Pergamon

Pattern Recognition, Vol. 27, No. 5, pp. 689–697, 1994
Elsevier Science Ltd
Copyright © 1994 Pattern Recognition Society
Printed in Great Britain. All rights reserved
0031-3203/94 $7.00 + .00

0031-3203(93)E0029-7

# LEARNING STRUCTURAL DESCRIPTIONS OF PATTERNS: A NEW TECHNIQUE FOR CONDITIONAL CLUSTERING AND RULE GENERATION

WALTER F. BISCHOF† and TERRY CAELLI‡§

† Department of Psychology, University of Alberta, Edmonton, Alberta, T6G 2E9, Canada
‡ Department of Computer Science, Curtin University of Technology, Box U 1987, Perth 6001, Western Australia, Australia

(*Received* 11 *March* 1993; *in revised form* 24 *November* 1993; *received for publication* 2 *December* 1993)

**Abstract**—A deterministic technique is developed for generating rules which can optimally classify patterns (for example, in 3D object recognition) in terms of the bounds on unary (single part) and binary (part relation) features which constitute different types of patterns. This technique, termed Conditional Rule Generation (CRG), was developed to take into account the label-compatibilities which should occur between unary and binary rules in their very generation, a condition which is generally not guaranteed in well-known rule generation and machine learning techniques.

Clustering  Machine learning  Pattern recognition  Rule generation

## 1. INTRODUCTION

Traditionally, pattern recognition has been concerned with two problems: pattern encoding and the generation of decision rules for pattern classification. Patterns are typically encoded as vectors of characteristic features which are chosen to optimize representational uniqueness of patterns belonging to different classes and to preserve uniqueness under specific feature transformations. Pattern classification is then achieved by partitioning feature space into regions associated with different pattern classes. Classification rules should minimize misclassification while, at the same time, maximizing the simplicity of feature space partitions in order to improve the probability of correct classification of new, unseen samples.

It goes without saying that traditional pattern recognition has been quite successful for simple isolated patterns. However, as pattern complexity increases, as is, for example, the case in 3D object recognition, traditional methods become increasingly unsuccessful. This can be attributed to a number of reasons. Descriptions of complex patterns in terms of features characterizing the whole pattern are often inadequate to encode the variability of class samples. Typically, patterns are best described as being composed of constituent parts and so pattern descriptions involve enumeration of both part (unary) features, and relationships between parts (binary features).(1) In 3D object recognition, for example, an object may be described by features characterizing surface parts (unary features) such as average curvatures or mean boundary length, and by features describing part relations (binary features), such as centroid distance or normal angle differences.

Mapping such structural descriptions onto simple feature vectors as is required in traditional pattern recognition approaches, leads to several problems. First, and foremost, is the label-compatibility problem: two patterns may be identical with respect to all unary and binary features (attributes) yet be structurally different, i.e. they differ with respect to the occurrence of specific *labeled* parts and their relationships. Second, and this is typical in 3D object recognition, patterns of different classes may share common feature states and, in general, there can be as much within-class sample variation as there is between classes, leading to poor class discrimination performance. Third, uniform treatment of pattern descriptors as feature-value tuples ignores the problem of representational adequacy and consistency for predicates of different arity.

Some of these problems have been dealt with in recent evidence-based recognition systems (EBS) that encode patterns as "rules" defined by region (volume) bounds in unary and binary feature spaces which are derived to optimally "evidence" different patterns or classes by "evidence" weights. Such weights are typically derived from the relative frequencies of different classes per region(2) or, more recently, by minimum entropy and neural network techniques.(3) In either case, the problem of generating rules subject to label-compatibility constraints was not considered. Indeed, the authors know of no technique, from the EBS perspective, which generate rules satisfying label-compatibility constraints.

The simplest representation for patterns which takes into account the label-compatibility of unary and bi-

§ Author to whom correspondence should be addressed.

nary features is a labeled graph. From this, standard graph matching techniques can then be used to solve the recognition problem. A sample pattern structure (for example, new data for classification) is matched to a model structure by searching for a label assignment that maximizes some objective similarity function. Pattern classes are represented by sets of instances and classification is thus achieved by searching through all model graphs to determine the best match (see references (4–7)).

In the following sections we focus on the development of a new technique, termed Conditional Rule Generation (CRG) which generates a tree of conditional rules for learning structural descriptions of patterns involving generalizations of training sample unary and binary attributes. The proposed approach can be characterized as follows:

(1) Rule conditions are generated as clusters in unary and binary feature spaces.

(2) Unification of pattern classes is achieved via conditional clustering of uniquely discriminable subgraphs.

(3) Deterministic classification rules are generated through controlled decision tree expansion and cluster refinement.

## 2. CONDITIONAL RULE GENERATION

### 2.1. *Cluster tree generation*

In the following, we present the conditional clustering technique, first in an informal way, and then we present the algorithm. Classification of a set of classes (in vision, 2D patterns or 3D objects) is learned in non-incremental batch mode. Each object is composed of a number of parts (pattern components). Each part $p_i$, $i = 1, \dots, N$ is described by a set of unary features $\mathbf{u}(p_i)$, and pairs of parts $(p_i, p_j)$ belonging to the same sample (but not necessarily all possible pairs) are described by a set of binary features $\mathbf{b}(p_i, p_j)$. Below, $S(p_i)$ denotes the sample (in 3D object recognition, a "view") a part $p_i$ belongs to, $C(p_i)$ denotes the class $S(p_i)$ belongs to, and $H(i)$ refers to the information, or cluster entropy statistic of cluster $i$

$$H(i) = -\sum_j q_{ij} \ln q_{ij}$$

where $q_{ij}$ defines the probability of elements of cluster $i$ belonging to class $j$.

We start constructing a unary feature space for all parts $U = \{\mathbf{u}(p_i), i = 1, \dots, N\}$ and cluster this feature space into clusters $U_i$. Clusters that are unique with respect to class membership (with entropy $H = 0$) provide a simple classification rule for some patterns (e.g. $U_3$ in Fig. 1). Each non-unique cluster $U_i$ is further analyzed with respect to binary features by constructing the (conditional) binary feature space $UB_i = \{\mathbf{b}(p_r, p_s) | \mathbf{u}(p_r) \in U_i \text{ and } S(p_r) = S(p_s)\}$. This feature space is clustered with respect to binary features into clusters $UB_{ij}$. Again, clusters that are unique with respect to class membership provide classification rules for some objects (e.g. $UB_{11}$ in Fig. 1). Each non-unique cluster $UB_{ij}$ is then analyzed with respect to unary features of the second part and the resulting feature space $UBU_{ij} =$

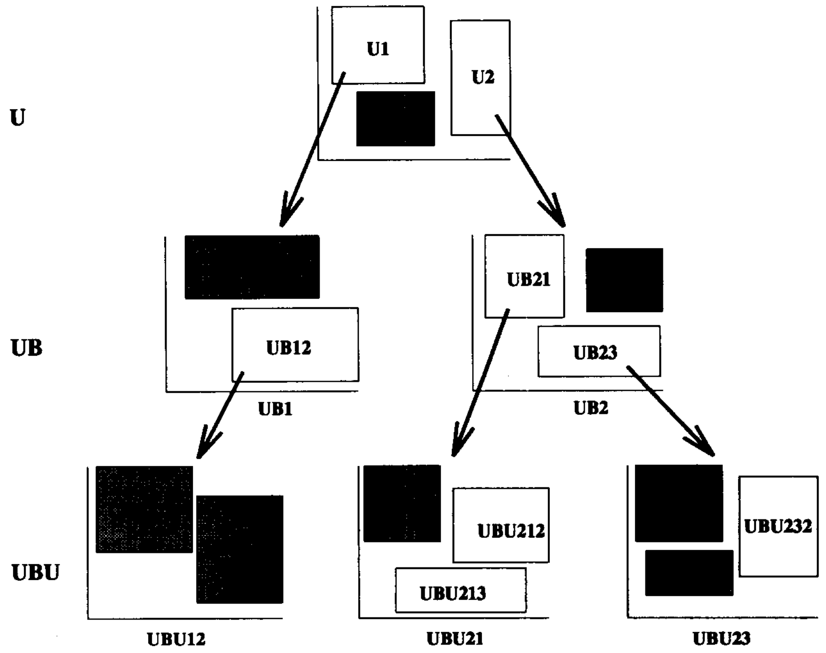

Fig. 1. The conditional cluster tree generated by CRG. The unresolved unary clusters ($U_1$ and $U_2$)—having more than one class represented in each cluster—are expanded to binary feature spaces $UB_1$ and $UB_2$. This process of clustering and expansion is continued until either all rules are resolved or the predetermined maximum rule length is reached, in which case rule splitting occurs.

$\{\mathbf{u}(p_s)|\mathbf{b}(p_r, p_s) \in UB_{ij}\}$ is clustered into clusters $UBU_{ijk}$. Again, unique clusters provide class classification rules for some objects (e.g. $UBU_{121}$ in Fig. 1), the other clusters have to be further analyzed, either by repeated conditional clustering involving additional parts at levels *UBUB*, *UBUBU*, etc. or through cluster refinement, as described later.

In this implementation of CRG, clustering in each feature space, both initially and for cluster refinement, is achieved by using a simple splitting-based procedure, similar to those used in decision trees.

The basic structure of the cluster-tree generation algorithm is shown in Table 1. Cluster trees are generated in a depth-first manner up to a maximum level of expansion and clusters that remain unresolved at that level are split in a way described in the following section. It should be noted that, in this implementation, we have assumed symmetry in conditional rule structures. That is, activation of *UBU* rules could involve any ordering of triples.

## 2.2. *Cluster refinement*

All non-unique (unresolved) clusters remaining at a given level of the cluster tree generation (e.g. clusters $UBU_{212}$, $UBU_{213}$, and $UBU_{232}$ in Fig. 1) have to be analyzed further to construct unique decision rules. One way of doing this is by further expanding the cluster tree, analyzing unary and binary attributes of additional parts to generate rules of the form "*UBUBUB*...". However, from the original clustering

Table 1. Cluster tree generation

```
level(root):= 0
push(root, queue)
while (queue not empty)
  $c$:= pop(queue)
  if level($c$) = maxlevel then
    split($c$)
  else
    $f$:= ConditionalFeatureSpace($c$)
    clist:= cluster($f$)
    foreach cluster $c' \in$ clist
      if unresolved ($c'$) then
        level($c'$):= level($c$) + 1
        push ($c'$, queue)
      endif
    endforeach
  endif
endwhile
ConditionalFeatureSpace($c$)
if $c$ = root then
  type($f$):= $U$
  elements($f$):= $\{\mathbf{u}(p_i), i = 1, \ldots, N\}$
else
  if type($c$) = $U$ then
    type($f$):= $B$
    elements($f$):= $\{\mathbf{b}(p,q)|\mathbf{u}(p) \in c$ and $(p,q)$ in a path$\}$
  else
    type($f$):= $U$
    elements($f$):= $\{\mathbf{u}(q)|\mathbf{b}(p,q) \in c\}$
  endif
endif
return ($f$)
```

ranges this may never give completely "resolved" branches in the cluster tree. Alternatively, the clusters in the tree can be refined or broken into smaller, more discriminating feature bounds or rules as described below. Both approaches have their respective disadvantages. Cluster refinement leads to an increasingly complex feature-space partitioning and thus may reduce the generality of classification rules. Cluster tree expansion, on the other hand, successively reduces the possibility of classifying objects from partial views or partial data. In the end, a compromise has to be established between both approaches.

In cluster refinement, two issues must be addressed, the method used for cluster refinement and the level at which cluster refinement is performed. Consider the cluster tree shown in Fig. 1 with non-unique clusters $UBU_{212}$, $UBU_{213}$ and $UBU_{232}$. One way to refine clusters (e.g. cluster $UBU_{232}$) is to recluster the associated feature space ($UBU_{23}$) into a larger number of clusters. However, classification rules associated with sibling clusters ($UBU_{231}$ and $UBU_{233}$) are lost and have to be recomputed. Alternatively, given that each cluster is bounded by a hyper-rectangle in feature space, refinement of a cluster can be achieved by splitting this rectangle along some optimal boundary. This ensures that sibling clusters remain unaffected.

Consider splitting the elements of an unresolved cluster $C$ along a (unary or binary) feature dimension $F$. The elements of $C$ are first sorted by their feature value $f(c)$, and then all possible cut points $T$ midway between successive feature values in the sorted sequence are evaluated. For each cut point $T$, the elements of $C$ are partitioned into two sets, $P_1 = \{c|f(c) \leq T\}$ with $n_1$ elements and $P_2 = \{c|f(c) > T\}$ with $n_2$ elements. We define the partition entropy $H_P(T)$ as

$$H_P(T) = n_1\, H(P_1) + n_2\, H(P_2).$$

The cut point $T_F$ that minimizes $H_P(T_F)$ is considered the best point for splitting cluster $C$ along feature dimension $F$ (see also reference (8)). The best split of cluster $C$ is considered the one along the feature dimension $F$ that minimizes $T_F$.

Rather than splitting an unresolved leaf cluster $C_L$ (e.g. $UBU_{232}$ in Fig. 1), one can split any cluster $C_i$ in the parent chain of $C_L$ (in this case, $UB_{23}$ or $U_2$ in Fig. 1). The cluster $C_i$ that minimizes $T_F$ is considered the optimal level for refining the cluster tree. Clusters above $C_L$ may contain elements of classes other than those that are unresolved in $C_L$. However, in computing $H_P$ for those clusters, we consider only elements of classes that are unresolved in $C_L$.

## 2.3. *Rule ordering*

Once the cluster tree has been completely resolved, the tree is reordered in such a way that the effort for sequential evaluation of rules associated with the leaf clusters is minimized. We let $n_i$ be a cluster conditional on a cluster $n$, and let $pa_i$ be the conditional probability prob(paths from $n_i$| paths in $n$). It should be remembered that such probabilities are determined from the fact that CRG may generate many rules for each class and that different rule paths can share common clusters in a given feature space but that the rule path is critical in defining a pattern. We then let $c_i$ be the cost of evaluating the rules under the cluster tree $n_i$. The cost of evaluating a leaf cluster is proportional to the number of feature dimensions, and the expected cost of evaluating $n$ is

$$c = c_1 + (1 - pa_1)c_2 + (1 - pa_1 - pa_2)c_3 + \cdots.$$

The expected cost $c$ is minimal if the clusters $n_i$ are sorted such that $c_i/pa_i \leq c_{i+1}/pa_{i+1}$. Note that this is strictly correct only for disjoint clusters. For non-disjoint clusters the cost $c$ is somewhat lower and more complex to evaluate, but this approximation is adequate given that most clusters in the conditional-cluster tree generation are disjoint. Disjoint clusters are guaranteed when splitting (refinement) alone is used as opposed to a combination of agglomeration clustering and splitting. Furthermore, in almost all cases CRG has produced more compact rules (in the sense described below) when splitting alone was used.

## 3. TESTS ON THE CRG TECHNIQUE

The CRG method is illustrated here with two different pattern classification problems of increasing complexity. The first example involves a small number of parts and objects and are used to illustrate different aspects of the CRG method. The second is more typical of 3D object recognition problems, involving many objects per class, a substantial intra-class variation of both unary and binary features, and each sample not incorporating all aspects of the class—analogous to an object view only encoding part of the object.

The first example, shown in Fig. 2, consists of four classes with four examples ("views") each. Each example consists of two parts (lines) and is described by the unary features "line length" and "orientation", and the binary features "distance between line centers" and "intersection angle". CRG was run on this example (without feature ordering) with maximum rule length (maxlevel in Fig. 3) set to 1 ($U$), 2 ($UB$) and 3 ($UBU$). The cluster trees produced for each condition are shown in Fig. 3. A summary of the rules produced for each condition is given in Table 2. The following measures are used to describe the rules: number of rules (number of leaves in the cluster tree), average number of paths per rule where "path" is used as described in Section 2.1, and the average feature space volume $V$ per rule where $V$ is an indicator for rule generality with $V = 0$ for rules covering a single case. As can be seen both from Fig. 3 and Table 2, bounds on features can be increased with increasing rule length, without affecting rule uniqueness.

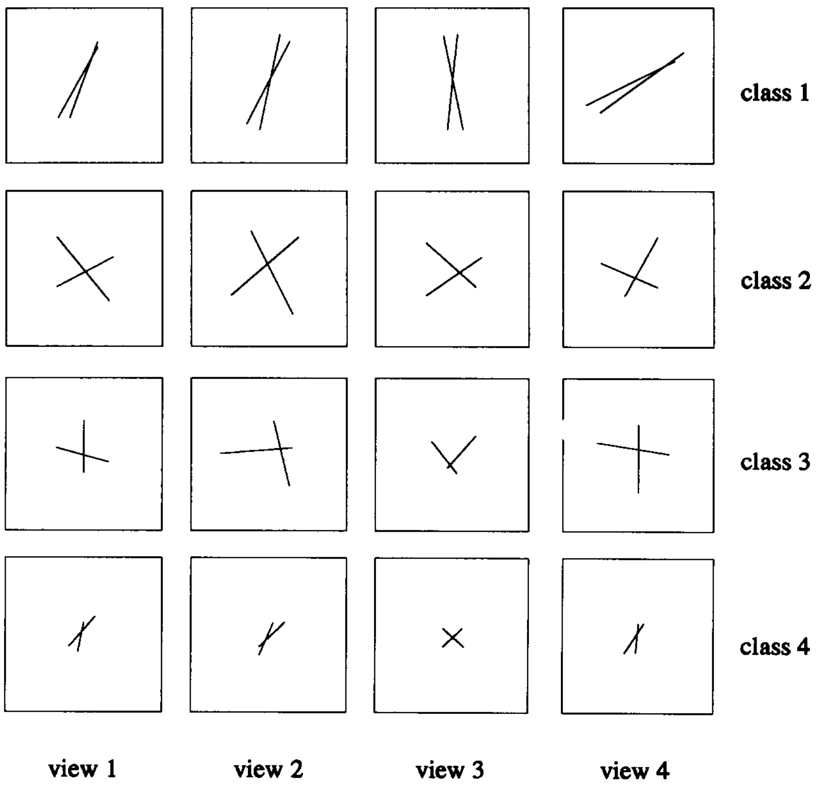

Fig. 2. Four examples ("views") of four different classes for learning. The samples are described by the unary features "line length" and "orientation" and by the binary features "distance between line centers" and "intersection angle".

Table 2. Rule summary for example 2 (Fig. 2) for maximum rule lengths of 1 (*U*), 2 (*UB*) and 3 (*UBU*)

| | Maxlevel = 1 | Maxlevel = 2 | Maxlevel = 3 |
|---|---|---|---|
| Number of rules | 11 | 7 | 8 |
| *U*-rules | 11 | 2 | 2 |
| *UB*-rules | — | 5 | 2 |
| *UBU*-rules | — | — | 4 |
| Average paths/rule | 2.91 | 4.57 | 4.00 |
| Average volume/rule | 0.06 | 0.48 | 0.54 |

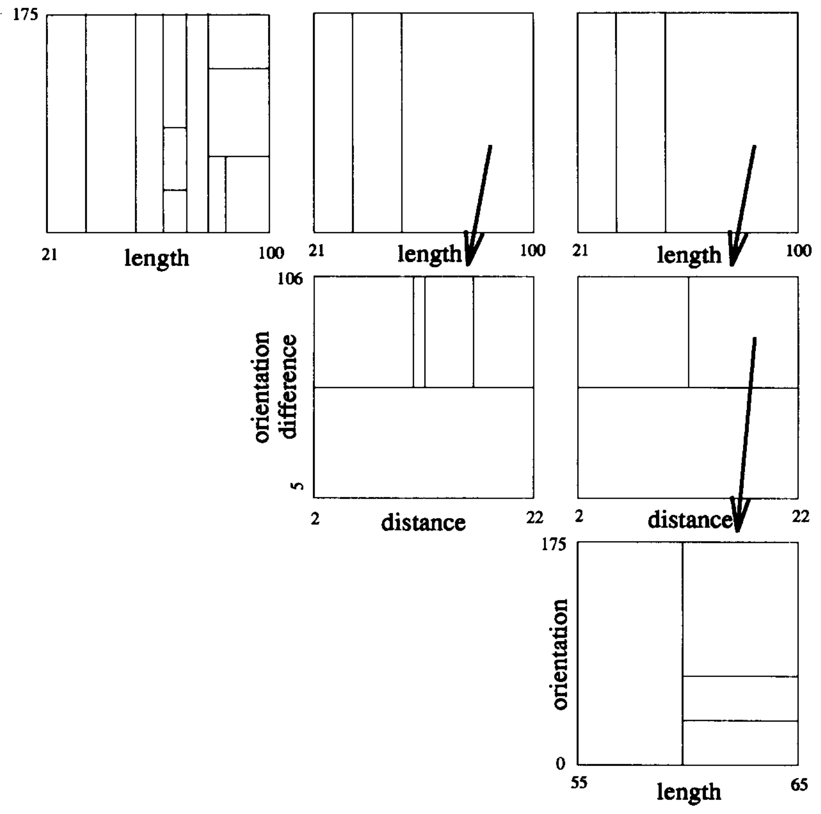

Fig. 3. Conditional clustering solutions three different maximum rule lengths (maxlevel = 1, 2, or 3). Unary feature axes correspond to length (abscissa) and orientation (ordinate). Binary feature axes correspond to distance between line centers (abscissa) and intersection angles (ordinate). Regions correspond to rules in each of the 6 conditional feature spaces and arrows point from an unresolved cluster to the conditional feature space.

Table 3. Rule summary for example 2 (Fig. 4) for maximum rule lengths of 5 (*UBUBU*), 7 (*UBUBUBU*) and 9 (*UBUBUBUBU*)

| | Maxlevel = 5 | Maxlevel = 7 | Maxlevel = 9 |
|---|---|---|---|
| Number of paths | 1608 | 3872 | 7708 |
| Number of rules | 431 | 544 | 562 |
| *U* | 0 | 0 | 0 |
| *UB* | 26 | 4 | 1 |
| *UBU* | 12 | 14 | 14 |
| *UBUB* | 316 | 111 | 68 |
| *UBUBU* | 77 | 94 | 92 |
| *UBUBUB* | — | 256 | 124 |
| *UBUBUBU* | — | 65 | 66 |
| *UBUBUBUB* | — | — | 157 |
| *UBUBUBUBU* | — | — | 40 |
| Average paths/rule | 3.73 | 7.12 | 13.72 |
| Average volume/rule | 0.59 | 0.99 | 1.30 |

The second example, shown in Fig. 4, consists of 5 classes with 4 "views" (examples) each consisting of 5–10 parts with a total of 143 parts. Parts are described by the unary features "number of corners" and "perimeter", and by the binary features "centroid distance" and "sum of distances between corners". Note that, in this example, all classes share parts with the same unary features, and hence discrimination between classes relies mostly on binary features. A summary of the rule produced by CRG (without feature ordering) is shown in Table 3, for maximum rule lengths of 5 (*UBUBU*), 7 (*UBUBUBU*) and 9 (*UBUBUBUBU*). A majority of rules are of even length, i.e. of the form "...*UB*", reflecting the fact that class discrimination relies mostly on binary features. Again, as with example 2, rule generality, as measured by the average feature space volume per rule, increases with increasing length of the rules.

Like any evidenced-based system, the rules generated by CRG will classify new patterns or pattern fragments, provided that they are sufficiently similar to patterns presented during training and contain enough parts to instantiate rules. Here we investigate the use of cluster trees and associated classification rules with partial rule instantiation. A rule of length $m$ (for example, a $UBUBU$-rule with $m = 5$) is said to be partially instantiated by any shorter ($l < m$) sequence of unary and binary features (for example, a $UBU$-sequence with $l = 3$). From the cluster tree shown in Fig. 1 it is clear that a partial instantiation of rules (for example, to the $UB$-level) cannot only lead to unique classification of certain pattern fragments (for example, those matched by the rules associated with $U_3$ or $UB_{11}$) but may also *reduce* classification uncertainty for other pattern fragments (for example, those matched by the rule associated with $UB_{23}$).

This is further illustrated in Fig. 5 which shows the expected classification performance for different lengths of sequences of unary and binary features for the two examples presented in this section. Expected classifi-

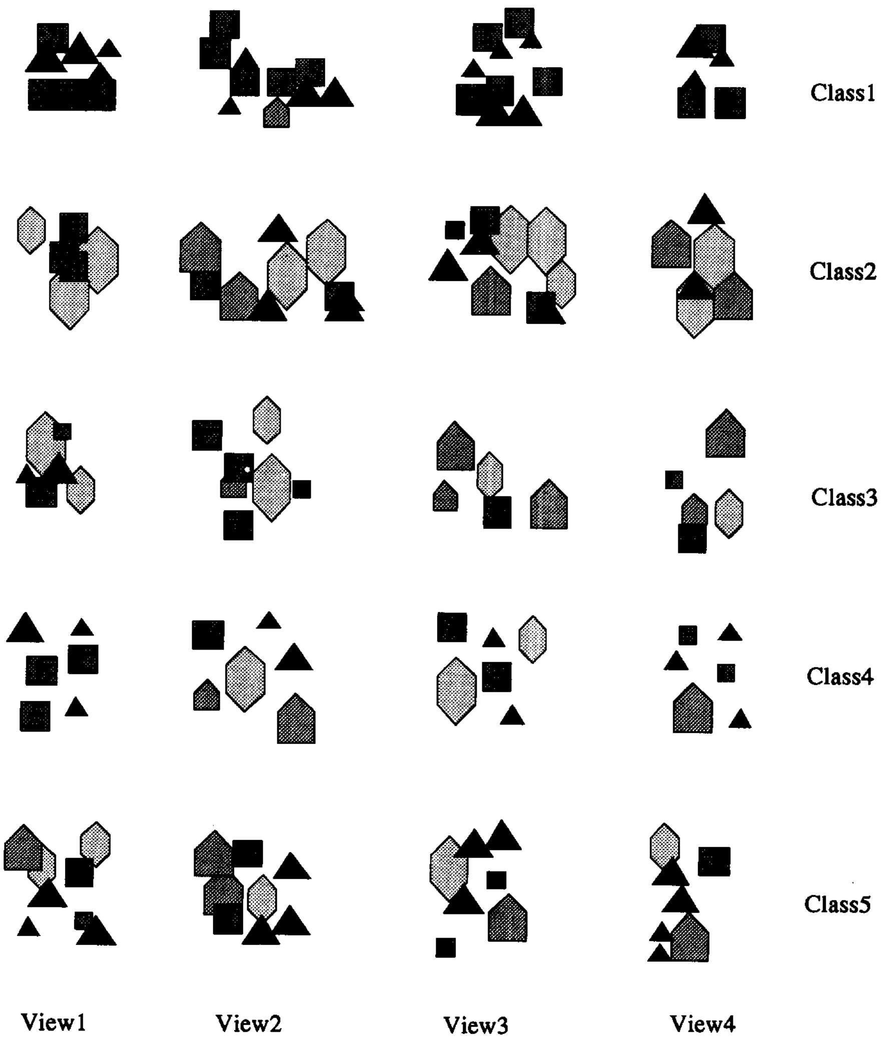

Fig. 4. Complex patterns—four samples ("views") for each of five classes used to train CRG. The samples share common micropatterns and are mainly differentiated by their relative positions.

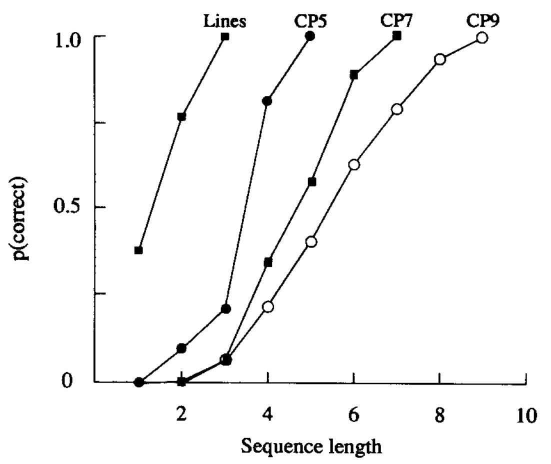

Fig. 5. Probability of correct classification ($P(C)$) for partial data for four different classification problems. "Sequence length" refers to the length of the "*UBU*.." sequence used for classification. The maximum lengths were 3 for lines (Fig. 2), 5 for CP5, 7 for CP7 and 9 for CP9 (complex patterns (CP) in Fig. 4).

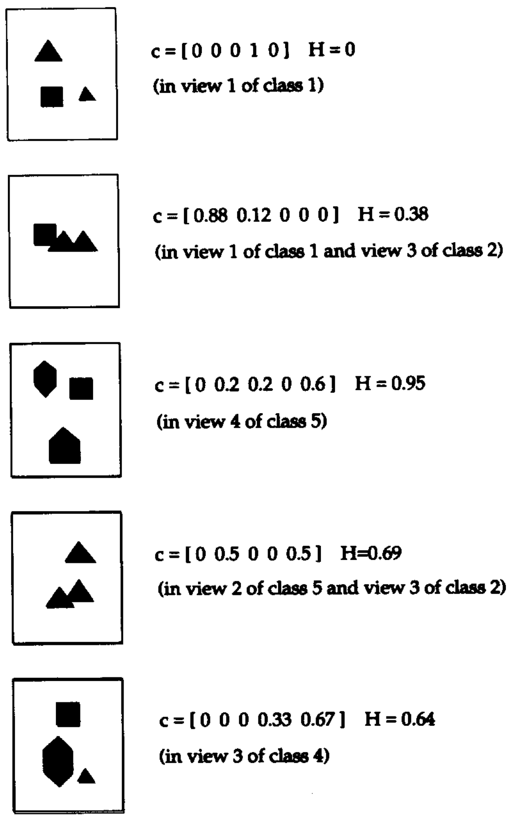
c = [0 0 0 1 0] H = 0
(in view 1 of class 1)
c = [0.88 0.12 0 0 0] H = 0.38
(in view 1 of class 1 and view 3 of class 2)
c = [0 0.2 0.2 0 0.6] H = 0.95
(in view 4 of class 5)
c = [0 0.5 0 0 0.5] H=0.69
(in view 2 of class 5 and view 3 of class 2)
c = [0 0 0 0.33 0.67] H = 0.64
(in view 3 of class 4)

Fig. 6. Classification vectors ($c$) and entropy ($H$) of the classification vectors for each of five degraded data samples for the complex patterns shown in Fig. 4. Here the most likely views from which the samples came, are also identified.

cation performance corresponds to the proportion of correctly classified sequences of a given length, computed over the complete set of training samples. As can be seen from Fig. 5, expected classification performance reaches moderate levels even for relatively short sequences. This then makes it feasible to use partial rule instantiations for pattern classification, as is discussed below.

At runtime, a sequence of unary and binary features of a path through a test pattern is classified by a breadth-first search through the conditional cluster tree. At each node, the next feature of the "*UBU*..." sequence is tested against the feature bounds of the node. The search stops when a (resolved) leaf of the cluster tree has been reached, or when the full length of the test path has been tested. In the former case, classification of the test path is unambiguous; in the latter case, several branches of the cluster tree may have been instantiated. In this case, the classification vector with the lowest entropy is taken as the best classifier.

Figure 6 shows best classification performance (defined by minimum entropy of the classification vectors over all paths) for a selection of partial and distorted patterns from example 2 (see Fig. 4). Here, the classification vectors correspond to the least ambiguous interpretation of the data with respect to what was learned by CRG and what evidence was available from the data. In each case, we have identified the class "view" from which the sample was most likely to come from. We have deliberately chosen difficult samples to illustrate the power of CRC in dealing with partial and distorted data. This is not to imply that CRG is not capable of being successful at such levels of difficulty. From Fig. 6, we have shown that it can learn to "structurally describe" each class perfectly—given the appropriate trade-off between rule length and generalization in the various feature spaces. Rather, we have endeavored to construct quite difficult data to demonstrate how the system works under limited data conditions.

It is also important to note that, in this example, binary rules appear to be more discriminating. This is due to two reasons: first, the discrete nature of the unary features such as "number of corners"; and second, the fact that most samples had similar parts and that classes thus were more easily defined by the relative positions of these parts.

## 4. DISCUSSION

Since CRG develops structural descriptions of patterns in the form of decision trees (see Fig. 1) on attribute bounds on ordered predicates, it is useful to compare it with other techniques from machine learning which attain similar ends symbolically. First, CRG shares with ID3[9] and related techniques similar methods for the search and expansion of decision trees. These techniques, however, were not designed to generate rules satisfying label-compatibility between unary and binary predicates. CRG, on the other hand, is explicitly designed to develop rules for unique identification of classes with respect to their "structural" (linked unary and binary feature) representation.

Recently, however, Quinlan[10] and Muggleton and Buntine[11] have investigated general methods for learning symbolic relational structures in the form of Horn clauses. In FOIL, Quinlan[9] considers the problem of learning, from positive examples (closed world) or positive and negative examples, conjunctions of literals of the form

$$C \leftarrow L_1, \ldots, L_m$$

where $C$ would correspond, in our case, to a class label. FOIL solves such problems by expanding the literals—adding predicates and their variables—to the right-hand side to maximize the capturing of positive instances and minimize negative ones from the training database. In this framework, then, CRG is also concerned with generating similar class descriptions of the specific forms:

$$\begin{aligned}
&C_1^1 \leftarrow U^1(X), B^1(X,Y), U^2(Y), B^2(Y,Z), U^3(Z), \ldots\\
&\vdots\\
&C_1^{n_1} \leftarrow U^1(X), B^1(X,Y), U^2(Y), B^2(Y,Z), U^3(Z), \ldots\\
&\vdots\\
&C_m^1 \leftarrow U^1(X), B^1(X,Y), U^2(Y), B^2(Y,Z), U^3(Z), \ldots\\
&\vdots\\
&C_m^{n_m} \leftarrow U^1(X), B^1(X,Y), U^2(Y), B^2(Y,Z), U^3(Z), \ldots
\end{aligned}$$

However, CRG differs significantly from FOIL insofar as: (1) the choice of unary ($U$) and binary ($B$) rules—as bounded attribute (feature) states—is determined within the continuous unary and binary feature spaces; (2) the *ordering* of literals must be satisfied in the rule generation; (3) the search technique uses backtracking and recursive splitting; and (4) the resultant rules are not only Horn clauses but each literal *indexes* bounded regions in the associated feature space (as shown in Fig. 1).

This does not imply that FOIL could not be run on our data—with every part and relation correspond to a very large set of instantiated literals. Rather, we argue that problems involving numerical and continuous data, as they typically occur in pattern recognition, can be solved more efficiently by the type of technique developed here.

The CRG method is an example of the general solution to complex pattern recognition problems involving the generation of rules, as bounded predicate Horn clauses, which are linked together in ways that determine "structure" uniquely enough to identify classes but enabling maximum generalization to tolerate maximum distortions. Both aims, uniqueness and generalization, are not explicitly guaranteed in other methods, such as neural networks (see reference (12)) or decision trees. Further, in CRG, they explicitly constitute the equivalent of a "cost" function and a search technique has been developed to satisfy these constraints.

What we have not solved here, as yet, is the problem of using CRG to recognize classes of objects in complex montages of other objects. This is the subject of current work. However, we should emphasize that the current system does function with partial class data—both in learning and run time modes.

Finally, CRG raises the question as to what constitutes a "structural description" of a set of patterns. For a set of patterns, CRG generates conditional classification rules which are general in the sense that they are as short as necessary and that they constrain feature bounds a little as necessary for discriminating different classes. For classes with complex and highly variable patterns, CRG may generate many rules which give a set of *equivalent structural descriptions* for the data. One can even introduce a notion of *typicality* through the number of paths that are covered by a branch in the conditional cluster tree. However, as can be particularly true with the complex patterns in Fig. 4, this may not really be a meaningful definition of structure and a "structural description" may have to be defined as the *minimum set of equivalent rules* that CRG generates for patterns, particularly if generalization is necessary and the run time data are incomplete.

*Acknowledgement*—This project was funded by grants from the Center for Intelligent Decision Systems and the Natural Science and Engineering Research Council of Canada.

## REFERENCES

1. L. Shapiro and R. Haralick, Structural descriptions and inexact matching, *IEEE Trans. Pattern Analysis Mach. Intell.* **3**, 504–519 (1981).
2. A. Jain and D. Hoffman, Evidence-based recognition of objects, *IEEE Trans. Pattern Analysis Mach. Intell.* **10**, 783–802 (1988).
3. T. Caelli and A. Pennington, An improved rule generation method for evidence-based classification systems, *Pattern Recognition* **26**, 733–740 (1993).
4. P. Flynn and A. K. Jain, 3D object recognition using invariant feature indexing of interpretation tables. *Comput. Vision, Graphics and Image Process.* **55**, 119–129 (1992).
5. P. Flynn and A. K. Jain, Three-dimensional object recognition. *Handbook of Pattern Recognition and Image Processing, Vol. 2: Computer Vision.* Tzay Y. Young, ed., Academic Press, New York (1993).
6. W. E. L. Grimson, *Object Recognition by Computer*, MIT Press, Cambridge, Massachusetts (1990).
7. K. Ikeuchi and T. Kanade, Automatic generation of object recognition programs, *Proc. IEEE* **76**, 1016–1035 (1988).
8. R. C. Bolles and P. Horaud, 3DPO: A three-dimensional part orientation system, *Int. J. of Robotics Research* **5**, 3–26 (1986).
9. M. A. Fischler and R. A. Elschlager, The representation and matching of pictorial structures, *IEEE Trans. Comput.* **22**, 67–92 (1973).
10. R. Mohan and R. Nevatia, Using perceptual organization to extract 3-D structures, *IEEE Trans. Pattern Analysis Mach. Intell.* **11**, 1121–1139 (1989).
11. D. G. Lowe, Three-dimensional object recognition from single two-dimensional images, *Artif. Intell.* **31**, 355–395 (1987).
12. R. Michalski and R. E. Stepp, Automated construction of classifications: conceptual clustering vs numerical taxonomy. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **5**, 396–409 (1983).
13. A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data.* Prentice-Hall, Englewood Cliffs, New Jersey (1988).
14. U. Fayyad and K. Irani, On the handling of continuous-

valued attributes in decision tree generation, *Machine Learning* **8**, 87–102 (1992).
15. J. R. Quinlan, Learning logical definitions from relations, *Mach. Learning* **5**, 239–266 (1990).
16. J. R. Quinlan, Induction of decision trees, *Mach. Learning* **1**, 81–106 (1986).
17. S. Muggleton and W. Buntine, Machine invention of first-order predicates by inverting resolution, *Proc. Fifth Int. Conf. on Machine Learning*, pp. 339–352. Morgan Kaufmann, San Mateo (1988).
18. R. P. Lippmann, An introduction to computing with neural nets, *IEEE ASSP Mag.* April (1987).

**About the Author**—WALTER BISCHOF received his Ph.D. from the University of Bern in 1982. Since then he has been active in research in the areas of human and machine vision. He is Associate Professor of Psychology and Computer Science at the University of Alberta, Alberta, Canada, and a Senior Research Fellow at the Collaborative Information Technology Research Institute, The University of Melbourne, Victoria, Australia. His research interests, at present, focus on modelling aspects of human and machine motion processing, machine pattern and object recognition and medical image processing.

**About the Author**—TERRY CAELLI received his Ph.D. from the University of Newcastle in 1975. Since then he has been active in research in the areas of human and machine vision. He is Professor of Computer Science at The University of Melbourne and interests lie in human and machine pattern/object recognition and machine learning.