learning*, 1994.

[4] L. Dent, J. Boticario, J. McDermott, T. Mitchell, and D. Zabowski, "A Personal Learning Apprentice," *Proceedings of 1992 National Conference on Artificial Intelligence*, Rutgers, NJ., July 1992.

[5] N. Draper and H. Smith, *Applied Regression Analysis,*, John Wiley & Sons, 2nd edition, 1981.

[6] K. Kira and L. Rendell, "The Feature Selection Problem: Traditional Methods and a New Algorithm," *AAAI-92 proceedings, 10th International Conference on Artificial Intelligence,*, 1992.

[7] I. Kononenko, "Estimating Attributes: Analysis and Extensions of Relief," *Proceedings of the European Conference on Machine Learning,* 1994.

[8] G. John, R. Khavi, and K. Pfleger, "Irrelevant Features and the Subset Selection Problem," *The Proceedings of the Eleventh International Conference on Machine Learning,*, Rutgers, NJ., July 1994.

[9] J. Jourdan, L. Dent, J. McDermott, T. Mitchell, and D. Zabowski, "Interfaces that Learn: A Learning Apprentice for Calendar Management," CMU Technical Report CMU-CS-91-135, Carnegie Mellon University, 1991.

[10] P. Langley, S. Sage, "Scaling to Domains with Many Irrelevant Features," presented at the workshop on Computational Learning and Natural Learning, 1993.

[11] P. Langley, S. Sage, "Oblivious Decision Trees and Abstract Cases," to be presented at the AAAI94 Workshop on Case-Based Reasoning, Seattle, WA., 1994.

[12] N. Littlestone, "Learning quickly when irrelevant attributes abound," *Machine Learning,* **2**:4, pp. 285–318

[13] T. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski, "Experience With a Personal Learning Assistant," to appear in Communications of the ACM, 1994.

[14] A. Moore and M. Lee, "Efficient Algorithms for Minimizing Cross Validation Error,", *The Eleventh International Conference on Machine Learning,* Rutgers, NJ., July, 1994.

[15] J.R. Quinlan, *C4.5: Programs for Machine Learning,* Morgan Kaufman, 1993.

[16] S. Salzberg, "Improving Classification Methods via Feature Selection," John Hopkins TR, 1992.