Classification basée sur l'agrégation d'opinions par la méthode de recuit simulé

M. Boubou[1] , A. Bounekkar[1] et M. Lamure[1]

Université Lyon I, LIRIS-MA2D
43 Boulevard du 11 Novembre 1918
69622 VILLEURBANNE CEDEX, France
(boubou,bounekkar,lamure)@univ-lyon1.fr

Résumé

Dans ce papier, nous allons présenter une méthode de classification basée sur l'agrégation d'opinions. La méthode proposée consiste à associer à chaque variable une fonction de classement qui va jouer le rôle de juge. Ce dernier va classer les individus selon ses propres critères. A partir de l'ensemble de classement de toutes les variables, on cherche à construire sur l'ensemble des individus un classement collectif qui soit la meilleure agrégation possible.
Afin de résoudre le problème d'optimisation rencontré, nous avons utilisé la méthode de recuit simulé.

Mots clés : classification, optimisation, recuit simulé, agrégation, partition.

1 Introduction

Dans ce papier, une méthode de classification basée sur l'agrégation d'opinions. Dans ce contexte, nous proposons une solution du problème d'agrégation des relations binaires en relation d'équivalence par l'utilisation de la méthode du recuit simulé.
Ce problème a été posé par *Régnier* [5] repris ensuite par *Marcotorchino et al.* [4], *Amorim et al.* [2] et *Barthélemy et al.* [1].

2 Présentation du problème

Nous considérons deux ensembles : L'ensemble des variables, $\mathcal{V} = \{V_1, V_2, \ldots, V_p\}$ et l'ensemble des individus, $\mathcal{I} = \{w_1, w_2, \ldots, w_n\}$. Nous associons à une variable V_k une application $A_k(.,.) : \mathcal{I}^2 \rightarrow \{0,1\}$ tel que $\forall (w_i, w_j) \in \mathcal{I}^2; w_i \neq w_j$

- $A_k(w_i, w_j) = 1$ Si pour V_k, w_i et w_j sont dans une même classe.
- $A_k(w_i, w_j) = 0$ Si pour V_k, w_i et w_j sont dans deux classes différentes.

Étant donné les classements des p variables $A_1, A_2, \ldots, A_p$ nous cherchons à construire sur $\mathcal{I}$, un classement collectif qui soit la meilleure agrégation possible qui va maximiser le nombre des concordances entre le classement collectif et les ensembles de classement des variables.
Une partition de l'ensemble $\mathcal{I}$ peut être assimilée à un vote défini sur $\mathcal{I}$ qui peut prendre autant des modalités qu'il y a d'éléments dans la partition.
Nous identifierons cette partition à une application $X(.,.)$ de $\mathcal{I}^2$ dans $\{0,1\}$.
Étant données, deux applications $X(.,.)$ et $Y(.,.)$ de $\mathcal{I}^2$ dans $\{0,1\}$, on appelle concordance de $X(.,.)$ et de $Y(.,.)$ notée $C(X,Y)$ le nombre entier défini par :

$$C(X,Y) = Card\{(w_i, w_j)\,;\, X(w_i, w_j) = Y(w_i, w_j)\}$$

Cette concordance peut se décomposer en deux catégories de concordances :

– Une concordance positive : $C^+(X,Y) = Card\{(w_i,w_j)\,;\, X(w_i,w_j) = Y(w_i,w_j) = 1\}$
– Une concordance négative : $C^-(X,Y) = Card\{(w_i,w_j)\,;\, X(w_i,w_j) = Y(w_i,w_j) = 0\}$

Étant donnée une partition caractérisée par l'application $X(.,.)$ de $\mathcal{I}^2$ dans $\{0,1\}$ et le classement d'une variable V_k, la concordance entre cette partition et le classement de la variable V_k est donnée par $C(X,A_k)$. Les différents classements de variables $A_k; k = 1,\ldots,p$ étant données, nous cherchons X qui maximise l'expression :

$$C(X,\mathcal{A}) = \sum_{k=1}^{p} C(X,A_k) \tag{1}$$

Ce problème de maximisation peut être formulé en terme de programmation linéaire en nombre entiers. En effet, pour tout $(w_i,w_j) \in \mathcal{I}^2$ on pose $x_{ij} = X(w_i,w_j)$ et pour tout $k = 1,\ldots,p$ on pose $a_{ij}^k = A_k(w_i,w_j)$

$$C(X,A_k) = Card\{(w_i,w_j)\,;\, X(w_i,w_j) = A_k(w_i,w_j)\} = Card\{(w_i,w_j)\,;\, x_{ij} = a_{ij}^k\}$$

Le couple (w_i,w_j) contribue donc à la cohérence dans tous les cas suivants :
– Cas 1 : $x_{ij} = a_{ij}^k = 1$ qui est équivalent à : $x_{ij}a_{ij}^k = 1$
– Cas 2 : $x_{ij} = a_{ij}^k = 0$ qui est équivalent à : $(1-x_{ij})(1-a_{ij}^k) = 1$

Par suite :

$$C(X,\mathcal{A}) = \frac{1}{n^2}\sum_{k=1}^{p}\sum_{(i,j)\in\mathcal{I}^2}(1-x_{ij})(1-a_{ij}^k) + x_{ij}a_{ij}^k$$

Etude du critère d'optimisation

Un algorithme de résolution de ce problème, connu sous le nom de "Clique partitioning", a été proposé dans [3, 2] en utilisant la programmation linéaire. Pour résoudre ce problème, nous proposons une approche heuristique : l'algorithme du recuit simulé.
Nous associons à une application X définie sur $\mathcal{I}^2$ deux vecteurs (x_{ij}) et $(\bar{x}_{ij})$ définis comme suit : $\forall (w_i,w_j) \in \mathcal{I}^2$:

$$\begin{cases} x_{ij} = 1 \text{ Si } X(w_i,w_j) = 1 \\ x_{ij} = 0 \text{ Sinon} \end{cases} \quad \text{et} \quad \begin{cases} \bar{x}_{ij} = 1 \text{ Si } X(w_i,w_j) = 0 \\ \bar{x}_{ij} = 0 \text{ Sinon} \end{cases}$$

Nous associons à la variable $V_k; (k = 1,\ldots,p)$ les deux vecteurs (a_{ij}^k) et $(\bar{a}_{ij}^k)$ définis comme suit $\forall (w_i,w_j) \in \mathcal{I}^2$:

$$\begin{cases} a_{ij}^k = 1 \text{ Si } A_k(w_i,w_j) = 1 \\ a_{ij}^k = 0 \text{ Sinon} \end{cases} \quad \text{et} \quad \begin{cases} \bar{a}_{ij}^k = 1 \text{ Si } A_k(w_i,w_j) = 0 \\ \bar{a}_{ij}^k = 0 \text{ Sinon} \end{cases}$$

Il y a une concordance entre la partition donnée par X et la partition donnée par A_k si : $(x_{ij} = 1 \text{ et } a_{ij}^k = 1)$ ou si $(\bar{x}_{ij} = 1 \text{ et } \bar{a}_{ij}^k = 1)$. Notons :

$$\begin{cases} q_{ij}^k = 1 \text{ s'il y a une concordance entre } X(w_i,w_j) \text{ et } A_k(w_i,w_j) \\ q_{ij}^k = 0 \text{ Sinon} \end{cases}$$

Le nombre de concordances $C(X,A_k) = \sum_{(i,j)\in\mathcal{I}^2} q_{ij}^k$ et le nombre de concordance entre X et les partitions données par $A_k; (k = 1,\ldots,p)$ peut s'écrire :

$$C(X,\mathcal{A}) = \sum_{k=1}^{p} C(X,A_k) = \sum_{k=1}^{p}\sum_{(i,j)\in\mathcal{I}^2}[x_{ij}a_{ij}^k + \bar{x}_{ij}\bar{a}_{ij}^k] = \sum_{k=1}^{p}\sum_{(i,j)\in\mathcal{I}^2}[x_{ij}a_{ij}^k + (1-x_{ij})\bar{a}_{ij}^k] = \sum_{k=1}^{p}\sum_{(i,j)\in\mathcal{I}^2}\bar{a}_{ij}^k + \sum_{k=1}^{p}\sum_{(i,j)\in\mathcal{I}^2}[a_{ij}^k - \bar{a}_{ij}^k]x_{ij}$$

$\forall (w_i,w_j) \in \mathcal{I}^2$ nous posons ; $(r_{ij} = \sum_{k=1}^{p} a_{ij}^k)$, $(\bar{r}_{ij} = \sum_{k=1}^{p} \bar{a}_{ij}^k)$ et $(s_{ij} = r_{ij} - \bar{r}_{ij})$. L'expression (1) devient alors

$$C(X,\mathcal{A}) = \sum_{(i,j)\in\mathcal{I}^2}\bar{r}_{ij} + \sum_{(i,j)\in\mathcal{I}^2} s_{ij}x_{ij} \tag{2}$$

3 Formulation du problème

La solution du problème revient à maximiser l'expression $C(X, \mathcal{A})$ donné par (2). Ceci revient à maximiser la quantité linéaire : $\sum_{(i,j)\in\mathcal{I}^2} s_{ij}x_{ij}$. Pour poser la propriété de transitivité imposée à X par des contraintes linéaires sur les x_{ij}, notons : $T = \{(i,j,k) \in I^3 | 1 \leq i < j < k \leq n\}$. Donc la propriété de transitivité est équivalente aux trois conditions linéaires suivantes :

$$\forall (i,j,k) \in T \begin{cases} x_{ij} + x_{jk} - x_{ik} \leq 1 \\ x_{ij} - x_{jk} + x_{ik} \leq 1 \\ -x_{ij} + x_{jk} + x_{ik} \leq 1 \end{cases} \tag{3}$$

Le problème devient un problème de programmation linéaire en nombres entiers suivant : $max \sum_{(i,j)\in\mathcal{I}^2} s_{ij}x_{ij}$, sachant que $x_{ij} \in \{0,1\}$ avec les contraintes linéaires citées dans le formule (3)

Remarque :
Le nombre de variables est de l'ordre de n^2, et le nombre de contraintes est de l'ordre de n^3. C'est ce qu'on appelle "click partitioning problem". Il est un NP-difficile [1],[2]. Plusieurs algorithmes ont été proposés pour la résolution de ce problèmes [3],[2]. Dans ce sens, nous proposons une autre méthode de résolution basée sur l'algorithm du recuit simulé.

4 Résolution du problème

Nous avons trouvé que la donnée d'une partition X vérifiant la propriété de la transitivité est équivalente à celle d'un vecteur $(x_{ij})_{(i,j)\in\mathcal{I}^2}$, chaque élément x_{ij} étant à valeur dans $\{0,1\}$ et vérifiant le condition (3).
Remarquons que la donnée d'un tel vecteur x_{ij} est équivalente à celle d'une partition $\mathcal{P}$ sur $\mathcal{I}$, selon le schéma suivant : $x_{ij} = 1 \Leftrightarrow \exists G \in \mathcal{P}$ tel que $i \in G$ et $j \in G$. Conformément à (2), nous notons : $val(\mathcal{P}) = \sum_{(i,j)\in\mathcal{I}^2} s_{ij}x_{ij}$
Si $\mathcal{P} = \{G_1, \ldots, G_m\}$, nous pouvons écrire :

$$val(\mathcal{P}) = \sum_{h=1}^{m} \sum_{\substack{(i,j)\in G_h \\ i<j}} s_{ij}$$

Notre problème prend alors la formulation suivante :

$$max\{val(\mathcal{P}) | \mathcal{P} \in \mathbb{P}\} \tag{4}$$

où $\mathbb{P}$ désigne l'ensemble des partitions de I.
Avant de résoudre ce problème (4), nous allons préciser la description d'un voisinage $\mathcal{V}(\mathcal{P})$ d'une partition $\mathcal{P} = \{G_1, \ldots, G_m\}$.

Définition 1 *Une partition $\mathcal{P}'$ appartient à $\mathcal{V}(\mathcal{P})$, si elle dérive de $\mathcal{P}$ en déplaçant un seul individu i_0 comme suit :*

- *Soit i_0 est transféré dans un groupe $G_l (l \neq h)$, de $\mathcal{V}(\mathcal{P})$,*
- *Soit i_0 constitue un nouveau groupe G_l à lui tout seul.*

Il s'en suit que : $\delta = val(\mathcal{P}') - val(\mathcal{P})$ et, en posant, pour simplifier l'écriture :

$$S_{ij} = \begin{cases} s_{ij} & Si \ \ i < j \\ s_{ji} & Si \ \ i > j \\ 0 & Si \ \ i = j \end{cases} \Longrightarrow \delta = \sum_{j\in G_l} S_{i_0 j} - \sum_{j\in G_h - \{i_0\}} S_{i_0 j}$$

Le schéma de la procédure est décrit par l'algorithme suivant :

```
Données : T_0, K_Max, r_0 , Résultat : P*
Begin
    Initialisation T := T_0
        choisir au hasard une partition P de I ; P* = P
    Répéter
        change :=FAUX
        Pour k := 1 à K_Max faire
            choisir au hasard P' dans V(P)
            calculer δ = val(P') − val(P)
            tirer au hasard une valeur R dans [0,1] (loi uniforme)
            si δ > 0 ou R < e^(δ/T) alors
                    P := P' ; change :=vrai ;
                si val(P) > val(P*) alors P* := P ;
                fin si
            fin Si
        Fin Pour
    T := T * r_o
    jusqu'à Not change
end
```

Remarques :

1. Remarquons que la définition de $\mathcal{V}(\mathcal{P})$ permet d'atteindre toutes les partitions possibles à partir d'une quelconque partition initiale en utilisant le procédé itératif.
2. Les paramètres T_0, K_{Max}, r_o sont respectivement la température initiale, le nombre d'itérations dans la boucle interne, et le coefficient de refroidissement. Nous avons choisi les valeurs de ces paramètres conformément aux recommandations de [3],[2], et plus précisément $K_{Max} = 10 \times card(I), r_o = 0.98$.
3. La valeur de T_0 doit garantir que $e^{(\delta/T_0)}$ soit proche de 1 en moyenne avec $\delta < 0$.

5 conclusion

L'algorithme du recuit simulé comporte un aspect aléatoire, ce qui explique que deux exécutions successives sur le même exemple ne donnent pas forcément le même résultat. Pour cela nous appellerons "Résultat" de l'algorithme, le meilleur des résultats obtenus lors de plusieurs exécutions successives.
La principale avantage de cette méthode de classification, est que les données ne sont pas forcement quantitatives.
Elle est basée sur l'opinion de l'expert du domaine, qui définit les fonctions de classement.

Références

[1] J.-P. Barthélemy et B. Leclerc, "The median procedure for partitions", *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **19**, (1995) 3–33.

[2] S.G. de Amorim, J.-P. Barthélemy et C.C. Ribeiro, "Clustering and clique partitioning : simulated annealing and tabu search approaches", *Journal of Classification*, **9**,(1992) 17–41

[3] M. Grötschel, Y. Wakabayashi, "A cutting plane algorithm for a clustering problem", *Mathematical Programming*, Series B, **45** (1989) 59–96

[4] J.F. Marcotorchino, P. Michaud, "Agrégation de similarités en classification automatique", *Revue de Statistique Appliquée*, Tom **30** no. 2 (1982)

[5] S. Regnier, "Sur quelques Aspects Mathématiques des Problèmes de Classification Automatique", *I.C.C. Bulletin*, No **4**, 175, (1965).