Control and Cybernetics

vol. **24** (1995) No. 4

Dynamic Programming: A valuable algorithm for optimization approaches in clustering?

by

Thierry Gafner

Sous le Chêne 2
2043 Boudevilliers
Switzerland

This paper is meant to develop an idea showing how the dynamic programming algorithm, originally described by Jensen (1969) can be a valuable algorithm for optimization approaches in clustering. The original algorithm is reliable, but it is too slow, because it has to deal with a great number of redundant partial solutions. The way to speed up this algoritm is then to relax its complexity and to reduce the needed amount of memory.

To achieve these goals, some elements taken from Rousseeuw's partitioning around medioids method (1987) and from Rao's integer programming algorithm (1971) were combined.

1. Introduction

This paper considers the problem of *optimal* partitioning a set N, of n entities or objects, into m disjoint and nonempty subsets (clusters). Let $x_{i,j}$ denote the value of characteristic c of entity j for $c = 1, 2, \ldots, p$ characteristics, observations or properties under study and $x_{ij} \in R^Z$ for all entities of the set N.

Let us define what is meant by "optimal solution". The approach used here is based on finding the global optimum (minimum or maximum) to the problem. A numerical criterion is to be optimized in order to determine the degree of *homogeneity* among clusters. Most clustering methods use a function of distance as their criterion. This function measures the *dissimilarity* between each pair of objects $i, j = 1, 2, \ldots, n$. Let $d_{i,j}$ denote the distance between entities i and j of the set N, so that d can be called the *dissimilarity* matrix.

One familiar dissimilarity measure in cluster analysis is the Euclidean distance :

$$d(X_i, X_j) = \left(\sum_{k=1}^{n} (x_i^k - x_j k)^2 \right)^{\frac{1}{2}} \tag{1}$$

Recent studies often use the Manhattan distance or L_1 Norm as their metric:

$$d(X_i, X_j) = \sum_{k=1}^{n} |x_i^k - x_j^k| \tag{2}$$

Given a subset A of N, a real valued function $r(A)$ of the elements belonging to A can be defined. In many cases, this function $r(A)$ is a function of the distances. Such a criterion could be:

$$r(A) = \begin{cases} \frac{1}{\lambda} & \sum_{i<j} d_{ij}^2 \text{ for } \lambda \geq 2 \; i, j \in A \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Then the *optimal partitioning* of the set N into m disjoint and non-empty clusters means choosing the best partition of N according to a certain objective criterion C. This criterion C is a real valued function defined over R^M, where M denotes the set of the m clusters. The problem can be defined by:

$$\text{Optimize } C(r_j) \text{ with } j = 1, \ldots, m \tag{4}$$

There are some well known objective criterions in cluster analysis such as:

Minimizing the total of the average within-group sum of squares of the distances

or

Minimising the total within-group sum of squares of the distances

or

Minimising the maximum of the within-group distances

As Hartigan (1975) noticed: "All clustering algorithms are procedures for searching through the set of all possible clusterings to find one that fits the data well. Frequently, there is a numerical measure of fit which the algorithm attempts to optimize, but many useful algorithms do not explicitly optimize a criterion...".

There is a lot of methods available for clustering. Most of them may not produce an optimal solution, as they trade off optimality for computational ease. On the other hand, the optimal clustering problems are computationally very complex, so that these methods have not been often used, because they need to much system ressources (processor time, main memory, etc...) to run.

1.1. Jensen's dynamic programming algorithm

A general dynamic programming problem can be explained by assuming that one is observing a *system* and that the system moves from one state to another depending on the *action* taken at the beginning of each *stage* of decision-making. On account of the decision made one can have a *reward*, which may be profit made, losses incurred or cost of the action. Any specification of action depending on the state and the stage is called a *policy* and if the policy optimizes the total

reward, it is called an *optimal policy.* The approach for such problems is due to Bellman (1965).

Applied to the clustering problem the above definitions can be denoted as follows:

> A *state* represents a subset A of N, that is one of the many clustering alternatives;
>
> A *stage* represents the "construction" of a given cluster, so that stage 1 denotes all clustering possibilities to build cluster number 1 and stage m denotes the same for cluster m. Stage 0 denotes all separate entities and a final stage is obtained after cluster m has been built;
>
> An *action* is then the clustering of 1 or more entities in a given cluster;

Jensen's original algorithm supposes that there are n_k entities in the set g_k of entities within cluster k of a given alternative for partitioning n entities into m clusters. It uses the following criterion as its homogeneity measure:

$$W = \sum_{k=1}^{m} T(y_k) \tag{5}$$

where

$$T(y_k) = \left(\frac{1}{n_k}\right) \sum_{i,j \in y_k} (d_{ij}^2) \tag{6}$$

is the 'transition cost' of cluster k. This transition cost will be the reward for building cluster k. It is the increase of the objective function. Jensen used the following recursive formula as his objective function for his dynamic programming formulation, using the forward value iteration algorithm:

$$Wk^*(z) = \begin{cases} 0 & \text{for } k = 0 \\ \min_{zk=1,\ldots,M_0} [T(z-y) + W_k + 1^*(y)] & \text{for } k = 1, \ldots, M_0 \end{cases} \tag{7}$$

with

M = number of clusters
$M_0 = M$ if $N = 2M$, and $N - M$ if $N < 2$
k = index of stage
z = index of objects at stage $k+1$
y = index of objects at stage k
$z - y$ = all objects in z, but not in y
$T(z-y)$ = Transition cost between stage k and stage $k+1$

Some other definitions are useful. All clustering alternatives of the n entities in m clusters can be themselves classified according to various *distribution forms.* For example, suppose $n = 4$ and $m = 2$. There are in this case 7 clustering alternatives, each of wich has one of the following distribution forms:

1. Distribution form {3} {1}, where 3 entities are assigned to one cluster and a single one to the other cluster. With this distribution form, there are the following clustering alternatives:

$$\begin{array}{ll} \{1,2,3\} & \{4\} \\ \{1,2,4\} & \{3\} \\ \{1,3,4\} & \{2\} \\ \{2,3,4\} & \{1\} \end{array}$$

2. Distribution form {2} {2}, where 2 entities are assigned to each cluster with the following clustering alternatives:

$$\begin{array}{ll} \{1,2\} & \{3,4\} \\ \{2,3\} & \{1,4\} \\ \{2,4\} & \{1,3\} \end{array}$$

The distribution forms are to be found so that $n_1 \geq n_2 \geq, \ldots, \geq n_m$. In the algorithm the distribution forms are very useful to compute the clustering alternatives, or states.

1.2. Computational problems of the dynamic programming algorithm

Jensen's dynamic programming algorithm assures convergence on the optimal solution. Although a great number of redundant solutions are eliminated in comparison to total enumeration, there are still too many clustering alternatives to be evaluated. For example, if a clustering alternative should contain the entities {1,2,3}, then the symetric alternatives {1,3,2}, {2,3,1}, {2,3,1}, {3,2,1} and {3,1,2} are also evaluated, if no precautions are taken.

Jensen saw this problem and described an alternative formulation to reduce the redundancy of the symetric alternatives. To achieve this, a selective criterion has to be applied. He studied the efficiency of such a criterion. It seemed to him that during large computational problems, it was meaningless to spend time reducing this redundancy, because it was a relatively small additional gain in comparison to total enumaration.

"Even when arc redundancy is not eliminated, the number of transition calculations required under dynamic programming is substantially less than the number of such calculations required under total enumeration in large problems". Jensen (1965).

This redundancy means that a program, based on this algorithm, uses a great amount of memory to store all these states and transition costs, on the other hand "useless" computations are needed, even if the alternative formulation described by Jensen is implemented.

In order to explain what is ment with storing too much data, a comparison between a PC version of Rousseeuw's PAM and an implementation of Jensen's dynamic algorithm can be done. Rousseeuw's program can compute a problem

of size 100 objects and 80 observations, clustering the 100 objects in 20 groups. Jensen's algorithm is limited to 10 objects with 25 observations and can build up to 10 clusters, and has to store a lot of data in temporary files, so that it cannot be runned efficiently. The size of both programs is around 500 KB.

2. Algorithm

The basic idea to relax the complexity of the algorithm is to consider that it is not necessary to compute all the possible clustering solutions generated by Jensen's definitions. For his integer programming algorithm, Rao (1971) introduced the concepts of string condition and leaders. The string condition states that *"in an optimal solution, each group should consist of a least one entity, which for convenience will be denoted as the leader of the group, such that the distance between the leader and any entity that does not belong to the same group is not less than the distance between the leader and any entity within the same group"*. A leader is then the first object that is clustered in a given group. The string condition will be used in the new algorithm to select the entities that are to be clustered in a given group at a given stage of the algorithm, instead of considering any possible combination of the entities, as Jensen did. Rao's string condition will be the selective criterion used to reduce the states' redundancy.

Assume that one state, that is to be generated, should contain the 3 entities {1}, {2}, {3}, that entity {1} is to be considered as the leader and that the distances are so that: $d_{12} \leq d_{13} \leq d_{23}$. Then the only clustering alternative generated in respect to the string condition is: {1,2.3}. Because $d_{12} \leq d_{13}$, the state {1,3,2} is not a valid one with this formulation. But entities {2} or {3} can also be considered as leader. Assume that if entity {2} is the leader, then the only valid clustering alternative is {2,1,3}, and if entity {3} is the leader, then {3,2,1} is the cluster generated for it. For this clustering alternative, three states are then generated. In comparison to Jensen's definition, it is a gain of 50%.

A first trial program was written using this string condition. It has been called DYNARAO, and was very useful to verify, that Rao's string condition was a good selective criterion to reduce the states' redundancy. It will not be exposed in its details in this paper, but will be used in the comparison tables at the end.

With this formulation, all the n entities are considered as *leaders*, wich is surely not very efficient, because only m clusters are to be built. The second idea to reduce the states' redundancy is then to find a way to determine, wich entities are the best *leaders* to form the m clusters.

After Hartigan (1975) developped his solution with the K-Mean algorithm, more technics such as H-Mean, or H-median, were proposed. In the new algorithm, the method used by Rousseeuw, *the medioids*, was preferred to others, because it had been proven that it was more robust.

Assume that the same entities as above have to be clustered. The medioids'

computation will then determine wich entity between {1},{2} and {3} is the best, so that at the end only one clustering alternative has to be considered in the computations for these three entities, instead of the six with Jensen's formulation.

Then the new algorithm can be defined as follows:

Given d a dissimilarity measurement between the objects:

1. Determine the leader (medioids) with Rousseeuw's PAM method:
   The first one is found by:
   $$\text{Find } i | \sum d_{ij} = \min_i \sum d_{ij} \; i, j = 1, \ldots, n \tag{8}$$
   The next ones are computed by the iterative formula:
   $$\text{Find } i | C_i = \max(C_j) \; j = 1, \ldots, n \tag{9}$$
   with
   $$C_j = \sum_q \max(d_{pq} - d_{pj}, 0) \tag{10}$$
   $q = 1, \ldots, n$ (-medioids)
   $p|$medioids
   The set of medioids is then controlled and eventually enhanced:
   While a permutation can be done:
   $$\text{Find } i | C_i > 0 \tag{11}$$
   with
   $$C_i = \sum_q (d_{pq} - d_{qi}) \tag{12}$$
   $$q = 1, \ldots, n(\text{-medioids}) \tag{13}$$
   $q =$ the nearest medioid for p
   $p = 1$ given medioid
   $i = 1$ given object
2. Find the distribution forms of the n objects in m groups so that:
   $$n_1 \geq n_2 \ldots \geq n_m \tag{14}$$
3. Compute the solution
   For f varying from 1 to number of forms dó:
   For x varying from 1 to m do:

   generate the state for medioid x in respect to Rao's string condition and with the number of entities required by distribution form f

   compute
   $$W k^*(z) = \begin{cases} 0 & \text{for } k = 0 \\ \min_{z\, k=1,\ldots,M_0} & [T(z-y) + W_k + 1^*(y)] \\ & \text{for } k = 1, \ldots, M_0 \end{cases} \tag{15}$$
   with
   M = number of clusters

$M_0 = M$ if $N = 2M$, and $N - M$ if $N < 2M$

f = index of distribution form

x = index of medioid

k = index of stage

z = index of objects at stage $k + 1$

y = index of objects at stage k

$z - y$ = all objects in z, but not in y

$T(z - y)$ = Transition cost between stage k and stage $k + 1$

If $Wk^* < W'$, then $W' = Wk^*$ memorize the solution that generated Wk^*. At the end the memorized solution is the optimal one. The DYNARAO solution beginns at step 2 and considers all n entities as leaders, so that loop over x runs n times instead of m.

The homogeneity of a partition is measured with the within cluster sum of squares, as Jensen did in his formulation:

$$W = \sum_{k=1}^{m} T(y_k) \tag{16}$$

where

$$T(y_k) = \left(\frac{1}{n_k}\right) \sum_{i,j \in y_k} (d_{ij}^2) \tag{17}$$

3. Results and conclusion

As a first result, the size of the greatest problem, that can be computed with the new algorithm, is 100 objects, 60 observations and 10 clusters.Rousseeuw's programm can compute a problem of size 100 objects and 80 observations, clustering the 100 objects in 20 groups. The programs used for the following tables were all written in FORTRAN.

The first table shows the goodness of the algorithm by clustering different data sets generated as normal random deviates or by a Monte Carlo method. The result given is the percent of problems that were correctly solved among 10'000. As these simulations prove it, the new algorithm still finds an optimal solution, as Jensen's original does. All these simulations have been done on a VAX-11 machine.

Table 2 shows the average CPU times needed by an AT/286 class machine to cluster some data sets, as an example 6 objects in 2 clusters. It proves that the new algorithm runs faster as integer programming and the DYNAROA solution. No times are given for Jensen's algorithm, as it cannot compute all datasets, because of its limited capabilities (only 10 objects).

The third table shows the CPU time needed to cluster a real dataset with 62 objects and 25 observations each in different numbers of clusters using some modern microprocessors. This table shows, that with the rapid microprocessors,

Method	Normal	Monte-Carlo
PAM	62	65
Rao (Integer Program.)	98	98
Jensen 's algorithm	100	100
Dynamic + Rao's string condition	100	100
NEW DYNAmic algorithm	100	100

Table 1. Efficience of the algorithms

Dataset	Pam	Rao	Dynarao	Newdyna
6/2	1"	1"	1"	0"57
10/2	1"	1"	1"	1"
20/2	2"	15"	1"	1"
6/3	1"	1"	1"	0"81
9/3	1"5	53"	1"60	0"90
12/3	1"79	12"	2"37	0"90
18/3	2"76	118"	11"03	2"35

Table 2. Performance of the algorithms on an 80286 12 Mhz

one can use a complex algorithm without having to wait to long to obtain good results.

In comparison with the classical methods it is still a bit slow. With the PAM algorithm the CPU time never exceeds 30". But as simulations proved it, the new algorithm finds an optimal solution.

Number of clusters	80286 12 Mhz	80486/DX2 66Mhz	Pentium 60Mhz
2	38"	2"	2"
3	2'46"	10"	7"
4	9'31"	33"	25"
5	21'32"	1'11"	51"
6	34'21"	1'24"	57"

Table 3. Performance of NEW-DYNAMIC

References

ARTHANARI, T.S. AND DODGE, Y. (1981) Mathematical Programming in Statistics, Wiley, New York.

BELLMAN, R.E., AND DREYFUS, S.E. (1965) La programmation dynamique et ses applications, Dunod, Paris.

COOPER, M.C. AND MILLIGAN, G. W. (1985) *An Examination of Procedures Determining the Number of Clusters in a Data Set*, Psychometrica, **50**, 159-179.

EVERITT, B. (1974) Cluster Analysis, Social Science Research Council, London.

GAFNER, T. (1992) Analyse critique des méthodes classiques et nouvelle approche par la programmation mathématique en classification automatique, Thèse de doctorat, Université de Neuchâtel, Imprimerie de l'Evole, Neuchâtel.

GARFINKEL, R. S. AND NEMHAUSER, G.L. (1972) Integer Programming, Wiley, New York.

HARTIGAN, J.A. (1975) Clustering Algorithms, Wiley, New York.

JENSEN, R.E. (1969) *A Dynamic Programming Algorithm for Cluster Analysis*, Operation Research, **17**, 103.

RAO, M.R. (1971) *Cluster Analysis and Mathematical Programming*, Journal of the American Statistical Association.

ROUSSEEUW, P. (1987) Cluster Analysis, Wiley, New York.

TRICOT, M. AND DONEGANI, M. (1987) Optimisation en classification automatique sur une famille d'indices de proximité en classification hié rarchique ascendante, EPFL-DMA rapport 8702.

VINOD, H.D. (1964) *Integer Programming and the Theory of Grouping*, Journal of the American Statistical Association.