Co-classification sous contraintes par la somme des résidus quadratiques

Ruggero G. Pensa*, Jean-François Boulicaut**

*KDD-Lab, ISTI-CNR - Via Giuseppe Moruzzi, 1 - I-56124 Pisa, Italy
ruggero.pensa@isti.cnr.it
**INSA-Lyon, LIRIS CNRS UMR5205, F-69621 Villeurbanne cedex, France
jean-francois.boulicaut@insa-lyon.fr

Résumé. Dans de nombreuses applications, une co-classification est plus facile à interpréter qu'une classification mono-dimensionnelle. Il s'agit de calculer une bi-partition ou collection de co-clusters : chaque co-cluster est un groupe d'objets associé à un groupe d'attributs et les interprétations peuvent s'appuyer naturellement sur ces associations. Pour exploiter la connaissance du domaine et ainsi améliorer la pertinence des partitions, plusieurs méthodes de classification sous contraintes ont été proposées pour le cas mono-dimensionnel, e.g., l'exploitation de contraintes "must-link" et "cannot-link". Nous considérons ici la co-classification sous contraintes avec la gestion de telles contraintes étendues aux dimensions des objets et des attributs, mais aussi l'expression de contraintes de contiguité dans le cas de domaines ordonnés. Nous proposons un algorithme itératif qui minimise la somme des résidus quadratiques et permet l'exploitation active des contraintes spécifiées par les analystes. Nous montrons la valeur ajoutée de ce type d'extraction sur deux applications en analyse du transcriptome.

1 Introduction

Dans de nombreux domaines applicatifs, l'analyste se trouve devant des jeux de données matriciels dans lesquels un certain nombre d'objets sont décrits par un certain nombre d'attributs qui prennent leurs valeurs dans un domaine numérique, éventuellement restreint au domaine 0/1. L'une des techniques phares pour l'étude exploratoire de tels jeux de données est la classification, i.e., le calcul de partitions, soit sur l'ensemble des objets, soit sur l'ensemble des attributs. On peut aussi vouloir faciliter l'interprétation des groupements calculés en développant des méthodes de co-classification. Dans ce cas, les partionnements selon les deux dimensions sont couplés et les algorithmes comme ceux présentés dans Robardet et Feschet (2001); Dhillon et al. (2003); Ritschard et Zighed (2003); Jollois et al. (2003) produisent une bi-partition, i.e., une collection de co-clusters. Chacun des co-clusters est un groupe d'objets associé à un groupe d'attributs et la co-classification apparaît comme une méthode de classification conceptuelle. La co-classification a été particulièrement étudiée dans le contexte de l'analyse du transcriptome (voir, e.g., Cheng et Church (2000); Madeira et Oliveira (2004)). En effet, les technologies à haut débit permettent de construire des matrices d'expression de (tous

les) gènes d'un organisme dans différentes situations expérimentales. Dans ce type de matrices Expériences $\times$ Gènes, un co-cluster apparaît comme un ensemble de gènes ayant des profils d'expression similaires dans un ensemble de situations expérimentales. Il est alors possible de considérer chaque co-cluster comme un module de transcription putatif, i.e., une hypothèse sur un mécanisme de régulation génique et donc une réponse (partielle) au but de l'analyse du transcriptome qui a motivé la collecte des données d'expression.

Nous nous intéressons à la pertinence des bi-partitions. Il nous semble important que les analystes puissent spécifier leurs attentes (intérêt subjectif dérivé de la connaissance du domaine) au moyen de contraintes et que les techniques de co-classification puissent produire des résultats cohérents vis-à-vis de ces spécifications. Une co-classification est alors vue comme le calcul de $\{\phi \in \mathcal{L} \mid q(r, \phi) \text{ est vrai}\}$ où r est une matrice, $\mathcal{L}$ désigne le language des bi-partitions sur r, et le prédicat q spécifie des propriétés attendues sur ϕ. Une vision classique est que q va exprimer une contrainte d'optimisation sur une fonction objectif, e.g., la perte d'information mutuelle dans Dhillon et al. (2003). On peut également trouver d'autres contraintes comme la définition du nombre de co-clusters, le fait que certains objets (resp. attributs) doivent (resp. ne doivent pas) être ensembles (i.e., des contraintes habituellement désignées sous les noms de "must-link" et "cannot-link"). Combiner les méthodes d'optimisation sur la fonction objectif avec la satisfaction des autres contraintes est clairement difficile. L'introduction des contraintes comme "must-link" et "cannot-link" dans des processus de classification mono-dimensionnelle (e.g., K-Means, classification hiérarchique) a motivé de nombreux travaux ces 5 dernières années (Wagstaff et al., 2001; Klein et al., 2002; Bilenko et al., 2004; Davidson et Ravi, 2005a,b). Par contre, l'exploitation de contraintes pour la co-classification n'a été que très peu étudiée. Dans Pensa et al. (2006a,b), nous introduisions certains types de contraintes pour la co-classification. À coté des extensions de "must-link" et "cannot-link" pour qu'elles s'appliquent aux objets et/ou aux attributs, nous avons considéré le cas des dimensions ordonnées. Par exemple, dans le contexte de l'analyse de données d'expression, nous pouvons enregistrer l'évolution de l'expression des gènes au cours du temps. On peut alors spécifier des contraintes de contiguité (contrainte "Interval"). Dans nos travaux antérieurs, le calcul des bi-partitions était traité comme un post-traitement de collections de motifs qui capturent des associations localement fortes. Dans ce contexte, l'idée était de pouvoir "pousser" certaines contraintes sur la bi-partition jusque dans l'extraction des motifs locaux à post-traiter.

Nous présentons ici une approche de co-classification sous contraintes originale et alternative aux propositions décrites dans Pensa et al. (2006a,b). Ces dernières étaient réservées au traitement de données 0/1. Ici, nous travaillons sur des données numériques et sans passer par des collections de motifs locaux. Pour celà,, nous étendons la technique de minimisation des résidus quadratiques proposée dans Cho et al. (2004) à un cadre de classification sous contraintes. La Section 2 formalise le problème. La Section 3 présente notre algorithme de co-classification sous contraintes. La Section 4 est dédiée à une validation expérimentale sur deux jeux de données réelles en analyse du transcriptome. La Section 5 est une brève conclusion.

2 Co-classification sous contraintes

Désignons par $X = [x_{ij}] \in \mathbb{R}^{m \times n}$ une matrice de nombres réels à m lignes et n colonnes. Dans la suite, le jeu de données à fouiller sera la matrice X et nous parlerons de lignes et de colonnes plutôt que d'objets et d'attributs. Soit x_{ij} l'élément correspondant à la ligne i et à la

colonne j. Par exemple, x_{ij} peut contenir le niveau d'expression du gène i dans la condition expérimentale j. Nous noterons $x_{i.}$ et $x_{.j}$ les vecteurs associés respectivement à la ligne i et à la colonne j. Une co-classification $C^{k\times l}$ sur X produit simultanément un ensemble de $k \times l$ co-clusters (une partition C^r en k groupes de lignes associée à une partition C^c en l groupes de colonnes). Soit I_r l'ensemble des indices des lignes appartenant à la classe de lignes r, et J_c l'ensemble des indices des colonnes dans la classe de colonnes c. La sous-matrice de X déterminée par I_r et J_c est nommée co-cluster. Pour avoir un premier critère de qualité sur les co-classifications, nous cherchons toujours à optimiser une certaine fonction objectif.

Définition 1 (contrainte d'optimisation) *Soit $f(X, C^{k\times l})$ une fonction objectif, la contrainte d'optimisation $c_{opt}(f, X, C^{k\times l})$* est satisfaite ssi *$C^{k\times l} = \text{argmin}_{\phi\in\mathcal{L}_{\mathcal{C}^{k\times l}}} f(X, \phi)$ où $\mathcal{L}_{\mathcal{C}^{k\times l}}$ est l'ensemble de toutes les co-classifications possibles.*

Des exemples de fonction objectif sont le coefficient τ de Goodman-Kruskal utilisé dans Robardet et Feschet (2001) ou la perte d'information mutuelle exploitée dans Dhillon et al. (2003). Dans cet article, nous utilisons la somme des résidus quadratiques introduite dans Cho et al. (2004). Pour des raisons de faisabilité calculatoire, les algorithmes de co-classification relaxent ces contraintes d'optimisation avec une mise en œuvre d'heuristiques d'optimisations locales. En sus des contraintes d'optimisation qui sont souvent implicites, on veut pouvoir spécifier d'autres types de contraintes qui sont maintenant définis.

Définition 2 (contraintes must-link et cannot-link) *Si deux lignes i_a et i_b (resp. colonnes j_a et j_b) sont impliquées dans une contrainte **must-link**, notée $c_{=}(i_a, i_b)$ (resp. $c_{=}(j_a, j_b)$), elles doivent être dans la même classe de $C^r = \{I_1, \ldots, I_k\}$ (resp $C^c = \{J_1, \ldots, J_l\}$). Si deux lignes i_a, i_b (resp. colonnes j_a et j_b) sont impliquées dans une contrainte **cannot-link**, notée $c_{\neq}(i_a, i_b)$ (resp. $c_{\neq}(j_a, j_b)$), elles ne peuvent pas être dans la même classe de $C^r = \{I_1, \ldots, I_k\}$ (resp $C^c = \{J_1, \ldots, J_l\}$).*

Ces formes de contraintes ont été très étudiées dans le cadre de la classification semi-supervisée (Bilenko et al., 2004). Nous les généralisons ici pour qu'elles puissent s'appliquer aussi bien à l'ensemble des lignes qu'à l'ensemble des colonnes. Dans une matrice d'expression, on peut ainsi exploiter des connaissances sur les gènes et/ou sur les conditions expérimentales. Par exemple, si l'on sait que le gène i_a et le gène i_b ont la même fonction (disons F) dans un processus biologique, on peut vouloir forcer une contrainte must-link entre ces deux gènes afin de privilégier la recherche d'un co-cluster associant des gènes ayant cette fonction F et donc d'identifier un module de transcription qui serait à l'origine de cette fonction. Nous pourrions également ajouter des contraintes cannot-link pour éviter d'associer dans les co-clusters des situations expérimentales que l'on souhaite séparer (e.g., séparer différents type de tissus ou de lignées cellulaires).

Nous faisons aussi l'hypothèse qu'une valeur réelle $s_c(j)$ (resp. $s_r(i)$) est associée à chaque colonne j (resp. ligne i). Nous avons donc $s_r : \{1, 2, \ldots, m\} \to \mathbb{R}$ et $s_c : \{1, 2, \ldots, n\} \to \mathbb{R}$. Par exemple, $s_c(j)$ (resp. $s_r(i)$) pourrait être une mesure temporelle ou spatiale liée à j (resp. i). Dans des données issues de Puces ADN, où chaque colonne correspond à une puce (i.e., une expérience) et chaque ligne désigne un gène, $s_c(j)$ pourrait être le temps d'échantillonnage liée à la puce ADN j. Un second exemple serait de considérer $s_r(i)$ comme une mesure de la position (spatiale) absolue du gène i dans la séquence d'ADN de l'organisme étudié. Les deux fonctions s_r et s_c permettent de définir un ordre $\preceq$ sur l'ensemble des colonnes et/ou

des lignes. On dit alors que $j_a \preceq j_b$ ssi $s_c(j_a) \leq s_c(j_b)$. Dans la suite, on considère que, si une fonction s_c existe, alors tous les éléments j sont ordonnés, i.e., $\forall j_a, j_b$ tels que $j_a < j_b$, $s_c(j_a) \leq s_c(j_b)$ (idem pour les lignes). Il devient alors intéressant de rechercher des co-clusters qui soient cohérents avec les ordres définis par les fonctions s_r et s_c. Par exemple, si l'on s'intéresse aux différentes étapes du développement d'une cellule et si l'on veut découvrir les gènes majoritairement impliqués dans chaque étape, nous pouvons chercher des classes qui soient contiguës dans le temps. Pour cela, on peut forcer la contrainte d'intervalle introduite dans Pensa et al. (2006a,b).

Définition 3 (contrainte d'intervalle) *Si un ordre ($\preceq$) est défini sur l'ensemble des colonnes (resp. des lignes), une contrainte **interval** sur cet ensemble, notée $c_{int}(C^c)$, exige que chaque classe dans C^c soit un intervalle : $\forall J_c \in C^c$, si $j_a, j_b \in J_c$ alors $\forall j_c$ tel que $j_a \preceq j_c \preceq j_b$, $j_c \in J_c$.*

La satisfaction des contraintes must-link, cannot-link et interval entraîne en général une diminution de l'optimum théorique de la fonction objectif. Nous voulons un algorithme de co-classification qui puisse prendre en compte de telles contraintes tout en essayant d'optimiser la fonction objectif retenue. Notez que, la satisfaction d'une conjonction de contraintes $c_=$, $c_\neq$ et c_{int} n'est pas toujours faisable. Par exemple, pour trois objets i_1, i_2, i_3 tels que $s(i_1) < s(i_2) < s(i_3)$, la conjonction $c_=(i_1, i_3) \wedge c_\neq(i_1, i_2) \wedge c_{int}(C^r)$ ne peut jamais être satisfaite même si les sous-contraintes de cette conjonction ne posent aucun problème. Dans cet article, nous ferons l'hypothèse que les conjonctions de contraintes traitées par notre approche sont toujours faisables. Le problème de la faisabilité des contraintes pour les méthodes partitionnelles et hiérarchiques a été largement traité dans Davidson et Ravi (2005a,b).

3 Exploitation de la somme des résidus quadratiques

Nous présentons notre approche de co-classification sous contraintes en proposant un algorithme itératif qui minimise la somme des résidus quadratiques. Cette fonction objectif a été introduite dans Cho et al. (2004) pour la co-classification sans contraintes appliquée au contexte des matrices d'expression. Il s'agit d'une adaptation de la mesure introduite dans Cheng et Church (2000) pour la découverte de motifs locaux dans les matrices d'expression.

Soit $X \in \mathbb{R}^{m \times n}$ la matrice de données, nous cherchons une partition de X en k classes de lignes, et l classes de colonnes. Utilisons la définition de résidu de Cheng et Church (2000).

Définition 4 (résidu) *Soit $x_{ij} \in X$, le résidu de x_{ij} dans le co-cluster contenant x_{ij} et déterminé par les ensembles d'indices I_r et J_c, de cardinalités respectives $|I_r|$ et $|J_c|$ est donné par*

$$h_{ij} = x_{ij} - x_{I_r j} - x_{i J_c} + x_{I_r J_c} \tag{1}$$

où $x_{I_r J_c} = \frac{\sum_{i \in I_r, j \in J_c} x_{ij}}{|I_r| \cdot |J_c|}$, $x_{I_r j} = \frac{\sum_{i \in I_r} x_{ij}}{|I_r|}$, *et* $x_{i J_c} = \frac{\sum_{j \in J_c} x_{ij}}{|J_c|}$.

Soit $H = [h_{ij}] \in \mathbb{R}^{m \times n}$ la matrice des résidus calculés avec la définition précédente. La fonction objectif que l'on veut minimiser est la somme des résidus quadratiques de Cho et al. (2004) calculée de la manière suivante :

$$||H||^2 = \sum_{r,c} ||h_{I_r J_c}||^2 = \sum_{r,c} \sum_{i \in I_r, j \in J_c} h_{ij}^2 \tag{2}$$

On peut réécrire la matrice des résidus dans une forme plus compacte si l'on introduit les deux matrices $R \in \mathbb{R}^{m \times k}$ et $C \in \mathbb{R}^{n \times l}$ ainsi définies : chaque élément (i, r) $(1 \leq r \leq k)$ de R est égal à $m_r^{-1/2}$ si i est dans la classe r ($m_r = |I_r|$ étant le nombre de lignes dans la classe r), 0 autrement. Chaque élément (j, c) $(1 \leq c \leq l)$ de la matrice C est égal à $n_c^{-1/2}$ si j est dans la classe c ($n_c = |J_c|$ étant le nombre de colonnes dans la classe c), 0 autrement. La matrice des résidus dévient alors :

$$H = (I - RR^T)X(I - CC^T) \tag{3}$$

La démonstration de la validité de cette équation est dans Cho et al. (2004). Les auteurs démontrent d'abord que $(RR^TX)_{ij} = x_{I_r j}$, $(XCC^T)_{ij} = x_{iJ_c}$ et $(RR^TXCC^T)_{ij} = x_{I_r J_c}$, avant de montrer que (3) est vraie. Ils concluent que, si l'on considère la projection $(I - RR^T)X$ de la matrice X, alors $||H||^2$ donne la fonction objectif de K-MEANS pour cette matrice modifiée.

Considérons maintenant notre contribution algorithmique. Notre approche utilise une technique dite "ping-pong" pour traiter de manière alternée (avec la méthode K-MEANS) les colonnes et les lignes. Ainsi, la matrice C n'est mise à jour qu'après que chaque colonne ait été affectée à la classe de colonnes la plus proche (similairement pour les lignes). Nous proposons donc de décomposer le calcul de la fonction objectif. Soit $X^P = (I - RR^T)X$, $X^C = (I - RR^T)XC$, et $\hat{X}^P = (I - RR^T)XCC^T = X^C C^T$, nous obtenons la réécriture suivante :

$$\begin{aligned} ||X^P - \hat{X}^P||^2 &= \sum_{c=1}^{l} \sum_{j \in J_c} ||X^P_{.j} - \hat{X}^P_{.j}||^2 \\ &= \sum_{c=1}^{l} \sum_{j \in J_c} ||X^P_{.j} - (X^C C^T)_{.j}||^2 \\ &= \sum_{c=1}^{l} \sum_{j \in J_c} ||X^P_{.j} - n_c^{1/2} X^C_{.c}||^2. \end{aligned} \tag{4}$$

De la même manière, en posant $X^P = X(I - CC^T)$, $X^R = R^T X(I - CC^T)$, et $\hat{X}^P = RR^T X(I - CC^T) = RX^R$, nous obtenons comme décomposition en termes de lignes :

$$||X^P - \hat{X}^P||^2 = \sum_{r=1}^{k} \sum_{j \in I_r} ||X^P_{i.} - m_r^{1/2} X^R_{r.}||^2. \tag{5}$$

Les matrices X^C et X^R correspondent alors aux centres de masse des classes, respectivement pour les colonnes et pour les lignes.

Nous pouvons maintenant introduire notre algorithme de co-classification sous contraintes. Nous donnons d'abord une version qui traite la satisfaction d'une conjonction de contraintes must-link et cannot-link puis une version traitant la contrainte interval. Nous proposons enfin une stratégie pour l'intégration de ces deux traitements.

3.1 Satisfaction de conjonctions must-link et cannot-link

La transitivité des contraintes must-link est bien connue. On peut alors transformer un ensemble de contraintes must-link sur les lignes en une collection $\mathcal{M}_r = M_1, \ldots, M_N$,

où chaque M_i est un ensemble de lignes impliquées par la même fermeture transitive de contraintes must-link. Notons $\mathcal{M}_c$ le même ensemble construit pour les colonnes et soit $\mathcal{C}_r$ et $\mathcal{C}_c$ les ensembles des contraintes cannot-link respectivement pour les lignes et les colonnes.

Algorithme 1 : ConsCoClust(X,k,l,$\mathcal{M}_r$,$\mathcal{M}_c$,$\mathcal{C}_r$,$\mathcal{C}_c$)

Entrées : Une matrice de données X, k, l, les ensembles de contraintes cannot-link $\mathcal{C}_r$ et $\mathcal{C}_c$, et les collections d'ensembles de lignes $\mathcal{M}_r$ et $\mathcal{M}_c$
Sorties : Matrices R et C
Initialiser R et C;
$\Delta = 1\,; \tau = 10^{-2}||X||^2\,; t = 0\,; obj^t = ||(I - RR^T)X(I - CC^T)||^2$;
tant que $\Delta > \tau$ **faire**
  $t = t + 1\,; X^C = (I - RR^T)XC\,; X^P = (I - RR^T)X$;
  pour chaque $1 \leq j \leq n$ **faire**
    $L = \emptyset$;
    si $\exists M_v \in \mathcal{M}_c$ *t.q.* $j \in M_v$ **alors**
      pour chaque $j_v \in M_v$ **faire**
        $L = L \cup \{1 \leq c \leq l \mid \nexists j_c \mid \gamma^t[j_c] = c \wedge c_{\neq}(j_v, j_c) \in \mathcal{C}_c\}$;
      fin
      $\gamma^t[M_v] = \text{argmin}_{c \in L} \frac{\sum_{j_v \in M_v} ||X^P_{.j} - n_c^{-1/2} X^C_{.c}||^2}{|M_v|}$;
    sinon
      $L = \{1 \leq c \leq l \mid \nexists j_c \mid \gamma^t[j_c] = c \wedge c_{\neq}(j, j_c) \in \mathcal{C}_c\}$;
      $\gamma^t[j] = \text{argmin}_{c \in L} ||X^P_{.j} - n_c^{-1/2} X^C_{.c}||^2$;
    fin
  fin
  Mettre à jour C à partir de γ;
  $X^R = R^T X(I - CC^T)\,; X^P = X(I - CC^T)$;
  {Affectation des lignes}($\star$);
  $obj^t = ||(I - RR^T)X(I - CC^T)||^2\,; \Delta = |obj^t - obj^{t-1}|$;
fin

L'algorithme 1, traite la co-classification en présence de conjonctions de contraintes must-link et cannot-link (la partie concernant le traitement des lignes ($\star$) est omise). Il commence par initialiser (e.g., aléatoirement) les matrices C et R. À chaque itération, l'algorithme affecte chaque colonne (ligne) à la classe de colonnes (lignes) la plus proche qui n'introduit pas une violation des contraintes cannot-link. Si une colonne (ligne) est impliquée dans une contrainte must-link, nous affectons l'ensemble des colonnes (lignes) impliquées par la fermeture transitive de cette contrainte à la classe de colonnes (lignes) pour laquelle la moyenne des distances est minimum, en contrôlant toujours qu'il n'y ait pas de contrainte cannot-link violée par cette opération. Ensuite, l'algorithme met à jour la matrice C (R) selon le schéma d'affectation résultant des opérations décrites précédemment. Le processus est réitéré jusqu'à ce que la diminution de la fonction objectif devienne très petite (i.e., inférieure à un facteur de tolérance τ). Notons que la phase d'initialisation peut ne pas prendre en compte les contraintes, car leur satisfaction est assurée par la première itération. Une amélioration possible consiste à utiliser un meilleur critère d'affectation pour les objets impliqués dans des contraintes cannot-link. On

sait d'ailleurs que la satisfaction d'un ensemble de contraintes cannot-link pour un nombre de classes donné est un problème **NP**-complet (Davidson et Ravi, 2005b).

3.2 Satisfaction de la contrainte "interval"

Algorithme 2 : IntCoClust(X,k,l)

Entrées : Une matrice de données X, k et l, int_r, int_c
Sorties : Matrices R et C
Initialiser R, C, $gauche$, $droite$; ($\star$)
$\Delta = 1$; $\tau = 10^{-2}||X||^2$; $t = 0$; $obj^t = ||(I - RR^T)X(I - CC^T)||^2$;
tant que $\Delta > \tau$ **faire**
  $t = t + 1$; $X^C = (I - RR^T)XC$; $X^P = (I - RR^T)X$;
  pour chaque $1 \leq c \leq l$ **faire**
    $fin = faux$;
    tant que $c > 1 \wedge fin = faux \wedge droite[c] > gauche[c]$ **faire**
      $j = gauche[c]$;
      si $||X^P_{.j} - n_{c-1}^{-1/2} X^C_{.c-1}||^2 < ||X^P_{.j} - n_c^{-1/2} X^C_{.c}||^2$ **alors**
        $\gamma^t[j] = c - 1$; $gauche[c] = gauche[c] + 1$;
        $droite[c-1] = droite[c-1] + 1$;
      sinon
        $fin = vrai$;
      fin
    fin
    {Traitement des frontières droites};
  fin
  Mettre à jour C à partir de γ;
  $X^R = R^T X(I - CC^T)$; $X^P = X(I - CC^T)$;
  {Affectation des lignes}($\star\star$);
  $obj^t = ||(I - RR^T)X(I - CC^T)||^2$; $\Delta = |obj^t - obj^{t-1}|$;
fin

L'algorithme 2 permet de résoudre le problème de la satisfaction de la contrainte "interval" (la partie concernant le traitement des lignes ($\star\star$) est omise). Dans ce cas, l'initialisation ($\star$) des partitions concernées par cette contrainte doit produire un nombre l (resp. k) d'intervalles sur les colonnes (resp. les lignes). Ensuite, le processus d'affectation s'intéresse uniquement aux frontières entre les intervalles. Plus particulièrement, il traite itérativement d'abord la frontière gauche, puis la frontière droite. Une colonne (resp. ligne) peut être affectée à l'intervalle adjacent si la distance est inférieure à celle calculée sur l'intervalle de départ. Dans ce cas, on continue à traiter les colonnes (resp. lignes) restantes. Lorsque la frontière gauche et la frontière droite d'un intervalle correspondent à la même colonne (resp. ligne), l'algorithme passe à la frontière suivante. Dans le cas où il n'est pas nécessaire de réaffecter la colonne (resp. ligne), l'algorithme termine le traitement de cette frontière et passe à la frontière suivante. Notez que, contrairement à Pensa et al. (2006b), la satisfaction de la contrainte interval est bien garantie sur la co-classification résultat.

L'intégration des deux algorithmes pour traiter une conjonction de contraintes must-link, cannot-link et interval n'a pas encore été traitée. Donnons cependant une piste de recherche. Il s'agit d'abord de faire en sorte que chaque ensemble $M_r \in \mathcal{M}_r$ (ou $M_c \in \mathcal{M}_c$) soit un intervalle. Par exemple, pour un ensemble d'objets $\{i_1, i_2, i_3, i_4, i_5\}$, et un ensemble $M_r = \{i_2, i_4\}$, nous serions obligés d'inclure l'objet i_3 dans M_r par la définition même d'un intervalle. Ensuite, il faut que l'initialisation produise une partition qui prenne en compte l'ensemble des contraintes (notons que la satisfaction d'une conjonction de contraintes cannot-link est un problème **NP**-complet). Enfin, il est possible d'utiliser la stratégie de l'algorithme 1 uniquement sur les frontières, suivant le schéma présenté dans l'algorithme 2.

Complexité Concernant l'algorithme 1, notons que pour calculer $(I - RR^T)X(I - CC^T)$, le nombre d'opérations nécessaires est celui qu'il faut pour calculer $R^T XC$, i.e., $kn(m + l)$ opérations. Ce calcul peut donc être effectué dans un temps $O(N)$ (quand $N = mn$) dans l'hypothèse vraisemblable où $k \simeq l << m \simeq n$. La phase d'affectation des lignes et colonnes aux nouvelles classes, nécessite un temps $O(N(k + l)$ à chaque itération. La complexité totale de l'algorithme est donc en $O(N(k + l)t)$, où t est le nombre total d'itérations nécessaires à l'algorithme pour compléter la co-classification. La complexité de l'algorithme 2, est trivialement la même que celle de l'algorithme 1. Noter que, en général, le fait de travailler uniquement sur les frontières, se traduit par une meilleure efficacité dans la phase d'affectation.

4 Validation expérimentale

Nous avons étudié le comportement de nos algorithmes dans deux jeux de données "Puces ADN" nommés plasmodium et drosophila. Le premier décrit dans Bozdech et al. (2003) concerne le transcriptome du cycle de développement intraerythrocytique du Plasmodium Falciparum, i.e., un agent responsable de la malaria humaine. Les données fournissent le profil d'expression de 3 719 gènes dans 46 échantillons biologiques. Chaque échantillon correspond à un moment dans le cycle de développement : il commence avec l'invasion des globules rouges du sang par le mérozoïte, et il est divisé en trois phases : anneau, trophozoïte et schizonte (concernant respectivement le moustique, le foie, et le sang). Après 48 heures, le cellule se réplique et se divise. Aux instants marqués 17h et 29h, on observe deux transitions brusques. Le second jeu de données est décrit dans Arbeitman et al. (2002). Il concerne l'expression des gènes de la Drosophile melanogaster durant son cycle de vie. Les niveaux d'expression de 3 944 gènes sont mesurés pour 57 périodes séquentielles de temps divisés en stade embryonnaire, larvaire et pupaire.

Dans toutes nos expériences, la valeur du paramètre d'arrêt τ a été fixée à $10^{-4}||X||^2$. L'initialisation des partitions étant aléatoire, nous avons exécuté nos algorithmes 20 fois pour chaque groupe de contraintes.

4.1 Contraintes must-link et cannot-link

Nous avons d'abord étudié le traitement des contraintes must-link et cannot-link sur l'ensemble des gènes uniquement. Pour cela, nous avons utilisé le jeu de données plasmodium, pour lequel on dispose d'un certain nombre d'information sur les gènes impliqués dans les différentes étapes du développement. En particulier, nous avons considéré le groupe *cytoplasmic*

translation machinery (159 gènes), actif dans la première phase du cycle de vie de la bactérie, le groupe *merozoite invasion* (87 gènes) actif dans la seconde phase, et le groupe *early ring transcripts* (34 gènes), caractérisant la dernière phase de son développement. Tous ces groupes fonctionnels sont décrits dans Bozdech et al. (2003). Nous avons sélectionné aléatoirement 20 ensembles de contraintes sur la base des trois groupes de gènes précédemment décrits. Chaque ensemble contient un nombre variable de contraintes, et le nombre de gènes impliqués dans chaque ensemble varie entre le 20% et le 50% des gènes impliqués dans les trois groupes fonctionnels. Pour cette expérience, nous avons utilisé $k = 3$ and $l = 3$, afin de pouvoir identifier les trois étapes du développement de Plasmodium Falciparum.

	CTM	MI	ERT
C1	**94.43%**	0.40%	20.44%
C2	0	**80.11%**	1.32%
C3	5.57%	19.49%	**78.24%**

(a)

	CTM	MI	ERT
C1	**95.87%**	0.39%	19.26%
C2	0.01%	**86.18%**	2.09%
C3	4.12%	13.43%	**78.65%**

(b)

TAB. 1 – *Pourcentage de gènes dans les trois classes pour l'approche sans contrainte (a) et avec contraintes (b).*

Les résultats présentés dans le tableau 1 montrent, pour chaque groupe fonctionnel, le pourcentage de gènes impliqués dans chaque classe. L'amélioration est beaucoup plus sensible pour le second groupe de gènes (merozoite invasion). Le dernier groupe (early ring transcripts) ne semble pas bénéficier de l'exploitation des contraintes.

Pour mesurer l'impact de l'utilisation combinée de contraintes sur l'ensemble des lignes et l'ensemble des colonnes, nous avons choisi une co-classification cible parmi les résultats obtenus sans l'utilisation des contraintes. En particulier, nous avons sélectionné la co-classification avec la valeur minimale de la fonction objectif obtenue à la fin du processus itératif. Cette valeur était de 1.99×10^4. Nous avons ensuite généré aléatoirement 20 ensembles de contraintes impliquant gènes et conditions expérimentales. Le nombre de gènes impliqués dans les contraintes varie entre 5% et 10% de la taille totale de l'ensemble de gènes. Pour les conditions expérimentales, ce nombre varie entre 15% et 25%. Pour évaluer la conformité entre la bi-partition choisie et les partitions découvertes par l'algorithme de co-classification, nous avons utilisé l'indice de Rand corrigé (Hubert et Arabie, 1985). Si $\mathbf{C} = \{C_1 \dots C_z\}$ est la structure issue de la classification et que $\mathbf{P} = \{P_1 \dots P_z\}$ est une partition prédéfinie, chaque paire de points peut être affecté à la même classe ou à deux classes différentes. Soit a le nombre de paires appartenant à la même classe de $\mathbf{C}$ et à la même classe de $\mathbf{P}$, soit

$$exp(a) = \frac{\sum_{k=1}^{z} |C_k|(|C_k| - 1)/2 \cdot \sum_{k=1}^{z} |P_k|(|P_k| - 1)/2}{p(p-1)/2}$$

l'espérance de a (p étant le nombre de points), et

$$max(a) = \frac{1}{2}\left(\sum_{k=1}^{z} \frac{|C_k|(|C_k| - 1)}{2} + \sum_{k=1}^{z} \frac{|P_k|(|P_k| - 1)}{2}\right)$$

la valeur maximale de a. La conformité entre $\mathbf{C}$ et $\mathbf{P}$ peut être estimée au moyen de la formule :

$$RC(\mathbf{C}, \mathbf{P}) = \frac{a - exp(a)}{max(a) - exp(a)}$$

Lorsque $RC(\mathbf{C}, \mathbf{P}) = 1$, les deux partitions sont identiques.

	RC_r	RC_c	$\|\|H\|\|^2$	N.Itérations
Const.	**0.88**	**0.73**	**2.16×10^4**	**9.18**
Unconst.	0.70	0.43	2.21×10^4	9.35

TAB. 2 – *Index de Rand corrigé, valeur finale de la fonction objectif et nombre d'itérations (valeurs moyennes).*

Nous avons comparé les résultats obtenus avec l'utilisation des contraintes avec ceux qui ont été obtenus sans aucune spécification de contrainte. Le résumé de cette expérience est présenté dans le tableau 2. On voit que l'utilisation de contraintes produit une amélioration très nette de la conformité des deux partitions, en entraînant une légère diminution du nombre moyen d'itérations nécessaire pour compléter la co-classification. Dans le même temps, la spécification de contraintes entraîne une amélioration de la valeur finale de la fonction objectif (diminution d'environ 2%).

4.2 Contrainte interval

Pour évaluer la valeur ajoutée de la contrainte interval, nous avons appliqué notre algorithme au jeu de données drosophila. Notre objectif est ici de redécouvrir les trois phases du cycle de vie de la drosophile en utilisant comme seule connaissance le nombre de classes ($k = l = 3$).

	RC	$\|\|H\|\|^2$	N.Itérations
Const.	**0.76**	8.23×10^4	**11.60**
Unconst.	0.41	7.73×10^4	14.60

TAB. 3 – *Index de Rand corrigé, valeur finale de la fonction objectif et nombre d'itérations (valeurs moyennes).*

Nous avons donc comparé l'index de Rand obtenu avec et sans l'utilisation de la contrainte interval pour une collection de 20 exécutions. Les résultats (voir tableau 3) montrent clairement que l'utilisation de la contrainte interval permet de retrouver de façon plus nette les trois phases du cycle de vie de la drosophile (l'amélioration mesurée sur l'index de Rand est d'environ 85%). De plus, le nombre d'itérations nécessaires pour compléter la co-classification est sensiblement inférieur à celui que l'on obtient si l'on n'utilise pas la contrainte interval. Notons que la valeur finale de la fonction objectif et meilleure dans la version non contrainte. Cela signifie que la structure découverte par notre algorithme est loin d'être l'optimum global pour ce jeu de données. Notons aussi que, dans aucun des cas, l'algorithme sans contrainte n'a pu retrouver des intervalles.

5 Conclusion

La co-classification permet une interprétation plus facile des groupements qu'une classification mono-dimensionnelle. Nous nous sommes posés le problème de l'exploitation de contraintes dans une co-classification. Permettre la définition de contraintes, c'est autoriser l'exploitation de connaissances du domaine pour obtenir des groupements plus pertinents. Encore faut-il être capable de combiner l'optimisation des fonctions objectifs (au cœur des algorithmes de classification) et la satisfaction de contraintes comme des contraintes must-link ou cannot-link étendues au contexte de la co-classification. Contrairement à la seule proposition de co-classification sous contraintes que nous connaissons (Pensa et al., 2006a,b), nous proposons ici une méthode qui travaille directement sur les données numériques et qui garantit le respect des contraintes spécifiées. À cette fin, nous nous appuyons sur la fonction objectif des résidus quadratiques introduite dans Cho et al. (2004). L'une des perspectives à court terme de ce travail consiste à valider les pistes identifiées pour l'intégration entre la résolution des contraintes d'intervalle et celle des autres types de contraintes.

Remerciements Ce travail a été réalisé en grande partie alors que Ruggero Pensa était ATER à l'Université de Saint-Etienne. Ce travail est partiellement financé par le contrat européen IST FET IQ (FP6 516169).

Références

Arbeitman, M., E. Furlong, F. Imam, E. Johnson, B. Null, B. Baker, M. Krasnow, M. Scott, R. Davis, et K. White (2002). Gene expression during the life cycle of drosophila melanogaster. *Science 297*, 2270–2275.

Bilenko, M., S. Basu, et R. J. Mooney (2004). Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings ICML 2004*, Banff, Canada, pp. 81–88.

Bozdech, Z., M. Llinás, B. L. Pulliam, E. Wong, J. Zhu, et J. DeRisi (2003). The transcriptome of the intraerythrocytic developmental cycle of plasmodium falciparum. *PLoS Biology 1*(1), 1–16.

Cheng, Y. et G. M. Church (2000). Biclustering of expression data. In *Proceedings ISMB 2000*, San Diego, USA, pp. 93–103. AAAI Press.

Cho, H., I. S. Dhillon, Y. Guan, et S. Sra (2004). Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings SIAM SDM 2004*, Lake Buena Vista, USA.

Davidson, I. et S. S. Ravi (2005a). Agglomerative hierarchical clustering with constraints : Theoretical and empirical results. In *Proceedings PKDD 2005*, Volume 3721 of *LNCS*, Porto, Portugal, pp. 59–70. Springer.

Davidson, I. et S. S. Ravi (2005b). Clustering with constraints : Feasibility issues and the k-means algorithm. In *Proceedings SIAM SDM 2005*, Newport Beach, USA.

Dhillon, I. S., S. Mallela, et D. S. Modha (2003). Information-theoretic co-clustering. In *Proceedings ACM SIGKDD 2003*, Washington, USA, pp. 89–98. ACM Press.

Hubert, L. et P. Arabie (1985). Comparing partitions. *Journal of Classification 2*(1), 193–218.

Jollois, F.-X., M. Nadif, et G. Govaert (2003). Classification croisée sur des données binaires de grande taille. In *Actes EGC 2003*, Lyon, France, pp. 213–218. Hermes Science Publications.

Klein, D., S. D. Kamvar, et C. D. Manning (2002). From instance-level constraints to space-level constraints : Making the most of prior knowledge in data clustering. In *Proceedings ICML 2002*, Sydney, Australia, pp. 307–314.

Madeira, S. C. et A. L. Oliveira (2004). Biclustering algorithms for biological data analysis : A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 1*(1), 24–45.

Pensa, R., C. Robardet, et J.-F. Boulicaut (2006a). Co-classification sous contraintes. In *Actes CAp 2006*, Trégastel, France, pp. 155–170. Presses Universitaires de Grenoble.

Pensa, R., C. Robardet, et J.-F. Boulicaut (2006b). Towards constrained co-clustering in ordered 0/1 data sets. In *Proceedings ISMIS 2006*, Volume 4203 of *LNCS*, Bari, Italy, pp. 425–434. Springer.

Ritschard, G. et D. A. Zighed (2003). Simultaneous row and column partitioning : Evaluation of a heuristic. In *Proceedings ISMIS 2003*, Volume 2871 of *LNCS*, Maebashi City, Japan, pp. 468–472. Springer.

Robardet, C. et F. Feschet (2001). Efficient local search in conceptual clustering. In *Proceedings DS 2001*, Volume 2226 of *LNCS*, Washington, USA, pp. 323–335. Springer.

Wagstaff, K., C. Cardie, S. Rogers, et S. Schrödl (2001). Constrained k-means clustering with background knowledge. In *Proceedings ICML 2001*, Williamstown, USA, pp. 577–584.

Summary

In many applications, the expert interpretation of co-clustering is easier than for mono-dimensional clustering. Co-clustering aims at computing a bi-partition or a collection of co-clusters: each co-cluster is a group of objects associated to a group of attributes and these associations can support interpretations. Many constrained clustering algorithms have been proposed to exploit the domain knowledge and improve partition relevancy in the mono-dimensional case, e.g., by using "must-link" and "cannot-link" constraints. Here, we consider constrained co-clustering for these constraints extended to both dimensions of objects and attributes, but also for interval constraints that enforce properties of co-clusters when considering ordered domains. We propose an iterative co-clustering algorithm which exploit user-defined constraints while minimizing the sum-squared residues. We show the added value of our approach in applications in transcriptomics.