Detecting Feature Interactions from Accuracies of Random Feature Subsets*

Thomas R. Ioerger
Department of Computer Science
Texas A&M University
ioerger@cs.tamu.edu

Abstract

Interaction among features notoriously causes difficulty for machine learning algorithms because the relevance of one feature for predicting the target class can depend on the values of other features. In this paper, we introduce a new method for detecting feature interactions by evaluating the accuracies of a learning algorithm on random subsets of features. We give an operational definition for feature interactions based on when a set of features allows a learning algorithm to achieve higher than expected accuracy, assuming independence. Then we show how to adjust the sampling of random subsets in a way that is fair and balanced, given a limited amount of time. Finally, we show how decision trees built from sets of interacting features can be converted into DNF expressions to form constructed features. We demonstrate the effectiveness of the method empirically by showing that it can improve the accuracy of the C4.5 decision-tree algorithm on several benchmark databases.

Introduction

One of the most challenging aspects of applying machine learning algorithms to difficult real-world problems is choosing an appropriate representation for examples. The choice of features often has a significant impact on the accuracy of many learning algorithms. Most learning algorithms are effective only when there exist some attributes that are fairly directly relevant to (or correlated with) the target concept. Numerous anecdotal examples have been reported where shifting the representation of examples has been the key to increasing accuracy, in domains from chess (Flann & Dietterich 1986) to splice junctions in DNA (Hirsh & Japkowicz 1994), for example.

The initial selection of features is often done with the assistance of a domain expert, and there are usually many options for smoothing, quantizing, normalizing, or otherwise transforming the raw data. This process is often called "feature engineering." To facilitate decision-making, it would be convenient to have an automated way of quantifying the *utility* of features. Unfortunately, this has proven to be a tricky task. Some approaches, called "wrapper" methods, evaluate features by running the learning algorithm itself on subsets of features and examining the marginal impact on accuracy of adding or dropping features (John, Kohavi, & Pfleger 1994). However, highly relevant features can mask the utility of less relevant or correlated features, which might otherwise have information to contribute. Other approaches, called "filter" methods, attempt to evaluate the relationship between features and the target class using an independent measure, such as conditional entropy, or class-similarity between nearest neighbors along each dimension (Kira & Rendell 1992). In either approach, the utility of an individual feature is often not apparent on its own, but only in combination with just the right other features.

The root of these difficulties lies in a larger issue - *feature interaction* - which is one of the central problems facing machine learning today. Feature interaction is informally defined as when the relationship between one feature and the target class depends on another. As a result, the utility of a feature might not be recognizable on its own, but only when combined with certain other ones, which are needed to form a context in order to reveal its relationship to the target concept. Feature interaction often disperses positive examples in instance space, making learning more difficult, as exemplified by *parity* functions. In fact, feature interaction causes problems for almost all standard learning algorithms (Rendell & Seshu 1990), from decision-trees to perceptrons, which tend to make greedy decisions about one attribute at a time. Feature interaction is also related to lack of conditional independence among features, which violates a basic assumption of the Naive Bayes algorithm, but can be addressed by searching for a Bayesian network with a specific structure that captures any dependencies via restricted joint probability distribution tables at each node (Singh & Provan 1996).

If the interacting features were known, then feature construction techniques could be used to augment the initial representation in such a way as to circumvent the interaction by forcing the features to combine together. This change of representation effectively merges instances back together in instance space and allows greedy algorithms to make more complex and informed decisions. A wide variety of feature construction techniques have been proposed (for a survey, see (Matheus 1989)). In many cases, the operators available for constructing features are limited, and the main difficulty lies in determining which features to combine. Previous feature construction algorithms each use a different approach to avoiding a combinatorial search (such as FRINGE exploiting the replication problem in decision trees, or LFC using lookahead). However, the complexity of feature construction in general remains a challenge.

In this paper, we describe a new approach to detecting feature interactions. Our main idea is to use random sampling to select subsets of features, and then to evaluate whether these features produce higher than expected accuracy when used during learning. The key technical details are: a) how to define the expected accuracy for comparison, and b) how to organize the sampling to provide a fair and balanced search for interacting features within a limited amount of time. Then we describe a simple method for constructing new features from sets of interacting features discovered during the search. Finally, we demonstrate the effectiveness of this approach by showing that it can improve the accuracy of C4.5 on three benchmark databases.

Defining Interactions

We start by developing an operational definition for feature interactions. Informally, a set of features is said to interact if their utility for classifying the target concept is only apparent when they are all combined together. We take an empirical approach to capturing this idea formally: we run the underlying learner on some representative examples using various sets of features, and observe the average accuracy of the constructed decision trees by cross-validation. While this should work for any learner, we focus on decision-tree algorithms in this paper. We define a (positive) interaction among a set of features as when the observed accuracy for all the features taken together is larger than expected, based on the accuracies of trees built using any known subsets of those features.

Definition 1 *A set of features constitutes an* interaction *when the observed accuracy of the underlying learning algorithm with all of the features is higher than "expected."*

At the very least, we can try to estimate what the accuracy of the subset will be based on the performance of the individual features (i.e. one-level decision trees). However, the estimates will clearly be more accurate if we have some information about the performance of various combinations of features.

Bayesian Expectations for Accuracies

The key to this definition is determining what a reasonable expectation is for the accuracy of the whole combination, since interactions are defined in contrast to this (i.e. when the observed accuracy differs from it). One might expect that the accuracy of decision trees built from a set of features would be at least be equal to the maximum for any subset, since the same features are available and presumably could be used to reconstruct the tree. However, it is in general hard to predict how new features will interleave with other ones in being selected for splits, which could improve the accuracy, or possibly even decrease it.

We can use a simple Bayesian analysis to provide a reasonable estimate of the accuracy of a combination of features. To simplify, assume we have two disjoint sets of features, F and G, and we are interested in predicting the accuracy of trees built from the union $F \cup G$. We do not know the exact structure of the trees built from each set of features; they can change with each run of cross-validation. However we can treat the trees built by the decision-tree algorithm with each set of features as a black box, and combine them as independent learners. Call trees built with the first set of features L_F and trees built with the second set of features L_G. To classify a new example with a hypothetical composite learner, we would get the predicted class for the example by running it through L_F, get the class predicted by L_G, and then make the optimal decision based on the probabilities of mis-classification for each of the learners.

For example, if the class predicted by L_F is c_1 and the class predicted by L_G is c_2, then the Bayes-optimal decision would be to output the class that is most likely, given these states of the component learners: $argmax_{c_i}\ Prob[c(x) = c_i \mid L_F(x) = c_1, L_G(x) = c_2]$. The estimated accuracy of this approach would be equal to the probability of the chosen class, which is simply the *max*. This is summed over all combinations of class labels that could be output by L_F and L_G, weighted by the probabilities of these cases:

$$\mathrm{P}[L_{F\cup G}(x) = c(x)] = \sum_{c_j,c_k} \mathrm{P}[L_F(x) = c_j, L_G(x) = c_k] \times$$

$$max_{c_i}\ \mathrm{P}[c(x) = c_i \mid L_F(x) = c_j, L_G(x) = c_k]$$

Next, re-write the conditional probability using Bayes Rule and cancel the denominators with the weights:

$$\mathrm{P}[L_{F\cup G}(x) = c(x)] = \sum_{c_j,c_k} \mathrm{P}[L_F(x) = c_j, L_G(x) = c_k] \times$$

$$max_{c_i} \frac{\mathrm{P}[L_F(x) = c_j, L_G(x) = c_k | c(x) = c_i]\ \mathrm{P}[c(x) = c_i]}{\mathrm{P}[L_F(x) = c_j, L_G(x) = c_k]}$$

$$= \sum_{c_j,c_k} (\ max_{c_i}\ \mathrm{P}[L_F(x) = c_j, L_G(x) = c_k \mid c(x) = c_i]$$

$$\times\ \mathrm{P}[c(x) = c_i]\ )$$

Finally, invoke the Independence Assumption to break apart the conditional probabilities into values that are easy to determine experimentally:

$$\mathrm{P}[L_{F \cup G}(x) = c(x)] =$$

$$\sum_{c_j, c_k} (\ max_{c_i}\ \mathrm{P}[L_F(x) = c_j \mid c(x) = c_i] \times$$

$$\mathrm{P}[L_G(x) = c_k \mid c(x) = c_i] \cdot \mathrm{P}[c(x) = c_i]\ )$$

The values required for this computation can be derived directly from the average rates of true and false positives and negatives for trees built from each subset of features (along with prior class probabilities). Specifically, they can be extracted from the confusion matrix averaged over multiple runs of cross-validation. This approach can easily be extended to predicting the accuracy for partitions with more than two subsets of features. We note that this method of calculating expected accuracies often produces the maximum of the accuracies of the combined subsets, consistent with what is often observed in decision trees, though the expected accuracy can also be higher than the maximum for synergistic combinations.

So this Bayesian approach provides reasonable estimates of the performance of decision trees built from a combination of features, based on the performance of subsets. An important consequence of this approach is that it essentially equates feature interactions with *non-independence*. If the whole combination of features produces an accuracy higher than expected, then the Bayesian explanation for this would be that the probabilities used in the calculation were not in fact independent. The observed accuracy can also be *lower* than expected, which we call a *negative* interaction. This can happen in cases where features interfere with each other in terms of selection of splits in trees.

Apical Interactions

This definition for interactions so far is awkward because the appropriate partition to base the calculation on is undefined, and essentially, the more subsets whose performance we know, the better the estimate will become. However, we can make a pragmatic improvement by restricting the definition to a special class of features interactions. We note that, since the accuracy produced by a subset of features is usually at least as great as the accuracy of the best feature by itself (in the subset), interactions among lesser features will often be masked. Therefore, any observed interaction will typically involve the most accurate individual feature in the subset, which we call an *apical* interaction.

Definition 2 *An* apical interaction *is an interaction among a set of features such that the most accurate feature by itself is required for the interaction.*

Of course, any feature can participate in an apical interaction with features of lower accuracy than it.

Detecting these types of interactions is much easier. Suppose $F_1 ... F_m$ is a set of features sorted in increasing order such that F_m has the highest accuracy by itself. Consider the partition $\{\{F_1 ... F_{m-1}\}\{F_m\}\}$. Empirically determining the accuracy of the combination of the first $m-1$ features would reveal the utility of almost all of them together, including possible interactions among lesser features. This can be combined with the accuracy of the single most accurate feature to produce a reasonable Bayesian expectation for the whole set. And yet, this calculation will probably underestimate the true accuracy for apical interactions, since it does not include the added boost in accuracy that occurs only when F_m is combined with the others.

Searching for Interactions

Now that we have an operational definition for feature interactions, we need to determine a way of finding them efficiently. To establish a frame of reference, we start by considering the amount of work it takes to do an exhaustive search. Suppose we have a total of n features and we are looking for interactions of order b (i.e. a set of b features constituting an apical interaction). To determine whether such an interaction exists, and if so to identify it, we would have to run the learning algorithm on all possible combinations of n features taken b at a time. This gives the observed accuracy for each of the combinations, but we have to compare it to the expected accuracy. To compute the expected accuracy, we would have to evaluate the accuracy of all but the highest feature ($b-1$ total) and then calculate the expected value of the full combination using the Bayesian formula. Apical interactions would thus be detected in cases where the observed accuracy is higher than this expected value.

To fairly account for the work required, we need to consider how the underlying algorithm itself is affected by the number of features involved. In practice, decision-tree algorithms typically have run-time roughly linear in the number of features (though theoretically they could be quadratic in the worst case, since they depend on the size of the trees) (Quinlan 1986). Therefore, to a first approximation, we can treat the run-time as a constant times the number of features being evaluated. The constant can be determined empirically by running the decision-tree algorithm on various randomly chosen sets of features, using the actual number of examples and cross-validation procedure that will be used throughout the experiment. We call this constant u, and hereafter concern ourselves only with units of work based on numbers of features evaluated.

Hence, the work for exhaustive search requires ${}_nC_b$ runs of the algorithm on b features at a time (b units of work) plus ${}_nC_{b-1}$ runs of the algorithm on $b-1$ features at a time ($b-1$ units of work), or:

$$work_{exh} = b \cdot {}_nC_b + (b-1) \cdot {}_nC_{b-1}$$

This work scales up badly as n increases, and very badly as b increases. It is usually only feasible to exhaustively search for at most binary interactions in real-world datasets.

Random Sampling

To reduce the complexity of the search, we propose sampling subsets larger than the postulated interaction. The idea is that, if we are looking for an interaction of size 3, for example, then instead of testing them all, which is essentially exponential, we can try randomly evaluating subsets of size 6, for example. The probability of combining the right 3 features is higher because the larger subsets each contain multiple triples which are being evaluated simultaneously.

Of course, not all of these triples are easily identifiable as interactions in the set of 6 features. We discuss this in more detail below. But for now, it is useful to point out in general that the probability of selecting the right b features of an interaction in a subset of size $k > b$ is:

$$p(n,k,b) = \frac{{}_{n-b}C_{k-b}}{{}_{n}C_{k}} = \frac{k(k-1)...(k-b+1)}{n(n-1)...(n-b+1)}$$

where the numerator represents the number of ways that the remaining $k-b$ features can be chosen out of $n-b$ total if the b interacting ones are forced to be included, and the denominator represents all possible combinations of n things taken k at a time.

Since we are generating subsets by random sampling rather than exhaustive search, we cannot guarantee that we will observe an arbitrary interaction of size b. However, if we repeat this process some number of times, eventually the probability becomes quite high. For example, if we repeat the sampling of subsets of size k for r times, the chances that we will observe an arbitrary combination of b features (the postulated interacting ones) becomes:

$$p(n,k,b,r) = 1-(1-p(n,k,b))^r$$

In the approach we will be describing, we will allow the user to specify the minimum acceptable probability, such as $p > 0.95$, to set the level of completeness required in the search. For a given value of k, determined by our approach, we can then solve for the smallest number of repetitions of sampling required to meet the requirement specified by p. Generally, as k grows, many fewer repetitions are required.

To use this idea of sampling to search for interactions requires a bit more sophistication. First, even if a given set of features contains a subset of interacting features, it might be masked by other features with higher accuracies. Also, we must consider that sampling larger sets of features costs more. Therefore, we break up the search into sub-searches for specific apical interactions. Apical interactions are easy to identify since they necessarily involve the top feature in the subset. Such an interaction will generally not be affected by the inclusion of additional (non-interacting) features of lower accuracy, and the only penalty is that we have to repeat these tests for each feature as the candidate apical feature.

In detail, we sort all of the features by their individual accuracies (assume w.l.o.g. that $F_1...F_n$ are in sorted order). Starting with $m = b$, we look for apical interactions of F_b with the $b-1$ features below it. Then we increment m to search for apical interactions where F_{b+1} is the top feature, and so on, up to F_n.

If we were to structure the exhaustive search in levels like this, it would require ${}_{m-1}C_{b-1}$ subsets to be evaluated (running the decision-tree algorithm) with and without F_m, for each m from b up to n. However, at each level m, we can use our idea of sampling subsets larger than $b-1$ to reduce the amount of work. What size subsets should be sampled, and how many should we generate? Since the number of features to choose from grows with m, we suggest increasing the size of the sampled subsets linearly. The growth rate γ can be anything between 0 and 1. For example, using $\gamma = 1/2$, finding apical interactions with the 12^{th} (least accurate) feature F_{12} would involve evaluating combinations of 6 features at a time including F_{12} (or F_{12} combined with 5 of the 11 lesser features). In any case, if $m \cdot \gamma < b$, we always evaluate subsets of size at least b (exhaustive for small cases). When $\gamma = 1$, this is essentially equivalent to asking for each feature: when combined with all the features with lower accuracy than it, is there an interaction? However, the problem with this extreme case is that it risks confusion if multiple sets of interacting features are mixed together. We would prefer to make γ as small as possible, so that the subsets of features provide an isolated context and hence the best chance to identify a given interaction, with as little distraction from additional features as possible.

Making γ smaller means we have to repeat the sampling at each level a greater number of times to achieve the desired probability p. However, evaluating smaller subsets for apical interactions requires less time in running the decision-tree algorithm. Although it is not immediately obvious, it turns out that as γ decreases, the overall cost (summed over all levels m) increases, taking into account both the number and size of samples required. For a given value of γ, the cost of evaluating each sample to identify apical interactions for each feature F_m is approximately: $m \cdot \gamma + (m \cdot \gamma - 1) = 2m\gamma - 1$ (for running the cross-validation once with $m \cdot \gamma$ features and once without F_m). This has to be repeated enough times to ensure with probability $> p$ that an interaction among b arbitrary features will be observed. So we compute the minimum number of repetitions r_m required by increasing r until $p(m-1, m \cdot \gamma, b-1, r) > p$, since we have $m-1$ features to choose from, we are sampling subsets of a fraction γ of that size, and we are trying to identify the right $b-1$ features that complete the apical interaction. Hence the total cost for determining whether F_m is involved in an apical interaction is $r_m \cdot (2m\gamma - 1)$. This is summed up for all levels m:

$$Work_{sampling} = \sum_{m=b}^{n} r_m \cdot (2m\gamma - 1)$$

This cost estimate is used to determine the optimal value for γ. Suppose we have a fixed amount of time t (in units of work, or $t \cdot u$ seconds) to devote to searching for interactions. Our approach involves determining the minimum value for γ such that the overall cost of the experiments does not exceed t. This is done by estimating the cost for various values of γ distributed uniformly between 0 and 1, and selecting the smallest value of γ that allows the total work to be completed within t.

Example

Table 1 shows an example that gives the schedules for several values of γ (0.2, 0.4, 0.6, and 0.8), designed to identify interactions up to order $b = 4$ among $n = 20$ features. Each column gives a prescription for testing features (in increasing order of accuracy) for apical interactions. Each entry shows the size of the random subsets to generate and combine with the corresponding apical feature for the evaluation, along with the number of repetitions required to guarantee with at least 95% confidence that a hypothetical interaction will be contained in at least one of the subsets. For example, in the row for $m = 8$ and the column $\gamma = 0.2$, we see that 104 random subsets of size 3 among the 7 features with lower accuracy than F_8 would have to be generated. Each of these would be tested for apical interactions with F_8 by calculating the cross-validated accuracy of the decision tree algorithm on the random set of features with and without F_8, and comparing to the expected accuracy from the Bayesian analysis of confusion matrices. As γ increases across a row, proportionally larger subsets of lesser features must be sampled (e.g. for F_{16}: 3/15, 6/15, 9/15, and 12/15).

In the table, we also indicate the work (units of runtime) estimated for evaluating each feature for apical interactions. These are shown at the bottom. Note how the total work required drastically decreases as γ increases. Figure 1 gives a better picture of how the work decreases for increasing values of γ. For comparison, the amount of computation required for an exhaustive search of all four-way interactions among 20 features would be 22,800 units of time. Therefore, if one did not have enough time to do an exhaustive search, one could use random sampling with a value for γ that requires less time (e.g. $>\sim$ 0.3 in this case).

To use this analysis to generate a search schedule for a specific time bound, we could compute the work required for a uniform distribution of values of γ, such as in increments of 0.05, and take the lowest one that meets the time bound. For example, suppose we only wanted to allot 5,000 units of time for the search for interactions. Based on the above graph, $\gamma = 0.55$ would suffice. In fact, by sampling for apical interactions using 55% of the features with accuracy less than each one being tested for apical interactions, the work can be accomplished in 4991 units of time.

Table 1: Example schedule for searching for interactions of order 4 among 20 features. W=work.

	$\gamma = 0.2$		$\gamma = 0.4$		$\gamma = 0.6$		$\gamma = 0.8$	
m	Sz	Rp	Sz	Rp	Sz	Rp	Sz	Rp
4	3	1	3	1	3	1	3	1
5	3	11	3	11	3	11	4	1
6	3	29	3	29	3	29	4	6
7	3	59	3	59	4	14	5	5
8	3	104	3	104	4	25	6	4
9	3	167	3	167	5	16	7	4
10	3	251	4	62	6	12	8	3
11	3	358	4	89	6	17	8	5
12	3	493	4	123	7	13	9	5
13	3	658	5	65	7	18	10	4
14	3	856	5	85	8	14	11	4
15	3	1089	6	54	9	12	12	4
16	3	1362	6	67	9	15	12	5
17	3	1677	6	83	10	13	13	5
18	3	2036	7	57	10	16	14	4
19	3	2444	7	69	11	14	15	4
20	4	725	8	51	12	12	16	4
W		87690		12122		3638		1376

Feature Construction

Once we have discovered a feature interaction, we would like to take advantage of this knowledge to construct a new feature based on it, hopefully to improve the overall accuracy of the learning algorithm. If the features interact, then there must be something about the way that they combine together in a decision tree that produces a higher accuracy than expected based on a simple Bayesian analysis. If this is the case, then we would like to create a new feature that instanta-

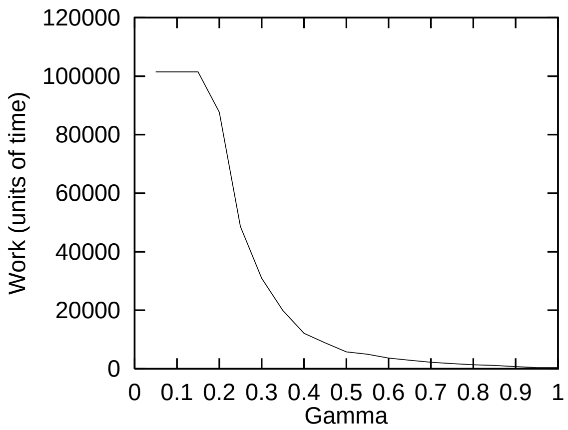

Figure 1: Work (in units of time) required to search for interactions using different values of γ.

neously captures the information in this complex decision, effectively allowing the splitting algorithm to realize the benefit (information gain) of several tests in one step and avoid the pitfalls of making greedy, less informed decisions based on one attribute at a time.

The method we propose for constructing the new feature is to build a decision tree using the interacting features and then extract a DNF expression that represents a grouping of the positive leaves. Let $T(S)$ be a tree built from a subset $S \subset \mathcal{F}$ of the initial features. Let $L_1...L_n$ be the leaf nodes of $T(S)$. Each leaf node can be expressed as a conjunction of attribute tests $(a_1 = v_1) \wedge (a_2 = v_2) \wedge ...$ on the path from the root to that node in the tree. Now, we could construct a DNF expression $L_{i_1} \vee L_{i_2} \vee ...L_{i_d}$ for the d leaf nodes L_i that are positive ($P[c(x) = + \mid L_i] > 0.5$, assuming a two-class model). However, this does not necessarily guarantee to form a new feature with maximum information gain. It is possible that it might be better to group some positive leaves with negative leaves to avoid diluting the impact of a few very pure leaves.

Therefore, the algorithm we use is to sort the leaves according to their purity (100% to 0% positive), and incrementally try shifting the proportion p^* that separates the two groups. For each division, we form the DNF expression of the group of more positive leaves and evaluate the information gain (Quinlan 1986) of this new candidate feature. Call the leaves in sorted order $L_{s_1}...L_{s_n}$, such that the fraction of positives decreases monotonically $p(L_{s_1}) \geq p(L_{s_2}) \geq ....$ We begin by separating out the most positive leaf and calculating the information gain of the partition: $\{\{L_{s_1}\}\{L_{s_2} \vee ...L_{s_n}\}\}$. Then we shift the next most positive leaf over: $\{\{L_{s_1} \vee L_{s_2}\}\{L_{s_3} \vee ...L_{s_n}\}\}$, recalculate the information gain, and so on. Finally, we return the disjunction $L_{s_1} \vee L_{s_2} \vee ...L_{s_q}$ that produces the maximum information gain. The value of this new feature is calculated for each example and appended to its attribute vector.

Experiments

In this section, we describe the results of some experiments with this approach for discovering interactions and constructing new features from them, using C4.5 (Quinlan 1993) as the underlying learning system. We applied our approach to three databases from the UC Irvine repository: *congressional-voting* (with **physician-fee-freeze** feature removed), *census*, and *heart-disease*. In each of the databases, we were able to discover interactions among features that, when used in feature construction, boosted the accuracy of C4.5 by $3 - 8\%$ using cross-validation.

The general course of the experiments is as follows:

1. First, the database is randomly divided into three equal subsets, which we call the "training," "pruning," and "testing" sets, respectively.
2. Next a schedule is generated that is estimated to take about $10 - 15$ minutes on a typical workstation. (This may require some preliminary tests to evaluate how long it takes to run C4.5 with cross-validation on various numbers of features.)
3. Then this schedule is followed to test random subsets of features for apical interactions using the training examples. A given subset with its apical feature are evaluated for interaction by determining the accuracy of C4.5 on the set of features with and without the apical feature, and comparing the observed accuracy to that expected by a Bayesian analysis.
4. When an interacting subset is discovered, a decision tree based on the features is built by C4.5 using all of the training data, and is pruned with the pruning set using *reduced-error pruning* (Quinlan 1987).
5. This pruned tree is used to construct a new feature by converting its paths into a DNF expression and finding the subset of disjuncts (in linear order) that produces the highest entropy reduction.
6. Finally, the new feature is added to the original features in the database, and the change in accuracy is determined over the testing examples using a paired T-test (i.e. running cross-validation within the testing examples to see if the new feature leads C4.5 to generate more accurate trees).

It is important to note that the accuracies we report are slightly lower (but not much) than have been reported by others in the literature, but this is because we are evaluating the utility of constructed features by running C4.5 on only one-third of the data.

Voting Database

The *Voting* database contains the voting records for 435 US Congresspersons during 1984 on 15 key bills. The attributes are all discrete, consisting of three possible votes. The goal is to learn to how to discriminate between Democrats and Republicans. In the original database, one feature, **physician-fee-freeze**, could be used to achieve 95% classification accuracy on its own. We removed this feature, which make the problem a little more challenging; C4.5 can usually only get $85 - 90\%$ accuracy using the remaining features.

Table 2 shows the 15 features, their individual accuracies cross-validated within the training set, and a schedule established for searching for interactions. This schedule was designed for searching for interactions of up to third order within 3000 units of time.

During the search, a total of 389 subsets with three to six features were evaluated for apical interactions. 108 subsets exhibited some degree of interaction, where the accuracy when the most accurate feature was added to the rest was higher than expected, by a difference ranging from just over 0 up to 8.0% (note: we do not require the increase to be statistically significant at this stage). Each of these was used to construct a new feature by building a decision tree, pruning it, converting to a DNF expression, and finding a set of disjuncts

Table 2: Features, accuracies, and sampling schedule for the Voting database. 'Sz' means how many of the features with lower accuracy should be selected in random subsets and tested for apical interactions, and 'Rp' means how many times this should be repeated.

num	feature	acc	Sz	Rp
F1	Religious-Groups-In-Schools	0.589		
F2	Synfuels-Corporation-Cutback	0.605		
F3	Water-Projects-Cost-Sharing	0.642	2	1
F4	Immigration	0.648	2	8
F5	Exports-To-South-Africa-Act	0.704	2	17
F6	Handicapped-Infants	0.707	2	29
F7	Duty-Free-Exports	0.710	2	44
F8	Superfund-Right-To-Sue	0.729	2	62
F9	Crime	0.753	3	27
F10	Anti-Satellite-Test-Ban	0.774	3	35
F11	Aid-To-Nicaraguan-Contras	0.796	3	44
F12	MX-Missle-Program	0.831	4	26
F13	Education-Spending	0.843	4	32
F14	El-Salvador-Aid	0.860	4	38
F15	Adoption-Of-The-Budget-Res	0.863	5	26

with maximal entropy reduction. Many of these new features, when added to the original 15 attributes, did not produce a measurable increase in the performance of C4.5. Often this was because interactions were detected among less relevant features, the combinations of which were masked by the performance of better features in the decision trees anyway.

However, there were several interactions that produced feature constructions that improved the accuracy of C4.5 over the testing set. One example involved an interaction between EL-SALVADOR-AID (F14) and SYNFUELS-CORP-CUTBACK (F2). This was discovered when evaluating a subset of features including these two plus HANDICAPPED-INFANTS (F6), IMMIGRATION (F4), and EXPORTS-SOUTH-AFRICA (F5). On the training data, the accuracy of F14 was highest (86.3%), and the accuracy of the others combined (F2, F4, F5, and F6) was 81.6%. Based on the Bayesian analysis of confusion matrices, the accuracy for all five features together was expected to be 89.6%, but when the accuracy of C4.5 on all five features was determined by cross-validation on the training set, the accuracy was observed to be 91.2%, which was 1.7% higher than expected. Hence this is assumed to be an apical interaction.

A decision tree was constructed with these five features using the training examples, and was then pruned using the pruning examples. Here is the pruned tree:

```
EL-SALVADOR=N : DEMOCRAT
EL-SALVADOR=? : DEMOCRAT
EL-SALVADOR=Y :
  |  SYNFUELS-CORP-CUTBACK=Y : DEMOCRAT
  |  SYNFUELS-CORP-CUTBACK=N : REPUBLICAN
  |  SYNFUELS-CORP-CUTBACK=? : REPUBLICAN
```

The pruned tree was used to construct the following feature (DNF expression), which covered mostly Democrats:

```
(EL-SALVADOR=N) or (EL-SALVADOR=?) or
((EL-SALVADOR=Y) and (SYNFUELS-CORP-CUTBACK=Y))
```

When this constructed feature was added to the original 15 features, the cross-validated accuracy of C4.5 on the *testing* examples rose from 81.7% to 87.1%, and the increase of 5.4% in accuracy was significant based on a paired T-test ($Z = 4.91, p < 0.05$). Table 3 shows several other features, constructed from interactions, that were found to improve the accuracy of C4.5 on the testing set. It should be noted that these types of paired T-tests have been shown to occasionally lead to high type I error (Dietterich 1998), and more robust methods are being explored to assess the significance of the constructed features more accurately.

Table 3: Results on the Voting database. The features constructed from some detected interactions are shown. The 'acc w/o' column represents the accuracy of C4.5 on the original features, using cross-validation on the set of testing examples, while the 'acc with' column shows the accuracy when the constructed feature is added. The last two columns show the difference and the Z-score, which is the test statistic for a paired T-test; Z$>$ 1.83 is significant for $p < 0.05$.

acc w/o	acc with	diff	Z
(EL-SALVADOR=?) or (EL-SALVADOR=N) or ((EL-SALVADOR=Y) and (SYNFUELS=Y))			
81.7%	87.1%	5.4%	4.91
(MX-MISSLE=Y) or (MX-MISSLE=?) or ((MX-MISSLE=N) and (SYNFUELS=Y))			
80.8%	84.6%	3.8%	3.29
(EL-SALVADOR=?) or (EL-SALVADOR=N) or ((EL-SALVADOR=Y) and (BUDGET=Y or ?))			
80.2%	86.2%	6.0%	3.26

Census Database

The *Census* database consists of thousands of records of people taken during a recent census. For each individual, a total of 14 attributes, including both discrete and continuous ones, are given, such as age, sex, education, type of occupation, marital status, etc. The goal is to predict from these attributes whether the annual income of an individual is $>$ or $\leq$ \$50,000. We used a subset of 1000 randomly selected examples in our experiment. A schedule similar to the one for the Voting database was developed for searching for interactions.

A total of 681 feature subsets were evaluated, and of these, 54 were found to have apical interactions; the accuracies of these subsets were greater than expected by up to 3.8%. Table 4 shows some features that were constructed from these detected interactions which produced significant gains in the accuracy of C4.5 (each by around 3.5%) for predicting income levels on the set of testing examples (about 300 examples)

Table 4: Results on the Census database.

acc w/o	acc with	diff	Z
(RELATIONSHIP≠Husband) or ((RELATIONSHIP=Husband) and (YEARS-EDUCATION≤9))			
75.2%	78.4%	3.2%	2.23
(MARITAL-STAT≠MarriedToCivilian) or ((MARITAL-STAT=MarriedToCivilian) and (YEARS-EDUCATION≤9))			
76.6%	80.3%	3.7%	2.50
(CAPITAL-GAIN≤5013) and (CAPITAL-LOSS≤1741)			
76.9%	80.4%	3.5%	5.4

Table 5: Results on the Heart Disease database.

acc w/o	acc with	diff	Z
(NUM-COLOR-VESSELS=0) or ((NUM-COLOR-VESSELS=1) and (SEX=Fem))			
70.4%	78.8%	8.4%	4.25
((EX-ANGINA=T) and (CHEST-PAIN=None or Asymp)) or ((EX-ANGINA=F) and (COLORED-VESSELS≤1))			
69.7%	74.8%	5.1%	2.13

Heart Disease Database

The *Heart* database consists of medical records taken from approximately 300 patients in a clinic in Cleveland. It contains 13 attributes, both continuous and discrete, on personal history and various test results. The state of health for each of the patients is given as a number from 0 to 4 indicating level of heart disease, with 0 representing absence and 4 being worst. For this experiment, we grouped all of the levels from 1 to 4 together, making the goal to distinguish between health and sickness. A schedule similar to the one for the Voting database was developed for searching for interactions by sampling random subsets and evaluating their accuracy with and without the apical feature. Out of 422 feature subsets evaluated, 27 revealed apical interactions, with accuracies that were greater than expected by up to 5.8%. Table 5 shows two of the features that were constructed from detected interactions which produced significant gains (5.1% and 8.4%) in the accuracy of C4.5 on the set of testing examples.

Conclusion

We have shown that feature interactions can be effectively discovered by random sampling. We introduced the notion of an apical interaction, in which the accuracy a learning algorithm can achieve with a given subset of features is higher than the expected accuracy from a Bayesian combination of the most accurate feature with a hypothesis learned from the rest of the features. This definition essentially equates interactions with feature non-independence. An important consequence of defining interactions this way is that they are dependent on the learning algorithm being used. Thus an interaction discovered with a decision-tree algorithm might not necessarily interact in a neural network, for example. We also provided a method for adjusting the sampling for a limited amount of CPU time so that testing for interactions remains fair and balanced. Exhaustive testing for interactions is generally infeasible. By randomly selecting slightly larger subsets to test for apical interactions, we can increase the probability of observing an interaction with less work. We provide a prescription for determining the optimal size of subsets to sample, and how many, to make a probabilistic guarantee of adequate search within resource bounds. Finally, we describe a method for converting observed interactions into new features by extracting DNF expressions from pruned decision trees built with the interacting features. The resulting constructed features combine multiple attribute tests that allow the learning algorithm to make more complex and informed decisions with each test, which can potentially overcome the greediness of many learning algorithms.

References

Dietterich, T. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation* 10:1895–1924.

Flann, N., and Dietterich, T. 1986. Selecting appropriate representations for learning from examples. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, 460–466.

Hirsh, H., and Japkowicz, N. 1994. Bootstrapping training-data representations for inductive learning. In *Proc. AAAI*, 639–644.

John, G.; Kohavi, R.; and Pfleger, K. 1994. Irrelevant features and the subset selection problem. In *Eleventh International Conf. on Machine Learning*, 121–129.

Kira, K., and Rendell, L. 1992. A practical approach to feature selection. In *Proceedings of the Ninth International Workshop on Machine Learning*, 249–256.

Matheus, C. 1989. *Feature Construction: An Analytic Framework and an Application to Decision Trees.* Ph.D. Dissertation, University of Illinois, Department of Computer Science.

Quinlan, J. 1986. Induction of decision trees. *Machine Learning* 1:81–106.

Quinlan, J. 1987. Simplifying decision trees. *International Journal of Man-Machine Studies* 27:221–234.

Quinlan, J. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufmann: Palo Alto, CA.

Rendell, L., and Seshu, R. 1990. Learning hard concepts through constructive induction: Framework and rationale. *Computational Intelligence* 6:247–270.

Singh, M., and Provan, G. 1996. Efficient learning of selective bayesian network classifiers. In *Thirteenth International Conf. on Machine Learning*, 453–461.