Constructive Induction of Cartesian Product Attributes

Michael J. Pazzani pazzani@ics.uci.edu
Department of Information and Computer Science
University of California, Irvine ph: +1-714-824-5888
Irvine, CA 92717 USA fx: +1-714-824-4056

Abstract: *Constructive induction is the process of changing the representation of examples by creating new attributes from existing attributes. In classification, the goal of constructive induction is to find a representation that facilitates learning a concept description by a particular learning system. Typically, the new attributes are Boolean or arithmetic combinations of existing attributes and the learning algorithms used are decision trees or rule learners. We describe the construction of new attributes that are the Cartesian product of existing attributes. We consider the effects of this operator on a Bayesian classifier an a nearest neighbor algorithm.*

Keywords: Bayesian Classifier, Nearest Neighbor, Dependencies.

Area of Interest: Concept Formation and Classification.

1 Introduction

In classification learning, a variety of methods have been explored for changing the representation of examples to improve the accuracy of learned concepts. One method is to select a subset of the attributes (Caruana and Freitag, 1994; John, Kohavi, & Pfleger 1994; Moore & Lee, 1994; Skalak, 1994). In the subset selection problem, examples are represented by a set of attributes (e.g., {`height, age, shoe-size, weight`}) and the goal is to find a subset of these attributes (e.g., {`height, age, weight`}) such that when a classifier is applied to examples represented by a subset of the attributes the accuracy of the classifier is maximized. This problem has also recently received attention in the context of Bayesian classifiers (Langley & Sage, 1994), Bayesian networks (Provan & Singh, 1995) and nearest neighbor algorithms (Aha & Bankert, 1995; Townsend-Webber & Kibler, 1994). The subset selection problem has also been studied in pattern recognition (e.g., Kittler, 1986). It has proved important in applying machine learning to real world tasks (Salzberg, Chandar, Ford, Murthy & White, 1995; Kubat, Flotzinger and Pfurtscheller, 1993) in which the attributes were not selected by hand by experts.

Constructive induction (Michalski, 1978) is another method for improving the accuracy of learned concepts. Constructive induction is the process of changing the representation of examples by creating new attributes from existing attributes. The new attributes are typically Boolean or arithmetic combinations of existing attributes and the learning algorithms used are typically decision trees or rule learners (e.g., Matheus & Rendell, 1989; Pagallo & Haussler, 1989; Ragavan & Rendell, 1993; Sanger, Sutton, & Matheus 1992; Wnek & Michalski, 1994) although neural nets have also been used (e.g., Cherkauer & Shavlik, 1993). Here, we consider the use of different operator for con-

structive induction: the Cartesian product. We will use the term "joining" to refer to the process of creating a new attribute whose values are the Cartesian product of two other attributes. For example, if `height` with values `tall` and `short` were joined to `weight` with values `heavy` and `light`, a new attribute would be formed `height_weight` with values `tall_heavy`, `tall_light`, `short_heavy` and `short_light`.[1] Note that joining two attributes differs from conjoining them that is commonly used in some induction systems (e.g., Schlimmer, 1987; Ragavan & Rendell, 1993) because the resulting attribute is not a Boolean attribute.

In this paper, we first propose an algorithm based on the "wrapper model" (John, Kohavi & Pfleger, 1994) for the construction of Cartesian product attributes and describe the effect of constructive induction of Cartesian product attributes on three learning algorithms: a naive Bayesian classifier (Duda & Hart, 1973); ID3, a decision tree learner (Quinlan, 1986); and PEBLS, a nearest neighbor algorithm (Cost & Salzberg, 1993). We show experimentally that constructing Cartesian product attributes is not generally beneficial to decision tree learners but can be beneficial to Bayesian classifiers and PEBLS.

2 A Wrapper Approach for Creating Cartesian Product Attributes

A template of the general algorithm for changing example representation is shown in Table 1. In summary, it starts with the provided example representation and performs hill-climbing search to find a new representation by applying two operators:

a. Replacing a pair of attributes in the representation with a new attribute that is the Cartesian product of the two attributes.

b. Deleting an attribute used by the representation.

There are several items that require further elaboration and explanation about the algorithm in Table 1. First, to determine which single change of representation is best, the accuracy of the learned description that would result from this representation is estimated by using leave-one-out cross validation on the training data. We choose this particular method of accuracy estimation because it may be computed quickly for the naive Bayesian classifier and the nearest neighbor algorithm. Second, we terminate the process of changing the example representation when no change results in an improvement over the current representation as determined by leave-one-out cross-validation. Third, a new attribute replaces the attributes from which it was formed. This is done since some learners (e.g., Bayesian classifiers) assume independence of attributes within each class and the Cartesian product of two attributes is clearly dependent on the two attributes.[2] Fourth, we consider deleting attributes as well as joining attributes since this can easily be accomplished at the same time and it is useful both in eliminating original irrelevant attributes and attributes created with constructive induction. Finally, we use hill-climbing search to explore the search space, since an exhaustive search would not be feasible[3]. In

[1] Note that if the value of either attribute is unknown, then the value of the joined attribute is unknown. Joining is only defined on discrete attributes. If an attribute has continuous values, it must be discretized to be joined. In this paper, all continuous attributes are discretized into five values.

[2] On decision trees, which do not make the same independence assumption, retaining the original attributes had no significant effect in our experiments.

[3] The number of possible changes to representation grows faster than exponential in the number of attributes. One way to view this problem is to find a partitioning of the attributes, such that each attribute appears in at most one block of the partition. For example, one partitioning of the attributes {`height, age, shoe-size, weight`} would be {{`height, weight`}, {`age`}}. When two or more attributes appear in the same block, the cross product

```
0.  Start with the initial example
representation
1.  Consider joining each pair of
attributes
2.  Consider deleting each attribute
3.  If there is a ''good'' change of
representation
Then:  Make the ''best'' change of
representation and go to 1
Else:  Return the current example
representation.
```

Table 1: A template of the algorithm for changing representation.

the future we may explore other search algorithms such as best-first search or genetic algorithms but in this paper we provide evidence that the Cartesian product operator is useful in constructive induction and that hill-climbing search guided by accuracy estimation can improve the accuracy of some learning algorithms.

There are two reasons for introducing the Cartesian product operator in constructive induction. First, by rerepresenting examples, some concepts that are not capable of being expressed in some learners representation language can now be expressed. For example, the naive Bayesian classifier is limited in expressiveness in that it cannot learn nonlinearly separable functions (Langley & Sage, 1994; Rachlin, Kasif, Salzberg & Aha, 1994). In addition, it cannot learn with 100% accuracy some *m-of-n* concepts (Kohavi, 1995). Second, the naive Bayesian classifier and PEBLS make independence assumptions that are violated by some data sets. When attributes are treated individually, joint probabilities are computed by taking the product of individual probabilities estimated from data. When two or more attributes are replaced by the Cartesian product of the attributes, the joint probability is estimated from the training data.

We call the constructive induction algorithm, BSEJ, for Backward Sequential Elimination and Joining by analogy with the backward sequential elimination algorithm for attribute selection. In the remainder of this paper, we first report on experiments using BSEJ on both naturally occurring and artificial datasets with three learning algorithms and explain why it is effective with some algorithms but not others. Finally, we compare BSEJ to related work and elaborate on several design choices made in BSEJ:

- Creating Cartesian product attributes rather than deleting attributes.
- Searching backward rather than forward.

3 Experiments

To investigate the effects of BSEJ on the three learning algorithms we will use 10 databases from the UCI Repository of machine learning databases and two artificial concepts. The artificial concepts used are exclusive-or with two relevant attributes and 6 irrelevant attributes and parity with 4 relevant attributes and 4 irrelevant attributes. Both artificial concepts are cases where there are dependencies among attributes. The parity concept should be more difficult for all algorithms to deal with since four attributes are involved in the interaction rather than just two in exclusive-or. The experiments on the naturally occurring databases are included to determine whether the problems that BSEJ is intended to address occur in practice as well as in artificial concepts.

The domains we used are summarized in Table 2. In each experiment, we ran 24 trials of randomly selecting training examples, running the learning algorithm with and without the BSEJ algorithm for constructive induction, and evaluating the accuracy of the algorithm on a test set consisting of all examples

of those attributes is used in the example representation. There is not a closed form for the number of partitionings but for a fixed number of blocks, this is a Stirling number of the second kind. In contrast, the hill-climbing algorithm considers at most $O(n^3)$ Cartesian product attributes where n is the number of attributes.

Domain	Training Examples	Attributes
glass	150	9
krkp	300	36
mushroom	800	22
voting	300	16
credit	250	15
horse-colic	200	21
iris	100	4
pima-diabetes	500	8
wine	125	13
wisc-cancer	500	9
xor	128	8
4-Parity	128	8

Table 2: Summary of domains used in the experiments.

not in the training set. We will use a paired, two-tailed t-test at least at the .05 level to determine whether BSEJ has a significant effect on the accuracy of each learning algorithm.

3.1 Naive Bayesian Classifier

The Bayesian classifier (Duda & Hart, 1973) is a probabilistic method for classification. It can be used to determine the probability that an example j belongs to class C_i given values of attributes of an example represented as a set of n nominally-valued attribute-value pairs of the form $A_1 = V_{1_j}$:

$$P(C_i|A_1 = V_{1_j} \& \ldots \& A_n = V_{n_j})$$

If the attributes are independent, this probability is proportional to:

$$P(C_i) \prod_k P(A_k = V_{k_j}|C_i) \qquad (1)$$

This formula is well suited for learning from data, since the probabilities $\hat{P}(C_i)$ and $\hat{P}(A_k = V_{k_j}|C_i)$ may be estimated from the training data. To determine the most likely class of a test example, the probability of each class is computed. A classifier created in this manner is sometimes called a simple (Langley, 1993) or naive (Kononenko, 1990) Bayesian classifier.

Domain	Naive	BSEJ
xor	.461	1.00 +
4-parity	.424	.475 +
glass	.417	.768 +
krkp	.868	.931 +
mushroom	.940	.992 +
voting	.904	.920 +
credit	.840	.833
horse-colic	.810	.802
iris	.931	.938
pima	.755	.749
wine	.980	.975
wisc-cancer	.973	.971

Table 3: The accuracy of the naive Bayesian classifier with and without constructive induction. A "+" indicates that using BSEJ results in a significant increase in accuracy. There are no significant decreases in accuracy.

Table 3 shows the accuracy of the Bayesian classifier with and without the constructive induction algorithm.

The exclusive-or problem is a dramatic example of why adding Cartesian product attributes results in a significant increase in accuracy. For simplicity, in this discussion we will assume there are 2 relevant attributes, A_1 and A_2, and only 1 irrelevant attribute, A_3. Without constructive induction, the probability of class C_i is computed to be proportional to

$$\hat{P}(A_1 = V_{1_j}|C_i)\hat{P}(A_2 = V_{2_j}|C_i)\hat{P}(A_3 = V_{3_j}|C_i) \qquad (2)$$

However, on every trial the BSEJ algorithm forms the Cartesian product of A_1 and A_2 and the probability of class C_i is proportional to:[4]

$$\hat{P}(A_1 = V_{1_j} \& A_2 = V_{2_j}|C_i)\hat{P}(A_3 = V_{3_j}|C_i) \qquad (3)$$

A more accurate classifier can result from forming the Cartesian product of A_1 and A_2. For example, consider classifying an example

[4]While BSEJ also has the capability of deleting irrelevant attributes, this never occurs on the exclusive-or problem since it is possible to achieve 100% accuracy on the training set without deleting attributes.

in which A_1, A_2 and A_3 were equal to 1. In this case, $\hat{P}(A_1 = 1|C_i = T)$, $\hat{P}(A_2 = 1|C_i = T)$, $\hat{P}(A_3 = 1|C_i = T)$, $\hat{P}(A_1 = 1|C_i = F)$, $\hat{P}(A_2 = 1|C_i = F)$, $\hat{P}(A_3 = 1|C_i = F)$, $\hat{P}(C_i = T)$, and $\hat{P}(Class = F)$ would all be close to 0.5 if estimated from randomly selected examples, and the naive Bayesian classifier that did not use a Cartesian product attribute would perform at near chance levels. However, if [3] were used, the classifier would be accurate since $\hat{P}(A_1 = 1 \& A_2 = 1|C_i = T)$ would be 1 and $\hat{P}(A_1 = 1 \& A_2 = 1|C_i = F)$ would be 0.

Although in this simple example, Cartesian product attributes work out well, in practice, it may cause problems. If independent attributes are joined and the probabilities of joined attributes are estimated from training data, a less accurate classifier may result because the probability estimates of the joined attributes are less reliable than the estimates of individual attributes. There is a trade-off between inaccuracies caused by not having enough data to reliably estimate joint probabilities from the data and using more accurate estimates of individual probabilities, but incorrectly computing the joint probability as the product of dependent individual probabilities. The wrapper approach is one way of dealing with this trade-off since it only joins attributes when the joint probability is estimated accurately enough to improve the accuracy of the learned concept as measured by leave-one-out cross validation.

Forming Cartesian product attributes results in a significant increase in accuracy on four of the naturally occurring domains. This suggests that there are violations of the conditional independence assumption made by the naive Bayesian classifier in naturally occurring domains, and that BSEJ can detect and correct for violations of this assumption by changing the example representation.

The parity example illustrates a potential shortcoming of the hill-climbing approach to building Cartesian product attributes. Although constructive induction results in a small statistically significant increase in accuracy, it is not better than simply guessing randomly. In this case, a Cartesian product of four attributes is needed and none of the Cartesian products of two or three of these attributes appear beneficial to the evaluation metric. Although it would be possible to consider joining three (or more attributes) in one step, the computational complexity makes it impractical for most databases.

3.2 ID3

To conserve space, we will assume that the reader is familiar with decision tree learning algorithms such as ID3 (Quinlan, 1986). In order to minimize differences between algorithms in our experiments, the decision tree learner uses the exact same example representation as the naive Bayesian classifier (i.e., partitioning numeric attributes into discrete categories). Although there are better ways of dealing with numeric attribute values in decision trees (e.g., Quinlan, 1994; Dougherty, Kohavi & Sahami, 1995), we concentrate solely on the issue of constructive induction of Cartesian product attributes. Similarly, we will not use any of the more advanced decision tree options such as post-pruning or value subsetting (Quinlan, 1994). Therefore, the differences between algorithms that we report are the effects of the representation and search bias.

Table 4 shows the results of using constructive induction of Cartesian product attributes in the context of decision trees. These experiments are run in the same manner as the earlier experiment. A decision tree learner with constructive induction is significantly more accurate on only one naturally occurring problem (iris) and the two artificial problems.

A closer investigation of the significant difference on the iris data set revealed that the deletion of an attribute and not the creation of a Cartesian product attribute is responsible for the improvement. Section 4 demonstrates that this is not the case with the naive Bayesian classifier.

Domain	Tree	BSEJ
xor	.926	1.00 +
4-parity	.648	.971 +
glass	.786	.788
krkp	.961	.959
mushroom	.996	.996
voting	.930	.934
credit	.798	.806
horse-colic	.783	.780
iris	.912	.931 +
pima-diabetes	.709	.702
wine	.934	.926
wisc-cancer	.952	.951

Table 4: The effect of constructive induction of Cartesian product attributes on ID3.

Although Cartesian product attributes substantially improve the accuracy of the decision tree on exclusive-or and parity they do not help on any real problems encountered so far. Parity and exclusive-or are extreme cases of attribute interaction where no single attribute has information while combinations of certain attributes are very informative. Note that unlike the naive Bayesian classifier Cartesian product attributes do not change the expressiveness of the decision tree classifier. A test on a Cartesian product attribute in a decision tree is equivalent to a test on one of the attributes followed by a test on the other attribute on all branches. Unless individual attributes have no information gain, it would be capable of learning such a sequence of tests as well as having the flexibility to choose a different test under some branches. Furthermore, the change in the search bias caused by Cartesian product attributes could be achieved by allowing the decision tree to exploit look-ahead when building tests. Like Cartesian product attributes, look-ahead in decision trees has been shown to be beneficial on artificial problems, but it is not usually helpful and sometimes harmful on naturally occurring problems (Murthy & Salzberg, 1995). Cartesian product attributes require that continuous attributes be discretized, losing the ordering information that can be exploited by the decision tree learner. Therefore, we would not recommend using Cartesian product attributes with decision tree learners.

One final observation is worth noting on the results of the previous two experiments. On the six problems where constructive induction improved the accuracy of the naive Bayesian classifier, the naive Bayesian classifier is also significantly less accurate than a decision tree learner. Since the naive Bayesian classifier is the Bayes optimal classifier when attributes are conditionally independent given the class, this suggests that the reason the naive Bayesian classifier is not as accurate as the decision tree learner on these problems is the violation of the independence assumption.

3.3 PEBLS

PEBLS (Cost & Salzberg, 1993) is a nearest neighbor algorithm that makes use of a modification of the value difference metric, MVDM, (Stanfill & Waltz, 1986) for computing the distance between two examples. This distance between two examples is the sum of the value differences of all attributes of the examples. The value difference between two values V_{k_x} and V_{k_y} of attribute A_k is given by Equation 4:

$$\delta(V_{k_x}, V_{k_y}) = \sum_i |\hat{P}(C_i|A_k = V_{k_x}) - \hat{P}(C_i|A_k = V_{k_y})| \quad (4)$$

PEBLS assigns a test example to the class of the training example that has the minimum distance to the test example as measured by the value difference metric. PEBLS makes an independence assumption and its accuracy degrades on problems in which this assumption is violated (Rachlin, Kasif, Salzberg & Aha, 1994). In Table 5, we report on experiments run in the same manner as the previous experiments.

Equation 4 is designed to be an estimate of the relevance of an attribute toward making a classification. Difference in values of relevant attributes greatly affect the distance measured by the similarity metric, while dif-

Domain	PEBLS	BSEJ
xor	.740	.999 +
4-parity	.475	.989 +
glass	.821	.816
krkp	.941	.946
mushroom	.999	.999
voting	.882	.946 +
credit	.805	.803
horse-colic	.451	.710 +
iris	.882	.887
pima-diabetes	.479	.676 +
wine	.960	.956
wisc-cancer	.957	.960

Table 5: The effect of constructive induction of Cartesian product attributes on PEBLS.

ference in values of irrelevant attributes do not. However, without constructive induction of Cartesian product attributes, it considers the relevance of an attribute in isolation. A constructed Cartesian product attribute allows the relevance of a combination of attributes to be determined.

The nature of the independence assumption in PEBLS and the naive Bayesian classifier is quite different. In the Bayesian classifier, joint probabilities are computed as the product of individual probabilities. In PEBLS, the overall distance is the sum of the individual attribute distances which are a function of the conditional probability of the class given the attribute. Nonetheless, Cartesian product attributes are beneficial since they change how the distance is computed. For example, on the exclusive-or problem discussed earlier, as the number of examples increases, PEBLS without BSEJ will assign a distance of 0 between all examples, since for all attributes and values will be the probability of a class given an attribute value will be 0.5 (Rachlin, Kasif, Salzberg & Aha, 1994). However, if a Cartesian product attribute is formed from the two relevant attributes, a test example will be considered most similar to those training examples that share the two relevant attributes. For example, consider classifying an example in which the relevant attributes A_1 and A_2 are 1. In this case, $|\hat{P}(C_i = T|A_1 = 1\&A_2 = 1) - \hat{P}(C_i = T|A_1 = 1\&A_2 = 1)|$ would be 0. In contrast, for other pairs of values of A_1 and A_2 such as 1 and 0 $|\hat{P}(C_i = T|A_1 = 1\&A_2 = 1) - \hat{P}(C_i = T|A_1 = 1\&A_2 = 0)|$ would be 1. Therefore, PEBLS will correctly classify exclusive-or examples if the two relevant attributes are joined. Inspection of traces running the algorithm and the near perfect accuracy show that the two relevant attributes are joined by BSEJ.

The accuracy improvement in PEBLS is not restricted to just artificial problems. On three of the naturally occurring databases, there is a significant and substantial increase in accuracy. For example, on the diabetes diagnosis problem, the accuracy increased from .479 to .676 by using the BSEJ algorithm.

The approach implemented in BSEJ to deciding which attributes to join is useful for two problems with Bayesian classifiers: representational inadequacies and conditional dependencies among variables. Since different learners have different representational biases, the attributes constructed for one may not be best for another. There is only one naturally occurring problem, congressional voting on which BSEJ significantly improves the accuracy of the Bayesian classifier and PEBLS. Next, we ran an experiment to determine whether the attributes found by BSEJ for the Bayesian classifier will improve the accuracy of PEBLS and whether the attributes found by BSEJ for the PEBLS will improve the accuracy of the Bayesian classifier. We ran 24 paired trials of each algorithm using 300 examples of congressional voting for training and the remainder of the examples for testing. In this case, the average accuracy of the naive Bayesian classifier was 90.1%, the accuracy of the Bayesian classifier using BSEJ was significantly higher at 92.6% (using a paired two-tailed t-test at the .05 level), and the accuracy of the Bayesian classifier using the attributes found by using BSEJ with PEBLS was 90.4%. This last figure does not significantly differ from the naive Bayesian classifier and is significantly less than using the attributes found specifically for the Bayesian classifier. The av-

erage accuracy of PEBLS classifier using the original attributes was 89.1%, the accuracy of PEBLS using BSEJ was significantly higher at 94.0%, and the accuracy of the PEBLS using the attributes found by using BSEJ with the Bayesian classifier was significantly less than the other two at 86.0%. Violations of independence assumptions and representational biases that are important to correct for in one learning algorithm are not necessarily important for another.

One might be tempted to explain the difference in performance between the algorithms by the different conditional probabilities used by each. In particular, Equation 1 of the Bayesian classifier uses $P(A_k = V_{k_j}|C_i)$ while Equation 4 of the similarity metric for neareset neighbor uses $P(C_i|A_k = V_{k_j})$. However, we should note that there is an alternative equivalent form of the Bayesian classifier, in which the probability of each class is proportional to Equation 5.

$$P(C_i)\prod_k \frac{P(C_i|A_k = V_{k_j})}{P(C_i)} \qquad (5)$$

Equation 5 can be derived from Equation 1 with several applications of Bayes Theorem and ignoring constant factors that do not affect the assignment to the most probable class.

3.4 Efficiently estimating error with leave-one-out cross-validation

We introduce an optimization that greatly speeds up the leave-one-out cross validation. The examples are reordered such that the examples misclassified by the classifier using the current partition are tested first. When calculating the error of a classifier with a new partition, we stop leave-one-testing as soon as it is certain that it will have at least as many errors as the current partition.

Moore and Lee (1994) propose a method for improving the efficiency of leave-one-out comparisons of several candidates by stopping the testing of an "unpromising" candidate when it is statistically unlikely to be more accurate than another candidate. This method is quite general and could be used to select among a heterogeneous set of learners (such as deciding whether to use a decision tree or a neural net on a given problem). They have also used this method in backward sequential elimination on a set of artificial problems and found that using this method required evaluating 67% of the examples evaluated by the full backward search.

Our method is best used with hill-climbing refinement methods such as BSEJ in which it is likely that most small changes to current hypothesis will not make it more accurate. On the mushroom problem with a Bayesian classifier this reordering results in testing only 4.7% of the examples required by the full search (averaged over 24 trials). On the krkp problem 9.6% of the evaluations of the full search were required. Because the Moore and Lee (1994) technique assumes errors are uniformly distributed, it cannot take advantage of example reordering. Furthermore, our method is an admissible heuristic that returns the same result as the full search. In contrast, the Moore and Lee method may eliminate attributes that would not have been eliminated by the full search and as a consequence sometimes actually does more search than the full method when it eliminates more attributes than the full method would.

4 Attribute deletion

Langley & Sage (1994) have used a Forward Sequential Selection (FSS) method for eliminating attributes from Bayesian classifiers. They advocate the use of attribute selection to deal with dependencies in Bayesian classifier. Another alternative to eliminating attributes is Backwards Sequential Elimination (BSE) (Kittler, 1986) In the next experiment, we compare BSEJ to FSS and BSE to see if there is an advantage in using constructive induction to create attributes. In addition, we

Domain	BSEJ	BSE	FSS	FSSJ
xor	1.00	.450 -	.481 -	.789 -
4-Parity	.475	.442 -	.471	.471
glass	.768	.706 -	.737 -	.765
krkp	.931	.862 -	.739 -	.793 -
mushroom	.992	.952 -	.984 -	.984 -
voting	.920	.902 -	.956 +	.949 +
credit	.833	.847	.830	.836
horse-colic	.802	.806	.811	.809
iris	.938	.939	.941	.942
pima	.749	.756	.741	.740
wine	.975	.980	.957 -	.954 -
wisc-cancer	.971	.972	.956	.959

Table 6: A comparison of the subset selection methods (without joining) and the constructive induction selection methods with backward and forward search. A "+" or "-" indicates that the method is significantly more or less accurate than BSEJ.

investigate a "forward" approach to constructive induction that we call Forward Sequential Selection and Joining (FSSJ). FSSJ initializes the set of attributes to be used by the classifier to the empty set. Next, two operators are used to generate new example representations until no improvement is found:

a. Consider adding each attribute not used by the current example representation.

b. Consider joining each attribute not used by the current example representation with each attribute currently used.

Table 6 shows the results of these algorithms run in the same manner as the previous experiments. For brevity, we present results only on the Bayesian classifier, but similar results were obtained with PEBLS. Table 6 indicates when one of the methods for improving upon the naive Bayesian classifier is significantly better than another at the .05 level by placing a "+" after the accuracy. None of the algorithms do particularly well on the parity problem, since they are limited by hill-climbing search. The BSEJ method is more accurate than BSE on 6 problems and there are no significant differences on the remainder. Note that the problems on which BSEJ is more accurate than BSE are the same problems in which a decision tree learner is more accurate than the naive Bayesian classifier, giving additional support for the hypotheses that the constructive induction operator is needed to achieve the increased accuracies. The comparison between FSS and BSEJ is similar. BSEJ is usually more accurate, or at least as accurate, except FSS is significantly more accurate on the congressional voting problem. In this problem there is a single very predictive attribute and forward methods have an advantage over backward when there are many attributes that must be eliminated. Although Langley and Sage advocate using the FSS method for dealing with attribute dependencies, the experimental results indicate that joining a pair of correlated attributes is more useful than ignoring one of the attributes. Like BSEJ, FSSJ has the capability of constructing Cartesian product attributes. However, it is more limited, since to use this operator, one of the attributes must be beneficial individually. Since this is not true on exclusive-or, FSSJ does not perform as well on this problem nor three naturally occurring problems.

Before leaving the topic of attribute selection, we should note that we also tried BSEJ on the standard nearest neighbor algorithm with the overlap metric. PEBLS (Cost & Salzberg, 1993) is usually more accurate that the standard nearest neighbor algorithm. BSEJ provided no additional benefits over attribute selection alone with BSE on standard nearest neighbor. Since the overlap metric weights all attributes equally, it does not suffer from the same problem as PEBLS which tries to weight relevant attributes heavily in similarity calculations.

5 Related Work

Kononenko (1991) implemented a method for identifying that attributes are conditionally

independent based upon a statistical test that determines the probability that two attributes are not independent. The algorithm joins two attributes if there is a greater than 0.5 probability that the attributes are not independent. Experimental results with this method were disappointing. On two domains, the modified Bayesian classifier had the same accuracy as the naive Bayesian classifier, and on the other two domains tested, the modified Bayesian classifier was one percent more accurate. It is not clear whether this difference is statistically significant.

Langley (1993) proposes the use of "recursive Bayesian classifiers" to address the nonlinearly separable problem of Bayesian classifiers. Although the algorithm worked on an artificial problem, it did not provide a significant benefit on any naturally occurring databases including ones on which BSEJ had considerable increases in accuracy.

Naive Bayesian classifiers are a special case of Bayesian network in which the class is viewed to be the only "cause" of the attribute values (e.g., a disease causes a patient to have particular symptoms). Figure 1 (left) shows an example of such a Bayesian network. Note that joining two attributes corresponds to the creation of a "hidden" variable as shown in Figure 1 (right). If classification is the only goal, the Bayesian classifier with Cartesian product attributes are a special case of a Bayesian network that has a less computationally intensive procedure for creating a network and a fast inference procedure.

Our work differs from most in constructive induction in two related ways. First, the constructed attributes are not Boolean or arithmetic combinations of existing attributes and the learning algorithms used are not decision trees or rule learners (e.g., Matheus & Rendell, 1989; Pagallo & Haussler, 1989; Ragavan & Rendell, 1993; Sanger, Sutton, & Matheus 1992). Cartesian product attributes were shown to have little value for decision trees, but they were important in improving the accuracy on naive Bayesian classifiers and PEBLS.

6 Conclusions

We have investigated the constructive induction of Cartesian product attributes. We proposed an algorithm based on the "wrapper model" for deciding which attributes to join. We showed that this approach was superior to forming Cartesian product attributes guided by an entropy-based measure of conditional independence. This occurs because BSEJ is sensitive to representational biases in addition to violations of independence assumptions. We showed that a backward search method is more appropriate than a forward for creating Cartesian product attributes, although like all hill-climbing algorithms it can have a problem with local minima. We used three artificial concepts to illustrate the types of problems for which BSEJ is beneficial and showed that BSEJ can result in substantial increases in accuracy on a variety of naturally occurring problems for two different learning algorithms.

7 Acknowledgments

The research reported here was supported in part by NSF grant IRI-9310413 and ARPA grant F49620-92-J-0430 monitored by AFOSR. I'd like to thank Pedro Domingos, Dennis Kibler, Ron Kohavi, Pat Langley and Kamal Ali for advice on Bayesian classifiers, Cullen Schaffer for advice on cross-validation, Steven Salzberg and David Aha for advice on instance-based learners and Catherine Blake, Subramani Mani, David Schulenburg and Clifford Brunk for comments on an earlier draft of this paper.

8 References

Aha, D. & Bankert, R. (1995). A Comparative evaluation of sequential feature selection algorithms. *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics.* Ft. Lauderdale, Fl.

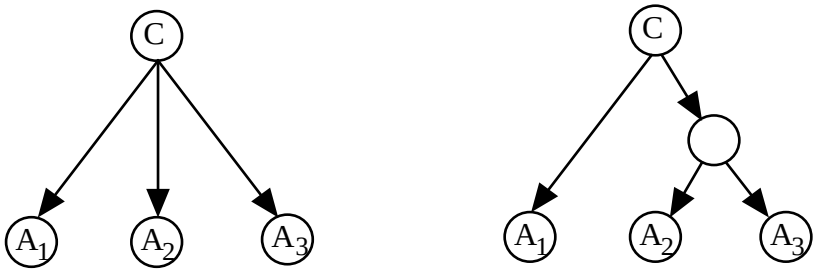

Figure 1: Left: A Bayesian classifier is a special case of a Bayesian network. Right: Joining attributes in a Bayesian classifier corresponds to introducing a hidden variable in a Bayesian network

Almuallim, H., and Dietterich, T. G. (1991). Learning with many irrelevant features. In *Ninth National Conference on Artificial Intelligence*, 547-552. MIT Press.

Caruana, R., & Freitag, D. (1994). Greedy attribute selection. In Cohen, W., and Hirsh, H., eds., *Machine Learning: Proceedings of the Eleventh International Conference*. Morgan Kaufmann

Cherkauer, K & Shavlik (1993). Protein Structure Prediction: Selecting Salient Features from large candidate pools. Internation Conference on Intelligent Systems in Molecular Biology. AAAI Press.

Cooper, G. & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9, 309-347.

Cost, S. & Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features *Machine Learning, 10*, 57-78.

Craven M.W., & Shavlik J.W. (1993). Learning to Represent Codons: A Challenge Problem for Constructive Induction, in Bajcsy R.(ed.), *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, San Mateo, CA, pp.1319-1324.

Danyluk, A. & Provost, F. (1993). Small disjuncts in action: Learning to diagnose errors in the telephone network local loop. *Machine Learning Conference*, pp 81–88.

Dougherty, J., Kohavi, J., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. *Machine Learning Conference*, pp 194–202

Duda, R. & Hart, P. (1973). *Pattern classification and scene analysis.* New York: John Wiley & Sons.

John, G. Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. *Proceedings of the Eleventh International Conference on Machine Learning.* New Brunswick, NJ.

Kittler, J. (1986). Feature selection and extraction. In Young & Fu, (eds.), *Handbook of pattern recognition and image processing.* New York: Academic Press.

Kohavi, R. (1995). *Wrappers for performance enhancement and oblivious decision graphs.* Ph.D. dissertation. Stanford University.

Kononenko, I. (1990). Comparison of inductive and naive Bayesian learning approaches to automatic knowledge acquisition. In B. Wielinga (Eds.), *Current trends in knowledge acquisition.* Amsterdam: IOS Press.

Kononenko, I. (1991). Semi-naive Bayesian classifier. *Proceedings of the Sixth European Working Session on Learning.* (pp. 206–219). Porto, Portugal: Pittman.

Kubat, M., Flotzinger., D., & Pfurtscheller, G. (1993). Discovering patterns in EEG signals: Comparative study of a few methods. *Proceedings of the 1993 European Conference on Machine Learning.* (pp. 367–371). Vienna: Springer-Verlag.

Langley, P. (1993). Induction of recursive Bayesian classifiers. *Proceedings of the 1993 European Conference on Machine Learning.* (pp. 153-164). Vienna: Springer-Verlag.

Langley, P. & Sage, S. (1994). Induction of selective Bayesian classifiers. *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence.* Seattle, WA

Matheus C.J., & Rendell L.A. (1989). Constructive Induction On Decision Trees, in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, Los Altos, CA, 645–650.

Moore, A. W., and Lee, M. S. 1994. Efficient algorithms for minimizing cross validation error. In Cohen, W. W., and Hirsh, H., eds., Machine Learning: *Proceedings of the Eleventh International Conference.* Morgan Kaufmann.

Murthy, S. & Salzberb, S. (1995). Lookahead and Pathology in Decision Tree Induction *Proceedings of the International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, Los Altos, CA, 1025–1031.

Pagallo G., & Haussler D. (1989). Two algorithms That Learn DNF by Discovering Relevant Features, in Segre A.M.(ed.), *Proceedings of the Sixth International Workshop on Machine Learning*, Morgan Kaufmann, Los Altos, CA, pp.119–123.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference.* San Mateo, CA: Morgan Kaufmann.

Provan, G. & Singh, M. (1995). Learning Bayesian networks using feature selection. *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics.* Ft. Lauderdale, Fl.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning, 1*, 81-106.

Rachlin, Kasif, Salzberg & Aha, (1994). Towards a better understanding of memory-based reasoning systems. *Proceedings of the Eleventh International Conference on Machine Learning.* New Brunswick, NJ.

Ragavan, H. & Rendell, L. (1993). Lookahead feature construction for learning hard concepts. *Machine Learning: Proceedings of the Tenth International Conference.* Morgan Kaufmann

Salzberg, S., Chandar, R., Ford, H., Murthy, S. & White, R. (1995). Decision Trees for Automated Identification of Cosmic Ray Hits in Hubble Space Telescope Images. *Publications of the Astronomical Society of the Pacific.*

Sanger T.D., Sutton R.S., & Matheus C.J.(1992). Iterative Construction of Sparse Polynomial Approximations, in Moody J.E., et al.(eds.), *Neural Information Processing Systems 4*, Morgan Kaufmann, San Mateo, CA, pp.1064–1071.

Schlimmer, J. (1987). Incremental adjustment of representations for learning. *Machine Learning: Proceedings of the Fourth International Workshop.* Morgan Kaufmann

Schaffer, C. (1994). A conservation law of generalization performance *Proceedings of the Eleventh International Conference on Machine Learning.* New Brunswick, NJ.

Spiegelhalter, D., Dawid, P., Lauritzen, S. and Cowell, R. (1993). Bayesian Analysis in Expert Systems. *Statistical Science, 8,* 219-283.

Stanfill, C. & Waltz, D. (1986). Towards memory-based reasoning. *Communications of the ACM, 29*, 1213-1228.

Skalak, D. (1994) Prototype and feature selection by sampling and random mutation hill climbing algorithms. *Proceedings of the Eleventh International Conference on Machine Learning.* New Brunswick, NJ.

Townsend-Webber, T. & Kibler, D. (1994). Instance-based prediction of continuous values. *AAAI Workshop on Case-based Reasoning.*

Wan, S. & Wong S. (1989). A mesaure for concept dissimilarity and its application in Machine Learning. In R. Janicki and W. Koczkodaj (eds.) *Computing and Information* 267–274.