TD Méthodes de clustering
M1-ATD
Ioannis Partalas, Eric Gaussier

D'après M.-R. Amini & E. Gaussier, *Recherche d'information*, Eyrolles 2013

Le but de ce TD est d'étudier plus précisément les algorithmes associés au partitionnement hiérarchique et à plat. Nous nous intéresserons d'abord à l'algorithme des k-moyennes (*k-means*), un des algorithmes de partitionnement les plus populaires. Nous verrons ensuite comment implanter de façon efficace des algorithmes hiérarchiques et agglomératifs.

Les algorithmes sont volontairement simplifiés, dans le sens où toutes les structures de données ne sont pas explicitées. Elles devraient néanmoins être claires au vu du cours.

1. Algorithme des k-moyennes

Pour un ensemble partitionné en K classes $(G_1, \ldots, G_K)$ de centres respectifs $(\mathbf{r}_1, \ldots, \mathbf{r}_K)$, considérons la fonction de coût somme des carrées des résidus intervenant dans l'algorithme des k-moyennes :

$$\mathcal{L}(G_1, \ldots, G_K; \mathbf{r}_1, \ldots, \mathbf{r}_K) = \sum_{i=1}^{K} \sum_{d \in G_i} ||\mathbf{d} - \mathbf{r}_i||_2^2$$

L'algorithme des k-moyennes consiste alors en la répétition de deux étapes de réaffectation et de recalcul des centroïdes (1).

(a) Montrer qu'entre les itérations t et $t+1$ de l'algorithme 1 on a :

$$\mathcal{L}(G_1^{(t+1)}, \ldots, G_K^{(t+1)}; \mathbf{r}_1^{(t)}, \ldots, \mathbf{r}_K^{(t)}) \leq \mathcal{L}(G_1^{(t)}, \ldots, G_K^{(t)}; \mathbf{r}_1^{(t)}, \ldots, \mathbf{r}_K^{(t)})$$

(b) Montrer aussi que :

$$\mathcal{L}(G_1^{(t+1)}, \ldots, G_K^{(t+1)}; \mathbf{r}_1^{(t+1)}, \ldots, \mathbf{r}_K^{(t+1)}) \leq \mathcal{L}(G_1^{(t+1)}, \ldots, G_K^{(t+1)}; \mathbf{r}_1^{(t)}, \ldots, \mathbf{r}_K^{(t)})$$

(c) En déduire que la fonction de coût de l'algorithme k-moyennes décroît à chaque itération.

Entrée :
- $\mathcal{C} = \{d_1, \cdots, d_N\}$, collection de documents ;
- K, nombre de classes ;
- T, nombre d'itérations maximal admis;

Initialisation :
- $(\mathbf{r}_1^{(0)}, \cdots, \mathbf{r}_K^{(0)})$, ensemble de représentants initiaux;
- $t \leftarrow 1$;

tant que *(t < T) ou (l'ensemble des classes n'est pas stable)* **faire**

  pour chaque $d \in \mathcal{C}$ **faire**

    // Etape de réaffectation

    $G_k^{(t)} \leftarrow \{d : ||\mathbf{d} - \mathbf{r}_k^{(t-1)}||_2^2 \leq ||\mathbf{d} - \mathbf{r}_l^{(t-1)}||_2^2, \forall l \neq k, 1 \leq l \leq K\}$;

  fin

  pour chaque $k, 1 \leq k \leq K$ **faire**

    // Etape de recalcul des centroïdes

    $\mathbf{r}_k^{(t)} \leftarrow \frac{1}{|G_k^{(t)}|} \sum_{d \in G_k^{(t)}} \mathbf{d}$;

  fin

  $t \leftarrow t + 1$;

fin

Sortie : G, une partition de $\mathcal{C}$ en K classes

Algorithme 1 : Algorithme des k-moyennes

2. Classification par méthodes agglomératives hiérarchiques ascendantes

L'algorithme 2 est un algorithme générique de classification hiérarchique ascendante qui peut être utilisé avec différentes méthodes d'agrégation. Il utilise des files de priorité pour lesquelles le coût d'une insertion ordonnée (c'est-à-dire de l'insertion d'un élément dans une liste ordonnée) est de l'ordre de $\log n$, n désignant la taille de la file.

On se propose ici de voir un algorithme un peu plus efficace pour la méthode du lien unique. Cet algorithme exploite le fait que la méthode du lien unique est stable pour la meilleure fusion, dans le sens suivant :

Définition 1. Stabilité de la meilleure fusion
Soit $G_{k,r}^{mf}$ la classe représentant la meilleure fusion pour la classe $G_{k,r}$ à l'étape r de construction d'un dendrogramme par une méthode d'agrégation $\mathcal{M}$, et soit $G_{k,r}^{mf}$ et $G_{l,r}$ les deux classes fusionnées à l'étape r, avec $G_{l,r} \neq G_{k,r}$ et $G_r = G_{k,r}^{mf} \cup G_{l,r}$. Nous dirons que $\mathcal{M}$ est stable pour la meilleure fusion *si et seulement si :*

$$G_{k,r+1}^{mf} = G_r$$

Question 1 Montrer que la méthode du lien unique est stable pour la meilleure fusion. On rappelle que la méthode du lien unique est fondée sur la distance entre classes :

Entrée : $\mathcal{C} = \{d_1, \cdots, d_N\}$, collection de documents
//Initialisation (matrice de similarité et files de priorité)
pour chaque $i \in \{1, \ldots, N\}$ **faire**
  pour chaque $j \in \{1, \ldots, N\}$ **faire**
    $C[i][j] \leftarrow sim(d_i, d_j)$;
  fin
  $I[i] \leftarrow 1$ //indicateur de classe active;
  Construire $P[i]$, file de priorité pour d_i triée suivant $C[i][]$;
fin
$L \leftarrow \emptyset$;
//Construction du dendrogramme
pour chaque $k \in \{1, \ldots, N-1\}$ **faire**
  $i_1 \leftarrow \text{argmax}_{i:I[i]=1} P[i].MAX.\text{sim}$;
  $i_2 \leftarrow P[i_1].MAX.\text{index}$;
  $L \leftarrow L.(i_1, i_2)$ //ajout de (i_1, i_2) à l'ensemble des fusions
  $I[i_2] \leftarrow 0$;
  Supprimer $C[i_1][i_2]$ de $P[i_1]$;
  pour chaque i *tel que* $I[i] = 1 \wedge i \neq i_1$ **faire**
    Supprimer $C[i][i_1]$ de $P[i]$ et $P[i_1]$ et $C[i][i_2]$ de $P[i]$;
    $C[i][i_1], C[i_1][i] \leftarrow \text{sim}_{\mathcal{M}}(i, (i_1, i_2))$;
    Insérer (avec tri) $C[i][i_1]$ dans $P[i]$ et $P[i_1]$;
  fin
fin
return L

Algorithme 2 : Algorithme de partitionnement hiérarchique agglomératif

$$\text{sim}_{\text{lu}}(G_k, G_l) \quad = \quad \max_{d \in G_k, d' \in G_l} sim(d, d')$$

Question 2 Quelle est la complexité de l'algorithme 3 qui utilise un tableau de meilleure fusion en lieu et place des files de priorité ?

Question 3 Expliquer pourquoi la stabilité de la meilleure fusion est importante pour le bon déroulement de cet algorithme (donner un exemple simple).

Question 4 Montrer que la méthode par lien unique est monotone.

Définition 2. Monotonicité
Soit $G_{(r)} = G_1^{(r)} \cup G_2^{(r)}$ *la classe obtenue à l'étape* r *de construction d'un dendrogramme par une méthode d'agrégation* $\mathcal{M}$ *et soit* $G^{(r+1)} = G_1^{(r+1)} + G_2^{(r+1)}$ *la classe obtenue à l'étape* $r+1$. *Nous*

Entrée : $\mathcal{C} = \{d_1, \cdots, d_N\}$, collection de documents
//Initialisation (matrice de similarité et tableaux de meilleure fusion)
pour chaque $i \in \{1, \ldots, N\}$ **faire**
  pour chaque $j \in \{1, \ldots, N\}$ **faire**
    $C[i][j].sim \leftarrow sim(d_i, d_j)$;
    $C[i][j].index \leftarrow j$;
  fin
  $I[i] \leftarrow i$;
  $TMF[i] \leftarrow \text{argmax}_{X \in \{C[i][j], i \neq j\}} X.sim$;
fin
$L \leftarrow \emptyset$;
//Construction du dendrogramme
pour chaque $k \in \{1, \ldots, N-1\}$ **faire**
  $i_1 \leftarrow \text{argmax}_{i:I[i]=i} TMF[i].sim$;
  $i_2 \leftarrow I[TMF[i_1].index]$;
  $L \leftarrow L.(i_1, i_2)$ //ajout de (i_1, i_2) à l'ensemble des fusions
  pour chaque $i \in \{1, \ldots, N\}$ **faire**
    si $I[i] = i \wedge i \neq i_1 \wedge i \neq i_2$ **alors**
      $C[i_1][i].sim \leftarrow C[i][i_1].sim \leftarrow \max(C[i_1][i].sim, C[i_2][i].sim)$;
    fin
    si $I[i] = i_2$ **alors**
      $I[i] \leftarrow i_1$;
    fin
    $TMF[i_1] \leftarrow \text{argmax}_{X \in \{C[i_1][i], I[i]=i \wedge i \neq i_1\}} X.sim$;
  fin
fin
return L

Algorithme 3 : Algorithme de partitionnement hiérarchique agglomératif pour le lien simple

dirons que $\mathcal{M}$ est monotone si et seulement si :

$$sim_{\mathcal{M}}\left(G_1^{(r)}, G_2^{(r)}\right) \geq sim_{\mathcal{M}}\left(G_1^{(r+1)}, G_2^{(r+1)}\right)$$