In *Proceedings of the 14th IJCAI*, Morgan Kaufmann, 1064-1070, 1995.

Constructing Nominal X-of-N Attributes

Zijian Zheng
Basser Department of Computer Science
The University of Sydney, NSW 2006
Australia
(zijian@cs.su.oz.au)

Abstract

Most constructive induction researchers focus only on new boolean attributes. This paper reports a new constructive induction algorithm, called XofN, that constructs new nominal attributes in the form of X-of-N representations. An X-of-N is a set containing one or more attribute-value pairs. For a given instance, its value corresponds to the number of its attribute-value pairs that are true. The promising preliminary experimental results, on both artificial and real-world domains, show that constructing new nominal attributes in the form of X-of-N representations can significantly improve the performance of selective induction in terms of both higher prediction accuracy and lower theory complexity.

1 Introduction

A well-known elementary limitation of selective induction algorithms is that when task-supplied attributes are not adequate for describing hypotheses, their performance in terms of prediction accuracy and/or theory complexity is poor. To overcome this limitation, constructive induction algorithms [Michalski, 1978] transform the original instance space into a more adequate space by creating new attributes. By contrast to new attributes, the task-supplied attributes are called primitive attributes.

On real-world application domains, domain experts use three different types of attributes: binary, nominal, and continuous-valued attributes,[1] to describe their examples and concepts. Different types of attributes have different advantages. For example, boolean attributes are very simple; nominal and continuous-valued attributes are complex but more powerful for representing concepts. Note that attributes with more than two ordered discrete values can be specified as either nominal or continuous-valued attributes.

Most selective induction algorithms can accept attributes of these three kinds. However, many existing constructive induction algorithms such as FRINGE [Pagallo, 1990] and CITRE [Matheus and Rendell, 1989] construct new boolean attributes only by using logical operators such as $\wedge$, $\neg$, and $\vee$. ID2-of-3 [Murphy and Pazzani, 1991] creates, as new attributes, M-of-N representations stating whether at least M of N conditions are true. M-of-N representations are also boolean attributes.

A few systems such as BACON [Langley *et al.*, 1987] and INDUCE [Michalski, 1978] explore methods to construct new continuous-valued attributes using mathematical operators such as multiplication and division. Systems such as LMDT [Brodley and Utgoff, 1992] and Swap1 [Indurkhya and Weiss, 1991] construct linear discriminant functions as new attributes. To the best of the author's knowledge, few researchers have developed constructive induction systems that construct new nominal attributes. One exception is that INDUCE, AQ17-DCI, and AQ17-MCI construct *attribute counting attributes* for rule learning [Michalski, 1978; Bloedorn *et al.*, 1993]. They have ordered discrete values, but are used more like continuous-valued attributes rather than nominal attributes (see section 6 for the differences from X-of-N). Subsetting used by C4.5 groups discrete values of a single primitive nominal attribute to form a new test [Quinlan, 1993]. It can be thought as a method of constructing new nominal attributes. The author knows of no other decision tree algorithm that constructs new nominal attributes.

In this paper, we propose a new constructive induction algorithm that constructs nominal attributes in the form of X-of-N representations. Our implemented system, called XofN, uses decision trees as its theory description language. However, the idea of constructing new nominal attributes in the form of X-of-N representations is not limited to decision tree learning. It is not difficult to extend the idea to rule learning.

The following section describes X-of-N representations. Section 3 presents the approach to constructing new nominal attributes in the form of X-of-N representations and the constructive induction algorithm XofN. Section 4 reports experimental results of XofN on several artificial and real-world domains. Section 5 further discusses the X-of-N representations and the XofN algorithm. Section 6 discusses related work, and finally, the paper concludes with future work.

[1] Some more sophisticated attributes such as structured attributes may be used, but here we talk about only these three most commonly used types of attributes.

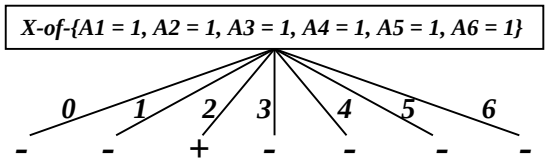

Figure 1: A tree on the Monks2 domain

2 X-of-N Representation

In short, an X-of-N is a set containing one or more attribute-value pairs. For a given instance, the value of an X-of-N representation corresponds to the number of its attribute-value pairs that are true.

Definition 1 ***The X-of-N representation***

Let $\{A_i \mid 1 \leq i \leq MaxAtt\}$ *be the set of attributes of a domain, and for each* A_i, $\{V_{ij} \mid 1 \leq j \leq MaxAttVal_i\}$ *be its value set,*[2] *where MaxAtt is the number of attributes, and* $MaxAttVal_i$ *is the number of different values of* A_i.

An X-of-N representation is a set, denoted as:

X-of-$\{AV_k \mid AV_k$ *is an attribute-value pair denoted as "*$A_i = V_{ij}$*",* $1 \leq k \leq N_+, N \leq N_+, 1 \leq N \leq MaxAtt\}$, *where* N_+ *is the number of attribute-value pairs in the X-of-N representation, called the size of the X-of-N representation, and N is the number of different attributes that appear in the X-of-N representation. The value of an X-of-N can be any number between 0 and N.*

Given an instance, its value is X iff X of the AV_k *are true. An attribute-value pair* AV_k *(*$A_i = V_{ij}$*) is true for an instance iff attribute* A_i *of the instance has value* V_{ij}.

Now, let us see an example of the X-of-N representation. The target concept of the Monks2 problem [Thrun *et al.*, 1991] is "*exactly two of the six attributes have their first value*". It can be represented using a tree as in Figure 1. The test at the root is a nominal X-of-N representation. Given instances ⟨3, 2, 1, 2, 1, 2; +⟩ and ⟨2, 1, 1, 1, 1, 1; -⟩, its values are 2 and 5 respectively.

As a nominal attribute, the X-of-N representation has one main advantage over conjunction, disjunction, and M-of-N representations: stronger expressive power without expanding the search space.

Many constructive induction algorithms such as FRINGE, CITRE, and CI [Zheng, 1992] use $\wedge$ and $\neg$ as constructive operators. They can only indirectly represent disjunctive concepts by using the negation of a conjunction. This makes it harder to construct new attributes in the form of disjunctions, especially on domains with primitive nominal attributes. Some algorithms also have $\vee$ as a constructive operator that makes it easier to create disjunctions. However, there are still some other concepts such as parity concepts, at-least, exactly, at-most M-of-N concepts, and their possible combinations, that cannot be effectively represented. ID2-of-3 and MoN [Ting, 1994] can only create at-least M-of-N representations. From the definition, we can see that X-of-N can directly represent all the following types of concepts.

- at-least M-of-N (as X-of-N in {M, M+1, ..., N}[3]),
- exactly M-of-N (as X-of-N = M),
- at-most M-of-N (as X-of-N in {0, 1, ..., M}),
- parity (for even parity, as X-of-N in {0, 2, 4, ...}),
- conjunction with internal disjunction (as X-of-N = N),
- disjunction with internal disjunction (as X-of-N in {1, 2, ..., N}), and
- possible combinations of the above six types of concepts.

Compared with M-of-N representations, the search space of X-of-N representations is smaller. This is because to create an M-of-N representation, we must first search for N_+ attribute-value pairs and then search for a value M. When the number N is large, searching for a good value M can be expensive. However, to create an X-of-N representation, we need only search for N_+ attribute-value pairs. X-of-N representations have as large a search space as conjunction and disjunction representations.

One disadvantage of X-of-N representations is a "fragmentation" problem. When X-of-N representations with large Ns are used as tests for decision trees, they quickly split training sets into a large number of small subsets. This makes it harder to find good tests for the subtrees. However, this situation occurs only when training sets are relatively small and target concepts are very complex in the sense that a few X-of-N representations are not enough. If target concepts are not complex or huge data sets are available, it is not a problem.

3 Constructing New Nominal Attributes for Decision Trees

Now, let us describe how to create and use X-of-N representations in constructive induction. XofN uses the well-known decision tree learning algorithm C4.5 [Quinlan, 1993] as its selective induction component. It consists of a single process. As shown in Table 1, XofN builds a decision tree by constructing, at each decision node, one new attribute based on primitive attributes using the local training set. If the new attribute constructed at a node is better than all primitive attributes and previously created new attributes, XofN uses it as the test for the node; otherwise discards it and uses the best of the primitive attributes and previously constructed new attributes. After growing a tree, XofN applies the pruning mechanism used by C4.5 [Quinlan, 1993].

Decision trees built by conventional tree learning algorithms such as C4.5 use a test based on one attribute at each decision node. They are called univariate trees [Brodley and Utgoff, 1992]. By contrast, XofN creates multivariate trees in which tests can refer to multiple attributes. We call them X-of-N trees as X-of-N representations are used as multivariate tests. At each decision node, besides creating one new X-of-N attribute, XofN considers reusing X-of-N attributes constructed previously for other decision nodes.

Function "Construct-X-of-N( )" in XofN constructs the best X-of-N representation using the local training set

[2] At the moment, the X-of-N representation is defined on binary and nominal attributes. Continuous-valued attributes are transformed into binary or nominal attributes by discretization.

[3] This means that X-of-N equals M, or M+1, ..., or N.

XofN-Tree($A_{primitive}$, A_{active}, $D_{training}$, C)
Input: $A_{primitive}$: a set of primitive attributes,
A_{active}: a set of primitive and X-of-N attributes for creating a test for the current decision node, initialized as $A_{primitive}$,
$D_{training}$: a set of training examples represented using A_{active},
C: majority class at the parent node of the current node, initialized as the majority class in the whole training set.
Output: a decision tree,
A_{active}: modified by adding one new X-of-N attribute constructed at this point.

IF ($D_{training}$ is empty)
THEN RETURN a leaf node labeled with C
ELSE
{ C := the majority class in $D_{training}$
IF (all examples in $D_{training}$ have the same class C)
THEN RETURN a leaf node labeled with C
ELSE
{ A_{best} := Best-Attribute(A_{active}, $D_{training}$)
A_{new} := **Construct-X-of-N**($A_{primitive}$, $D_{training}$)
IF (A_{new} is better than A_{best})
THEN
{ $A_{active} := A_{active} \cup \{A_{new}\}$
$A_{best} := A_{new}$
Rewrite $D_{training}$ using A_{active}
}
ELSE Dispose of A_{new}

Use A_{best} to partition $D_{training}$ into n subsets D_1, D_2, ..., D_n, one for each outcome of the test formed using A_{best}

RETURN the tree formed by a decision node with test A_{best} and subtrees
XofN-Tree($A_{primitive}$, A_{active}, D_1, C),
XofN-Tree($A_{primitive}$, A_{active}, D_2, C),
... ,
XofN-Tree($A_{primitive}$, A_{active}, D_n, C)
}
}

Table 1: Kernel of the XofN algorithm

at a decision node. It performs a simple greedy search in the instance space defined by primitive attributes. The starting point of the search is an empty X-of-N attribute. At each search step, it applies one of two operators: adding one possible attribute-value pair or deleting one possible attribute-value pair. To make the search efficient, if possible,[4] the deleting operator is applied first. During the search, XofN creates and keeps the best X-of-N representation for each possible size. Finally, function "Construct-X-of-N( )" returns the best of the X-of-N representations retained.

The information gain ratio[5] is used as the evaluation function for comparing and selecting new attributes. To avoid creating very complex new attributes that might over-fit the training data, another criterion is added, based on the coding cost of new attributes. By complex, we mean that an X-of-N representation has a large number of attribute-value pairs.[6] This kind of new attribute splits the training set into many subsets, so over-fitting is likely to occur. We use a similar coding method described in [Quinlan and Rivest, 1989]. The coding cost of a new attribute includes two parts. One is for coding the new attribute itself. The other is for coding the exceptions when applying the new attribute as a classifier to the local training data at the current decision node. There is no weight added to either part. The newly constructed X-of-N representation (new-X-of-N) will replace the current best X-of-N (best-X-of-N) to become the best one only if the following condition is true.

(Gain-ratio(new-X-of-N) > Gain-ratio(best-X-of-N) ∧
Coding-cost(new-X-of-N) ≤ Coding-cost(best-X-of-N)) ∨
(Gain-ratio(new-X-of-N) = Gain-ratio(best-X-of-N) ∧
Coding-cost(new-X-of-N) < Coding-cost(best-X-of-N))

With this condition, the algorithm accepts a new attribute with a higher gain ratio if its coding cost is not higher. The algorithm also accepts a new attribute with the same gain ratio but with a lower coding cost.

The stopping criterion for searching X-of-N representations is that the maximum possible size of X-of-N representations is reached. To reduce the search time, another restriction is applied: if no better new attribute has been found in five consecutive search steps,[7] the algorithm terminates.

From the definition of X-of-N representations, it may seem that X-of-N representations can only be directly constructed from binary and nominal attributes. To deal with continuous-valued attributes, the XofN algorithm has a preprocessor that discretizes primitive continuous-valued attributes. In the current implementation, we use a very simple method, although some better, but more complex, methods could be used. When there are some primitive continuous-valued attributes, XofN runs C4.5 on primitive attributes once and gets cut points for continuous-valued attributes. Then it discretizes the continuous-valued attributes using the cut points. The new attribute construction is carried out on the discretized attributes, binary attributes, and nominal attributes.

4 Experiments

The most commonly used measures of the behavior of learning systems are prediction accuracy of learned theories on test examples, theory complexity, and time complexity. Theory complexity (abbreviated to complexity later in this paper) is the size of a learned theory. For a rule set, it is the number of all conditions in the rule set. For a tree, it is the sum of sizes of all nodes of the tree. The size of a leaf is 1. The size of a decision node is 1 for a univariate tree, and is the number of attribute-value pairs in the test of the node for a multivariate tree such

[4] If the size of an X-of-N representation is less than or equal to two, the deleting operator cannot be applied, because the best X-of-N of size one has already been found.

[5] As used by C4.5.

[6] We talk about the situation that most primitive attributes in the X-of-N representation are different.

[7] This default setting is arbitrary, but it can be changed. There is no special reason for selecting "five".

Domain	Size	No. of Attributes				No. of Classes
		B	N	C	T	
Heart Disease	303	0	0	13	13	2
Hepatitis	155	13	0	6	19	2
Liver Disorders	345	0	0	6	6	2
Diabetes	768	0	0	8	8	2
Breast Cancer	699	0	0	9	9	2
Promoters	106	0	57	0	57	2
Phoneme[8]	5438	0	7	0	7	52
Stress[9]	5438	0	7	0	7	5
Letter[10]	5438	0	7	0	7	163
Tic-Tac-Toe	958	0	9	0	9	2

Table 2: Description of real-world domains

as a tree built by XofN, ID2-of-3, or CI. Time complexity is the execution time needed by a learning algorithm. We focus here on the first two measures, accuracy and theory complexity, because time results of other algorithms are not available or different computers and/or programming languages are used. As far as the time complexity is concerned, XofN is much slower than the selective tree learning algorithm C4.5, but is still acceptable. For example, on a DEC AXP 3000/500 workstation, the cpu time for XofN on the Cleveland Heart Disease domain (13 continuous-valued attributes, 2 classes, 303 cases) is 14.8 seconds, while it is 0.2 seconds for C4.5. On the Nettalk (Phoneme) domain (7 nominal attributes with 27 different values, 52 classes, 5438 cases), it is 9256.5 seconds for XofN and 4.3 seconds for C4.5.

To evaluate experimentally the effects of constructing new nominal attributes in the form of X-of-N representations, the results of XofN on a set of artificial and real-world domains are given. For comparison, we also give the results of some other constructive induction algorithms: AQ17-HCI, AQ17-DCI, AQ17-MCI, ID2-of-3, and our algorithm CI [Zheng, 1992]. CI creates new attributes for decision trees by using conjunctions of conditions from production rules generated by C4.5rules [Quinlan, 1993]. The default option setting of CI is used here. To create a new attribute, CI chooses two conditions, which are near the root of a tree, from a rule. With this setting, CI constructs new attributes based on two primitive attributes, and identifies relevant attributes [Zheng, 1992]. The results of C4.5 are given for reference.

4.1 Experimental domains and methods

Three Monks domains [Thrun *et al.*, 1991] are chosen because they are well-studied. There are published results for more than 25 different learning algorithms on them. They represent three different types of learning tasks with two binary and four nominal attributes. To make the Monks2 problem harder, especially for simple M-of-N learning methods, [Bloedorn *et al.*, 1993] creates the "Noisy and Irrelevant Monks2" problem by adding 5% random classification noise (by inverting the classes) in the training set, and adding seven random five-value irrelevant attributes in both the training and test sets. There is no classification noise in the test set. On all these four domains, the fixed training set (a subset of the whole universe) and test set (the whole universe) are given by the problem designers. In our experiments, we follow this methodology and run experiments once on the given training set and test set for each domain.

In addition, ten real-world domains are used, on which M-of-N like concepts are expected to be found [Spackman, 1988]. They are five medical domains (Cleveland Heart Disease, Hepatitis, Liver Disorders, Pima Indians Diabetes, and Wisconsin Breast Cancer), one molecular biology domain (Promoters), three linguistics domains (Nettalk(Phoneme), Nettalk(Stress), and Nettalk(Letter)), and one game domain (Tic-Tac-Toe). All are from the UCI repository of machine learning databases [Murphy and Aha, 1994]. Table 2 gives a brief summary of the domains, including the data set size, the number of binary (B), nominal (N), continuous-valued (C), total (T) attributes, and the number of classes.

Phoneme and Stress are two basic subproblems of the Nettalk domain. Letter is a combination of them. Their tasks are mapping a letter in an English word into a phoneme, a stress, and a phoneme-stress pair respectively. Following the method used in [Dietterich *et al.*, 1990] we generate data sets by using a window of length 7, but use the 1000 most common English words.

On each real-world domain, a 10-fold cross-validation [Breiman *et al.*, 1984] is conducted. In all the experiments reported here, C4.5, CI, ID2-of-3, and XofN are run with their default option settings, and are run on the same partitions for all the domains. No parameter-tuning is done here. Pruned trees are used for all the four algorithms.

4.2 Experimental results

This subsection compares the performance of the XofN algorithm with AQ17-HCI, AQ17-DCI, AQ17-MCI, CI, ID2-of-3, and C4.5.

The Monks domains

Table 3 summarizes the accuracy (Acc) and complexity (Com) of C4.5, AQ17-DCI, AQ17-HCI, CI, ID2-of-3, and XofN on three monks problems. The results of AQ17-DCI and AQ17-HCI are from [Thrun *et al.*, 1991]. The Table shows that only XofN solves all three problems with correct representations. As we expected, XofN finds a perfect representation of the target concept on Monks2.

To explore how noise and irrelevant attributes affect the performance of learning algorithms on the Monks2 domain, we give our results of C4.5, CI, ID2-of-3, and XofN, and the results of AQ17-DCI, AQ17-HCI, and AQ17-MCI from [Bloedorn *et al.*, 1993] in Table 4. Only XofN learns the correct concept. Because of noise, the learned concept is not the perfect representation. Instead, XofN finds two new attributes X-of-$\{A_4 = 1\}$ and X-of-$\{A_1 = 1, A_2 = 1, A_3 = 1, A_5 = 1,\ A_6 = 1\}$. However, it is still the most concise representation among those learned by these algorithms except for C4.5 and

[8] There are fifty four English phonemes, but phonemes for "Word Boundary" and "Period" do not appear in the dataset.

[9] There are six English stresses, but one for "Word Boundary" does not appear in the dataset.

[10] The number of all possible phoneme-stress pairs that appear in the dictionary of 20008 words is 163.

Algorithm	Monks1		Monks2		Monks3	
	Acc	Com	Acc	Com	Acc	Com
C4.5	75.7	18	65.0	31	97.2	12
AQ17-DCI	100	NA	100	NA	94.2	NA
AQ17-HCI	100	NA	93.1	NA	100	NA
CI	100	14	67.1	22	95.8	14
ID2-of-3	100	18	98.1	24	97.2	21
XofN	100	17	100	13	100	9

Table 3: Accuracy (%) and Complexity

Algorithm	Accuracy (%)	Complexity
C4.5	67.1	1
AQ17-DCI	81.5	122
AQ17-HCI	42.1	55
AQ17-MCI	90.2	23
CI	67.1	1
ID2-of-3	55.6	67
XofN	100	15

Table 4: On the Noisy and Irrelevant Monks2 domain

CI which return a tree having only one leaf. This illustrates that XofN can, to some extent, tolerate irrelevant attributes under conditions of noise, but this matter remains to be explored further.

The real-world domains

To demonstrate that the XofN algorithm can work well on real-world domains as well as artificial domains, Tables 5 and 6 give the performance of XofN on ten real-world domains. It uses only C4.5, CI, and ID2-of-3 as references, since very few directly comparable results of other constructive induction algorithms are available from publications. Each value given is the average of a ten-fold cross-validation. It is worth mentioning that LFC [Ragavan and Rendell, 1993] achieves higher prediction accuracy (78.8%) than XofN on Pima Indians Diabetes domain (see section 6).

To compare accuracies of XofN, ID2-of-3, CI, and C4.5, a two-tailed pairwise t-test is used. In Table 5, boldface indicates that CI, ID2-of-3, or XofN is better than C4.5 with the significance level of above 95%, while * and # denote that XofN is better than ID2-of-3 and CI respectively at the 95% significance level.

On seven out of ten domains, XofN achieves a significant improvement on prediction accuracy over C4.5. On the other three domains, the differences are not significant. The reason why XofN does not work well on these domains might be that there are no X-of-N representations in them, or there are some but XofN cannot find them due to the simple search strategy of the current implementation. Table 5 also shows that ID2-of-3 is significantly better than C4.5 on four domains, and there is no significant difference on the other six domains. CI is significantly better than C4.5 on six domains, significantly worse on the Nettalk(Letter) domain, and there is no significant difference on the other three domains. Compared to ID2-of-3, XofN is significantly better on four domains and there is no significant difference on the other six domains. Compared to CI, XofN is significantly better on six domains. The differences on the other domains are not significant.[11] On seven out of ten domains, XofN achieves the highest accuracy among the four algorithms. As far as the complexity is concerned, on most domains, XofN can learn a more concise tree than C4.5, especially on those domains where XofN achieves a higher accuracy. Compared to ID2-of-3, XofN learns smaller trees on seven out of ten domains.

Domain	C4.5	CI	ID2-of-3	XofN
Heart Disease	73.3	**77.8**	76.8	**79.8**
Hepatitis	78.2	**83.4**	77.0	79.4
Liver Disorders	62.1	62.4	63.2	*#**70.1**
Diabetes	71.5	**73.5**	69.2	70.8
Breast Cancer	94.8	95.8	94.4	94.9
Promoters	76.3	81.0	**87.6**	#**88.5**
Phoneme	81.1	**82.3**	**83.1**	#**83.9**
Stress	82.7	**83.8**	**86.2**	*#**87.6**
Letter	73.7	65.6	75.1	*#**76.9**
Tic-Tac-Toe	84.7	**94.2**	**94.9**	*#**98.4**

Table 5: Accuracy(%) on the real-world domains

Domain	C4.5	CI	ID2-of-3	XofN
Heart Disease	49.8	16.1	62.2	41.1
Hepatitis	13.6	10.1	24.6	13.4
Liver Disorders	79.4	28.1	108.9	89.1
Diabetes	128.8	41.6	191.4	153.6
Breast Cancer	20.6	15.0	37.3	26.3
Promoters	22.6	15.0	11.2	13.9
Phoneme	2339.2	1634.5	1188.4	1506.0
Stress	2077.3	1074.1	961.5	739.6
Letter	3394.9	1024.9	1654.8	2242.4
Tic-Tac-Toe	128.5	82.0	95.8	42.8

Table 6: Complexity on the real-world domains

5 Discussion

It has been found that XofN works quite well on a set of artificial and real-world domains. We expect that XofN can be applied to domains containing M-of-N concepts, such as biomedical domains [Spackman, 1988], linguistic domains, and domains containing parity concepts found for example in digital logic circuit design. To apply XofN to domains containing complex DNF concepts with long terms, some mechanisms are necessary to overcome the "fragmentation" problem. One approach is using subsetting of C4.5 [Quinlan, 1993]. Two other possible solutions are using subranging and building decision graphs [Oliver *et al.*, 1992] instead of decision trees. Subranging is similar to subsetting but also considers the order of the values of X-of-N representations. Subranging reduces the number of outcomes of an X-of-N test by combining some adjacent values of the X-of-N representation which are identical in terms of splitting data sets.

At the moment, XofN uses the cut points found by C4.5 to discretize primitive continuous-valued attributes.

[11] On the Diabetes domain, the significance level of the difference between CI and XofN is less than 95%, although the significance level of the difference between CI and C4.5 is above 95% and the average accuracy of C4.5 is higher than that of XofN. The reason is that the accuracy of XofN is higher than that of CI in some folds.

Other discretization methods that can be used are multi-interval discretization methods [Catlett, 1991; Fayyad and Irani, 1993], supervised/unsupervised methods [Van de Merckt, 1993], and an entropy method [Ragavan and Rendell, 1993]. The current XofN discretizes continuous-valued attributes statically in the sense that discretization occurs before new attribute construction. An alternative is dynamic discretization, i.e. doing discretization while constructing new attributes. This method might be able to create good discretizations but with an increased computational complexity.

Up to now, we have concentrated only on treating X-of-N representations as nominal attributes. Because their values are ordered, however, they can also be treated as continuous-valued attributes. Since the standard decision tree based learning algorithms transform continuous-valued attributes into binary tests, the continuous-valued X-of-N should work well when the target concept requires X-of-N representations with only one cut point (M-of-N concepts). However, on domains requiring X-of-N representations with more than one cut point, the continuous-valued X-of-N has weaker expressive power than the nominal X-of-N. For details on this issue, please see [Zheng, 1995].

6 Related Work

The closest related work is ID2-of-3 [Murphy and Pazzani, 1991]. It constructs new binary attributes in the form of M-of-N representations, while XofN constructs X-of-N representations. When building a decision tree, both ID2-of-3 and XofN construct one new attribute for each decision node using the local training set. Instead of building trees, MoN [Ting, 1994] creates M-of-N rules.

The production rule learning algorithms INDUCE [Michalski, 1978], AQ17-DCI, and AQ17-MCI [Bloedorn *et al.*, 1993] use the counting operator[12] #VarEQ(x) to construct new attributes that count the number of attributes in an instance which take the value x. For primitive boolean attributes, a boolean counting operator takes a vector of n boolean attributes ($n \geq 2$) and counts the number of true values for an instance. Like X-of-Ns, new attributes constructed by these two operators have ordered discrete values. However, when used to generate production rules, they are treated more like continuous-valued attributes than nominal attributes.[13] The boolean counting attribute is a special case of the #VarEQ(x) attribute, while the #VarEQ(x) attribute is a special case of the X-of-N representation.

Most hypothesis-driven constructive induction algorithms such as FRINGE [Pagallo, 1990], CITRE [Matheus and Rendell, 1989], CI [Zheng, 1992], and AQ17-HCI [Wnek and Michalski, 1994] construct and select a set of new attributes based on the entire training set. This strategy has a shortcoming: new attributes that have high values of the evaluation function for the entire training set might have lower values than other unselected new attributes for a training subset after a part of a decision tree or a rule set has been created [Matheus and Rendell, 1989]. To overcome this, XofN constructs one new attribute using the local training set for each decision node. Therefore, the new attribute constructed by XofN at each decision node is the best one in XofN's search space in terms of the evaluation function. Another difference between XofN and these algorithms is that the latter interleave the theory learning phase and the process of building new attributes, and generate new attributes by analyzing the theory learned previously, while XofN has only one iteration and constructs new attributes by analyzing data.

Like ID2-of-3 and XofN, LFC [Ragavan and Rendell, 1993] is also a data-driven constructive induction algorithm that builds multivariate trees, but it uses negation and conjunction as constructive operators. LFC creates one conjunction for each decision node by using a directed lookahead search. It achieved quite high prediction accuracy on a couple of real-world domains such as Pima Indians Diabetes, but the problem is that it has a sensitive parameter "Lookahead Depth" which needs to be set when applied to a domain. Another multivariate tree learning algorithm is LMDT [Brodley and Utgoff, 1992] that generates a linear machine at each decision node when building a tree.

7 Conclusions and Future Work

Selective induction algorithms build their theories by selecting primitive attributes and are thus limited for hard tasks whose primitive attributes are not sufficient for, or directly relevant to, representing the theories. To overcome this problem, constructive induction algorithms use various methods to generate powerful new attributes. However, most algorithms construct only binary attributes, while domain experts use binary, nominal, and continuous-valued attributes to describe their tasks. From this point, we first proposed a rich knowledge representation means, namely X-of-N representations, and then gave a new constructive induction algorithm XofN. It generates X-of-N representations as new nominal attributes for decision trees using a data-driven constructive strategy.

The experiments illustrate the learning power of the XofN algorithm on some artificial and real-world domains in terms of both higher prediction accuracy and lower theory complexity. In addition, it has been shown that XofN can achieve higher accuracy on some domains than other constructive induction algorithms that construct, as new attributes, conjunctions or M-of-N representations.

However, the current XofN is a preliminary implementation. A lot of research is worth doing and some is in progress. Examples are exploring more appropriate evaluation functions for comparing and selecting X-of-Ns, other search methods, other discretization methods for continuous-valued attributes, and implementing subranging and decision graph algorithms. In addition, examining XofN's ability to tolerate missing values, noise, and irrelevant attributes should be interesting.

[12] In INDUCE, it is called #v_COND.

[13] Generated rules have the form like (#VarEQ(1) >= 3) [Thrun *et al.*, 1991, p11].

Acknowledgments

This research was partially supported by an ARC grant (to J.R. Quinlan) and by a research agreement with Digital Equipment Corporation. The author is supported by EMSS scholarship. The author appreciates the advice and suggestions J.R. Quinlan has given. Many thanks to J.R. Quinlan for providing C4.5, P.M. Murphy for supplying the code of ID2-of-3, also to K.M. Ting, A. Varšek, N. Indurkhya, P. Brazdil, P. Langley, M. Cameron-Jones, and three anonymous reviewers for comments that improve the ideas and earlier drafts. Finally, P.M. Murphy and D. Aha are gratefully acknowledged for creating and managing the UCI Repository of ML databases.

References

[Bloedorn *et al.*, 1993] E. Bloedorn, R.S. Michalski, and J. Wnek, Multistrategy constructive induction: AQ17-MCI. *Proceedings of the Second International Workshop on Multistrategy Learning*, 188-203, 1993.

[Breiman *et al.*, 1984] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification And Regression Trees*, Belmont, CA: Wadsworth, 1984.

[Brodley and Utgoff, 1992] C.E. Brodley and P.E. Utgoff, Multivariate versus univariate decision trees. *COINS Technical Report 92-8, Department of Computer Science, University of Massachusetts, Amherst, Massachusetts, USA*, 1992.

[Catlett, 1991] J. Catlett, On changing continuous attributes into ordered discrete attributes. *Proceedings of the European Working Session on Learning*, 164-178, Springer Verlag, 1991.

[Dietterich *et al.*, 1990] T.G. Dietterich, H. Hild, and G. Bakiri, A comparative study of ID3 and backpropagation for English text-to-speech mapping. *Proceedings of the Seventh International Conference on Machine Learning*, 24-31, Morgan Kaufmann, 1990.

[Fayyad and Irani, 1993] U.M. Fayyad and K.B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning. *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1022-1027, Morgan Kaufmann, 1993.

[Indurkhya and Weiss, 1991] N. Indurkhya and S.M. Weiss, Iterative rule induction methods. *Journal of Applied Intelligence*, **1**, 43-54, 1991.

[Langley *et al.*, 1987] P. Langley, H.A. Simon, G.L. Bradshaw, and J.M. Zytkow, *Scientific Discovery: Computational Explorations of the Creative Processes*, Cambridge: MIT Press, 1987.

[Matheus and Rendell, 1989] C.J. Matheus and L.A. Rendell, Constructive induction on decision trees. *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, 645-650, 1989.

[Michalski, 1978] R.S. Michalski, Pattern recognition as knowledge-guided computer induction. *Technical Reports: 927, Department of Computer Science, The University of Illinois, Urbana,* 1978.

[Murphy and Aha, 1994] P.M. Murphy and D.W. Aha, *UCI Repository of machine learning databases* [Machine-readable data repository]. Irvine, CA: University of California, Department of Information and Computer Science, Available by anonymous ftp at ics.uci.edu in the pub/machine-learning-databases directory, 1994.

[Murphy and Pazzani, 1991] P.M. Murphy and M.J. Pazzani, ID2-of-3: Constructive induction of M-of-N concepts for discriminators in decision trees. *Proceedings of the Eighth International Workshop on Machine Learning*, 183-187, Morgan Kaufmann, 1991.

[Oliver *et al.*, 1992] J.J. Oliver, D.L. Dowe, and C.S. Wallace, Inferring decision graphs using the minimum message length principle. *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, 361-367, World Scientific Publisher, 1992.

[Pagallo, 1990] G. Pagallo, *Adaptive Decision Tree Algorithms for Learning from Examples,* Ph.D. thesis, University of California at Santa Cruz, 1990.

[Quinlan and Rivest, 1989] J. Ross. Quinlan and R.L. Rivest, Inferring decision trees using the minimum description length principle. *Information and Computation*, **80**, 227-248, 1989.

[Quinlan, 1993] J.R. Quinlan, *C4.5: Programs for Machine Learning*, San Meteo: Morgan Kaufmann, 1993.

[Ragavan and Rendell, 1993] H. Ragavan and L. Rendell, Lookahead feature construction for learning hard concepts. *Proceedings of the 10th International Conference on Machine Learning*, 252-259, 1993.

[Spackman, 1988] K.A. Spackman, Learning categorical decision criteria in biomedical domains. *Proceedings of the Fifth International Conference on Machine Learning*, 36-46, Morgan Kaufmann, 1988.

[Thrun *et al.*, 1991] S.B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jong, S. Džeroski, S.E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R.S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, W. Van de Welde, W. Wenzel, J. Wnek, and J. Zhang, The MONK's problems - a performance comparison of different learning algorithms. *Tech. Report: CMU-CD-91-197, Carnegie Mellon University*, 1991.

[Ting, 1994] K.M. Ting, An M-of-N rule induction algorithm and its application to DNA domain. *Proceedings of the 27th Annual Hawaii International Conference on System Sciences, Volume V: Biotechnology Computing*, 133-140, IEEE Computer Society Press, 1994.

[Van de Merckt, 1993] Thierry Van de Merckt, Decision trees in numerical attribute spaces. *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1016-1021, Morgan Kaufmann, 1993.

[Wnek and Michalski, 1994] J. Wnek and R.S. Michalski, Hypothesis-driven constructive induction in AQ17-HCI: a method and experiments. *Machine Learning*, **14:2**, 139-168, Kluwer Academic, 1994.

[Zheng, 1992] Z. Zheng, Constructing conjunctive tests for decision trees. *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, 355-360, World Scientific Publisher, 1992.

[Zheng, 1995] Z. Zheng, Continuous-valued X-of-N Attributes Versus Nominal X-of-N Attributes for Constructive Induction: A Case Study. *Proceedings of the Fourth International Conference for Young Computer Scientists*, Peking University Press, 1995.