USTHB,
FEI, Département Informatique
M2, Master SII

Bab-Ezzouar le 14 janvier 2017

Corrigé de l'EMD de Data Mining

Considérer le dataset suivant contenant 10 instances et 5 attributs nommés A, B, C, D et E. On s'intéresse à extraire des motifs fréquents pour déduire des règles d'association entre les attributs. Les instances font office de transactions et les valeurs des attributs d'items.

	A	B	C	D	E
I1	1	4	13	2	3
I2	1	2	12	0	7
I3	1	3	13	2	6
I4	1	4	11	2	7
I5	1	4	14	2	7
I6	0	4	15	2	7
I7	1	1	13	0	3
I8	1	4	14	0	7
I9	1	4	14	2	7
I10	1	4	12	2	7

1) Décrire avec clarté l'algorithme A-priori pour un quelconque dataset en précisant les points suivants :
   a. Les structures de données utilisées
   b. Les entrées- sorties
   c. Les techniques algorithmiques

Structures de données utilisées

Codage du dataset : coder chaque valeur d'un attribut X comme Xvaleur. En l'occurrence, la valeur 1 de l'attribut A sera codée A1.

Le dataset sera représenté par une matrice de codes. Nous aurons aussi besoin de 2 autres matrices C et L pour stocker respectivement les candidats potentiels et les motifs fréquents calculés à partir des candidats. Les deux matrices auront pour colonnes respectivement *itemset*, le motif fréquent à déterminer ainsi que #occurrences, le nombre d'occurrences de ces motifs dans la table de transactions. Une colonne nommée *ensemble* contiendra les éléments de la transaction (instance) et servira à faciliter le comptage des itemsets dans la base des transactions.

type
code : chaine de caractères ;
set : ensemble de code ;
élémentM : **enregistrement** attribut : **tableau** [1.. m] de code ; ensemble : set **fin** ;
M : tableau [1 .. n] d'élémentM ;
élément : **enregistrement** itemset : set ; #occurrences : **entier fin** ;

Entrées

- M, Matrice des codes représentant la table de transactions *M.attribut[n]* où *n* est le nombre d'instances.
- Support minimum

Sorties

L'ensemble des motifs fréquents sous forme d'une table.

```
Algorithme Apriori
Entrée : M[n], Matrice des codes représentant la table de transactions et MinSup, le support minimum
Sortie : L, l'ensemble des motifs fréquents.

Var C, L : tableau [1 .. max] d'élément ;
    i, j, k, max : entier;
    arrêt : booléen ;
début
        (* construction de C1 *)
        (* initialisation *)
        max := 100 ;
        pour k := 1 à max faire C.#occurrences[k] := 0;
        (* détermination des itemsets et calcul des nombres d'occurrences des itemsets *)
        pour i := 1 à n faire
        début M.ensemble[i] := ∅ ;
              pour j := 1 à m faire
                 début k := 0 ;
                       M.ensemble[i]  := M.ensemble[i]  ∪ {M.attribut[i][j]} ;
                       k := k + 1;
                       C.itemset[k] := M.ensemble[i];
                       C.#occurrences[k] := C.#occurrences[k] +1;
                 fin;
        fin;
        (* détermination de L1 *)
        j := 0
        pour i := 1 à k faire
           si C.#occurrences[i]>= MinSup alors début  j := j + 1 ;
                                                      L.itemset[j] := C.itemset[i];
                                               fin;
        (* itération du processus *)
        arrêt := faux;
        tant que non arrêt faire
        début
           (* détermination de l'ensemble courant des candidats *)
           C.itemset := jointure (L.itemset x L.itemset) ;
        (* détermination de l'ensemble courant des motifs fréquents *)
          si  |C.itemset|=0 alors arrêt := vrai
           sinon début
                   (* initialisation des nombres d'occurrences des candidats *)
                   pour i := 1 à |C.itemset| faire C.#occurrences[i] := 0 ;
                  (* calcul du nombre d'occurrences des itemsets *)
                  pour k := 1 à |C.itemset| faire
                    pour i := 1 à n faire
                       si C.itemset[k] est dans M.ensemble[i]
                               alors C.#occurrences[k] := C.#occurrences[k] +1 ;
                 (* élagage des motifs fréquents qui ne respectent pas le support minimum *)
                 j := 0
                 pour i := 1 à  |C.itemset| faire
                   si C.#occurrences[i] >= MinSup alors début  j := j + 1 ;
                                                               L.itemset[j] := C.itemset[i];
                                                        fin;
                 fin;
          fin;
fin;
```

2) En déduire sa complexité.

Construction de C1 :

- Initialisation : $O(n * m)$ car le nombre de candidats de C1 est au maximum égal à $(n * m)$

- Détermination des itemsets et calcul des nombres d'occurrences des itemsets : *O(n * m)* car toute la table des transactions est parcourue.

Détermination de L1 :

- *O(n * m)* car au maximum, le nombre de candidats est égal à $n * m$.

Boucle

- détermination de l'ensemble courant des candidats : la jointure se fait en $O(|L|^2)$. Mais comme $|L|$ est égal au maximum à (n * m), la complexité de la jointure est $O((n * m)^2)$.
- initialisation des nombres d'occurrences des candidats : *O(n * m).*
- calcul du nombre d'occurrences des itemsets : *O(n * m).*
- détermination des motifs fréquents : *O(n* m).*

La complexité globale est $O((n * m)^2 * x)$ où x est le nombre d'itérations de la boucle.

3) Appliquer l'algorithme A-priori sur le dataset ci-dessus avec un support minimal de 40%.

Pour appliquer l'algorithme A-priori sur le dataset, nous devons d'abord coder les valeurs des attributs comme suit :

	A	**B**	**C**	**D**	**E**	**ensemble**
I1	A1	B4	C13	D2	E3	{A1, B4, C13, D2, E3}
I2	A1	B2	C12	D0	E7	{A1, B2, C12, D0, E7}
I3	A1	B3	C13	D2	E6	{A1, B3, C13, D2, E6}
I4	A1	B4	C11	D2	E7	{A1, B4, C11, D2, E7}
I5	A1	B4	C14	D2	E7	{A1, B4, C14, D2, E7}
I6	A0	B4	C15	D2	E7	{A0, B4, C15, D2, E7}
I7	A1	B1	C13	D0	E3	{A1, B1, C13, D0, E3}
I8	A1	B4	C14	D0	E7	{A1, B4, C14, D0, E7}
I9	A1	B4	C14	D2	E7	{A1, B4, C14, D2, E7}
I10	A1	B4	C12	D2	E7	{A1, B4, C12, D2, E7}

La dernière colonne contient les ensembles des éléments de chaque instance.

Première itération :

Détermination des candidats C1 : parcours des transactions et comptage des occurrences de chaque item. Ce qui donne :

C1

itemset	**#occurrences**
{A0}	1
{A1}	9
{B1}	1
{B2}	1
{B3}	1
{B4}	7
{C11}	1
{C12}	2
{C13}	3
{C14}	3
{C15}	1
{D0}	3
{D2}	7
{E3}	2
{E6}	1
{E7}	7

Le support minimum étant égal à 40%, il équivaut à 10 x 40% = 4 transactions. Les motifs fréquents d'ordre 1, appartenant à L1 sont des candidats C1 qui satisfont le support minimum.

L1

itemset	#occurrences
{A0}	1
{A1}	9
{B1}	1
{B2}	1
{B3}	1
{B4}	7
{C11}	1
{C12}	2
{C13}	3
{C14}	3
{C15}	1
{D0}	3
{D2}	7
{E3}	2
{E6}	1
{E7}	7

L1

itemset	#occurrences
{A1}	9
{B4}	7
{D2}	7
{E7}	7

Deuxième itération :
Détermination de C2, des itemsets candidats contenant au plus deux items :

C2

itemset	#occurrences
{A1, B4}	6
{A1, D2}	6
{A1, E7}	6
{B4, D2}	6
{B4, E7}	6
{D2, E7}	5

Les itemsets retenus sont donc :

L2

itemset	#occurrences
{A1, B4}	6
{A1, D2}	6
{A1, E7}	6
{B4, D2}	6
{B4, E7}	6
{D2, E7}	5

Les items d'un même attribut ne peuvent pas apparaître dans un itemset car ils représentent le même attribut avec des valeurs différentes.

Troisième itération :

C3

itemset	#occurrences
{A1, B4, D2}	5
{A1, B4, E7}	5
{A1, B4, D2, E7}	4
{A1, D2, E7}	4
{B4, D2, E7}	5

L3

itemset	#occurrences
{A1, B4, D2}	5
{A1, B4, E7}	5
{A1, B4, D2, E7}	4
{A1, D2, E7}	4
{B4, D2, E7}	5

Quatrième itération :

C4 = ∅, l'ensemble vide. Le processus s'arrête. L'ensemble des motifs fréquents est donc comme suit :
$L = L1 \cup L2 \cup L3$.

4) Adapter l'algorithme k-means à un dataset constitué d'instances d'attributs en fournissant les précisions sur :

Algorithme *k-means*
entrée *: D, un dataset et k, le nombre de clusters*
sortie *: un ensemble de k clusters*

var *arrêt :* ***booléen*** *;*
début
choisir aléatoirement k instances du dataset ;
arrêt := ***faux*** *;*
tant que non *arrêt* ***faire***
début
affecter chaque instance au cluster le plus similaire ;
mettre à jour les centres des clusters ;
si *pas de changement* ***alors*** *arrêt :=* ***vrai*** *;*
fin *;*
fin *;*

---

a. Le choix des centroides.

Les centroides sont engendrés de manière aléatoire initialement.

b. Calcul de la similarité en tenant compte des types des attributs.

Nous sommes dans le cas où nous avons m variables ou attributs de différents types. La similarité dans ce cas se calcule comme suit :

$$distance(I_i, I_j) = \frac{\sum_{t=1}^{t=m} distance^t_{I_i I_j}}{m}$$

t est l'indice de l'attribut. Si l'attribut est de type :

- réel ou entier :

$$distance^t_{I_i I_j} = \left| \frac{I_i^t}{max_{k=1}^{k=m} I_{i,k}^t - min_{k=1}^{k=m} I_{i,k}^t} - \frac{I_j^t}{max_{k=1}^{k=m} I_{j,k}^t - min_{k=1}^{k=m} I_{j,k}^t} \right|$$

- booléen ou nominal : $distance^t_{I_i,I_j} = 0$ si I^t_i=I^t_j et $distance^t_{I_i,I_j} = 1$ si $I^t_i \neq I^t_j$.
- Ordinal : calculer les rangs $r^t_{I_i}$ et $z^t_i = \frac{r^t_{I_i}-1}{M_t-1}$ et considérer z^t_i comme réel.

c. Le choix du paramètre k.

Considérer k comme un paramètre empirique et le fixer après des expérimentations.

5) Rappeler sa complexité.

- *affecter chaque instance au cluster le plus similaire :*
    - *vider les clusters :* $O(k)$
    - *pour chaque instance I faire*
        - *pour chaque centroide c_i (i := 1 à k)*
            - *calculer la similarité de I avec c_i pour i := 1 à k;* $O((n-k)*k)$
        - insérer I dans le cluster le plus proche de I ; $O((n-k)*k)$
- *mettre à jour les centres des clusters :*
    - *pour chaque cluster*
    - *calculer la moyenne de ses éléments ;* $O(n)$
- *boucle : le processus est répété x fois, x étant le nombre d'itérations*

donc au total, la complexité de l'algorithme est $O((n-k)*k*x)$

6) Appliquer l'algorithme k-means sur les 6 premières instances du dataset pour k = 2 et en démarrant avec les instances I2 et I4 comme centroides initiaux. Considérer tous les types des attributs comme des entiers.

Initialisations :

C1 = {I2} C2 = {I4}

Première itération :

Calcul des distances entre les instances et I2 et I4 :

I2	1	2	12	0	7
I4	1	4	11	2	7

I1	1	4	13	2	3

Distance (I1, I2) = $|1-1|+|4-2|+|13-12|+|2-0|+|3-7| = 0+2+1+2+4 = 9$

Distance (I1, I4) = $|1-1|+|4-4|+|13-11|+|2-2|+|3-7| = 0+0+2+0+4 = 6$

C2 = {I4, I1}

I3	1	3	13	2	6

Distance (I3, I2) = $|1-1|+|3-2|+|13-12|+|2-0|+|6-7| = 0+1+1+2+1 = 5$

Distance (I3, I4) = $|1-1|+|3-4|+|13-13|+|2-2|+|6-3| = 0+1+0+0+3 = 4$

C2 = {I4, I1, I3}

I5	1	4	14	2	7

Distance (I5, I2) = $|1-1|+|4-2|+|14-12|+|2-0|+|7-7| = 0+2+2+2+0 = 6$

Distance (I5, I4) = $|1-1|+|4-4|+|14-13|+|2-2|+|7-3| = 0+0+1+0+4 = 5$

C2 = {I4, I1, I3, I5}

I6	0	4	15	2	7

Distance (I6, I2) = $|0 - 1| + |4 - 2| + |15 - 12| + |2 - 0| + |7 - 7| = 1 + 2 + 3 + 2 + 0 = 8$

Distance (I6, I4) = $|0 - 1| + |4 - 4| + |15 - 13| + |2 - 2| + |7 - 3| = 0 + 3 + 0 + 2 + 0 = 5$

C2 = {I4, I1, I3, I5, I6}

Mise à jour des centroides:

I2	1	2	12	0	7
c1	1	2	12	0	7

I1	1	4	13	2	3
I3	1	3	13	2	6
I4	1	4	11	2	7
I5	1	4	14	2	7
I6	0	4	15	2	7
c2	0.8	3.8	13.2	2	6

Deuxième itération :

Calcul des distances entre les instances et c1 et c2 :

c1	1	2	12	0	7

c2	0.8	3.8	13.2	2	6

I1	1	4	13	2	3

Distance (I1, c1) = $|1 - 1| + |4 - 2| + |13 - 12| + |2 - 0.66| + |3 - 7| = 1.67$

Distance (I1, c2) = $|1 - 0.8| + |4 - 3.8| + |11 - 13.2| + |2 - 2| + |7 - 6| = 0 + 0 + 2 + 0 + 4 = 0.72$

C2 = {I1}

I2	1	2	12	0	7

Distance (I2, c1) = $|1 - 1| + |2 - 2| + |12 - 12| + |0 - 0| + |7 - 7| = 0$

Distance (I2, c2) = $|1 - 0.8| + |2 - 3.8| + |12 - 13.2| + |0 - 2| + |7 - 6|$ =1.24

C1 = {I2}

I3	1	3	13	2	6

Distance (I3, c1) = $|1 - 1| + |3 - 2| + |13 - 12| + |2 - 0| + |6 - 7| = 1$

Distance (I3, c2) = $|1 - 0.8| + |3 - 3.8| + |13 - 13.2| + |2 - 2| + |6 - 6| = 0.24$

C2 = {I1, I3}

I4	1	4	11	2	7

Distance (I4, c1) = $|1 - 1| + |4 - 2| + |11 - 12| + |2 - 0| + |7 - 7| = 1$

Distance (I4, c2) = $|1 - 0.8| + |4 - 3.8| + |11 - 13.2| + |2 - 2| + |7 - 6| = 0.72$

C2 = {I1, I3, I4}

I5					
	1	4	14	2	7

Distance (I5, c1) = $|1 - 1| + |4 - 2| + |14 - 12| + |2 - 0| + |7 - 7| = 1.2$

Distance (I5, c2) = $|1 - 0.8| + |4 - 3.8| + |14 - 13.2| + |2 - 2| + |7 - 6| = 0.61$

C2 = {I1, I3, I4, I5}

I6					
	0	4	15	2	7

Distance (I6, c1) = $|0 - 1| + |4 - 2| + |15 - 12| + |2 - 0| + |7 - 7| = 1.6$

Distance (I6, c2) = $|0 - 0.8| + |4 - 3.8| + |15 - 13.2| + |2 - 2| + |7 - 6| = 0.64$

C2 = {I1, I3, I4, I5, I6}

Le processus s'arrête car les contenus des clusters ne changent pas. Le résultat est donc :

C1 = {I2}
C2 = {I1, I3, I4, I5, I6}