Hierarchical Clustering of Composite Objects with a Variable Number of Components

A. Ketterlin, P. Gançarski & J.J. Korczak
Laboratoire des Sciences de l'Image, d'Informatique et de Télédétection
URA 1871, CNRS, Université Louis Pasteur
7, rue René Descartes, F-67084 Strasbourg Cedex.
e-mail: {alain,gancars,jjk}@dpt-info.u-strasbg.fr

Abstract

This paper examines the problem of clustering a sequence of objects that cannot be described with a predefined list of attributes (or variables). In many applications, such a crisp representation cannot be determined. An extension of the traditionnal propositionnal formalism is thus proposed, which allows objects to be represented as a set of components. The algorithm used for clustering is briefly illustrated, and mechanisms to handle sets are described. Some empirical evaluations are also provided, to assess the validity of the approach.

1 Introduction

The basic process of clustering consists in finding coherent groupings from a given data set. Each grouping must exhibit *sufficient* internal cohesiveness, while maintaining enough *distance* to other members of the partition. Such a definition of clustering importantly conditions many techniques and algorithms devoted to the automated discovery of clusters [7].

The statistical view on the problem, namely the field of cluster analysis, essentially deals with numerical and/or categorical data, where objects are represented as feature vectors [2]. This representation is equivalent, in essence, to propositionnal logic (or attribute-value formalism). Derived clusters are usually represented as centroids in the instance-space, these being used for further classification of new data (either with the help of a distance measure, or with some implicit probability distribution).

On the other hand, machine learning research on conceptual clustering [11] conducted during last decade focused on the representation of instances and concepts, the goal being to derive some effective knowledge (i.e. *generalizations*) about the domain under observation. *Concept formation* is defined as incremental conceptual clustering [4]: incrementality is the ability to maintain a conceptual structure, which is updated after each observation.

Many machine learning clustering algorithms still use propositional logic to describe the data, but seek for an enrichment of the language used to describe concepts. Some systems use a extended form of propositional logic, where concepts are defined by logical formulae [11]. Others use *probabilistic concepts* [3], where a concept can be thought of as a prototype (a typical instance) along with a probability distribution on each dimension. More recent works extend clustering techniques to higher level languages: KBG [1] deals with first-order logic. KLUSTER [6] uses a KL-ONE-*like* language to avoid computationnal complexity and still keep comfortable representative power.

However, there are domains where many numerical aspects are to be considered: in such domains, logic-based formalisms apply difficultly, and their generality might be penalizing. This paper describes an extension of attribute-value formalisms which allows objects to be represented as somewhat *(un)structured*: in that framework, an object is built up from any number of *components*. Earlier work on such extensions of propositional formalisms introduced the name of *structured concept formation* (see[12]), because objects exhibit several levels of detail in terms of their structure.

Section 2 briefly reviews the algorithm used for clustering. Section 3 describes how the clustering can be done with objects described as sets. Section 4 gives three examples of sets clustering.

2 Concept Formation

The underlying concept formation algorithm used throughout this paper is COBWEB[1]. This algorithm forms a hierarchy of clusters from a sequence of objects. Each object is an attribute-value list. Derived clusters (or *concepts*) are represented in a probabilistic form. Attributes are *typed*: the algorithm is able to handle nominal variables as well as numerical ones. In the case of numerical attributes, an underlying normal distribution is assumed, whose parameters are estimated. Therefore, a concept stores for each numerical attribute the estimated mean and variance with respect to the covered objects. The variances help define a global predictivity score for one concept, named Π, which is an averaging of the inverse of the standard deviation over all attributes.

Each object is incorporated into the concept hierarchy in turn. When incorporating a new object, a search is performed by hill-climbing through a space of conceptual hierarchies. The object is sorted down through the current hierarchy. At each level, several operators are temptatively applied to the current partition, some of them having restructuring properties. The best of these operators is definitely applied, and the process restarts one level deeper (see [3, 5] for a complete description of the algorithm).

When several distinct partitions are generated, a heuristic called *category utility* is used to select between them. *Category utility* evaluates the global *quality* of a single partition. This evaluation is based on the individual predictivity of each concept involved. The partition $(C, \{C_1, \ldots, C_K\})$ is evaluated by:

$$\frac{1}{K}\sum_{k=1}^{K} P(C_k)[\Pi(C_k) - \Pi(C)]$$

where $\Pi(C_*)$ measures the individual predictivity of C_*. This expression is an averaging of the predictivity-gain when stepping from the concept C to one of its sub-concepts. The individual predictivity of a concept is defined as:

$$\Pi(C_*) = \frac{1}{I}\sum_{i=1}^{I} \Pi(A_i, C_*)$$

where $\Pi(A_i, C_*)$ quantifies how the attribute A_i is predictive in C_* (i.e. how precisely values of A_i can be predicted for members of C_*).

In this framework, an learning-problem is defined on a set of attributes. Observations must be represented by a value for all (or some) of the attributes. Concepts or clusters must represent some distribution of values for each attribute of the learning-problem. Attributes may be of different types. The definition of a type of attribute must thus include a value-space, a way to represent a distribution of such values, and a predictivity measure for such a distribution. As originally described in [3], COBWEB provides the definitions for nominal (categorical) attributes. CLASSIT [5] provides the definitions for numerical (continuous) attributes. The LABYRINTH system [12] further extends COBWEB to deal with composite objects by the way of "structured" attributes. The next section introduces a new type of attributes, called "set" attributes, defines the representation of values and distributions, and gives a predictivity evaluation measure for such distributions.

3 Clustering Sets

3.1 Representation of Objects

The aim of this paper is to describe a conceptual clustering system that allows objects to be represented as sets of *components* (i.e. sub-objects). The representation formalism is inspired by propositionnal logic. In that framework, each object is a list of attribute-value pairs. Our system allows a value to be a set of objects (instead of being one single numerical or categorical quantity), all these objects being described with the same set of attributes. Members of such a value are less abstract objects which are part of the description of the englobing (abstract) object. To illustrate this notion, an example domain from the litterature will be used. This domain is the one of *quadruped mammals*: each object is constituted of several sub-objects (cylinders), each of them being, in turn, described with several numerical features. An observation of that domain may be written down as:

```
Dog1 = [ cyls = {
                [r=13.9,l=25.1],
                [r=6.5,l=10.0],
                [r=4.7,l=6.3] } ]
```

In that case, the object `Dog1` is described with only one attribute, named `cyls`, whose value is a set containing three sub-objects.

[1] In this paper, we consider COBWEB to be the algorithm described in [3], with an extension to deal with numerical attributes developped in CLASSIT (see [5]), but without any other extension from CLASSIT.

Each of these sub-objects is described with two numerical attributes (r and l)[2].

The previous section briefly explained how to build a concept hierarchy. It was said that this problem can be cast into the problem of quantifying how predictive a concept is. In the case of the domain described above, the question is: "How can one quantify the predictivity of a set of animals ?".

The answer to this problem lies in the fact that a concept hierarchy can be built at each level of abstraction. In the animal domain it means that two conceptual hierarchies are maintained: one for the cylinders (the component-concept hierarchy) and one for the animals (the composite-concept hierarchy). The results of incorporating the cylinders forming an animal are used during the incorporation of the animal. That is, the same process is repeated at each level of structural abstraction. This is a *component-first* strategy, where parts are clustered before the whole. The algorithm may be summarized as follows :

- for each component
    - incorporate the component in the component-concept hierarchy
    - update the description of the composite (replace the component by a reference to the concept it reached)
- incorporate the composite object into the composite-concept hierarchy

Each "incorporate" operation is a recursive call to COBWEB. The first call operates in each component-space, the second call in the composite-space.

It follows from that sketch that the integration of a composite object (e.g. an animal) is done with each component replaced by a reference to a concept[3]. What is thus needed is a way to quantify the predictivity of a set of sets of component-concept references. This problem divides itself into two distinct phases, explained in the next two sections.

[2] In this paper, only single attribute domains will be considered. The mechanisms described here apply to one attribute, whose values are sets of objects. With no loss of generality, the method can be extended to domains described with several attributes, each of them having sets as values

[3] A reference to the most specific concept covering an object is also called the *conceptual signature* of the object

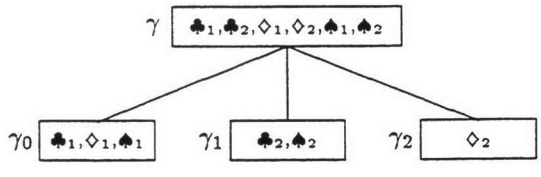
3.2 Conceptual Representation

The first step is to find an adequate representation for the concept. Since members of values (i.e. sub-objects, or components) are hierarchically clustered, they are replaced by their most specific covering concept: the initial instance space is replaced by a conceptual hierachy (which is a partially ordered space). A conceptual (or intensionnal) representation of a set of composite objects must rely on a conceptual "vocabulary" for the components. This language is provided by the component-concept hierachy. On the other hand, an intuitive way to characterize a set of sets is by the way of their mutual intersection (or overlapping). This intersection must be described in the language provided by the component-concepts.

The conceptual representation of a set of composite objects will thus be a set of component-concepts: these concepts are called *central-concepts*, and must have the following properties:

1. Each central concept must cover at least one member of each value. This means that each central concept must characterize the intersection between all the sets under consideration.
2. All the members of the sets must be covered by central concepts. This means that central concepts must cover all the elements of all the sets, thus avoiding "marginal" intersections between only parts of the sets (that would leave some other components —members— uncovered).
3. Central concepts have to be as specific as possible. This means that the set of central concepts is the most "precise" conceptual characterization of the intersection between all the sets.

As an illustration, let us consider the following part of a component-concept hierarchy. In this example, three sets are involved: $S_{\clubsuit}$, $S_{\diamondsuit}$ and $S_{\spadesuit}$. Each set has exactly two members.



Let us suppose now a composite-concept $C_{\clubsuit\diamondsuit\spadesuit}$ covering the three sets $S_{\clubsuit}$, $S_{\diamondsuit}$ and $S_{\spadesuit}$. Only

one central-concept can be found for $C_{\clubsuit\diamondsuit\spadesuit}$: γ is the only component-concept satisfying the three conditions expressed above (γ_1 and γ_2 violate the first condition, γ_0 the second one). Let us now consider the composite-concept $C_{\clubsuit\spadesuit}$. In that case, γ_0 and γ_1 are both central concepts (γ_2 is ignored because it does not cover any member of any value covered by $C_{\clubsuit\spadesuit}$).

It is important to note that, in the case where only one composite object is covered, the set of central concepts used to represent the composite-concept is equal to the set of conceptual signatures (component-concepts) of members of the object.

3.3 Predictivity Evaluation

The second step is to evaluate the predictivity of the set of central concepts. Since each central concept can be found in the component hierarchy, it is labelled with its individual predictivity (named Π). A straightforward way to compute predictivity for a set of concepts is to average their individual predictivity score. This was the approach undertaken in our implementation, even though other methods of combination could be explored. But, since all central concepts do not cover the same proportion of objects, the contribution of each central concept is weighted by the amount of objects it covers. The predictivity of a set of L central concepts $\{\gamma_1, \ldots, \gamma_L\}$ is thus:

$$\sum_{l=1}^{L} \frac{n_o(\gamma_l)}{N_o} \Pi(\gamma_l)$$

where $n_o(\gamma_l)$ is the number of members covered by γ_l, and $N_o = \sum_{l=1}^{L} n_o(\gamma_l)$ is the total number of components.

The weight associated to a central concept will be called its *coverage*. The overall predictivity is thus a tradeoff between the predictive ability and the coverage of central concepts.

3.4 Incrementality

The original COBWEB algorithm, which worked with a purely propositionnal formalism, was designed to handle the problem of concept formation, which is defined as incremental conceptual clustering. The incremental ability is crucial in most applications. It is thus important to check that incrementality is preserved when extending the knowledge representation language.

The problem may be stated as: "Having a composite-concept description and a new composite object to incorporate, how can one compute the new description of the composite-concept ?" The process can be divided in two phases:

- generalize any central concept that does not cover at least one of the components of the new object

- if any component remains uncovered by a central concept, generalize the "nearest" central concept

Again, this is only a sketch of the procedure: particularly, any generalization must be performed carefully. A generalization is the replacement of one central-concept by its parent in the component-concept hierarchy. This process is equivalent to the application of a "climb-generalization-tree" operator [10], with the property that the generalization tree is itself built and maintained by the system. Nevertheless, this simple procedure ensures that the definition of central concepts is maintained. The two phases correspond to the first two conditions in the definition of central concepts. It has to be noted that the addition of an object to a concept may only generalize the central concepts describing the composite-concept. This does not mean that the predictivity of that composite-concept decreases, since a loss in predictivity may be compensated by an increase in coverage.

3.5 Complexity

This section investigates the computational cost of the set-clustering process, and focuses particularly on the integration of a new composite object into an existing composite-concept.

The problem of structured concept formation has already been adressed by the LABYRINTH system [12][4]. The difference between LABYRINTH and the system described in this paper is that LABYRINTH is designed to find a binding between the components and a predefined set of attributes. Once the binding is determined, the task is reduced to composite-objects clustering. But this binding process has an extreme computational cost: $O(d!)$ if an exhaustive search is performed (d being the number of components per object, which is fixed in

[4] LABYRINTH's formalism also includes relationnal information between components, which are not discussed here.

LABYRINTH), $O(d^2)$ or $O(d^3)$ if heuristic solutions are used (see [12]). This binding is computed each time an object is added to a concept. Note that this cost may be penalizing when d is high (see next section for examples of such situations).

In contrary, our systems solves the mapping by finding a mutual intersection between all the sets. Hence, the mapping process is replaced by the search for central-concepts. Let us consider a composite-concept C covering N objects $S_1, \ldots, S_N$. Let $n_i = \mid S_i \mid$. It is easy to see that C will be represented by at most $\omega = \max_i\{n_i\}$ central concepts. Hence, the integration of the next composite object in C may lead to at most ω generalizations during the first step of the updating process. The second step may apply only $\mid S_{N+1} \mid$ times. The whole updating process is thus linear in the average number of components per composite object.

This notable decrease in complexity may be explained by the fact that, instead of considering all possible bindings between the components and a predefined set of attributes, the set-clustering mechanism uses the component-concept hierarchy to solve the partial matching problem. Moreover, the binding searched for by LABYRINTH appears as an epiphenomenon: better than searching for a fixed number of components, a global structure appears by the way of the representation of composite-concepts in terms of central concepts.

4 Empirical Results

4.1 The Simplified Quadruped Mammals Domain

This domain has already been used in the literature about structured concept formation, to demonstrate the abilities of the CLASSIT algorithm [5]. It has been simplified here for explanatory purposes. In this domain, objects represent some perceptual sketch of an animal: each object is composed of three cylinders (instead of eight in the original domain) representing the head, the torso and one leg of the animal. Each cylinder is described by two (instead of 9 in the original domain) numerical attributes (the radius and length).

Since these objects are artificially generated from four available models (cat, dog, horse or giraffe)[5], the goal is to discover these classes from the unlabelled data. In our experiment, 20 objects were randomly generated (five from each model), each one being made of three cylinders. The system was asked to build a conceptual hierarchy for animals: it thus built two conceptual hierarchies (one for the cylinders and one for the animals). Results are sketched on Figure 1: note that each terminal class on Figure 1 is in fact further developped.

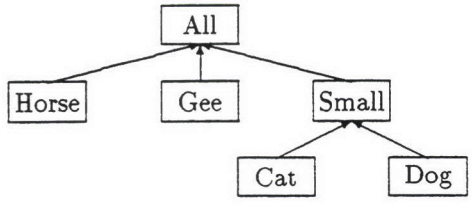


Figure 1: Results in the quadruped mammals domain.

The system found the four classes of animals, even though all of them are not at the first level. More important is the fact (not shown on Figure 1) that the conceptual representation of each of these classes included three central concepts, corresponding to the three cylinders. The concept labelled "Small" covers all small animals (dogs and cats), and its conceptual representation includes only two cylinder-concepts. The interpretation of such a conceptual representation is the following: a small animal is an animal made of two cylinders of one type (described by the first central concept) and one cylinder of another type. Such a representation could not have been obtained if a fixed structure (i.e. a predetermined number of attributes) had been used.

4.2 Object Recognition Domains

4.2.1 Character Recognition

The second experiment involves some simplified form of pattern recognition. The basic idea is to consider a pattern as a set of pixels. To compensate the loss of the information of the spatial arrangement of pixels, the representation of each individual pixel is based on several convolutions.

Data are drawn from a set of four alphabetic characters (shown on Figure 2), each of them printed in four directions. Each pattern was convolved with the laplacian of a gaussian

[5] The instance-generator is available at the "UCI Repository of Machine Learning Databases and Domain Theories" at Irvine, at `ftp.ics.uci.edu`, as `/pub/machine-learning-databases/quadrapeds`

at several different scales. Each pixel shown black in the original pattern is considered as a component of that pattern. Each component is described by the values of the convolutions at its position. These values can be seen as local characteristics of the pixel in the original pattern: in fact, as shown in [9], convolution by the laplacian of a gaussian can be used as an edge detector (a value near zero meaning that the pixel is on an edge in the original image). In the experiments reported here, three convolutions were used, with the σ parameter respectively equal to 5/2, 3 and 7/2 (see [9] for details about the meaning of this parameter). The convolutions were directly applied to the iconic images.

Note also that this means that the four sets representing one character (one set for each "direction") perfectly intersect. It means also that all objects do not have the same number of components, as shown on Figure 2.

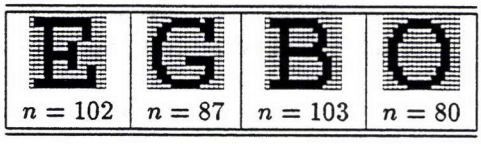


Figure 2: The four characters used.

The system was thus given 16 composite-objects, with a total of 1488 components. Results are shown on Figure 3.

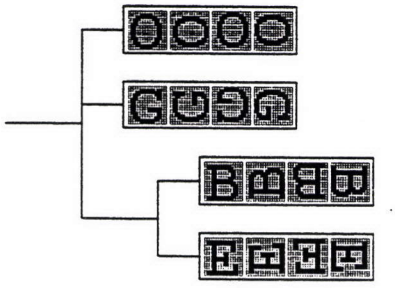


Figure 3: Results in the letter-recognition domain.

The system discovered one concept for each class of character, as was expected. In that case again, a fixed structure (i.e. a predefined set of attributes) cannot model such a problem.

4.2.2 Rotated Polygons

The third experiment was built on the same base as the second, but with more realistic rotation angles. The design was as follows: each pattern was computed from an underlying polygon, to which a rotation was applied, whose angle was randomly drawn. The rotated figure was then discretized, leading to a grey-levels icon. The icon was then treated the same way as the alphabetic characters in the previous experiment. Each non-white pixel was considered a component, and local characteristics (i.e. values of the convolution with the laplacian of a gaussian) were used to describe it. Parameters of the convolutions were set to the same values as in the previous experiment.

Three "models" were used to generate the data. Each model was randomly rotated by three different angles, and each time expressed as a set of pixels. Figure 4 shows the data, with the angle of rotation and the number of pixels. The result of the clustering is shown on Figure 5. Results are as good as in the naïve case, and suggest that the set clustering algorithm is able to discover rotational-invariant concepts of patterns.

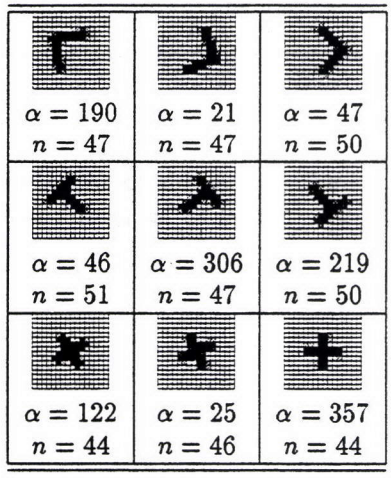


Figure 4: Nine rotated polygons.

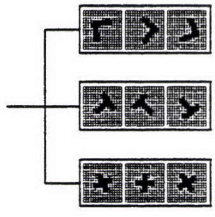


Figure 5: Results in the rotated-polygons domain.

5 Conclusion

The set clustering mechanism described in this paper allows concept formation to take place in domains where a structure can be extracted. Particularly, it resolves the task of concept formation from composite objects. In the case when each set is restricted to one member, and several sets are used to describe each object, the problem reduces to composite-objects clustering. In the general case, the overlapping between sets is used to cluster composite-objects. Experimental results have been given, that illustrate the performance of the algorithm and demonstrate abilities that can not be attained by purely propositional algorithms.

The experiments described in the previous section raise some questions about the nature of the discovered concepts, particularly in the case of the pattern-recognition problems. The representation of a pattern (as a set of pixels) is somewhat unusual, and the concepts that are derived are expressed in terms of pixel-classes. It is thus impossible to ask the system to "illustrate" what is a cross, for instance. But a new cross will be recognized as such: more precisely, a new cross will be recognized as being similar to previous ones. It seems that this is a case of a knowledge structure with no explicit, internal representation of known concepts. The component-concept hierachy may be said to reside at a sub-symbolic level (assuming that patterns are at the symbol level).

The work presented in this paper is a part of a project on "Learning and Image Processing". Earlier work on this project established the adequacy on concept formation techniques for image segmentation [8]. The next step is to study the abilities of the same algorithm on an object recognition problem. Preliminary results are described in this paper. The important point is that the same algorithm is used for both tasks (image segmentation and object recognition), and that this algorithm is totally unsupervised. Planned future works include the integration of both phases (i.e. object recognition working on discovered segments), and deeper studies of the effect of occlusion for the object recognition task.

References

[1] Bisson, G. (1992) Conceptual Clustering in a First Order Logic Representation. *Proceedings of the Tenth European Conference on Artificial Intelligence*. pp. 458–462. J. Wiley & Sons.

[2] Diday, E., Lemaire, J., Pouget, J. & Testu, F. (1982). Eléments d'Analyse de Données. Dunod.

[3] Fisher, D.H. (1987). Knowledge Acquisition Via Incremental Conceptual Clustering. *Machine Learning, 2*, pp. 139–172.

[4] Fisher, D.H., Pazzani, M.J., & Langley, P. (Eds.), (1991). Concept Formation: Knowledge and Experience in Unsupervised Learning. Morgan Kaufmann.

[5] Gennari, J.H., Langley, P., & Fisher, D.H. (1989). Models of Incremental Concept Formation. *Artificial Intelligence, 40*, pp. 11–61.

[6] Kietz, J-U., & Morik, K. (1994) A Polynomial Approach to the Constructive Induction of Structural Knowledge. *Machine Learning, 14*, pp. 193–217.

[7] Korczak, J.J., & Rymarczyk, M. (1993). Application of Classical Clustering Methods to Digital Image Analysis. Research Report. Université Louis Pasteur. Strasbourg.

[8] Korczak, J.J., Blamont, D., & Ketterlin, A. (1994) Thematic Image Segmentation by a Concept Formation Algorithm. *Proceeding of the European Symposium on Satellite Remote Sensing*, Roma, Sept. 1994.

[9] Marr, D., & Hildreth, E. (1980). Theory of Edge Detection. *Proc. R. Soc. Lond. B, 207*, pp. 187–217.

[10] Michalski, R.S. (1983). A Theory and Methodology of Inductive Learning. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach.* Morgan Kaufmann.

[11] Michalski, R.S., & Stepp, R.E. (1983). Learning from Observation: Conceptual Clustering. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach.* Morgan Kaufmann.

[12] Thompson, K., & Langley, P. (1991) Concept Formation in Structured Domains. In [4].