Tecniche di clustering nel data mining

Vincenzo Antonio Manganaro

vincenzomang@virgilio.it, www.statistica.too.it

Indice

Nel seguente articolo vedremo come le tecniche di cluster analysis si inseriscano in un contesto di DM. Tuttavia, non essendo questa tesi dedicata alle tecniche di cluster, ci concentreremo esclusivamente sul trattamento delle variabili categoriali, tra l'altro molto presenti in contesti di DM, con l'introduzione di un algoritmo che in qualche modo rappresenta una generalizzazione dell'algoritmo principe della cluster analysis e che permette appunto il trattamento di variabili categoriali.

1 Introduzione alle tecniche di clustering.

Molte applicazioni di DM richiedono il partizionamento dei dati in gruppi (o clusters) omogenei in modo tale da scoprire gruppi interessanti dai quali ricavare informazioni strategiche (si pensi alla possibilità di poter partizionare i clienti di una banca in un numero non prefissato di clusters). Vi sono tuttavia almeno due motivi per cui non è possibile estendere i tradizionali metodi di cluster analysis nelle applicazioni di DM:

- i tradizionali metodi di cluster analysis non sono efficienti quando si hanno di fronte i databases tipici di applicazioni DM, ovvero databases enormi con dimensioni dell'ordine dei terabytes caratterizzati da migliaia e migliaia di records e decine o centinaia di attributi;

- i tradizionali metodi di cluster non prevedono la presenza di variabili categoriali con il conseguente trattamento specifico delle stesse; di conseguenza il modo tradizionale di trattare le variabili categoriali come variabili numeriche non sempre produce risultati significativi, anche per il fatto che le modalità di una variabile categoriale non sono ordinate.

Neanche un metodo che permette la soluzione di uno dei due problemi di cui sopra potrebbe far pensare alla possibilità di estendere quel metodo nel caso di applicazioni di DM; ad esempio i metodi basati sul concetto delle $k-medie$ sono efficienti anche nel caso di grossi datasets ma presentano l'inconveniente di essere limitati a dati di tipo numerico. Essi infatti sono costituiti da algoritmi che hanno come obiettivo la ricerca di clusters omogenei minimizzando una particolare funzione di costo definita attraverso una distanza di tipo euclideo fra i punti del dataset (se k sono le variabili, i punti si trovano in uno spazio $k-$dimensionale) e le medie dei gruppi. Ciò limita chiaramente la possibilità di usare questi algoritmi per variabili categoriali. Tuttavia gli algoritmi basati sul concetto delle $k-medie$ rappresentano un buon punto di partenza per la costruzione di algoritmi che gestiscano anche dati di tipo categoriale, salvaguardando l'efficienza. In questo contesto noi descriveremo proprio un algoritmo, che deriva dall'algoritmo delle $k-medie$, in grado di gestire variabili categoriali, molto frequenti nel DM, e che si riconduce all'algoritmo delle $k-medie$ qualora considerassimo solo variabili di tipo numerico. Dal momento che in tale algoritmo gli oggetti o records sono raggruppati confrontandoli con k oggetti *prototipi* invece che con le k medie di gruppo, chiameremo tale algoritmo con il nome di algoritmo dei $k-prototipi$.

2 Trattamento delle variabili categoriali nel clustering.

Indichiamo con $X = \{X_1, X_2, \ldots, X_n\}$ un set di n oggetti e con $x_i = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$ un oggetto rappresentato da m attributi. Indicato con k un intero positivo, l'obiettivo di un'operazione di clustering è trovare una partizione che divide gli oggetti in X in k gruppi disgiunti. Un modo, non certo pratico, di effettuare questa operazione potrebbe consistere nell'analizzare tutte le possibili partizioni, che per un certo n ed il k considerato hanno un numero definito anche se estremamente elevato, allo scopo di trovare quella che meglio si adatta per un problema di classificazione (o meglio per l'individuazione di gruppi specifici). Tuttavia la soluzione più comune in questi casi è la scelta di un criterio di raggruppamento che possa guidare l'utente nella ricerca della partizione e quindi dei gruppi che da essa derivano.
Il criterio di raggruppamento di gran lunga più usato consiste nel definire una "funzione di costo", ad esempio nel modo seguente:

$$E = \sum_{l=1}^{k} \sum_{i=1}^{n} y_{il}\, d(X_i, Q_l) \tag{1}$$

dove $Q_l = [q_{l_1}, q_{l_2}, \ldots, q_{l_m}]$ è un "vettore rappresentativo" o "prototipo" per il cluster l, y_{il} è il generico elemento di una matrice di partizione Y e d è semplicemente una misura di similarità, spesso definita come la distanza euclidea in uno spazio, in tal caso, $m-$dimensionale. La matrice Y di dimensione $[n * k]$ e di generico elemento y_{il} è la matrice di partizione, poichè ad ogni matrice così definita corrisponde una ed una sola partizione degli n oggetti in k gruppi. La matrice di partizione presenta le seguenti due proprietà:

1. $0 \leq y_{il} \leq 1$
2. $\sum_{l=1}^{k} y_{il} = 1$

La partizione corrispondente ad una matrice Y nella quale $y_{il} \in \{0, 1\}$ è denominata *hard partition*, altrimenti si tratta di una *fuzzy partition*. Noi utilizzeremo essenzialmente hard partition nelle quali il generico elemento $y_{il} = 1$ indica che l'oggetto X_i è assegnato al cluster l dalla matrice Y. Il termine $\sum_{i=1}^{n} y_{il}\, d(X_i, Q_l) = E_l$ dell'equazione 5.1 rappresenta il costo totale nell'assegnare gli oggetti in X al cluster

l, cioè la dispersione totale degli oggetti nel cluster l rispetto al suo prototipo Q_l. L'espressione E_l è minimizzata se:

$$q_{l_j} = \frac{1}{n_l} \sum_{i=1}^{n} y_{il}\, x_{ij} \quad perj = 1, 2, \ldots, m \tag{2}$$

dove $n_l = \sum_{i=1}^{n} y_{il}$ è il numero di oggetti nel cluster l.
Ma cosa accade se X presenta degli attributi categoriali?
In tal caso è necessario sostituire la misura di similarità (che non avrebbe più senso in un contesto di variabili categoriali) con una misura alternativa data ad esempio dalla seguente espressione:

$$d(X_i, Q_l) = \sum_{j=1}^{m_r} (x_{ij}^r - q_{lj}^r)^2 + \gamma_l \sum_{j=1}^{m_c} \delta(x_{ij}^c, q_{lj}^c) \tag{3}$$

dove $\delta(p, q) = 0$ per $p = q$ e $\delta(p, q) = 1$ per $p \neq q$, x_{ij}^r e q_{lj}^r sono valori di attributi numerici mentre x_{ij}^c ed q_{lj}^c sono valori (o, sarebbe meglio dire, modalità) di attributi categoriali per l'oggetto i e il prototipo di cluster l; m_r ed m_c rappresentano il numero degli attributi numerici e categoriali rispettivamente; γ_l rappresenta il peso relativo all'importanza degli attributi categoriali per il cluster l.
Scegliendo dunque la 5.3 come misura di similarità E_l può essere espressa come:

$$E_l = \sum_{i=1}^{n} y_{il} (\sum_{j=1}^{m_r} (x_{ij}^r - q_{lj}^r)^2 + \gamma_l \sum_{j=1}^{m_c} \delta(x_{ij}^c, q_{lj}^c)) \tag{4}$$

da cui:

$$E_l = \sum_{i=1}^{n} y_{il} (\sum_{j=1}^{m_r} (x_{ij}^r - q_{lj}^r)) + \gamma_l \sum_{i=1}^{n} \sum_{j=1}^{m_c} \delta(x_{ij}^c, q_{lj}^c)) = E_l^r + E_l^c \tag{5}$$

dove E_l^r è il costo totale su tutti gli attributi numerici di oggetti nel cluster l che viene minimizzato se i q_{lj}^r per $j = 1, 2, \ldots, m$ vengono calcolati con l'equazione 5.2.
Consideriamo ora il set indicato con C_j contenente l'insieme di modalità che può assumere l'attributo categoriale j e indichiamo con $p(c_j \in C_j|l)$ la probabilità che la modalità c_j si verifichi nel cluster l. E_l^c nell'equazione 5.5 può essere scritto come:

$$E_l^c = \gamma_l \sum_{j=1}^{m_c} n_l (1 - p(q_{lj}^c \in C_j|l) \tag{6}$$

essendo n_l il numero di oggetti nel cluster l.
Il problema consiste ora nel minimizzare la quantità 5.6 e la soluzione si ottiene nel

seguente modo:

- per uno specifico cluster l la quantità E_l^c è minimizzata se e solo se $p(q_{lj}^c \in C_j|l) \geq p(c_j \in C_j|l)$ per $q_{lj}^c \neq c_j$ per tutte la variabili categoriali.

Infine possiamo scrivere E come:

$$E = \sum_{l=1}^{k}(E_l^r + E_l^c) = \sum_{l=1}^{k}(E_l^r) + \sum_{l=1}^{k}(E_l^c) = E^r + E^c \tag{7}$$

che rappresenta la funzione di costo nel raggruppamento di oggetti contenuti in un dataset con variabili numeriche e categoriali. Dal momento che E^r ed E^c sono entrambi non negativi la minimizzazione può essere ottenuta minimizzando E^r ed E^c ovvero il costo totale degli attributi numerici e categoriali rispettivamente su tutti i clusters. In particolare E^r può essere minimizzato calcolando gli elementi numerici dei k prototipi di clusters con l'equazione 5.2 mentre E^c può essere minimizzato selezionando gli elementi categoriali dei k prototipi di cluster secondo il lemma visto. Dunque l'equazione 5.2 ed il lemma definiscono un modo di scegliere i prototipi di cluster per minimizzare la funzione di costo totale dell'equazione 5.7. Vedremo successivamente come da questa base si sviluppi l'algoritmo dei $k - prototipi$.

3 Un approfondimento sulle misure di similarità.

La funzione di costo data dall'equazione 5.7 è definita sulla base di una misura di similarità (eq.5.3) che è a sua volta una combinazione lineare di misure di similarità che riguardano variabili numeriche e categoriali. In questo caso la misura di similarità nel caso di variabili numeriche è data dalla distanza euclidea, mentre nel caso di variabili categoriali dal numero di ineguaglianze tra oggetti e prototipi di cluster. Il peso γ_l è introdotto per distribuire in maniera opportuna l'importanza delle due tipologie di attributi nel cluster l.
Illustriamo ora l'influenza del valore γ_l nelle operazioni di clustering. A tal scopo assumiamo che i triangoli ed i quadrati della fig.5.1 rappresentino un set di oggetti descritti da un attributo categoriale e da due attributi numerici. I triangoli ed i quadrati rappresentano due modalità dell'attributo categoriale mentre i valori degli

attributi numerici sono rappresentati dalla posizione degli oggetti nel piano delimitato dalle frecce. Gli oggetti sono classificati in due clusters separati dalla linea verticale tratteggiata in figura. Se $\gamma_l = 0$ il raggruppamento dipende solo dagli attributi numerici ovvero dalla posizione degli oggetti. Il risultato consisterà in due clusters separati da una linea verticale a tratti (vedere figura). Se $\gamma_l > 0$ l'oggetto C può passare al cluster sulla destra poichè esso è vicino a quel cluster e il suo valore categoriale è lo stesso della maggior parte degli oggetti del cluster sulla destra. Allo stesso modo l'oggetto D può passare al cluster di sinistra. L'oggetto A può rimanere ancora nel cluster di sinistra poichè esso è troppo lontano dal cluster di destra, sebbene presenta la stessa modalità dell'attributo categoriale della maggior parte degli oggetti del cluster di destra. Lo stesso vale per l'oggetto E che può rimanere ancora nel cluster di destra. Il trattamento dell'oggetto B è incerto, esso dipende dal valore di γ_l ed in particolare da come questo sia orientato verso attributi numerici o categoriali. Se γ_l è orientato verso gli attributi categoriali, l'oggetto B può passare al cluster di destra (anche se nella figura ne risulta abbastanza) altrimenti esso rimane in quello di sinistra.
Ma come scegliere γ_l in una operazione di clustering? La scelta di γ_l dipende dalla distribuzione degli attributi numerici; essa è generalmente legata al valore σ_l, la deviazione standard media degli attributi numerici nel cluster l. Nella pratica dunque σ_l può essere usato come guida per determinare γ_l; tuttavia dal momento che σ_l è incognito prima delle operazioni di clustering, al posto di σ_l potremmo usare la deviazione standard media della totalità degli attributi numerici, indicata con σ. Inoltre in un algoritmo di tipo iterativo γ_l può essere calcolato in base ad una precedente operazione di clustering.

4 L'algoritmo dei k-prototipi.

L'algoritmo dei $k - prototipi$ può essere descritto dai seguenti passi:

1. selezionare k prototipi iniziali dal dataset X, uno per ogni cluster;
2. allocare ogni oggetto in X al cluster di cui il prototipo è più vicino secondo l'equazione 5.5 e ridefinire il prototipo del cluster dopo ogni allocazione;
3. dopo che tutti gli oggetti sono stati allocati ai rispettivi clusters, saggiare

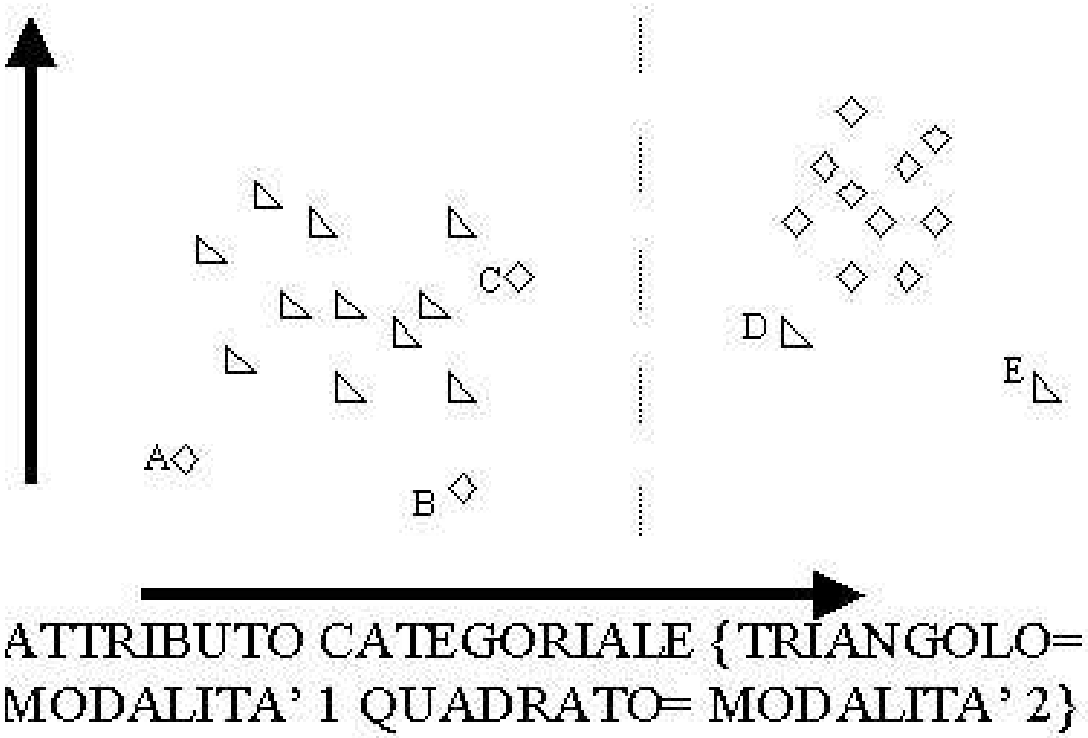

Figura 1: Influenza del valore γ_l nel clustering

nuoavamente la similarità di tali oggetti con tutti i prototipi di clusters. Se un oggetto risulta essere più vicino al prototipo di cluster cui non appartiene, riallocare l'oggetto a questo nuovo cluster e ridefinire nuovamente i prototipi di entrambi i clusters;

4. ripetere il passo 3 finché nessun oggetto subisce il cambiamento di cluster.

L'algoritmo è costruito dividendo il processo in tre sottoprocessi: "selezione iniziale dei prototipi", "allocazione iniziale" e "riallocazione". Il primo sottoprocesso seleziona a caso k oggetti come gli iniziali prototipi di clusters. Descriviamo ora dettagliatamente il cuore dell'algoritmo costituito dall'operazione di allocazione iniziale che, partendo da un iniziale set di prototipi di clusters, assegna ogni oggetto ad un cluster e ridefinisce il prototipo di cluster in seguito a ciascuna assegnazione di un oggetto al cluster. Il sottoprocesso in questione è rappresentato nel seguente prospetto:

```
for i = 1 to Numero di oggetti
    Mindistance= Distance(X[ i ],O_prototypes[1]) +
    gamma*Sigma(X[ i ],C_prototypes[1])
   for j=1 to Numero di clusters
      distance=Distance(X[ i ],O_prototypes[ j ]) +
      gamma*Sigma(X[ i ],C_prototypes[ j ])
      if(distance < Mindistance)
          Mindistance=distance
          cluster=j
      endif
   endfor
   Clustership[ i] =cluster
   ClusterCount[cluster]+1
   for j=1 to Numero di attributi numerici
      SumInCluster[ cluster,j] + X[ i,j]
      O_prototypes[cluster,j] =SumInCluster[ cluster,j]/ClusterCount[cluster]     endfor
   for j=1 to Numero di attributi categoriali
      FrequencyInCluster[ cluster,j,X[ i,j]]+1
      C_prototypes[cluster,j] =HighestFreq(FrequencyInCluster,cluster,j)     endfor
```

endfor

L'algoritmo è costituito da un ciclo for generale all'interno del quale operano tre successivi sottocicli for. Nel primo ciclo for l'algoritmo considera il numero di oggetti da raggruppare per mezzo di una variabile intera positiva i che va da 1 al numero di oggetti. Nel primo passo ad esempio i=1 e viene calcolata l'espressione:
Mindistance= **Distance**(X[1],O_prototypes[1]) + gamma***Sigma**(X[1],C_prototypes[1]) nella quale **Distance**() è la distanza euclidea fra le caratteristiche numeriche dell'oggetto 1 (X[1]) e le caratteristiche numeriche del prototipo 1 (O_prototypes[1]) e **Sigma** una implementazione della funzione $\delta()$ nell'equazione 5.5. Il primo sottociclo for considera invece il numero di clusters per mezzo di una variabile intera positiva j che va da 1 al numero di clusters, per ciascun j calcola la distanza fra l'oggetto 1 e il prototipo del cluster j (assegnando il valore alla variabile distance) e confronta il valore ottenuto con il valore Mindistance. Se distance < Mindistance l'algoritmo aggiornerà il valore Mindistance ponendo Mindistance=distance e assegnando alla variabile cluster il valore j corrispondente al cluster al quale l'oggetto 1 è più vicino. Il primo sottociclo for si conclude in tal modo. Le variabili Clustership[] e ClusterCount[] registrano rispettivamente l'appartenenza di oggetti ad un cluster e il numero di oggetti nei clusters. Se ad esempio nelle prima iterazione l'oggetto 1 è stato assegnato al cluster 3, la variabile cluster sarà uguale a 3 per cui si avrà Clustership[1] = 3 (appartenenza dell'oggetto 1 al cluster 3) e ClusterCount[3]+1 incrementerà di una unità il numero di oggetti nel cluster 3. Il successivo sottociclo for permette di ridefinire il prototipo di cluster cui è stato assegnato l'oggetto per quanto riguarda le variabili numeriche. La variabile intera positiva j va da 1 al numero di attributi numerici. L'espressione SumInCluster[cluster,j] + X[i,j] è analoga all'espressione SumInCluster[cluster,j] = SumInCluster[cluster,j] + X[i,j] cioè SumInCluster va a sommare gli attributi numerici del nuovo oggetto i ai corrispondenti attributi numerici già presenti nel cluster essendo X[i,j] il vettore di attributi numerici dell'oggetto i. Successivamente e sempre all'interno del medesimo ciclo, l'espressione O_prototypes[cluster,j] =SumInCluster[cluster,j]/ClusterCount[cluster] va ad aggiornare il vettore del prototipo di cluster tenendo conto del nuovo numero di oggetti nel cluster. L'ultimo sottociclo for permette di ridefinire il prototipo di cluster cui è stato assegnato l'oggetto per quanto riguarda le variabili categoriali.

La variabile intera positiva j va da 1 al numero di attributi categoriali. L'espressione FrequencyInCluster[cluster,j,X[i,j]]+1 registra le frequenze dei differenti valori delle variabili categoriali nei clusters ed infine la funzione **HighestFreq**() è una implementazione del lemma visto per la minimizzazione della distanza nel caso di attributi categoriali che permette di ridefinire gli attributi categoriali dei prototipi. Da ricordare che quanto visto deve essere ripetuto per tutti gli oggetti da raggruppare all'interno del ciclo for più generale.
Per quanto riguarda la fase di riallocazione, il processo è abbastanza simile al precedente algoritmo con la sola eccezione che dopo la riallocazione di un oggetto i prototipi per entrambi i clusters, quello in cui si trovava l'oggetto e il nuovo cluster di appartenenza, devono essere ridefiniti. L'algoritmo continua le iterazioni nella fase di riallocazione finché nessun oggetto cambia classe di appartenenza. Per avere un 'idea della performance di tale algoritmo la prova di esso su diversi datasets di produce delle curve di convergenza che presentano un comportamento simile e quindi tipico dell'algoritmo dei $k - prototipi$. In particolare si nota come il numero di oggetti che cambiano i clusters di appartenenza si riduce molto rapidamente durante le prime iterazioni; successivamente la velocità di riduzione si abbassa fino al punto in cui il numero di oggetti che cambiano clusters resta costante nonostante le ulteriori iterazioni. In seguito dopo un breve aumento tale numero tende definitivamente a 0. Ciò suggerisce che il processo di riallocazione può essere forzato a terminare prima, quando ad esempio la velocità di abbassamento del numero diventa piccola o magari è piccolo il numero di oggetti che modificano i clusters. Questi accorgimenti possono aiutare a ridurre il tempo necessario per l'esecuzione dell'algoritmo risultando in tal modo utili nel contesto di grossi datasets. Per concludere mostriamo in figura 5.2 il tipico andamento di una curva di convergenza dell'algoritmo dei $k - prototipi$. Sull'asse delle ascisse abbiamo il numero di iterazioni mentre su quello delle ordinate il numero di oggetti che modificano il cluster di appartenenza ad ogni iterazione.

Figura 2: Andamento tipico di una curva di convergenza nell'algoritmo dei $k-$ *prototipi* sull'asse delle ordinate è mostrato il numero di oggetti che modificano i clusters di appartenenza nelle diverse iterazioni.