Control and Cybernetics

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

Critical percolation properties to prove existence of $O(n)$ expected time single linkage clustering algorithm at a predetermined threshold

by

Philippe Lehert

Department of Computer Science,
Facultés Universitaires Catholiques de Mons
151, Chaussee de Binche,
B-7000 Mons, Belgium
E-mail: Lehert@message.fucam.ac.be

Ch. Dumortier

Department of Metallurgy,
Faculté Polytechnique de Mons
Rue de l'Épargne
B-7000 Mons, Belgium

Clustering is known to be demanding in terms of Space and Time complexity. In the difficult case of very large sample size ($n > 10000$), the choice of a particular algorithm is a compromise between optimality and time complexity. In this paper, we suggest the existence of an algorithm, based on hypercubic lattices on p-spaces proved to operate in $O(n)$ expected time, considered as the obvious lower bound of asymptotic complexity. Critical percolation theory was used for complexity demonstration. Experimental designs on simulated data were carried out with sample sizes from $n = 10^3$ to $n = 10^8$, and suggested that this algorithm is probably unique in combining exact output partition and minimum $O(n)$ complexity.

Key-words: complexity, connected components, hashing, minimum spanning tree, percolation, single linkage, square lattice.

1. Introduction

Clustering deals with the problem of grouping the elements of a sample into homogeneous and well separated clusters. Concerning all the proposed clustering algorithms, those converging to a partition with an optimal property, so-called

exact algorithms, are often characterized as demanding in terms of computing time and space allocation. Practically, their use is limited to small data sets (probably $n < 1000$). Recent developments decreased the computational complexity of clustering algorithms, particularly in the important and particular class of the hierarchical agglomerative scheme. By considering a sample size n in p-space, the Brute Force algorithm for hierarchical uglomerative scheme is clearly $O(n^3)$ and can be easily reduced to $O(n^2 \log(n))$ Anderberg (1973). Under utilization of certain ultrametrics, other particular techniques can reduce expected time complexity to $O(n^2)$: Murtagh (1984); Day (1984); Karchaf (1987). In particular, Single Linkage Clustering (SLC) method has interesting properties: it can be obtained through the Minimum Spanning Tree (MST) construction with $O(n^2)$ complexity, Gower, Ross (1969). However, this complexity remains much too demanding when clustering very large data sets. The main effort being connected with the $n(n-1)/2$ distance calculations, some authors suggested ways for alleviating evident useless distances through pre-processing of appropriate geometrical structures (Lattices, B-Sorts, $K-d$ trees), and the best approaches were able to achieve $O(n \log(n))$ expected time, Lehert (1985), and experimentally observed $O(n \log\log(n))$ expected time, Rohlf (1978).

In this paper, we focus on a particular problem of single linkage clustering, known as the clustering by Connected Components (CC) problem: Let E be a set of n points, $d(X,Y)$ a distance for any pair $(X,Y) \in E^2$, and a threshold μ. Let $G(E,F)$ be the undirected graph, with F a subset of E^2 such that $d(X,Y) < \mu$. In other words, the Connected Components $\{CC_1, \ldots, CC_k\}$ are the classes within E such that any two points X and Y belong to the same class if there is at least a path XY in F joining them. CC partition can be constructed on the MST basis, by deleting edges of length $> \mu$. In this respect, CC is one of the single linkage resulting partitions, and it has optimality property: By calling split of a partition, the minimum distance associated with the pair (X,Y), such that X and Y are in two different classes, CC has maximum split among all partitions into the same number of classes. This clustering problem appears in various disciplines. For example, under well defined circumstances, the alien atoms show a marked preference for bonding with their own species and this produces precipitates, which can have beneficial or detrimental effects on mechanical properties. Their influence is particularly strong when they are associated in connected components cluster. Particularly, oxide precipitates are acting in the nickel matrix as barriers to dislocation movement and causing hardening. Such CC distribution of precipitates can also play a major role in crack properties of material propagation and brittle failure.

The dendrogram provided by the MST enables $n-1$ partitions: For large data sets, this dendrogram becomes unreadable. Now, in many applications, μ is fixed – see Levinthal (1966); Rosenfeld (1969), and for such cases, dendrogram construction is no longer needed. Moreover, finding of CC is easier than that of MST. The brute force algorithm for finding CC is clearly $O(n^2)$, and requires the computation of n^2 distances. A particular algorithm has already

been proposed, associated with square lattice structure, and has been proved $O(n)$ expected time, with the limited use of Max-Norm or Tchebychev distance, Lehert (1981). In what follows, we propose a new algorithm for CC construction of $O(n)$ complexity and using any d_r Minkowsky distance.

2. Algorithm

Let there be E, a set of n points $X_i(x_{i1}, \ldots x_{ip})$ in p-space, and without loss of generality, we consider any $x_{ij} > 0$. We define a square lattice $\Omega(\tau)$ with mesh-size τ, as a partition of E into cells C_I, I designating integer n-tuple $(i_1, \ldots i_p)$, such that $C_I = \{X_s \in E | \lfloor x(s,j)/\tau \rfloor = i_j, j = 1, \ldots, p\}$, $\lfloor q \rfloor$ designating the smallest integer equal to or greater than q.

We will first define the direct proximity and full proximity concepts: Let Γ be the set of nonempty cells C_I, and by denoting a Minkowsky distance of order r by d_r (d_1 = City Block distance, d_2 = Euclidean distance, and d_∞ = Tchebychev or Max-Norm Distance), we define Direct Proximity $DP(I) = \{C_J \in \Gamma | d_1(I,J) = 1\}$, and Full Proximity $FP(I) = \{C_J \in \Gamma | d_\infty(I,J) \leq 1\}$. In the particular case of the plane (2-space), the direct proximity of a cell C_I is the set of non empty cells C_J sharing a common edge with C_I. Their maximum number can be 4 in the plane and $2p$ in p-space. The full proximity $FP(I)$ is the set of cells having a least a common point with C_I: 9 at most can exist in the plane and 3^p in p-space. The algorithm's rationale is based on the following elementary geometrical properties:

PROPERTY 2.1 *Under a Minkowsky d_r distance in p-space, a predetermined threshold μ and the associated CC partition, a square lattice $\Omega(\tau = \mu(p-1+2^{r)-1/r})$ determining a set of cells $\{C\}$, any point X belonging to a cell C_I and any other point Y belonging to C_J included in $DP(I)$ belong to the same class of the CC partition.*

Proof: In the case where C_J belongs to $DP(I)$, the maximum distance between two points X of C_I and Y of C_J is the length of the diagonal constructed on the hyper rectangle $\{C_I \cup C_J\}$. By fixing this length to μ, all the points C_I will be in the same CC-class. Now, in the plane and with the usual Euclidean distance, μ and τ are such that $\mu^2 = \tau^2 + (2\tau^2)$, and more generally in p-spaces with any Minkowsky distances, $\mu^r = (p-1)\tau^r + (2\tau)^r$.

We now consider a square lattice $\Omega(\tau)$ and construct a partition P as follows: Let P_1 be initialized by any C_I. From property 1, not only all the points of C_I are in the same CC class, but also any point $Y \in DP(I)$ is such that point X of C_I exists with $d_r(x,y) < \mu$, and therefore Y belongs to the same CC partition. In such a manner, all the points of $DP(I)$ can be affected automatically to P_1 class, without any distance calculation, by merely investigating the $2p$ possible cells of $DP(I)$. Any newly entered cell C_J will cause systematic $DP(J)$ annexion, until all the $DP(J)$ have been investigated. P_2 will be initialized by a nonempty cell

not yet annexed in P_1, and the same investigation is made, until all the cells have been annexed into one class. P partition $(P_1, \ldots P_k)$ can be defined by the undirected graph $G(\Gamma, \Phi)$, with $\Phi = \{(C_I, C_J) | C_I \in \Gamma,\ C_J \in \Gamma,\ C_J \in DP(I)\}$. Although P partition is close to the searched CC partition, there is still a difference: Any two points connected in P, will also be connected in CC, but one can find connected points in CC, not connected in P: indeed an hypersphere centered on a point X of C_I contains space areas not in $DP(I)$, that can contain a point Y such that $d_r(X,Y) < \mu$. Such pairs of points XY are erroneous links.

For a correct CC construction, it is required to detect these erroneous links by examining the neighbourhood of each point X, and assessing existence of a point $Y \in$ a different class of P, such that $d_r(X,Y) < \mu$. At this stage, it is no longer possible to avoid distance calculations, but in order to limit them, another square lattice nested or embedded on the original lattice will be successfully needed, as shown by the following property:

PROPERTY 2.2 *A new embedded square lattice $\Omega(\tau = \lfloor \mu/\tau \rfloor \tau \rfloor)$ producing the set of non empty cells $\{E\}$ is such that for any point $X \subset E_I$, the full proximity $FP(I)$ of a cell C_I contains the r-ball $B_r(X,\mu) = \{Y \in E | d_r(X,Y) < \mu\}$. $\Omega(\tau)$ based on the initial lattice allows to limit the distance calculations for finding the erroneous links in a the smallest possible space area: Space investigation for each point is limited to the maximum 3^p cells of $\Omega(\tau)$ constituting $FP(I)$.*

As a consequence, the CC algorithm has two stages: First, it builds a P partition, by simple direct proximity annexation, and second, there follows detection and suppression of erroneous links. A basic Pidgin Algol code is now derived:

Step 1. Finding of P partition on the graph $G(\Gamma, \Phi)$:

a) Pre-processing: Constructing square lattice $\Omega(\tau)$ defining the $\{C\}$ cell set; $k = 0$;

b) Determination of P partition
For any cell not annexed to P
do

$k = k + 1$;

$P_k = \{C_l\}$, where C_l designates a non annexed cell – for any $C_I - P_k$ to investigate,

do

$P_k = P_k + \{DP(I)\}$

end

end

Step 2. Finding erroneous links and final CC class search

a) Pre-processing: Transformation embedded $\Omega(\tau = \lfloor \mu/\tau \rfloor \tau \rfloor)$ producing $\{E\}$ set.

b) For any class P_k of P,
do

For any $X \in E_I$, within P_k
do

For every $Y \subset FP(I)$, $\subset P_l | l \neq k$,
do

if $d_r(X,Y) < \mu$, P_l annexed to P_k

end

end

end

3. Complexity

Let us prove the expected time complexity of the algorithm, under a general hypothesis of n points in p-space, and a fixed d_r distance of Minkowsky ($0 < r \leq \infty$). Step 1 constructs the original square lattice $\Omega(\tau)$: By using an appropriate hashing structure calculated on the keys I of the cells C_I, this requires $O(n)$ calculations Lehert (1991). Since direct access is provided to the cells, and $DP(I)$ contains $2p$ cell accesses $O(p)$, total computational time for step 1 and for N nonempty cells ($N < n$) is

$$T(\text{step } 1) = O(n) + 2NpO(p) < O(n) + 2pO(np) \leq O(np^2) \tag{1}$$

Step 2 investigates full proximities $FP(I)$ for each point X in the $\Omega(\tau)$ embedded lattice. At this stage, we need to define the discrete lattice (N^p, Θ), whose nodes are constituted by the integer p-tuples $I(i_1 \ldots i_p)$ of N^p and $\Theta = \{(I,J) | I \text{ and } J \in N^p \text{ and } d_1(I,J) = 1\}$. Let us consider a subset M of N^p, randomly selected from N^p so that we can consider spatial constant density λ and Θ' is the subset of Θ such that the extremities I and J are both in M. The random graph (Γ, Θ') includes a number of connected components which increases with density λ. In this context, we notice the following property:

PROPERTY 3.1 *In a discrete lattice defined on p-space, with uniform density, the expected number of connected nodes becomes infinite, once density is higher than a value π, known as the critical percolation threshold being only a function of the kind of lattice and the space dimensionality.*

This general property was demonstrated in R^2 et R^3 in Hammersley (1961); Reh (1979), and generalized to p-spaces Santalo (1976). A general analytic expression of π is intractable and can only be approached by simulations, Hammersley (1961).

A direct application of this property consists in observing that in an defined area of p-space, small enough to consider a uniform density λ, the undirected

graph (Γ, Ω') is such that Γ is the subgroup $\{J\}$ of N^p corresponding to the nonempty cells C_j. Let us now consider a bipartition of p-space into the low density area, where the density of nodes C_I is lower than π, and the high density area, where node density is larger than π.

1. Within the low density area ($\lambda < \pi$), investigation around each point X needs 3^p cell accesses, with each access requiring $O(p)$ effort. Within each cell, the maximum number of distance calculations between any point Y of the cell and X is bounded by π (finite, only dependent on p and independent of the data). By designating the computional time for step 2 in low density area by $T(\text{step } 2)|\lambda < \pi)$, we have:

$$T(\text{step } 2)|\lambda < \pi) < \pi(3^p(1 + \pi O(p)) \approx O(np3^p) \tag{2}$$

2. Within the high density area ($\lambda > \pi$), let us consider the graph (Γ, Θ'). Since it constitutes a discrete square lattice in p-space and the density is higher than the critical percolation value π, the expected number of connected cells is infinite, in other words, step 1 has already connected them without distances calculations. Finding erroneous links in step 2, should also involve a low number of distance calculations. By considering a linear linked list joining all the points of cell C_I, before all, all the points belonging to the cells annexed in the same class of P will be excluded for distance calculations. In conclusion, computational complexity for step 2 in high density areas $T(\text{step } 2)|\lambda > \pi)$ is limited to 3^p cell accesses, and a negligible expected number of distance calculations:

$$T(\text{step } 2)|\lambda > \pi) = n \cdot 3^p O(p) \tag{3}$$

From partial results $(1, 2, 3)$, we derive $O(np)$ expected time complexity, and as a consequence $O(n)$ linear expected complexity for a definite dimension p.

4. Experiments

Computer times were measured on artificial data sets whose values were obtained from mixtures of gaussian distributions themselves characterized by randomly generated parameters. The CC algorithm was implemented in two versions (version CC1 in Microsoft C++ 6.0 under DOS, version CC2 under F77/386 Fortran under DOS extender). In order to obtain results that would be hardware independent, the performance of the CC algorithm has been compared to BF-MST algorithm of Bentley and Friedman (1978), considered as probably the fastest for MST construction. Results are expressed as ratios between computational times (t_{MST}/t_{CC}) measured by the CC algorithm to those obtain by application of the Gower and Ross (1969) MST algorithm used as reference

method:

		$BF-MST$	$CC1$	$CC2$
$p=2$	$n=10^3$	33	290	315
	$n=10^5$	2807	39385	48960
	$n=10^6$	–	$5.32 \cdot 10^5$	$9.32 \cdot 10^5$
	$n=10^7$	–	$4.23 \cdot 10^6$	$7.89 \cdot 10^6$
$p=3$	$n=10^3$	16	134	199
	$n=10^5$	167	1345	1978
$p=5$	$n=10^3$	8	89	157
	$n=10^5$	189	867	1336

As an example, first line of this tables shows that BF-MST, CC1 and CC2 versions are 33, 290 and 315 times faster than the basic Gower-Ross Algorithm, respectively. These results confirm the linear behaviour of the algorithm with n for each value of p. The gains t_{MST}/t_{CC} decrease with p and are especially important for large data sets. For p exceeding 5, n must be greater than 10^5 to preserve an appreciable gain: the exponentially increasing number of cells accesses compensates the distance calculations economy and annihilates the benefits of the algorithm. The better performances of CC2, compared with CC1 is probably coming from better virtual memory allocation of the F77 compiler, and generally better optimized object code of Fortran for vector management, compared with C.

5. Conclusions

This paper described a clustering algorithm that combines an important property characteristic for the single linkage hierarchical scheme, and a time complexity that should probably be the lower bound complexity since clustering of n points requires at least reading them. As far as we know, no other algorithm has been proposed that combines such a low complexity while preserving optimality property. Another advantage of this algorithm is its possibility to work in paginated environment: pages are simply the cells. By doing so we do not need to store the data matrix in central memory. It means that this algorithm can analyse data sets with sizes only limited by the hard disk capacity. In order to compare the practical performances of the algorithm, experimentations have been realized by simulation in p-spaces ($p < 6$). Their results confirm the $O(n)$ expected time and the influence of the critical percolation value π. Now, the same simulations clearly exhibited the limits of the algorithm, in pointing out dramatically reduced performances when dimensionality p increases. As a consequence, although theoretically claiming a simultaneity between optimality and minimum complexity, which is an interesting result, this algorithm's interest is obviously limited to very large data sets ($n > 10000$) in low dimensionalities ($p < 5$).

References

ANDERBERG M.R. (1973) *Cluster Analysis for Applications*, New York, Academic Press.

BENTLEY J.L., STANAT D.F. AND WILLIAMS E.H. (1977) The Complexity of finding fixed-radius near neighbours, *Inf. Proc. Letters*, 6.6, pp. 209-213.

BENTLEY J., FRIEDMANN J.M. (1978) Fast Algorithms for Constructing Minimal Spanning Trees in Coordinate Spaces, *I.E.E.E. Trans. on Computers*, vol C-27, pp. 97-104.

BRUYNOOGHE M. (1978) Classification Ascendante Hiérarchique, un algorithme rapide fondé sur la construction des voisinages réductibles. *Les cahiers de l'analyse de données*, III, pp. vy-33.

DAY W.H.E. (1984) Efficient algorithms for agglomerative hierarchical clustering methods, *J. of Classification*, 1, pp. 7-24.

GOWER J.C., ROSS J.S. (1969) Minimum Spanning Trees and Single Linkage Cluster Analysis, *Applied Statistics*, 18, pp. 54-64.

HAMMERSLEY J.M. (1961) On the rate of convergence to the connective constant of the hypercubical lattice, *Quart. J. Math.* 2-12, pp. 250-256.

KARCHAF I. (1987) Sur la complexité des algorithmes de classification ascendante hiérarchique, *Les cahiers de l'analyse des données*, XII, pp. 195-197.

LEHERT PH. (1981) Clustering by Connected Components in $O(n)$ expected time, *R.A.I.R.O. Computer Science*, 28.

LEHERT PH. (1985) Ultramétrique inférieure maximale et Complexité, *Data Analysis and Informatics*, Diday, North Holland.

LEVINTHAL C. (1966) Molecular model building by computer, *Scientific American*, 214, pp. 42-52.

MURTAGH F.,A (1984) Complexities of hierarchical clustering: State of the art, *Computational Statistics Quartely*, 1, pp. 101-113.

REH W. (1979) First Passage Percolation under weak moment conditions, *J. App. Prob*, 16, pp. 750-763.

ROHLF F.J. (1978) A Probabilistic Minimum Spanning Tree Algorithm, *Information Processing Letters*, 7, pp. 44-48.

ROSENFELD A. (1969) *Picture Processing by Computer*, Academic Press, New York.

SANTALO L.A. (1976) Integral Geometry and Geometric Probability, *Encyclopedia of Mathematics and its Applications*, v.1. Addison Wesley, Reading, MA.