



Learning hierarchical clustering from examples

Edwin Diday, J.V. Moreau

► To cite this version:

Edwin Diday, J.V. Moreau. Learning hierarchical clustering from examples. RR-0289, INRIA. 1984. inria-00076269

HAL Id: inria-00076269

<https://inria.hal.science/inria-00076269>

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
CENTRE DE ROCQUENCOURT

Rapports de Recherche

N°289

**LEARNING
HIERARCHICAL CLUSTERING
FROM EXAMPLES**

**Edwin DIDAY
Jean Vincent MOREAU**

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél. 954 90 20

Mai 1984

LEARNING HIERARCHICAL CLUSTERING FROM EXAMPLES

Edwin DIDAY Jean Vincent MOREAU

RESUME

Un problème classique de reconnaissance des formes consiste à chercher de façon automatique un opérateur de classement ("classifier" en anglais) à partir d'un échantillon d'apprentissage sur lequel les classes sont connues. Un problème qui se pose fréquemment dans la pratique mais qui a été moins bien cerné par les différents auteurs consiste à chercher un opérateur de classification automatique (on pourrait dire en anglais "Clusterfier" par opposition à "classifier") à partir de classes connues sur un échantillon. Dans le premier problème il faut trouver un opérateur qui associe à chaque nouvel objet l'une des classes définie par l'échantillon d'apprentissage ; dans le second il faut trouver un opérateur qui détecte des classes dans la population complète en tenant compte au mieux de l'information donnée par les classes connues sur un échantillon. On propose une nouvelle approche permettant de construire une ultramétrie à partir des connaissances acquises sur les données de l'échantillon d'apprentissage ; l'ultramétrie ainsi obtenue permet de construire une hiérarchie qui "infère" les classes cherchées, sur toute la population.

Un algorithme de classification hiérarchique de type voisin réciproque pour induire la hiérarchie finale a été mis au point. On obtient un temps d'exécution nettement plus rapide qu'avec le programme de classification hiérarchique classique qui ne comporte pas d'inférence.

Ce programme a permis de construire des indices d'agrégation adaptés à des dispositions particulières de points (classes allongées, classes sphériques avec noyau central, demi-sphère avec noyau central, classes allongées ou sphériques bruitées...) et certains indices trouvés permettent de séparer des classes spécifiques qu'aucun indice classique ne reconnaît.

LEARNING HIERARCHICAL CLUSTERING FROM EXAMPLES

E. DIDAY J.V. MOREAU

ABSTRACT

A classical problem of Pattern recognition consist in looking for an operator of classification (a "classifier") induced from a learning set on which classes are known. A problem frequently encountered in practice is the one of looking for an operator of clustering (a "clusterfier", in opposition to "classifier") a learning set of which classes are also known. In the first case, we have to find an operator which allocate each new object to one of the classes defined by the learning set. In the second case, we have to find an operator which detect classes in the complete population, taking account as well as possible of the information given by the classes of the learning set. We propose a new approach permitting to induce an aggregation index from knowledge acquiring on the learning set ; the aggregation index thus obtained permit to induce a hierarchy which infers the desired classes on the whole population.

A nearest neighbours algorithm with validity constraints has been realized to induce the final hierarchy. We obtain a CPU time clearly shorter than with classical hierarchical ascending classification algorithm which doesn't use inference.

This program has permitted to find aggregation indices adapted to particular learning sets (elongated classes, spherical class with central kernel, half spherical class with central kernel, noising elongated classes ...), and some of the new indices permit to recognize more specific classes than the usual indices.

LEARNING HIERARCHICAL CLUSTERING FROM EXAMPLES

Application to the adaptive construction of dissimilarity indices

1) Introduction

It often happens in practice that a user wishing to make a hierarchical classification does not know which of the usual panoply of dissimilarity indices will be the best one for his data ; it can also happen that none of these indices satisfy the data that he must deal with ; so a problem arises with regard to the choice of one of the known indices and, possibly, the creation of new indices. Sometimes the data provide a natural index, for example, the demographic transfer flow from one district to another if the individuals to be classified are districts. Unfortunately, such an index may cause inversions to appear during the construction of the hierarchy, which then becomes hard to interpret ; an inversion comes about when the index associated with a level h is greater than the index associated with a level h' although h is included in h' . We provide simple rules that make it possible to deduce, from the value of the coefficients of Lance and Williams' recurrence formula (1967), generalized by Jambu (1978), the possible existence of inversions. The same sort of rules also make it possible to ascertain whether the condition of validity from the nearest neighbours algorithm (1982) is satisfied and whether it is therefore possible to use an accelerated hierarchical classification algorithm, without risk of distortion. If the user has at the outset some ideas on the classification that he wishes to obtain, the recurrence formula and the constraints obtained in order to ensure that no inversion takes place give equations of which the unknowns are the coefficients a_1, \dots, a_7 of the recurrence formula. If a solution exists, it provides a dissimilarity index well-adapted to the user's wishes.

2) Some definitions

Definition of a hierarchy

Let Ω be a finite set and H a set of parts (called levels) of Ω . H is a hierarchy on Ω if:

- 1) $\Omega \in H$ {i.e., the highest level contains all the individuals}
- 2) $\forall w \in \Omega : \{w\} \in H$ (the terminal points)
- 3) $\forall h, h' \in H : h \cap h' \neq \emptyset \Rightarrow h \subset h' \text{ or } h' \subset h$.

Binary hierarchy

This term thus designates a hierarchy H of which each level is formed by grouping together two elements of H , or more precisely:

$\forall B \in H : \text{card}(B) > 1, (B', B'') \in H \times H :$

$$B' \cap B'' = \emptyset \text{ and } B' \cup B'' = B.$$

Definition of an indexed hierarchy

An indexed hierarchy is a couple (H, f) in which H is a hierarchy and f an application of H in \mathbb{R}^+ such that:

- 1) $f(h) = 0$ if and only if h contains only one element.
- 2) for all h and h' in H , $h \subset h'$ (strict inclusion) implies $f(h) < f(h')$.

Other types of indexed hierarchies

In practice, certain algorithms of hierarchical classification (for example, the HAC algorithm, given in appendix 1, see [2] or [4]) may give rise to hierarchies of which the associated index does not exactly satisfy the definition we have just given: indeed it may happen that two elements h and h' of H satisfy the relationships $h \subset h'$, $h \neq h'$ and $f(h) = f(h')$. In this case, we shall say that the hierarchy is indexed "in the broad sense".

Definition of a hierarchy indexed in the broad sense

It is a couple (H, f) in which H is a hierarchy and f , an application of H in \mathbb{R}^+ such that:

- 1) $f(h) = 0$ if and only if h contains only one element.
- 2) for any h and h' in H , $h \subset h'$ implies $f(h) \leq f(h')$.

Definition of an inversion

It can also happen that the index associated with certain hierarchies gives rise to the existence of levels h and h' such that $h \subset h'$ and $f(h) > f(h')$. In such a case we say that there is an inversion. For an indexed hierarchy, or for a hierarchy indexed in the broad sense, there are no inversions.

The construction of a hierarchy requires the knowledge of a "measure of similarity" between groups, this "measure" being called the "dissimilarity index".

Definition of a dissimilarity index

A dissimilarity index between groups of individuals is by definition an application $\delta: P(\Omega) \times P(\Omega) \rightarrow \mathbb{R}$, $P(\Omega)$ being the set of parts of Ω , such that:

- 1) $\forall h_1, h_2 \in P(\Omega), \delta(h_1, h_2) \geq 0$ (positiveness)
- 2) $\forall (h_1, h_2) \in P(\Omega) \times P(\Omega), \delta(h_1, h_2) = \delta(h_2, h_1)$ (symmetry)

Let us consider a binary hierarchy H and a dissimilarity index between clusters; in order to index H and to be sure that there will be no inversion, one can define f in the following way:

$$\forall h_i, h_j \text{ in } H, f(h_i \cup h_j) = \text{Max} (\delta(h_i, h_j), f(h_i), f(h_j)).$$

In this way a hierarchy indexed in the broad sense is obtained. On the other hand, if f is chosen in the following way:

$$\forall h_i, h_j \text{ in } H, f(h_i \cup h_j) = \delta(h_i, h_j),$$

it is not certain that (H, f) is an indexed hierarchy, even in the broad sense. There may be some inversions.

3) Conditions that a dissimilarity index δ must satisfy in order for a hierarchy indexed by $f : f(h_1 \cup h_2) = \delta(h_1, h_2)$ to have no inversions

The visual representation of a hierarchy H can be interpreted more easily if it is indexed in such a way that the height of each level corresponds to the value taken on by the dissimilarity index for the two levels that have formed it. In other words, if f is chosen on the basis of δ in the following way:

$$\left. \begin{array}{l} f : H \rightarrow \mathbb{R}^+ \text{ such that } f(h) = \delta(h_1, h_2) \\ \text{for all } h_1, h_2 \text{ and with } h_1 \cap h_2 = \emptyset \text{ and } h = h_1 \cup h_2 \text{ in } H \end{array} \right\} \quad (1.)$$

Such a choice of f can lead to inversions, and this fact explains the importance of the following proposition, which makes it possible to state two necessary and sufficient conditions that δ must satisfy in order for there to be no inversions.

Let H be the hierarchy obtained with the help of the HAC algorithm (See Appendix 1) equipped with the dissimilarity index δ and f , the application defined by (1). We denote by P_{h_i} the partition that precedes the formation of $h_i = h_{i-1} \cup h_{i-2}$ as the algorithm unfolds (See Figure 1) :

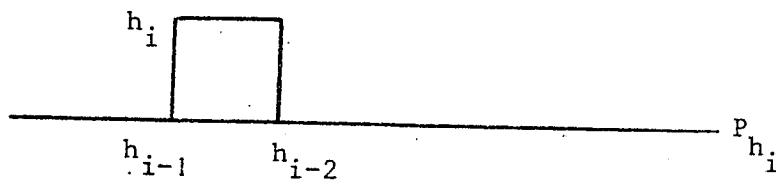


Figure 1

It is then possible to state the following result, see [2] for the proof :

PROPOSITION 1. The following three conditions are equivalent:

- ① (H, f) is a hierarchy indexed in the broad sense.
- ② For all $h_i \in H$, $h'_i \in H$ with $h_i \cup h'_i$ in H , we have $f(h_i \cup h'_i) \geq \max\{f(h_i), f(h'_i)\}$.
- ③ For all $h_j \in P_{h_i}$ with $h_i = h_{i-1} \cup h_{i-2}$ and $h_j \neq h_{i-1}$, $h_j \neq h_{i-2}$, we have $\delta(h_i, h_j) \geq f(h_i)$.

Remark : For clarification purposes, condition ② can be called the "local condition" and condition ③ the "global condition" (See Figure 2):

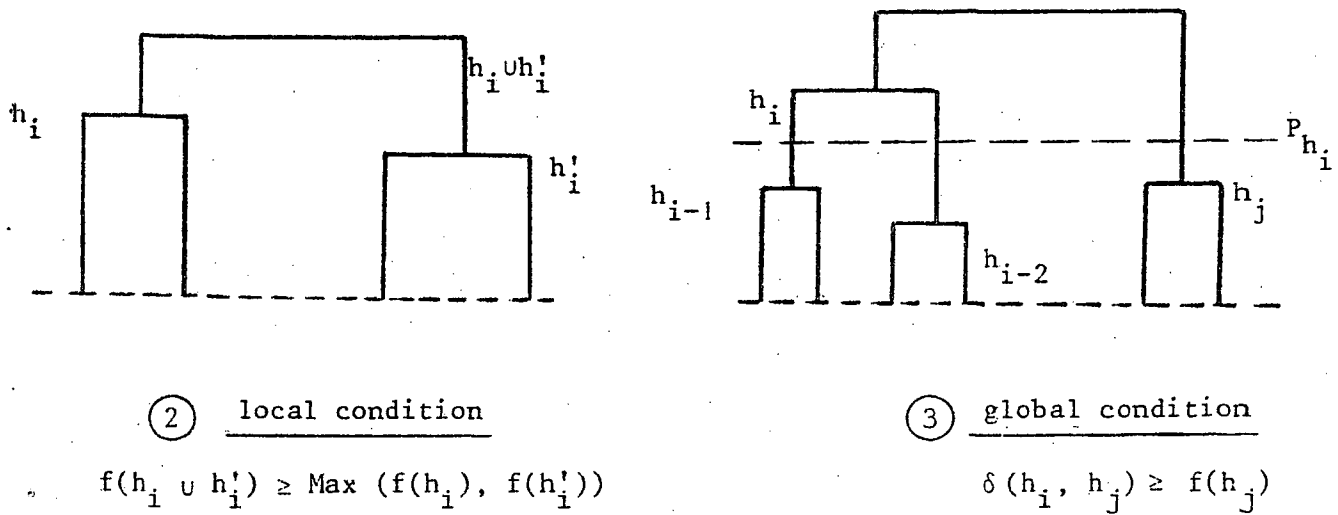


Figure 2

4) The general recurrence formula

When, during the unfolding of the HAC algorithm, two new clusters h_1 and h_2 are brought together in order to form a cluster $h_3 = h_2 \cup h_1 \in H$, it is necessary to perform an update of the $\delta(h_i, h_j)$ for h_i and h_j in P_{h_3} which is different from h_2 and h_1 . This update requires the calculation of $\delta(h, h_1 \cup h_2)$ for all h in P_{h_3} ; it turns out that, for classical choices of the dissimilarity index δ , it is possible to express $\delta(h, h_1 \cup h_2)$ with the help of a recurrence formula, by using $\delta(h, h_1)$, $\delta(h, h_2)$, $\delta(h_1, h_2)$, $f(h)$, $f(h_1)$, and $f(h_2)$, which have been calculated in previous iterations; this makes it possible to save much machine time and memory space; for this purpose, the following general recurrence formula (Jambu 1978) is used:

$$\delta(h, h_1 \cup h_2) = a_1 \delta(h, h_1) + a_2 \delta(h, h_2) + a_3 \delta(h_1, h_2) + a_4 f(h) + a_5 f(h_1) + a_6 f(h_2) + a_7 |\delta(h, h_2) - \delta(h, h_1)|$$

This formula generalizes the classical formula of Lance and Williams (1967), which does not consider the terms in a_4 , a_5 , and a_6 . Various classical choices of the coefficients a_i are given in Appendix 2.

5) A necessary and sufficient condition regarding the coefficients of the recurrence formula for the non-existence of inversions

The hierarchy H must be indexed in such a way that the height of each level corresponds to the value of the dissimilarity index for the two levels that have formed it. In other words, if the following condition is satisfied:

$$\{\text{for any } h_1, h_2, \text{ and } h_3 \text{ in } H \text{ with } h_3 = h_1 \cup h_2, \text{ we have } f(h_3) = \delta(h_1, h_2)\} \quad (2)$$

the following question may be raised:

Is there a necessary and sufficient condition regarding the coefficients (a_1, \dots, a_7) for the non-existence of inversions in the hierarchy H indexed by f ?

The following proposition, which generalizes Ducimetière's remark (1972), Milligan's proposition (1979), and Batagelj's proposition (1981), answer the question.

PROPOSITION 2

A necessary and sufficient condition for any δ satisfying a recurrence formula defined by the coefficients a_1, \dots, a_7 , to induce, using the HAC algorithm, a hierarchy indexed by f without inversions is that the following four conditions be satisfied :

- (a) $a_7 \geq -\text{Min}(a_1, a_2)$
- (b) $a_1 + a_2 \geq 0$
- (c) $a_1 + a_2 + a_3 \geq 1$
- (d) $a_4, a_5, \text{ and } a_6$ are positive.

Proof

To simplify the notations, we will say that condition A is true if relationships (a), (b), (c), and (d) are simultaneously satisfied. Condition B is true if, for any δ satisfying the recurrence formula defined by a_1, \dots, a_7 , the HAC algorithm gives a hierarchy H indexed by f that does not allow any inversions.

Given the symmetry of the recurrence formula, it may be supposed that $\delta(h, h_1) \leq \delta(h, h_2)$ (the proof is similar when the inequality goes in the opposite direction).

Let us first prove sufficiency, i.e., that A implies B.

$\delta(h, h_1) \leq \delta(h, h_2)$ implies that:

$$\delta(h, h_1 \cup h_2) = (a_1 - a_7) \delta(h, h_1) + (a_2 + a_7) \delta(h, h_2) + a_3 \delta(h_1, h_2) + a_4 f(h) + a_5 f(h_1) + a_6 f(h_2).$$

$$(a) \implies a_2 + a_7 \geq 0$$

$$(d) \implies a_4 \geq 0, a_5 \geq 0, a_6 \geq 0, \text{ whence it follows that}$$

$$\delta(h, h_1 \cup h_2) \geq (a_1 + a_2) \delta(h, h_1) + a_3 \delta(h_1, h_2)$$

By construction of the algorithm, we have: $\delta(h, h_1) \geq \delta(h_1, h_2)$

and (b) $\implies a_1 + a_2 \geq 0$, whence it follows that

$$\delta(h, h_1 \cup h_2) \geq (a_1 + a_2 + a_3) \delta(h_1, h_2)$$

Now (c) $\implies \delta(h, h_1 \cup h_2) \geq \delta(h_1, h_2)$, which proves that B is true according to Proposition 1.

Let us now show that $B \implies A$; more precisely, we shall now show that A is false \implies B is false; in other words, our aim is to show that if A is false, there exists a dissimilarity index δ , for which the HAC algorithm gives a hierarchy H indexed by f defined by (2), with inversions.

The fifteen possibilities ($\sum_{n=1}^4 C_4^n = 15$) that falsify A (in logical terms, $\text{not-A} = \text{not } a \wedge b \wedge c \wedge d = \overline{A} = \overline{a} \vee \overline{b} \vee \overline{c} \vee \overline{d}$) are covered by the following four cases:

- 1 $a_2 + a_7 < 0$, or $a_1 + a_7 < 0$.
- 2 $a_1 + a_2 < 0$;
- 3 $a_1 + a_2 + a_3 < 1$;
- 4 a_4, a_5 , and a_6 not all positive.

We shall show that in each one of these cases it is possible to construct a dissimilarity index δ such that

$$\delta(h, h_2) \geq \delta(h, h_1) \geq \delta(h_1, h_2)$$

and $\delta(h_1, h_2) > \delta(h, h_1 \cup h_2)$, i.e.:

$$(1 - a_3) \delta(h_1, h_2) > (a_1 - a_7) \delta(h, h_1) + (a_2 + a_7) \delta(h, h_2) + a_4 f(h) + a_5 f(h_1) + a_6 f(h_2) \quad (3)$$

Let us consider the four cases:

① $a_2 + a_7 < 0$ or $a_1 + a_7 < 0$.

We choose $\delta(h, h_1) \geq \delta(h_1, h_2)$ and

$$\delta(h, h_2) > \text{Max} \left(\frac{(1 - a_3) \delta(h_1, h_2) + (a_7 - a_1) \delta(h, h_1) - F}{a_2 + a_7}, \delta(h, h_1) \right)$$

where $F = a_4 f(h) + a_5 f(h_1) + a_6 f(h_2)$.

② $a_1 + a_2 < 0$

We can choose

$$\delta(h, h_1) = \delta(h, h_2) > \text{Max} \left(\frac{(1 - a_3) \delta(h_1, h_2) - F}{a_1 + a_2}, \delta(h_1, h_2) \right)$$

③ $a_1 + a_2 + a_3 < 1$

We can choose

$$\delta(h, h_2) = \delta(h, h_1) + \epsilon_1 = \delta(h_1, h_2) + \epsilon_2, \text{ with } \epsilon_1 \text{ and } \epsilon_2 \geq 0$$

We must have

$$(a_1 - a_7) \delta(h, h_1) + (a_2 + a_7) (\delta(h, h_1) + \epsilon_1) + F < (1 - a_3) (\delta(h, h_1) + \epsilon_1 - \epsilon_2)$$

or

$$(a_1 + a_2) \delta(h, h_1) + \epsilon_1 (a_2 + a_7) + F < (1 - a_3) \delta(h, h_1) + (1 - a_3) (\epsilon_1 - \epsilon_2)$$

$$C = \epsilon_1 (a_2 + a_7 + a_3 - 1) + \epsilon_2 (1 - a_3) + F < (1 - (a_1 + a_2 + a_3)) \delta(h, h_1)$$

$D = 1 - (a_1 + a_2 + a_3)$ being strictly positive, it is sufficient to take

$\delta(h, h_1) > \frac{C}{D}$ in order for inequality (3) to be satisfied.

⁴ a_4 , a_5 , and a_6 not all positive.

We choose $\delta(h, h_1) \geq \delta(h, h_2) \geq \delta(h_1, h_2)$ and inequality (3) is satisfied if we take $f(h)$, $f(h_1)$, or $f(h_2)$ sufficiently great when a_1 , a_2 , or a_3 is strictly negative. Such a choice is always possible, since there are $n(n-1)/2$ pairs of distances between individuals and $(n-1)(n-2)/2$ distances between clusters that are used, i.e., $(n-1)^2$ distances for only $(n-1)(n-2)/2$ equations (see § 7). \square

It follows from this proposition that a necessary condition for there to be an inversion is that at least one of conditions (a), (b), (c), or (d) not be satisfied.

Other results (for instance necessary and sufficient conditions for the existence of inversions) may be found in Diday (1982).

6) Inversion problems in hierarchical classification with the nearest neighbours algorithm

Two levels h_i and h_j of a hierarchy H are called "nearest neighbours" if h_j is the nearest neighbour of h_i and if h_i is also the nearest neighbour of h_j . On the basis of that concept, the nearest neighbours algorithm works according to the fact that each iteration, that is to say each computation of the matrix of distance, makes it possible to aggregate not only the two less distant nearest neighbours, but all the nearest neighbours existing at this iteration (see Appendix 2). Then the question is if the hierarchy we would obtain with this algorithm is always the same as the hierarchy we would obtain with the HAC algorithm. One can show that there is no deformation of the hierarchy, if the dissimilarity index δ satisfies a property that will be called "nearest neighbours validity condition".

6.1) Nearest neighbours validity condition

To have the same hierarchy with the HAC and nearest neighbours algorithm, it is necessary and sufficient to have the following condition : considering a partition P induced by the hierarchy : when two nearest neighbours nodes h_1 and h_2 merge together, we get a new partition P' such that all the nearest neighbours of P (except h_1 and h_2) stay nearest neighbours in P' .

Indeed, if from one iteration to the next, this condition is not satisfied, we can't merge all the nearest neighbours in each iteration, because a node would be merged with the new level created in P' rather than with his nearest neighbour of P .

PROPOSITION 3

A sufficient condition to get the same hierarchy with the CAH and nearest neighbours algorithms is that the following condition be satisfied :

$$(C) \left\{ \begin{array}{l} \forall h \in P, \quad \delta(h_1, h_2) \leq \inf (\delta(h_1, h), \delta(h_2, h)) \\ \implies \delta(h_1 \cup h_2, h) \geq \inf (\delta(h_1, h), \delta(h_2, h)) \end{array} \right.$$

Proof

Let h_1 and h_2 be two levels which have been merged from partition P to partition P' (they are then necessarily nearest neighbours) ; if two nodes h and h' nearest neighbours in P but not in P' would exist, we should have : $\delta(h, h') > \delta(h, h_1 \cup h_2)$ or $\delta(h, h') > \delta(h', h_1 \cup h_2)$. Let us suppose for example that this condition is true with h . because of condition (C) we then have :

$$\delta(h, h') > \inf (\delta(h_1, h), \delta(h_2, h))$$

and then h and h' would not be nearest neighbours in P .

Thus all the nearest neighbours of P stay nearest neighbours in P' and then the two hierarchies are the same.

Condition (C) will be called nearest neighbours validity condition.

This condition is realized by most of the usual dissimilarity indexes : minimum link, average, diametral, ward.

6.2) Connection between inversion and validity condition

PROPOSITION 4

If the validity condition is satisfied, a hierarchy constructed using the HAC algorithm or by the nearest neighbours algorithm has no inversion.

Proof

We shall show that if δ satisfies the validity condition, then condition 3 of Proposition 1 is satisfied. This so called "global" property can be stated

in the following way (See Figure 2) :

for any $h_j \in P_h$ such that $h_j \neq h_1$, $h_j \neq h_2$ and $h = h_1 \cup h_2 \in H$ } (3)
we have $\delta(h_j, h) \geq \delta(h_1, h_2)$

Now, if $h_j \in P_h$ with $h_j \neq h_1$, $h_j \neq h_2$ and $h = h_1 \cup h_2$, then :

$\delta(h_j, h_1) \geq \delta(h_1, h_2)$ and $\delta(h_j, h_2) \geq \delta(h_1, h_2)$, for P_h is the partition that precedes the merging in the algorithm of $h = h_1 \cup h_2$

Then : $\delta(h_1, h_2) \leq \inf (\delta(h_j, h_1), \delta(h_j, h_2))$

By applying the validity condition one can deduce that :

$\delta(h_j, h_1 \cup h_2) \geq \inf (\delta(h_j, h_1), \delta(h_j, h_2))$

So we have :

$\delta(h_j, h_1 \cup h_2) \geq \inf (\delta(h_j, h_1), \delta(h_j, h_2)) \geq \delta(h_1, h_2)$

then :

$\delta(h_j, h_1 \cup h_2) \geq \delta(h_1, h_2)$

Relationship (3) is thus satisfied ; according to proposition 1, we then have a hierarchy without inversion.

PROPOSITION 5

A necessary and sufficient condition for a dissimilarity index δ satisfying a recurrence formula defined by the coefficients a_1, \dots, a_7 , to verify validity condition of nearest neighbours algorithm, is that the following four conditions be satisfied :

$$(a) \ a_7 \geq -\min(a_1, a_2)$$

$$(b^2) \ a_1 + a_2 \geq 1$$

$$(c) \ a_1 + a_2 + a_3 \geq 1$$

$$(d) \ a_4, a_5, a_6 \text{ are positive}$$

Proof

As in proposition 2, we will say that condition A is true if relationships (a), (b²), (c), (d) are simultaneously satisfied and that condition B is true if for any δ satisfying the recurrence formula defined by a_1, \dots, a_7 , validity condition of

nearest neighbours algorithm is satisfied. We also supposed that $\delta(h, h_1) \leq \delta(h, h_2)$ (given the symmetry of the recurrence formula)

Let us prove A implies B :

considering a partition P induced by the hierarchy at a stage of the algorithm, one must show that :

$$\forall (h, h_1, h_2) \in P^3 :$$

$$[\delta(h_1, h_2) \leq \inf (\delta(h_1, h), \delta(h_2, h)) \Rightarrow \delta(h_1 \cup h_2, h) \geq \inf (\delta(h_1, h), \delta(h_2, h))]$$

or, as $\delta(h, h_1) \leq \delta(h, h_2)$:

$$\delta(h_1, h_2) \leq \delta(h_1, h) \Rightarrow \delta(h_1 \cup h_2, h) \geq \delta(h_1, h)$$

$$\delta(h_1, h_2) \leq \delta(h_1, h) \Rightarrow \delta(h_1 \cup h_2, h) - \delta(h_1, h) \geq 0$$

but, we have :

$$\begin{aligned} \delta(h_1 \cup h_2, h) - \delta(h_1, h) &= a_1 \delta(h_1, h) + a_2 \delta(h_2, h) + a_3 \delta(h_1, h_2) + a_4 f(h) + a_5 f(h_1) \\ &\quad + a_6 f(h_2) + a_7 (\delta(h_2, h) - \delta(h_1, h)) - \delta(h_1, h) \\ &= (a_1 - a_7 - 1) \delta(h_1, h) + (a_2 + a_7) \delta(h_2, h) + a_3 \delta(h_1, h_2) \\ &\quad + a_4 f(h) + a_5 f(h_1) + a_6 f(h_2) \end{aligned}$$

$$(a) \Rightarrow a_2 + a_7 \geq 0 \text{ and we know that } \delta(h_2, h) \geq \delta(h_1, h)$$

$$(d) = a_4 \geq 0, a_5 \geq 0, a_6 \geq 0 \text{ then } a_4 f(h) + a_5 f(h_1) + a_6 f(h_2) \geq 0$$

it follows that :

$$\delta(h_1 \cup h_2, h) - \delta(h_1, h) \geq (a_1 + a_2 - 1) \delta(h_1, h) + a_3 \delta(h_1, h_2)$$

And we have :

$$\delta(h_1, h) \leq \delta(h_2, h) \text{ (condition B)}$$

$$\text{and } (b^2) \Rightarrow a_1 + a_2 - 1 \geq 0, \text{ so it follows that :}$$

$$\delta(h_1 \cup h_2, h) - \delta(h_1, h) \geq (a_1 + a_2 + a_3 - 1) \delta(h_1, h_2)$$

$$(c) \Rightarrow a_1 + a_2 + a_3 - 1 \geq 0 \text{ and } \delta(h_1, h_2) \geq 0 \text{ then}$$

$$\delta(h_1 \cup h_2, h) - \delta(h_1, h) \geq 0$$

So, we have shown that : $\delta(h_1, h_2) \leq \delta(h_1, h) \Rightarrow \delta(h_1 \cup h_2, h) - \delta(h_1, h) \geq 0$

which proves that B is true

Let us show now that $B \Rightarrow A$:

once again, we shall show that $A \text{ false} \Rightarrow B \text{ false}$

so say that $\bar{a} \vee \bar{b}^2 \vee \bar{c} \vee \bar{d} \Rightarrow B \text{ false}$

or still that in the four next cases :

$$\textcircled{1} a_2 + a_7 < 0 \text{ or } a_1 + a_7 < 0$$

$$\textcircled{2} a_1 + a_2 < 1$$

$$\textcircled{3} a_1 + a_2 + a_3 < 1$$

$$\textcircled{4} a_4, a_5 \text{ and } a_6 \text{ not all positive}$$

it is possible to obtain a dissimilarity index δ such that

$$(4) \delta(h_1, h_2) \leq \delta(h_1, h) \text{ and } \delta(h, h_1 \cup h_2) < \delta(h_1, h) \text{ (B false)}$$

now, according to P_2 , cases $\textcircled{1}$, $\textcircled{3}$ and $\textcircled{4}$ each permit to obtain a dissimilarity index δ with inversion, which, according to P_4 insures that δ doesn't verify the validity condition, it's to say that B is false.

We still have to show : $\textcircled{2} \Rightarrow B \text{ false}$

then, we want condition (4) to be satisfied, in other words, to find δ such as :

$$\delta(h_1, h_2) \leq \delta(h_1, h)$$

and

$$\delta(h, h_1 \cup h_2) - \delta(h_1, h) < 0 \text{ i.e.}$$

$$(a_1 - a_7 - 1) \delta(h_1, h) + (a_2 + a_7) \delta(h_2, h) + a_3 \delta(h_1, h_2) + a_4 f(h) + a_5 f(h_1) + a_6 f(h_2) < 0$$

We choose $\delta(h_1, h) = \delta(h_2, h)$, then we want :

$$(a_1 + a_2 - 1) \delta(h_1, h) + a_3 \delta(h_1, h_2) + F < 0$$

$$\text{where } F = a_4 f(h) + a_5 f(h_1) + a_6 f(h_2)$$

$$\text{so : } \delta(h_1, h) > \frac{a_3 \delta(h_1, h_2) + F}{1 - (a_1 + a_2)}$$

we can choose

$$\delta(h_1, h) = \delta(h_2, h) > \text{Max} \left(\frac{a_3 \delta(h_1, h_2) + F}{1 - (a_1 + a_2)}, \delta(h_1, h_2) \right)$$

which proves (4) and then that B is false

According to propositions 4 and 5, conditions (a), (b²), (c) and (d) insure together the validity of nearest neighbours algorithm and non-existence of inversion.

7) Adaptive construction of dissimilarity indices

Given a recurrence formula, one can say, to simplify matters, that if conditions (a), (b), (c), and (d) of Proposition 2 are satisfied and if (b²) is false, then there are no inversions (Proposition 2), and CAH is available. If conditions (a) (b²), (c) and (d) of Proposition 5 are satisfied, then there are inversions and the nearest neighbours algorithm is available.

In appendix 3, we provide a series of classical recurrence formulas followed by a table giving the inversion and validity of nearest neighbours algorithm properties of the corresponding dissimilarity indices. The family of classical dissimilarity indices can be extended by calculating the a_i with the help of the recurrence formula starting from an example. Practically speaking, the user proposes, on the basis of an example drawn from his data, the desired hierarchy H by specifying the height of each of the levels. Beginning with this hierarchy, it is possible to calculate $\delta(h_i, h_j)$, where h_i and h_j are two arbitrary levels of H, by setting :

$$\delta(h_i, h_j) = \text{the height of the lowest level containing } h_i \text{ and } h_j.$$

With the help of these quantities and the general recurrence formula, one obtains a system of equations with 7 unknowns (the a_i , for $i = 1, \dots, 7$) and containing $(n-1)(n-2)/2$ equalities (n being the size of the population) ; indeed :

P_{h_i} being the partition that precedes the formation of the i^{th} level h_i , $n-2$ recurrence equations are used to construct h_2 ; if $n - m$ new equations are needed to construct h_m , then $n - (m + 1)$ are needed to construct h_{m+1} (if h_{m+1} contains h_m , then $n - m - 1$ or else $n - m - 2 + 1$ are needed). In all, therefore, $(n - 2) + (n - 3) + \dots + 1 = \frac{(n-1)(n-2)}{2}$ equations must be used to construct the hierarchy. It is necessary to find the a_i that best satisfy these equations and the constraints given by Proposition 2 to ensure the non-existence of inversions . As a general rule, there are more equations than unknowns; the solution, which is of the least square type with constraints, does not necessarily satisfy all the equalities : inversions may therefore appear ; to mitigate this drawback, the user may call into question his choices in an interactive way (by modifying, for example, the factors of the levels corresponding to these inversions) until he has obtained the coefficients that come closest to satisfying his wishes.

8) Computational Application.

* The search for this similarity index consists in solving an equations system with constraints, whose unknowns are the seven coefficients a_1, a_2, \dots, a_7 from the formula of Lance and Williams generalized by Jambu. So we have to solve a seven unknowns system ; The number of equations is generally greater than seven.

* As a general rule, there is no exact solution to such a system, and a least square type algorithm with constraints has been used to get an approximate solution.

* In addition the construction of the final hierarchy is realized with a nearest neighbours algorithm [4] adapted to the inferred similarity index. (see table 1).

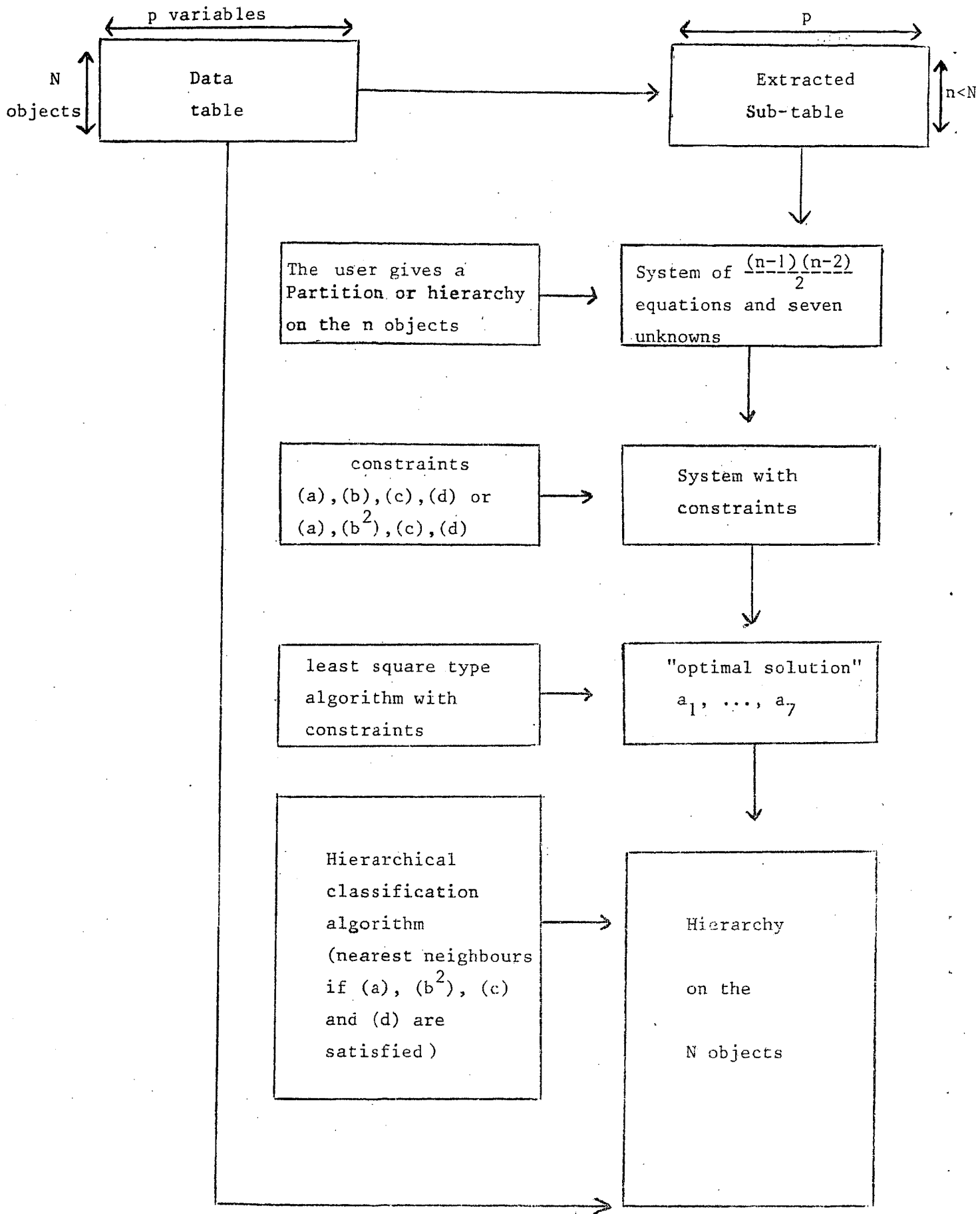


Table 1

* Program inputs :

- The number of objects N and the number of variables P
- A data table (objects, variables) or a table of distances (objects, objects)
- A sub-table $n \times P$ ($n < N$)
- A partition or hierarchy on this sub-table

* Program outputs :

- The inferred index of similarity (i.e coefficients a_1, \dots, a_7)
- Final hierarchy and its sub-classes
- Graphic representation of the hierarchy

* Technical specifications :

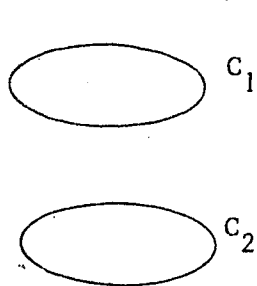
- Programs have been written in FORTRAN 6 on a 68 DPS computer from CII-HB, with MULTICS system.
- CPU Time : including search for the similarity index, construction of the final hierarchy and outputs.

Number of objects	Number of variables	Number of objects of the initial hierarachy	CPU time
20	2	10	13 seconds
185	2	10	46 seconds
40	13	10	31 seconds
120	13	10	50 seconds

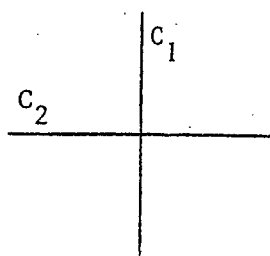
9) Example

The method has been tested on twelve two-dimensional pictures including 185 points of the plane . In these pictures, clusters are visible to the naked eye. We have tested classical aggregation indices and inference indices to try to recognize these pictures.

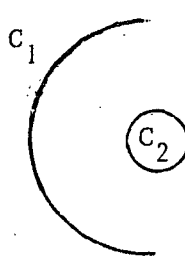
The 12 pictures are :



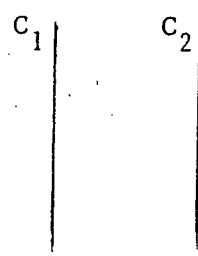
-a-



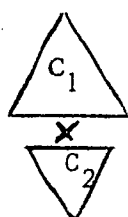
-b-



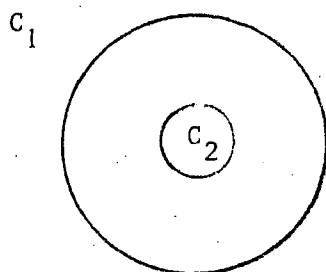
-c-



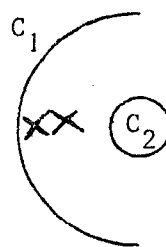
-d-



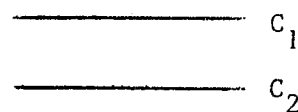
-e-



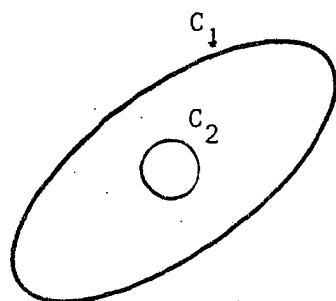
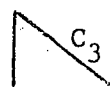
-f-



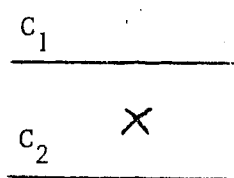
-g-



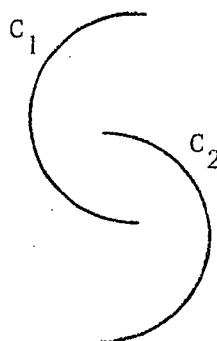
-h-



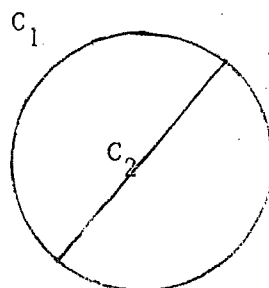
-i-



-j-



-k-



-l-

All the pictures have two clusters except picture h which have three clusters.

For pictures e , g and j , the clustering of the objects between the two clusters will be not taken into account.

Classical indices and inferred indices have been tested on the twelve examples. We have used the HAC algorithm in the case of classical indices and nearest neighbours for inferred indices. Distance between objects is euclidian.

A summary of the results is given in figures 3 and 4.

Aggregation indices	CPU time (in seconds)	Pictures recognized	Number of recognized pictures	Coefficients a_1, \dots, a_7
Minimum Link	125	a, c, d, f, k	5	see appendix 3
Diamétral	121	a, c, e	3	
Average	122	a, e, g	3	
Ward	122	a, c, e, g	4	
Least Inertia	123	a, c, e, g	4	
Least Variance	129	a, e, g	3	
Centroid	123	a, e	2	
Inférence on -a-	47	<u>a</u> , c, d, f, k	5	$a_1 = a_2 = 0,5$ $a_3 = a_4 = a_5 = a_6 = 0$ $a_7 = 0,5$ (minimum link)
Inférence on -b-		no indice recognizing -b- has been found	0	
Inférence on -c-	46	a, <u>c</u> , d, h, k	5	$a_1 = a_2 = 0,5$ $a_3 = a_4 = 0$ $a_5 = 0,32; a_6 = 0,22; a_7 = -0,5$
Inférence on -d-	47	a, c, <u>d</u> , f, k	5	$a_1 = a_2 = 0,5$ $a_3 = a_4 = a_5 = a_6 = 0$ $a_7 = -0,5$ (minimum link)
Inférence on -e-	56	a, c, d, <u>e</u> , f, i, j, k	8	$a_1 = a_2 = 0,59$ $a_3 = -0,1$ $a_4 = a_5 = a_6 = 0; a_7 = -0,55$
Inférence on -f-	79	a, c, d, e, <u>f</u> , i, j, k	8	$a_1 = a_2 = 0,59$ $a_3 = -0,17; a_4 = 0; a_5 = 0,15$ $a_6 = 0; a_7 = -0,54$
Inférence -g-	55	a, <u>g</u>	2	$a_1 = a_2 = 0,5$ $a_3 = 1,85; a_4 = 0,8; a_5 = 0$ $a_6 = 0,53; a_7 = 0,48$
Inférence -h-	46	a, c, d, e, f, <u>h</u> , k	7	$a_1 = a_2 = 0,5$ $a_3 = 0,003; a_4 = 0; a_5 = 0,12$ $a_6 = 0; a_7 = -0,5$
Inférence -i-	67	a, c, d, e, f, <u>i</u> , j, k	8	$a_1 = a_2 = 0,54$ $a_3 = -0,18$ $a_4 = a_5 = a_6 = 0; a_7 = 0,54$
Inférence -j-	49	a, c, d, e, <u>j</u> , k	6	$a_1 = a_2 = 0,5$ $a_3 = a_4 = a_5 = a_6 = 0$ $a_7 = -0,43$
Inférence -k-	46	a, c, d, e, f, h, <u>k</u>	7	$a_1 = a_2 = 0,5$ $a_3 = a_4 = 0; a_5 = 0,000001$ $a_6 = 0; a_7 = -0,5$
Inférence -l-	-	no indice recognizing -l- has been found	0	

Figure 3


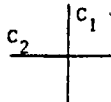

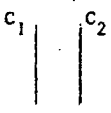
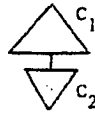
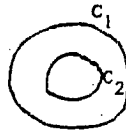
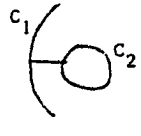
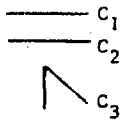
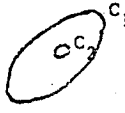
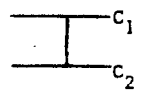
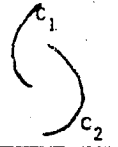
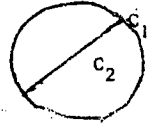
Picture	Classical indices recognizing the picture	number of indices	Inference indices recognizing the picture	number of indices	Total number
	Minimum-link, Diametral, Average, Ward, Least Inertia, Least Variance, Centroid	(sur 7) 7	<u>a</u> , c, d, e, f, g, h, i, j, k	(sur 12) 10	17
		0		0	0
	Minimum-link, Diametral, Ward, Least Inertia	4	<u>a</u> , <u>c</u> , d, e, f, h, i, j, k	9	13
	Minimum-link	1	<u>a</u> , c, <u>d</u> , e, f, h, i, j; k	9	10
	Diametral, Average, Ward, Least Inertia, Least Variance, Centroid	6	<u>e</u> , f, h, i, j, k	6	12
	Minimum-link	1	a, d, e, <u>f</u> , h, i, k	7	8
	Ward, Least Inertia, Least Variance	3	<u>e</u>	1	4
		0	c, <u>h</u> , k	3	3
		0	f, e, <u>i</u>	3	3
		0	e, f, i, <u>j</u>	4	4
	Minimum-link	1	a, c, d, e, f, h, i, j, <u>k</u>	9	10
		0		0	0

Figure 4

* Interpretation of results :

Let's remark first that classical and inference indices all recognize picture a in which the clusters were particularly obvious. Inference indices always recognized very clear clustering.

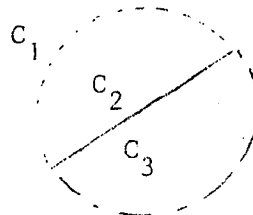
For classical indices, we remark the particularity of minimum link which doesn't recognize the same type of pictures as the others. CPU time is about 125 seconds for all these indices.

For inference indices, CPU time is including between 46 and 79 seconds, which is about two times less than for classical indices. Then we can note the much greater rapidity of the nearest neighbours algorithm in comparison with the classical HAC.

- Inference indices seem to be more adaptable to every sort of configurations than classical indices : six of the ten inference indices recognize more clusters than all classical indices. But some inference indices are, on the contrary, very specific to a special kind of picture ; the indice inferred on picture g, for example, recognize only this picture and the obvious clusters of picture a .

- Inference indices are also able recognize three clusters (pictures h , i , and j), which are never found by classical indices.

- For pictures b and l no indice (inferred or classical) has recognized the desired cluster, so we are never sure that an indice is possible to find, if clusters are very imbricated. For picture l , some inference indices did find the three clusters

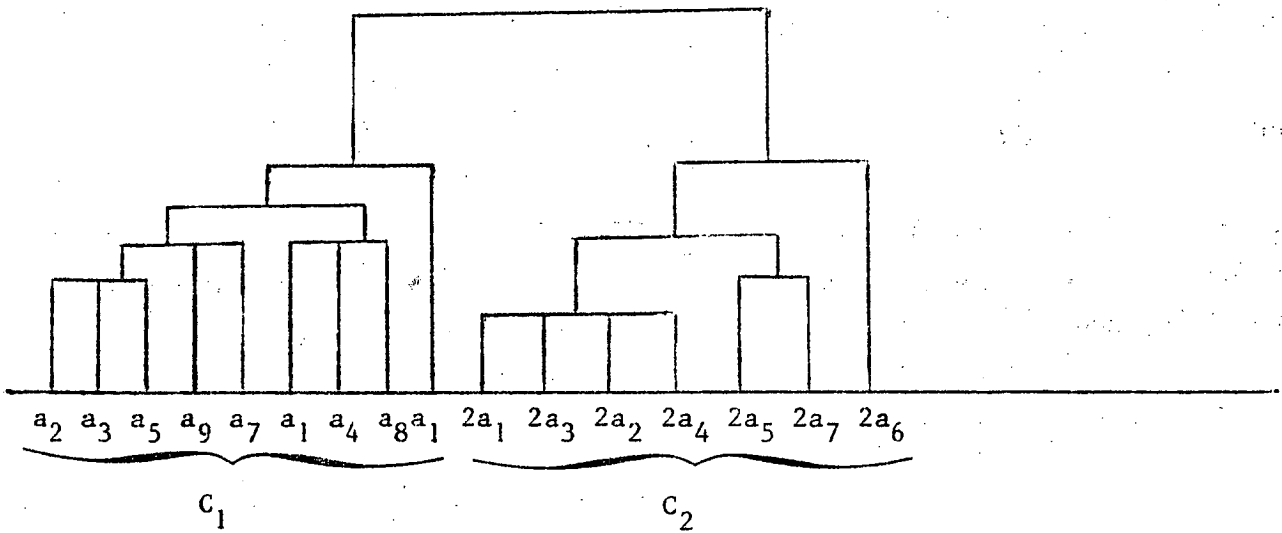


* Splitting threshold choice :

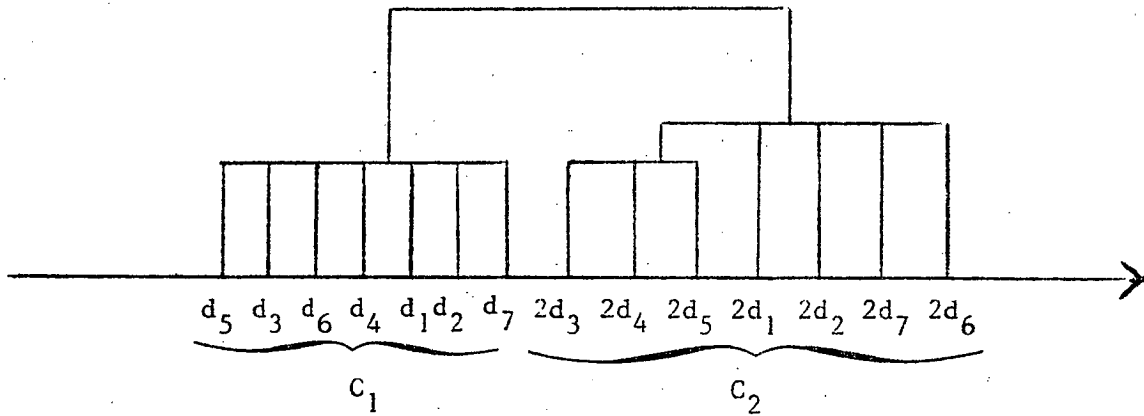
The output of our algorithm is in every case the inferred hierarchy and not a clustering. In order to obtain clusters, this hierarchy has to be truncated.

In the case of the twelve pictures, clusters were always very obvious to find. For example, the aggregation indice inferred on picture -a- is used to induce these hierarchies :

on picture -a-



on picture -d-



and we easily find the two clusters C_1 and C_2 in these hierarchies.

For a more complicated configuration of points, splitting threshold choice could be a difficult problem to solve for the user. It will then be possible to use usual splitting threshold methods from Lerman [8], or to adapt splitting thresholds to the topic under study.

Conclusion

Among all the results put forward, the following rules may be kept in mind : if conditions (a), (b), (c), and (d) are satisfied, the non-existence of inversions is assured ; if besides (b²) is satisfied, the nearest neighbours algorithm may be used. In all areas in which users have a precise idea of the clusters they desire on the basis of particular examples (image processing, for example), the given properties lead to programs for the automatic generation of hierarchies on the basis of examples.

In order to increase the possibilities of adequating the recurrence formula to the example proposed by the user, it would be interesting to study the following general recurrence formula :

$$\begin{aligned}\delta(h, h_1 \cup h_2) = & a_1 \delta(h, h_1) + a_2 \delta(h, h_2) + a_3 \delta(h_1, h_2) \\ & + a_4 f(h) + a_5 f(h_1) + a_6 f(h_2) \\ & + a_7 |f(h) - f(h_1)| + a_8 |f(h) - f(h_2)| + a_9 |f(h_1) - f(h_2)| \\ & + a_{10} |\delta(h, h_1) - \delta(h, h_2)|\end{aligned}$$

With $a_i = F_i(p(h), p(h_1), p(h_2))$, where $p(h_i)$ is a weight associated with level h_i .

APPENDIX I

The general algorithm of the Hierarchical Ascending Classification

This algorithm (called the HAC algorithm) consists in constructing, with the help of the chosen dissimilarity index δ , a sequence of partitions of decreasing fineness, the clusters of which form the desired hierarchy H . It can be stated in the following way :

- ① Start from partition P_0 , the clusters of which are reduced to one element.
- ② Construct a new partition by merging the two clusters of the previous partition that minimize δ .
- ③ Repeat procedure 2 until all the clusters are merged into one.

If in step ② there is more than one pair of clusters that minimize δ , let one be chosen at random ; thus the hierarchy obtained is not always unique. Note also that the hierarchy thus obtained is necessarily binary.

APPENDIX 2

The nearest neighbours hierarchical algorithm.

Two levels h_i and h_j of a hierarchy H are called "nearest neighbours" if h_j is the nearest neighbour of h_i and if h_i is also the nearest neighbour of h_j with a chosen dissimilarity index δ . On the basis of that concept, the algorithm works according to the fact that each iteration makes it possible to aggregate all the nearest neighbours existing at this stage of algorithm. It can be stated in the following way :

- ① Start from partition of individuals, the clusters of which are reduced to one element.
- ② Construct a new partition by combining all the nearest neighbours clusters and aggregate them.
- ③ Repeat procedure ② until all the clusters are combined into one

The hierarchy thus obtained is also binary.

APPENDIX 3

Values of the coefficients of the recurrence formula for some dissimilarity indices.

. Minimum link :

$$\delta_1(h_1, h_2) = \text{Min} \{d(w_1, w_2) / w_1 \in h_1, w_2 \in h_2\}$$

$$a_1 = a_2 = -\frac{1}{2} \quad a_3 = a_4 = a_5 = a_6 = 0 \quad a_7 = -\frac{1}{2}$$

. Single link :

$$\delta_2(h_1, h_2) = \text{Max} \{d(w_1, w_2) / w_1 \in h_1, w_2 \in h_2\}$$

$$a_1 = a_2 = -\frac{1}{2} \quad a_3 = a_4 = a_5 = a_6 = 0 \quad a_7 = \frac{1}{2}$$

. Average :

$$\delta_3(h_1, h_2) = \frac{1}{|h_1| |h_2|} \sum_{\substack{w_i \in h_1 \\ w_j \in h_2}} d(w_i, w_j), \text{ where } |h_i| = \text{card } h_i$$

$$a_1 = \frac{|h_1|}{|h_1| + |h_2|}, \quad a_2 = \frac{|h_2|}{|h_2| + |h_1|}, \quad a_i = 0 \text{ for } i > 2$$

. Centroid :

$$\delta_4(h_1, h_2) = d(G(h_1), G(h_2))$$

$$a_1 = \frac{p(h_1)}{p(h_1) + p(h_2)} \quad a_2 = \frac{p(h_2)}{p(h_2) + p(h_1)} \quad a_3 = \frac{-p(h_1) p(h_2)}{(p(h_1) + p(h_2))^2}$$

$$a_i = 0 \text{ for } i > 3$$

where $p(h) = \sum_{w \in h} p(w)$ is the weight associated with h .

. Least inertia :

$$\delta_5(h_1, h_2) = I(h_1 \cup h_2) \text{ where}$$

$I(h) = \sum_{w \in h} p(w) d(w, G(h))$: $G(h)$ is the mean of h ; $p(w)$ is a weight associated with each individual : $p(h_1 \cup h_2) = p(h_1) + p(h_2)$.

$$\begin{aligned} a_1 &= \frac{p(h) + p(h_1)}{T} & a_2 &= \frac{p(h) + p(h_2)}{T} & a_3 &= \frac{p(h_1) + p(h_2)}{T} \\ a_4 &= -\frac{p(h_1)}{T} & a_5 &= -\frac{p(h_2)}{T} & a_6 &= -\frac{p(h)}{T} & a_7 &= 0 \end{aligned}$$

$$\text{where } T = p(h) + p(h_1) + p(h_2)$$

. Least variance :

$$\delta_6(h_1, h_2) = \text{var}(h_1 \cup h_2) = \frac{1}{p(h_1) + p(h_2)} I(h_1 \cup h_2)$$

$$a_i = \left[\frac{p(h) + p(h_i)}{T} \right]^2 \text{ for } i = 1, 2.$$

$$a_3 = \left[\frac{p(h_1) + p(h_2)}{T} \right]^2 \quad a_4 = \frac{-p(h_1)^2}{T^2} \quad a_5 = \frac{-p(h_2)^2}{T^2} \quad a_6 = \frac{-p(h)^2}{T^2}$$

. Ward (1963) :

$$\delta_7(h_1, h_2) = I(h_1 \cup h_2) - I(h_1) - I(h_2) = \frac{p(h_1) p(h_2)}{p(h_1) + p(h_2)} d(G(h_1), G(h_2))^*$$

$$a_1 = \frac{p(h) + p(h_1)}{T} \quad a_2 = \frac{p(h) + p(h_2)}{T} \quad a_3 = \frac{-p(h)}{T}$$

$$a_4 = a_5 = a_6 = a_7 = 0$$

* If d is a Euclidean distance.

. Index of the weighted increase in variance

$$\delta_8(h_1, h_2) = \text{var}(h_1 \cup h_2) - \frac{p(h_1)}{p(h_1 \cup h_2)} \text{var } h_1 - \frac{p(h_2)}{p(h_1 \cup h_2)} \text{var } h_2.$$

$$a_i = \left[\frac{p(h) + p(h_i)}{T} \right]^2 \quad \text{for } i = 1, 2 \quad a_3 = - \frac{p(h)((p(h_1) + p(h_2)))}{T^2}$$

$$a_i = 0 \text{ for } i > 3.$$

Inversion and reducibility properties

In the following table, conditions (a), (b), (c), and (d) of Proposition 2 are noted as a, b, c, and d : (not -a) is noted as \bar{a} . Condition (b²) of proposition 5 is noted b².

Dissimilarity Indices	Conditions Satisfied	Properties Inversion	Validity of nearest-neighbours algorithm
δ_1	a b ² c d	No inversions [*] ,	yes ^{**}
δ_2	a b ² c d	No inversions [*] ,	yes ^{**}
δ_3	a b ² c d	No inversions [*] ,	yes ^{**}
δ_4	a b ² \bar{c} d	possibly ,	no
δ_5	a b ² c \bar{d}	No inversions ^o ,	yes ^o
δ_6	a b c \bar{d}	possibly ,	no
δ_7	a b ² c d	No inversions [*] ,	yes ^{**}
δ_8	a b \bar{c} d	No inversions ^o ,	yes ^o

* According to Proposition 2.

** According to Proposition 5.

o Since $I(h_1 \cup h_2) \geq I(h_1) + I(h_2)$ (Huyghens) where it follows that $\delta(h_1, h_2) \geq \text{Max}(f(h_1), f(h_2))$, assuming, of course, that d is a Euclidean distance.

BIBLIOGRAPHY

- [1] BATAGELJ V. "Note on ultrametric hierarchical clustering algorithms"
Psychometrika, Vol. 46 n° 3 (1981).
- [2] DIDAY E., LEMAIRE J., POUJET J., TESTU F., "Eléments d'analyse des
données", Dunod, (1982)
- [2bis] DIDAY E. "Inversions en classification hiérarchique, application à la
construction adaptative d'indices d'agrégations". RSA - 1983, Vol. XXXI, N° 1.
- [3] P. DUCIMETIERE, "Les méthodes de classification numérique"
Revue de Statistiques Appliquées, Volume 18, n° 4 p. 5 - 25 (1970)
- [4] J. JUAN, "Le programme Hivor de classification ascendante hiérarchique
selon les voisins réciproques et le critère de la variance",
Cahiers d'Analyse de Données, Vol. 7, n° 2, (1982)
- [5] JAMBU M., "Classification automatique pour l'analyse des données",
Dunod. (1978).
- [6] LANCE G.C., WILLIAMS W.T., "A general theory of classification sorting",
Computer Journal 9.10 and Computer Journal 10.3 (1967).
- [7] MILLIGAN G., "Ultrametric hierarchical clustering algorithms",
Psychometrika 44,3. (1979).
- [8] LERMAN I.C., "Classification et analyse ordinaire des données",
Dunod, Paris, (1981).

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique

