CLUSTERING CON MASSIMA SEPARAZIONE SU UN ALBERO

M. MARAVALLE, Facoltà di Economia e Commercio
Università degli Studi dell'Aquila
B. SIMEONE, Dip. di Statistica, Probabilità e Statistiche Applicate
Università "La Sapienza" - Roma

ABSTRACT

The present paper deals with the following problem: given a tree with n vertices and a dissimilarity d_{ij} for each pair (i,j) of vertices, partition its set of vertices into p classes such that each class induces a subtree and the split of the partition is maximized. Applications include paging of hierarchical data bases and districting of a tree-like distribution or communication network. We describe an $O(n^3)$ *greedy* algorithm for finding an optimal partition.

KEYWORDS : separation, trees, greedy algorithm.

1. INTRODUZIONE

Lo scopo fondamentale della *cluster analysis* è quello di classificare un insieme di oggetti in sottoinsiemi, o clusters, seguendo due criteri antitetici: OMOGENEITA' (oggetti dello stesso gruppo dovrebbero essere simili) e SEPARAZIONE (oggetti di gruppi differenti dovrebbero essere dissimili fra loro).

In questo articolo verrà trattato solo il secondo criterio. Si indichino gli n oggetti da classificare con i numeri 1,2,....,n, cosi' che l'insieme di oggetti sia immediatamente identificabile con l'insieme standard $V \equiv \{1,2,.....,n\}$. Si supponga che venga fornito un indice di dissimilarità d_{ij} per ogni coppia (i,j) di oggetti. Sia $\pi \equiv \{ C_1,C_2,....,C_p\}$ un'arbitaria partizione di V in p sottoinsiemi (*gruppi o clusters*), dove $1 \leq p \leq n$.

Una comune misura di separazione è il divario (split) . Il *divario* di π viene definito come il minimo indice di dissimilarità fra due oggetti appartenenti a clusters differenti :

$$s(\pi)=\min \{ d_{ij} : i \in C_h , j \in C_k , h \neq k \} . \qquad (1)$$

Come hanno dimostrato Delattre ed Hansen (1980) , il ben noto algoritmo del legame singolo dà luogo, per ciascun valore di p=1,2,.....,n, ad una partizione in p classi con massimo divario. Dal punto di vista pratico, però, non tutte le partizioni possono essere considerate come "accettabili". Per esempio se gli oggetti fossero città di una certa regione, verrebbe usualmente richiesto che ciascun gruppo sia formato da città geograficamente contigue. Se gli oggetti fossero "records" in un data base relazionale, ciascun record dovrebbe essere relazionalmente accessibile da ciascun altro record dello stesso gruppo e cosi' via.

Situazioni di questo tipo possono essere trattate in modo naturale con modelli di questo tipo : gli n oggetti da classificare vengono identificati come vertici di un grafo G , e una partizione $\pi \equiv \{ C_1,C_2,....,C_p\}$ dell'insieme V dei vertici di G viene dichiarata *ammissibile* se, per ciascun k=1,2.....,p il sottografo $G(C_k)$ indotto da C_k e' connesso (Per quanto riguarda la terminologia dei grafi ci si riferisce a Cerasoli, Eugeni e Protasi 1988). Per una rassegna sui metodi di clustering vincolato si può consultare Murtagh 1985.

Si indichi con $\Pi_p(G)$ l'insieme di tutte le partizioni ammissibili di V. Si cerca una $\overline{\pi} \in \Pi_p(G)$ tale che

$$s(\overline{\pi}) = \max [\, s(\pi) : \pi \epsilon \Pi_p(G)] \,. \qquad (2)$$

Data la partizione $\pi \epsilon \Pi_p(G)$, uno spigolo di G è chiamato *interno* (rispetto a π) se è uno spigolo di un qualsiasi sottografo $G(C_k)$; altrimenti lo spigolo è chiamato *esterno*.

In questo lavoro si considera il caso particolare in cui il grafo è un albero $T \equiv (V,E)$. Questo caso si presenta, ad esempio, allorché si abbia a che fare con un data base gerarchico, o con una rete di distribuzione o di comunicazione ad albero (gasdotto, rete locale, ecc.). Nel caso di un albero una partizione ammissibile è caratterizzata nel modo che segue: se si "tagliano" (cioè si eliminano) p-1 spigoli scelti arbitrariamente, si ottiene una foresta con p componenti connesse $C_1,C_2,....,C_p$. Allora $\pi \equiv \{ C_1,C_2,....,C_n\}$ è una partizione ammissibile di V in p gruppi. Per contro, ciascuna partizione $\pi \epsilon \Pi_p(T)$ può essere ottenuta in questo modo. Invero ci sono esattamente $p-1$ spigoli esterni rispetto a π : tagliandoli si ottiene precisamente π . Esiste pertanto una corrispondenza biunivoca tra insiemi di $p-1$ spigoli e partizioni ammissibili di V in p clusters. Se $X \subseteq E$ ed $|X| = p-1$, si indicherà con π_X la corrispondente partizione ammissibile.

Nel paragrafo 2 viene descritto un algoritmo *ghiotto* (greedy) per trovare una partizione $\pi \epsilon \Pi_p(T)$ con divario massimo. La complessità computazionale dell'algoritmo è $O(n^3)$, dove n e' il numero di vertici dell'albero T.

2. L'ALGORITMO

Questo paragrafo è dedicato alla descrizione di un algoritmo che, dato un albero $T=(V,E)$ con n vertici, una matrice di dissimilarità simmetrica $D_{nxn}= [d_{ij}]$, e un intero p , $1 \le p \le n$, permetta di calcolare una partizione $\pi \epsilon \Pi_p(T)$ avente massimo divario.
L'algoritmo consiste di tre fasi.

Nella fase 1 le $\mathcal{N} \equiv \binom{n}{2}$ coppie (i,j), $1 \le i<j \le n$, vengono ordinate in modo tale che

$$d_{i_1 j_1} \le d_{i_2 j_2} \le \ldots\ldots\ldots\ldots \le d_{i_{\mathcal{N}} j_{\mathcal{N}}}$$

Nella fase 2 a ciscuno spigolo è assegnata una etichetta (detta *rango* dello spigolo) con la seguente procedura in 5 passi:

Passo 1 : Si pone $k=1$ e $r=1$;

Passo 2 : Si genera l'insieme E_k degli spigoli del cammino (unico) su T con estremi i_k e j_k;

Passo 3 : Se tutti gli spigoli di E_k sono dotati di etichetta si va al Passo 5; altrimenti si assegna l'etichetta r a tutti gli spigoli di E_k che ne sono sprovvisti ;

Passo 4 : Si incrementa r di una unità;

Passo 5 : Se tutti gli spigoli di T sono stati etichettati, FINE ; altrimenti si incrementa k di una unità e si ritorna al Passo 2.

FINE

Nella fase 3 vengono eliminati i $p-1$ spigoli di T che hanno il rango più elevato. La partizione corrispondente risulterà avere il massimo divario. Un esempio della intera procedura viene riportato nel paragrafo 3.

Alla luce della fase 3 l'algoritmo può essere considerato *ghiotto* (greedy).

Allo scopo di implementare il Passo 2 della fase 2 è conveniente rappresentare l'albero T come un'arborescenza (*rooted tree*- viene utilizzata la terminologia di Aho,Hopcroft e Ullman 1975). Come radice dell'arborescenza viene scelto un vertice qualsivoglia di T. Ad eccezione della radice, ogni vertice i ha un unico predecessore *pred(i)*. Dunque l'arborescenza può essere rappresentata mediante la funzione pred($\bullet$) (indice dell predecessore). Usando l'indice del predecessore si può trovare l'antenato comune piú prossimo (*deepest common ancestor*) w_k di i_k e j_k . Il cammino (unico) che connette i_k a w_k è la concatenazione dei due cammini a ritroso (*backpaths*) da i_k a w_k e da j_k a w_k.

Prima di dimostrare la correttezza dell'algoritmo sarà utile introdurre alcune notazioni.

Per ogni $r=1,2,\ldots\ldots$ sia p(r) ilpiù piccolo indice k tale che esista almeno uno spigolo in E_k con rango r.

Per ogni $X \subseteq E$ sia $f(X) = s(\pi_X)$ e sia r(X) il minimo rango di uno spigolo in X.

La dimostrazione della correttezza dell'algoritmo si basa sul seguente lemma.

Lemma : Per tutti gli $X \subseteq E$ si ha:

$$f(X) = d_{i\,p(r(X))\,j\,p(r(X))} \qquad (3)$$

Dimostrazione. Sia $r=r(X)$. Per definizione di p(r) esiste almeno uno spigolo con rango r lungo il

cammino μ che collega i vertici $i_{p(r)}$ e $j_{p(r)}$; per di piu', tutti gli spigoli con rango r devono trovarsi sul cammino μ per come è definito il Passo 3. D'altra parte, f(X) è il divario della partizione ottenuto eliminando tutti gli spigoli in X. Tra questi spigoli ce ne sarà almeno uno con rango r . Poiche' questo spigolo è stato tagliato, i vertici $i_{p(r)}$ e $j_{p(r)}$ apparterrano a clusters differenti. Di conseguenza $f(X) \leq d_{i_{p(r)}j_{p(r)}}$

Supponiamo ora che $f(X) < d_{i_{p(r)}j_{p(r)}}$. Esiste un indice h tale che $f(X) = d_{i_h j_h}$.

Poichè la successione $\left\{ d_{i_k j_k} \right\}_{k=1,2.....}$ è non decrescente, si deve avere $h<p(r)$.

Per definizione di p(r), tutti gli spigoli lungo il cammino E_h devono avere rango $< r$. Ma almeno uno degli spigoli appartiene ad X, poichè i_h e j_h cadono in differenti clusters. Il rango di tale spigolo dovrebbe essere strettamente minore di $r=r(X)$ in contraddizione con quanto precedentemente detto. Riesce allora che

$$f(X) = d_{i_{p(r(X))}j_{p(r(X))}}$$

Teorema L'algoritmo è corretto e la sua complessità è dell'ordine di $O(n^3)$.

Dimostrazione Si noti dapprima che quando l'algoritmo termina tutti gli spigoli devono aver ricevuto un'etichetta. Allo scopo di trovare una partizione $\overline{\pi} \in \Pi_p(T)$ avente divario massimo, bisogna massimizzare f(X) tra tutti gli $X \subseteq E$ tali che $| X | = p - 1$. Perciò si deve massimizzare r(X) tra tutti gli $X \subseteq E$ tali che $| X | = p - 1$ (questo segue dalla (3) e dal fatto che la successione $\left\{ d_{i_{p(r)}j_{p(r)}} \right\}_{r=1,2.....}$ è non decrescente). Questo è precisamente quello che fa l'algoritmo nella fase 3. Conseguentemente l'algoritmo è corretto.

Passiamo ora ad analizzare la complessità dell'algoritmo. Nella Fase 1, l'ordinamento degli $\mathcal{N} \equiv \binom{n}{2}$ numeri d_{ij}, $1 \leq i<j \leq n$, richiede $O(n^2 \cdot \log_2 n)$ confronti. La complessità computazionale della Fase 2 è $O(n^3)$ poichè la generazione dell'insieme E_k nel Passo 2 richiede O(n) operazioni e bisigna generare al più $|\mathcal{N}|=O(n^2)$ insiemi E_k. Infine la Fase 3 richiede anch'essa O(n) confronti (si veda Aho,Hopcroft ed Ullman 1975). Quindi è possibile valutare globalmente la complessità computazionale dell'algoritmo che risulta $O(n^3)$.

3. UN ESEMPIO

Si consideri l'albero mostrato nella Fig. 1

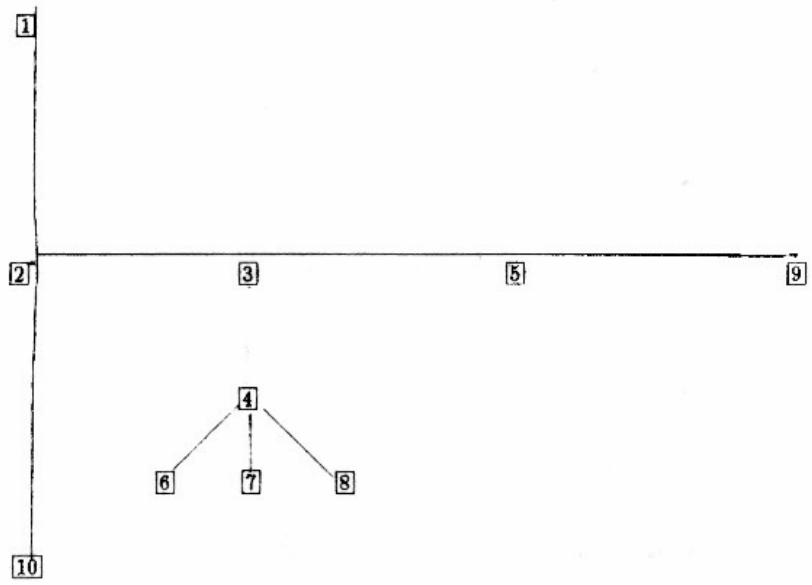

Fig.1 - Albero T.

Sia inoltre data la seguente matrice di dissimilarità $D_{jo,jo}$:

	1	2	3	4	5	6	7	8	9	10
1	0	15	23	12	45	68	90	13	24	37
2	15	0	21	39	22	77	85	4	71	45
3	23	21	0	24	69	40	25	83	27	38
4	12	39	24	0	29	34	55	86	19	28
5	45	22	69	29	0	36	8	59	47	66
6	68	77	40	34	36	0	20	50	18	54
7	90	85	25	55	8	20	0	25	20	60
8	13	4	83	86	59	50	25	0	49	23
9	24	71	27	19	47	18	20	49	0	15
10	37	45	38	28	66	54	60	23	15	0

In questo esempio si assumerà p=5.

La dissimilarità piu' piccola è $d_{2,8} = 4$. Allora tutti gli spigoli lungo il cammino con estremi 2 ed 8, cioè gli spigoli (2,3), (3,4),(4,8) avranno l'etichetta 1.

La dissimilarità successiva piu' piccola è $d_{5,7} = 8$. Gli spigoli lungo il cammino che ha come estremi 5 e 7 sono (3,5), (3,4) e (4,7). Lo spigolo (3,4) è stato già etichettato. Restano gli spigoli (3,5) e (4,7) che riceveranno l'etichetta 2.

Il successivo piu' piccolo indice di dissimilarità risulta $d_{1,4}$=12. Gli spigoli del cammino da 1 a 4 sono (1,2),(2,3) e (3,4). L'unico senza etichetta è lo spigolo (1,2) che prenderà il valore 3.

Il successivo più piccolo indice di dissimilarità risulta $d_{1,8} = 13$. Tutti gli spigoli lungo il cammino da 1 ad 8 sono etichettati.

Il successivo piu' piccolo indice di dissimilarità risulta $d_{9,10} = 15$. Gli spigoli lungo il cammino da 9 a 10 sono (5,9),(3,5),(2,3) e (2,10). Gli spigoli senza etichetta sono (5,9) e (2,10) che prenderanno l'etichetta 4.

Infine il piu' piccolo indice di dissimilarità risulta $d_{6,9} = 18$. Gli spigoli lungo il cammino da 6 a 9 sono (4,6),(3,4),(3,5) e (5,9). L'unico spigolo rimasto senza etichetta è (4,6) che prenderà il valore 5.

A questo punto l'algoritmo si interrompe poichè tutti gli spigoli sono stati etichettati.

Il numero posto accanto a ciascuno spigolo nella Fig.1 è il rango dello spigolo stesso.
I $p-1$ spigoli ($4=p-1$) di rango più elevato sono (4,6) con rango 5, (2,10) e (5,9) con rango 4, e (1,2) con rango 3.

Tagliando questi quattro spigoli si ottiene la partizione mostrata nella Fig.2.

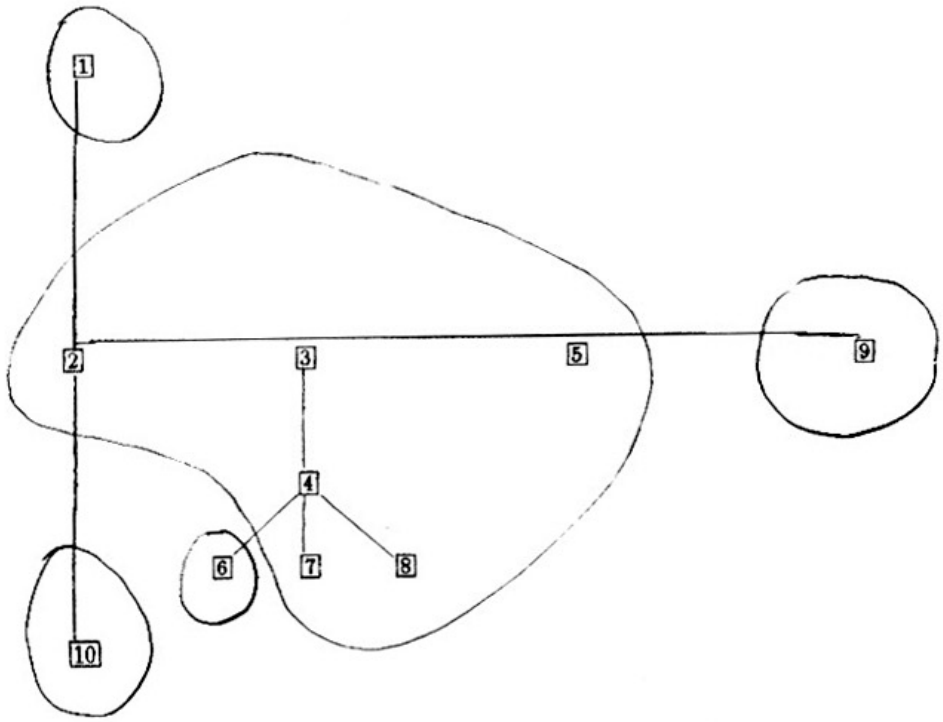

Fig.2 - Partizione in 5 gruppi .

Il *divario* di questa partizione è $d_{i_{p(3)}j_{p(3)}} = d_{1,4} = 12$, e nessuna partizione ammissibile può avere *divario* maggiore

BIBLIOGRAFIA

Aho A.V., Hopcroft J.E., Ullman J.D. (1975), *The design and analysis of computer algorithms* ,Addison Wesley, Reading.

Cerasoli M., Eugeni F. e Protasi M. (1988) *Elementi di matematica discreta*- Zanichelli - Bologna.

Delattre M., Hansen P., (1980)- *Bicriterion cluster analysis*. IEEE Trans. on Pattern Analysis and Machine Intelligence 2, n.4, 277-291 .

Murtagh, F. (1985) A Survey of Algorithms for contiguity-constrained clustering and related problems. *The Computer Journal*.28,82-88.