# Sélection de contraintes pour la classification topologique semi-supervisée

Kais Allab, Khalid Benabdeslem

# Sélection de contraintes pour la classification topologique semi-supervisée

Kais Allab et Khalid Benabdeslem

Université de Lyon, F69622-Lyon, France
Université de Lyon1 - Villeurbanne
allab.kais@gmail.com, kbenabde@univ-lyon1.fr

**Résumé** : Dans ce papier, nous proposons d'adapter la version batch de l'algorithme de Kohonen basée sur les cartes auto-organisatrices (SOM) aux informations éventuellement portées par les données étiquetées et imposées dans la classification des données non étiquetées (sous formes de contraintes). Il s'agit de la classification sous contraintes avec SOM dans un paradigme bien déterministe. Dans ce contexte nous adaptons la classification topologique à un ensemble de contraintes ; le faite qu'elles soit informatives et cohérentes entre elles permet de mesurer leur utilité dans le processus d'apprentissage semi-supervisé. Ces mesures représentent un moyen efficace pour sélectionner les contraintes les plus utiles pour l'algorithme proposé. Les expériences seront appliquées sur plusieurs bases de données pour valider notre approche en comparaison avec d'autres méthodes de classification sous contraintes.

**Mots-clés** : SOM, classification sous contraintes, sélection des contraintes.

## 1. Introduction

Les algorithmes traditionnels de classification ont seulement accès aux variables qui décrivent chaque donnée mais ils ne traitent jamais d'autre informations éventuellement données. La prise en compte de ces connaissances dans un processus de classification, si elles existent, représente un nouveau champ d'étude dans l'apprentissage automatique qui est la classification sous contraintes (Basu *et al.*, 2008). Cette approche semi-supervisée permet d'améliorer les performances de plusieurs jeux de données UCI (Frank & Asuncion, 2010) et celles des applications issus du monde réel, comme l'identification de personnes via des caméras de surveillance (Bar-Hillel *et al.*, 2005), le raffinement des cartes GPS (Wagstaff *et al.*, 2001) et la détection de paysage dans les données hyper-spectrales (Lu & Leen, 2005).

En outre, les dix dernières années ont vu beaucoup de travaux sur l'intégration de contraintes dans les méthodes de classification. Le premier travail a proposé une version modifiée de COBWEB qui mis en application les contraintes (Wagstaff & Cardie, 2000). Il a été suivi par une version améliorée l'algorithme k-means qui pourrait satisfaire des contraintes, appelée COP-kmeans (Wagstaff *et al.*, 2001). De plus, dans (Davidson & Ravi, 2005a), une exploration de l'utilisation des contraintes a été appliquée avec la classification hiérarchique. Dans (Elghazel *et al.*, 2007) les auteurs ont proposé un nouveau algorithme de classification sous contraintes basé sur la b-coloration de graphes appelé COPb-coloring où ils ont montré des améliorations de la qualité et la complexité informatique de la classification en utilisant des contraintes. Théoriquement, la classification sous contraintes souffre d'un problème de faisabilité (Davidson & Ravi, 2007, 2005b) pour trouver facilement une classification qui satisfait toutes les contraintes.

Les contraintes peuvent être générées à partir des connaissances préalables sur les données (Davidson *et al.*, 2006) ou d'un sous-ensemble de données étiquetées (Bilenko *et al.*, 2004). Dans la pratique, il a été illustré que les contraintes peuvent améliorer les résultats des algorithmes de classification. Cependant, il a été observé que les contraintes peuvent avoir des conséquences négatives même quand elles sont produites directement des étiquettes de données qui sont été utilisées pour l'évaluation. Donc ce comportement n'est pas causé par le bruit ou des erreurs dans les contraintes, mais plutôt, c'est un résultat de l'interaction entre le jeu de contraintes et l'algorithme utilisé (Davidson *et al.*, 2006).

Dans ce papier, nous proposons d'intégrer des contraintes dans une classification topologique auto-organisatrice (Kohonen, 2001) qui représente un outil prometteur pour l'analyse de données à grande dimension car il fournit une réduction de données substantielle qui peut être utilisée pour visualiser et explorer les propriétés de données. Notre première contribution est basée sur un modèle déterministe, où les contraintes sont intégrées et identifiées dans les nœuds des réseaux de neurones correspondants. La deuxième contribution concerne l'étude des deux mesures importantes, l'informativité et la cohérence, qui capturent les propriétés d'un jeu de contraintes. Ces mesures fournissent la perspicacité de l'effet d'un jeu de contraintes donné pour un algorithme de classification sous contraintes spécifié. Ils seront utilisés pour choisir les contraintes appropriées pour le processus classification.

## 2. Intégration de contraintes dans SOM

Les cartes auto-organisatrices SOM (topologiques ou dites de Kohonen) sont utilisées dans plusieurs domaines d'application pour leur rôle polyvalent (quantification, classification, codage et visualisation) sur les données multi-dimensionnelles. En effet, outre leur capacité à produire des données similaires au moyen de prototypes comme en quantification vectorielle et/ou en classification, elles autorisent la conservation de la topologie, d'où sa capacité à reproduire des représentations ordonnées, qu'on appelle prototypes, ou vecteurs référents, sur un espace de faible dimension qu'on appelle carte. Le processus d'apprentissage de SOM ressemble à k-means. La distinction la plus importante est qu'en plus du neurone gagnant, ses voisins sur la carte sont aussi mis à jour. Le résultat final est que ces neurones voisins sur le réseau correspondent à des régions voisines dans l'espace d'entrée.

### 2.1. Contraintes

L'intuition qui est derrière la classification sous contraintes, est que les connaissances portées par les données étiquetées peuvent guider un processus de classification non supervisée à une partition de données meilleure que celle obtenue avec des données non-étiquetées. Les contraintes fournissent des indications sur la partition souhaitée et mettent en œuvre ces indications dans des algorithmes de classification afin d'augmenter leur performance (Davidson & Ravi, 2005b). Soit $X = x_1, ..., x_n$ le jeu des observations qui doivent être regroupées dans $K$ classes, qu'on note par $u_1, ..., u_K$. Pour chaque paire d'observations $x_i, x_j$ dans $X$, on note la distance entre eux par $d(x_i, x_j)$. Dans ce papier, nous proposons d'adapter la classification topologique à l'intégration de contraintes de type : (1) **Must-Link** (***ML***) : qui force deux observations $x_i$ et $x_j$ a être dans la même classe. (2) **Cannot-Link** (***CL***) : qui force deux observations $x_i$ et $x_j$ a être dans deux classes différentes.

### 2.2. L'algorithme topologique sous contraintes

L'algorithme de SOM est proposé en deux versions : stochastique ou batch. Nous optons dans ce papier pour la deuxième version pour son déterminisme et sa rapidité bien qu'elle est adaptée à notre proposition d'optimiser la fonction objective pour une éventuelle intégration de contraintes. La version batch de SOM est un algorithme itératif dans lequel le jeu de données est entièrement présentées à la carte avant que l'adaptation ne soit faite.

Formellement, on définie la fonction d'affectation $f$ de $\mathcal{R}^D$ (l'espace d'entrée) à $C$ (l'espace de sortie), qui associe chaque élément $x_i$ de $\mathcal{R}^D$ au neurone dont le vecteur référence " est le plus proche" de $x_i$. Cette fonction induis une partition $P = P_c; c = 1...K$ de l'ensemble des observations tel que chaque classe $P_c$ est définie par $P_c = \{x_i \in X; f(x_i) = c\}$. La qualité de la partition $(P_c)_{c\in C}$ et ses vecteurs prototypes associés $(w_c)_{c\in C}$ est donnée par la fonction d'énergie suivante (Cheng, 1997) :

$$E^T((P_c)_{c\in C}, (w_c)_{c\in C}) = \sum_{x_i\in X}\sum_{c\in C} h^T(\delta(f(x_i), c)) \|w_c - x_i\|^2 \qquad (1)$$

Tel que $f$ représente la fonction d'affectation : $f(x_i) = c$ si $x_i \in P_c$, $h^T(.)$ [1] est le noyau de voisinage autour du neurone gagnant $c$ et $T$ [2] représente le rayon de voisinage dans la carte. En générale, $f(x_i) = r$ tant que $r \in C$, donc (1) peut être écrite :

$$E^T((P_c)_{c\in C}, (w_c)_{c\in C}) = \sum_{r\in C}\sum_{x_i\in C_r}\sum_{c\in C} h^T(\delta(r, c)) \|w_c - x_i\|^2 \qquad (2)$$

avec $h^T(0) = 1$, $E^T$ peut être décomposée en deux termes :

$$E_1^T = \sum_{r\in C}\sum_{x_i\in C_r} \|w_r - x_i\|^2 \qquad (3)$$

$$E_2^T = \sum_{r\in C}\sum_{c\neq r}\sum_{x_i\in C_c} h^T(\delta(r, c)) \|w_r - x_i\|^2 \qquad (4)$$

$E_1^T$ correspond à l'altération utilisée dans les algorithmes de classification basés sur le partitionnement comme k-means. $E_2^T$ est spécifique à SOM.

Notre première contribution consiste à adapter SOM à des contraintes $ML$ et $CL$ par l'optimisation de l'equation (1). Pour cela, nous utilisons la version proposée par Heskes et Kappen (Heskes & Kappen, 1993) qui procède en deux étapes : l'étape d'affectation pour calculer $f$ et l'étape d'adaptation pour calculer $w_c$. L'étape d'affectation consiste à minimiser $E^T$ avec $w_c$ fixé et l'étape d'adaptation minimise la même fonction objective mais avec les prototypes fixés. Bien que les deux optimisations soient exécutées correctement, nous ne pouvons pas garantir que l'énergie est totalement réduite au minimum par cet algorithme.

[1] Dans la pratique, $h^T(x) = e^{-\frac{x^2}{T^2}}$.

[2] Fixé ou décroît d'une valeur initiale $T_{max}$ à une valeur finale $T_{min}$.

Cependant, si $T$ est fixe, l'algorithme converge vers un état stable après un nombre d'itérations fini (Cheng, 1997). La formulation de (1) montre que l'énergie est construite comme la somme sur toutes les observations d'une mesure d'adéquation de $R^D \times C$ dans $R^+$ définie par :

$$\gamma^T(x, r) = \sum_{c \in C} h^T(\delta(r, c)) \, \|w_c - x\|^2 \tag{5}$$

$$\text{Qui donne :} \quad E^T((P_c)_{c \in C}, (w_c)_{c \in C}) = \sum_{x_i \in X} \gamma^T(x_i, f(x_i)) \tag{6}$$

Pour optimiser $E^T$ soumise à des contraintes $ML$ et $CL$ en fixant les prototypes, il suffit de minimiser indépendamment chaque somme avec l'incorporation d'une nouvelle procédure $V(.)$ pour contrôler la violation des contraintes pendant le processus d'affectation :

$$f(x_i) = \begin{cases} c^* = arg_{r \in C} Min \, \gamma^T \, (x_i, r) \text{ Tant que } V(x_i, c^*) \text{ est Faux} \\ \textbf{Ø} : \text{ Sinon} \end{cases} \tag{7}$$

$$V(x_i, c^*) = \begin{cases} \textbf{Vrai} : \text{Si } \forall x_j \in X / ML(x_i, x_j), \; f(x_j) \neq c^* \text{ Ou} \\ \forall x_j \in X / CL(x_i, x_j), \; (f(x_j) = c^*) \vee (f(x_j) \in N(c^*)) \\ \\ \textbf{Faux} : \text{Sinon} \end{cases} \tag{8}$$

Tel que $N(c^*)$ représente l'ensemble des voisins de $c^*$ dans la carte.

De même, quand les prototypes sont fixés, l'optimisation de $E^T$ peut être donnée par la minimisation de l'énergie associée à chaque neurone soumis à des contraintes $ML$ et $CL$ :

$$E_c^T(w) = \sum_{x_i \in X} h^T(\delta(f(x_i), c)) \, \|w - x_i\|^2 \tag{9}$$

Nous pouvons facilement résoudre ce problème en donnant une solution unique comme moyen pondéré des observations avec la définition d'une nouvelle fonction de contrôle $g(.)$ :

$$w_c = \frac{\sum_{x_i \in X} h^T(\delta(f(x_i), c)) \, x_i \, g(x_i)}{\sum_{x_i \in X} h^T(\delta(f(x_i)), c) \, g(x_i)} \tag{10}$$

$$\text{telque} : g(x_i) = \begin{cases} \textbf{0} : \text{ Si } \exists x_j \in X / (f(x_j) = c) \wedge CL(x_i, x_j) \\ \textbf{1} : \text{ Sinon} \end{cases} \tag{11}$$

Les équations (7) et (10) représentent la version modifiée de la version batch de l'algorithm SOM (qu'on va appeler CrSOM) avec notre modifications dans les deux étapes, l'étape d'affectation et l'étape d'adaptation.

L'algorithme prend dans un jeu de données ($X$) un ensemble de contraintes must-link ($\Omega_{ML}$) et un ensemble de contraintes cannot-link ($\Omega_{CL}$) et retourne une partition des observations dans $X$ qui satisfait tout les contraintes spécifiées. La modification majeur est qu'il faut s'assurer qu'aucune des contraintes spécifiées n'est violée pendant l'affectation des observations aux classes. Nous tentons d'affecter chaque point $x_i$ au neurone le plus proche $c$ dans la carte sans qu'une contrainte ne soit violée. Si nous ne trouvons pas un neurone qui peut légalement (sans violation de contraintes) accueillir $x_i$, la partition vide (Ø) est retournée.

Notons qu'avec l'utilisation de l'équation(11), chaque vecteur référence $w_c$ n'est mis à jour qu'avec les observations $x_i \in X$ qui n'ont pas une contrainte $CL$ avec un élément $x_j$ appartenant à $c$. La version batch de l'algorithme SOM adaptée au contraintes CrSom est décrite dans l'algorithme 1.

**Algorithm 1** CrSOM

**Entrée :** Données $X = \{x_i\}$, Contraintes : $\Omega_{ML}$, $\Omega_{CL}$
Initialiser $(w_c)_{c \in C}$
**for** $it = 1$ **à** $it_{max}$ **do**
  **for** $i = 1$ **à** $|X|$ **do**
    calculer $f(x_i)$ utilisant Eq.(7)
  **end for**
  **for** $c = 1$ **à** $|C|$ **do**
    calculer $w_c$ utilisant Eq.(10)
  **end for**
**end for**

## 3. Sélection de contraintes

Partant de la supposition que les contraintes vont contribuer plus ou moins à l'amélioration de la classification, il a été trouvé que quelques jeux de contraintes diminuent les performances de la classification. Pour Cela, Davidson *et al.* (2006) ont défini deux mesures importantes, l'informativité et la cohérence, qui capturent les propriétés d'un jeux de contraintes, et peuvent expliquer ce constat.

### 3.1. L'informativité

Cette mesure est basée sur le nombre de contraintes que l'algorithme de classification ne puisse pas prévoir en utilisant sa prédiction par défaut. Étant donné un jeu de contraintes $\Omega$ et un algorithme $A$, nous générons la partition $P_A$ par l'exécution de $A$ sur les données sans aucune contrainte. Nous calculons alors la fraction de contraintes dans $\Omega$ qui sont insatisfaites par $P_A$ :

$$I_A(\Omega) = \frac{1}{|\Omega|} \sum_{\alpha \in \Omega} unsat(\alpha, P_A) \tag{12}$$

Tel que $unsat(\alpha, P)$ égale à 1 si $P$ ne satisfait pas $\alpha$ et 0 sinon.

### 3.2. La cohérence

En supposant qu'une contrainte $ML(x, y)$ (ou $CL(x, y)$) impose une force attractive (or répulsive) dans l'espace des variables au long d'une ligne formée par $(x, y)$ dans le voisinage de $x$ et $y$. Deux contraintes, une $ML$ $(m)$ et une $CL$ $(c)$, sont incohérentes si elles exercent des forces contradictoires dans le même voisinage. Soit $\overrightarrow{m}$ et $\overrightarrow{c}$ les vecteurs reliant les points concernées par les contraintes $m$ et $c$ respectivement. La cohérence d'un jeu de contraintes $\Omega$ est définie comme la fraction des paires de contraintes que leurs projections ne sont pas chevauchées :

$$Coh_d(\Omega) = \frac{\sum_{m \in \Omega_{ML}, c \in \Omega_{CL}} \delta(over_c m = 0 \wedge over_m c = 0)}{|\Omega_{ML}|\,|\Omega_{CL}|} \tag{13}$$

Tel que $over_c m$ représente la distance entre les deux points résultats de la projection des deux points concernées par $m$ sur $\overrightarrow{c}$. $\delta$ est le nombre des projections chevauchées. Voir plus de détails dans (Davidson *et al.*, 2006).

De l'équation **??**, nous pouvons facilement définir une mesure de cohérence pour chaque contrainte comme suit :

$$Coh_d(m) = \frac{\sum_{c \in \Omega_{CL}} \delta(over_c m = 0)}{|\Omega_{CL}|} \tag{14}$$

$$Coh_d(c) = \frac{\sum_{m \in \Omega_{ML}} \delta(over_m c = 0)}{|\Omega_{ML}|} \tag{15}$$

### 3.3. Sélection stricte (Hard)

Pour qu'elle soit sélectionnée, une contrainte $\alpha_i$ doit être informative et totalement cohérente, Cette démarche stricte de la sélection de contraintes peut être décrite par l'algorithme 2.

**Algorithm 2** Sélection stricte (Hard)

**Entrée :** jeu de contraintes $\Omega = \{\alpha_i\}$, $\Omega_s = \emptyset$
**for** $i = 1$ **à** $|\Omega|$ **do**
  **if** $unsat(\alpha_i, P_{SOM}) = 1$ **then**
    **if** $Coh_d(\alpha_i) = 1$ **then**
      $\Omega_s = \Omega_s + \{\alpha_i\}$
    **end if**
  **end if**
**end for**

### 3.4. Sélection souple (Soft)

La sélection stricte est très sélective et peut réduire considérablement le nombre de contraintes utilisées $\Omega_s$. Elle pénalise les contraintes ayant une forte cohérence. Pour cela, nous proposons une méthode moins stricte (Soft) pour sélectionner les contraintes, décrite par l'algorithme 3.

**Algorithm 3** Sélection souple (Soft)

**Entrée :**Jeu de contraintes $\Omega = \{\alpha_i\}$, $\Omega_s = \emptyset$, $\Omega'_s = \emptyset$
**for** $i = 1$ **à** $|\Omega|$ **do**
  **if** $unsat(\alpha_i, P_{SOM}) = 1$ **then**
    $\Omega_s = \Omega_s + \{\alpha_i\}$
  **end if**
**end for**
**for** $i = 1$ **à** $|\Omega_s|$ **do**
  **if** $Coh_d(\alpha_i) > Coh_d(\Omega_s)$ **then**
    $\Omega'_s = \Omega'_s + \{\alpha_i\}$
  **end if**
**end for**
Trier décroissement $\Omega'_s$ selon leurs cohérence.

TAB. 1: Bases de données utilisées

| Base de données | $N$ | $D$ | $\#l$ | Dimensions de la carte |
|---|---|---|---|---|
| Glass | 214 | 9 | 6 | $11\times 7$ |
| Ionosphere | 352 | 34 | 2 | $13 \times 7$ |
| Iris | 150 | 4 | 3 | $16 \times 4$ |
| Leukemia | 72 | 1762 | 2 | $9 \times 5$ |

### 3.5. L'approche proposée

Notre approche nommée S3OM (pour Semi-Supervised SOM), décrite dans l'algorithme 4 réalise une classification topologique sous contraintes. Pour la construction de la carte, une heuristique proposée par Kohonen (Kohonen, 2001) basée sur l'Analyse en Composantes Principales (ACP) calcule automatiquement les dimensions des cartes (Table 1) et initialise les vecteurs références associés. L'algorithme CrSOM est utilisé pour l'apprentissage de la carte en respectant un jeu de contraintes sélectionné strictement ou souplement. Une fois la carte apprise, elle est classée par une Classification Ascendante Hiérarchique (CAH) pour optimiser le nombre de classes (en regroupant les neurones) (Dash & Liu, 1997).

En général, l'indice interne Davies Bouldin (Kalyani & Sushmita, 2003) est utilisé pour trouver la meilleure troncature du déndrogramme. Dans notre approche, nous avons adapté l'algorithme (CAH) de tel façon qu'il arrête la classification dés qu'il viole une contrainte. Nous cherchons le nombre minimal de classes qui satisfait toutes les contraintes.

**Algorithm 4** S3OM

1: **Entrée :** Données $X = \{x_i\}$,
2: Initialiser la carte utilisant l'heuristique de Kohonen ($r$ nœuds) ;
3: Sélection de contraintes : $\Omega_{ML}$ et $\Omega_{CL}$(Hard ou Soft)
4: $\Gamma$ (partition résultat) = CrSom $(X, \Omega_{ML}, \Omega_{CL})$ ;
5: $\Gamma = CAH(\Gamma, k, \Omega_{ML}, \Omega_{CL})$ : Optimiser la carte, $k$ est le nombre optimal de classes sans violer aucune contrainte ;

TAB. 2: Comparaison des performances (Indice de Rand) des algorithmes de classification sous contraintes.

| Base | CKM | MKM | PKM | MPKM | **S3OM** |
|---|---|---|---|---|---|
| | Inc / Con | Inc / Con | Inc / Con | Inc / Con | Inc / Con |
| Glass | 69.0 **69.4** | 39.5 **56.6** | 43.4 **68.8** | 39.5 **67.8** | 64.3 **68.4** |
| Ionosphere | 58.6 **58.7** | 58.8 **58.9** | 58.8 **58.9** | 58.9 **58.9** | 50.5 **61.3** |
| Iris | 84.7 **87.8** | 88.0 **93.6** | 84.3 **88.3** | 88.0 **91.8** | 91.2 **92.2** |

## 4. Résultats expérimentales

Plusieurs expériences ont été effectuées sur les bases de données cités dans la tableau 1, obtenus du dépôt UCI (Frank & Asuncion, 2010) sauf Leukemia, qu'on peut trouver dans (Golub *et al.*, 1999). Les étiquettes de chaque donnée est utilisée pour générer et évaluer les contraintes mais restent invisibles pour l'algorithme de classification. Ces bases sont choisies pour évaluer les performances de S3OM et des méthodes : COP-KMeans (CKM) (Wagstaff *et al.*, 2001) ; PC-KMeans (PKM) (Bilenko *et al.*, 2004) ; M-KMeans (MKM) (Bilenko *et al.*, 2004) ; MPC-KMeans (MPKM) (Bilenko *et al.*, 2004) et Cop-Bcoloring (CBC) (Elghazel *et al.*, 2007).

### 4.1. Évaluation de l'approche proposée

Pour évaluer S3OM, nous utilisons l'indice de Rand (Rand, 1971). Cet indice mesure la correspondance entre la partition obtenue et la partition correcte. Le Tableau 2 compare les résultats moyennes de 1000 exécutions pour chaque algorithme sans et avec contraintes (25 contraintes aléatoirement tirées). Nous avons ajouter les résultats de notre approche S3OM à ceux déjà montrées dans (Davidson *et al.*, 2006; Wagstaff & Cardie, 2000; Wagstaff *et al.*, 2001; Elghazel *et al.*, 2007). La meilleure performance de chaque combinaison algorithme/Données est en gras.

D'une part, S3OM montre que l'intégration de contraintes dans le modèle SOM résulte une amélioration considérable de la classification. D'autre part, les résultats obtenues par S3OM sont similaires et parfois meilleures que celles obtenues par les autres méthodes. Cependant, S3OM réalise des grandes améliorations par rapport aux autres méthodes. Par exemple, dans le Tableau 2, pour la base Glass, S3OM (68.4%) n'est pas meilleure que CKM

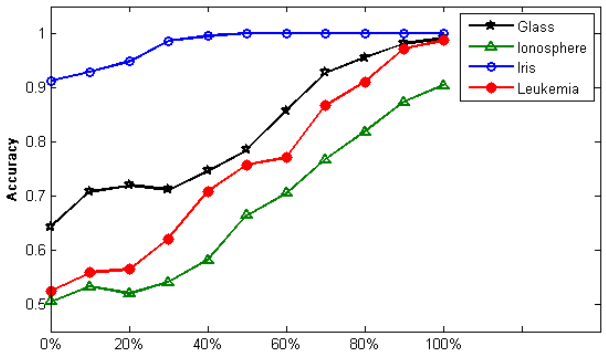


FIG. 1: Performances de S3OM selon le taux de cohérence, avec des contraintes toutes informatives.

(69.4%), mais S3OM réalise une amélioration de 4.1%, tandis que CKM ne réalise que 0.4%. Les propriétés topologiques et les relations de voisinage de SOM ont permet la séparation des points reliées par une contrainte $CL$, en les mettant dans deux neurones (classes) différents et surtout distincts. Par contre dans les autres méthodes, les points sont affectées à la classe la plus proche qui ne viole pas la contrainte.

### 4.2. Résultats de la sélection des contraintes

Dans le tableau 2, en général, les contraintes améliorent les performances de la classification. Cependant, ces contraintes ont été générées aléatoirement, donc quelques unes ne sont pas informatives et d'autres ne sont pas totalement cohérentes. Pour comprendre l'effet des propriétés d'un jeu de contraintes sur la partition obtenue par notre algorithme, on fait les expériences suivantes. Premièrement, nous avons mesuré le Rand en utilisant des jeux de contraintes informatives, mais avec des taux différents de cohérence (calculés par (13)).

La Figure (1) montre que toutes les bases exposent une amélioration importante de Rand quand la cohérence du jeu de contraintes augmente. Cependant, le Rand augmente rapidement quand l'algorithme réalise une bonne performance sans contraintes (Iris). Par contre, avec les autres bases, les contraintes peuvent améliorer le Rand comme elle peuvent parfois causer son diminution surtout lorsque la cohérence est faible.

Deuxièmement, nous appliquons S3OM sur quelque bases en choisissant aléatoirement 100 contraintes. Dans la Figure 2, la courbe rouge est obtenue en utilisant toutes les contraintes, la courbe bleu est obtenue en n'utilisant

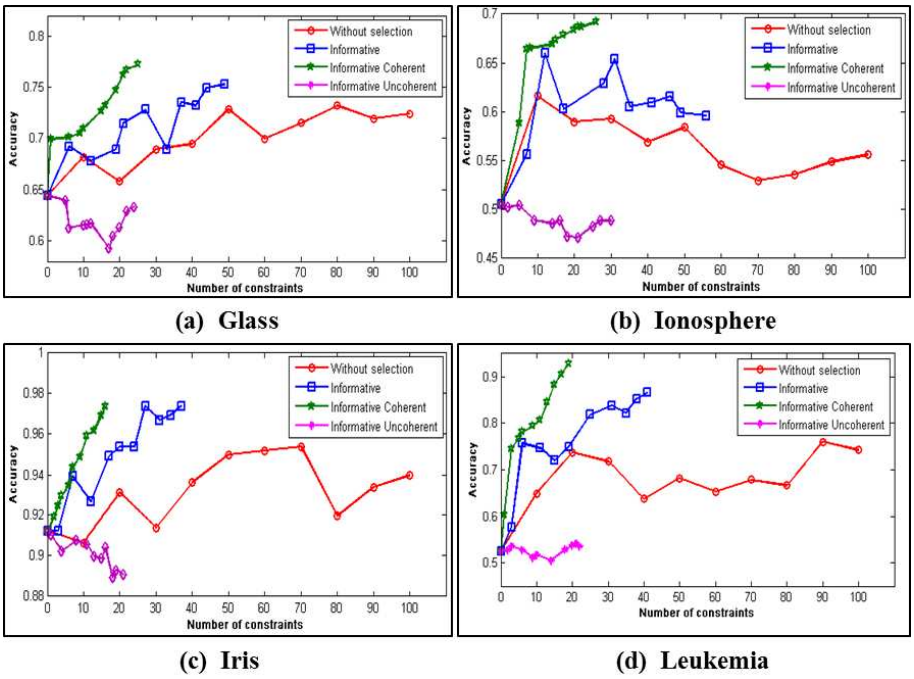


FIG. 2: Performances de S3OM CrSOM selon l'informativité et la cohérence.

que les contraintes informatives, parmi ces dernières, nous n'utilisons que les contraintes cohérentes pour obtenir la courbe verte et que les contraintes incohérentes pour obtenir la courbe rose.

En général, S3OM a le même comportement avec toutes les bases. Prenons Glass pour exemple, S3OM réalise un Rand de 64.3% sans contraintes. Le Rand augmente avec l'incorporation des contraintes arrivant à 73.9% après 100 contraintes. Cependant, la performance est meilleure lorsque on n'intègre que les 53 contraintes informatives (75.3%). Elle est encore meilleur quand on n'utilise que les 25 contraintes informatives et totalement cohérentes (77%), mais mauvaise quand on n'utilise que les 28 contraintes informatives mais incohérentes (64%). Cela signifie que l'intégration des contraintes non informatives et des contraintes incohérentes pourraient être parfois dramatique pour les performances de la classification et parfois inutile.

TAB. 3: Performances de S3OM sur la base Leukemia avec une sélection stricte (Hard) vs une sélection souple (Soft).

| Hard | | | Soft | | |
|---|---|---|---|---|---|
| #ML | #CL | Rand | #ML | #CL | Rand |
| 1 | 4 | 62.0 | 2 | 7 | 55.9 |
| 5 | 5 | 70.9 | 6 | 9 | 66.5 |
| **11** | **4** | **75.7** | **13** | **7** | **77.0** |
| 11 | 9 | 86.7 | 14 | 17 | 84.7 |
| **8** | **17** | **89.6** | **11** | **21** | **91.1** |
| 14 | 16 | 97.2 | 16 | 21 | 88.6 |

### 4.3. Résultats de la sélection pour la base Leukemia

La base de données Leukemia est constituée d'un ensemble de 72 individus, correspondant à deux types de Leucémie nommés *ALL (Acute Lymphocytic Leukemia)* et *AML (Acute Myelogenous Leukemia)*, avec 47 ALL et 25 AML. Le jeu de données contient initialement 7929 variables. Dans (Golub *et al.*, 1999) il a été proposé de supprimer les variables de contrôle "affymetrix" et les variables ayant au moins une valeur inférieur à 20, pour obtenir finalement 1762 variables. Nous testons la sélection stricte (Hard) et la sélection souple (Soft) avec le jeu de données obtenu ($72 \times 1762$).

Notre algorithme S3OM sans contraintes réalise un Rand de 52.5%. A partir des 72 individus étiquetés(AML et ALL), nous générons 2556 contraintes (1381 ML et 1175 CL). En utilisant l'informativité, ce nombre est réduit à 1215 (540 ML et 675 CL). Parmi eux, nous ne trouvons que 3 contraintes totalement cohérentes. La sélection stricte réalise un Rand de 54.3%, qui représente une augmentation de 2%. Mais, est ce qu'on peut améliorer ce score en ajoutant à ces 3 contraintes d'autres contraintes qui ne sont pas totalement cohérentes, mais ayant des fortes valeurs de cohérence ?

Premièrement, pour savoir est ce que la qualité de la classification est meilleure avec un grand nombre de contraintes sélectionnées souplement ($softly$) ou avec un petit nombre de contraintes sélectionnées strictement ($hardly$), nous mesurons le Rand pour 1000 exécutions avec des différents jeux de contraintes $ML$ et $CL$, d'une taille variant entre 5 et 37. Le tableau 3 compare les résultats d'une sélection stricte et celles d'une sélection souple.

Dans ce tableau, on peut remarquer en général, que la sélection stricte est

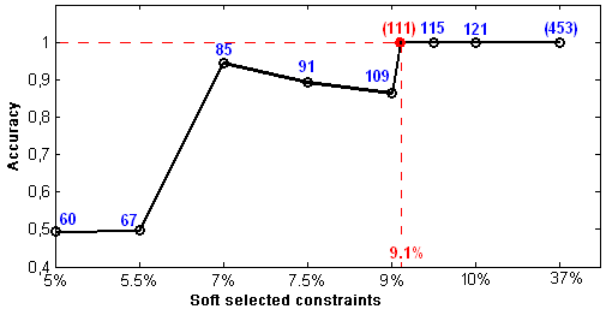


FIG. 3: Les contraintes informatives triées selon leurs cohérence.

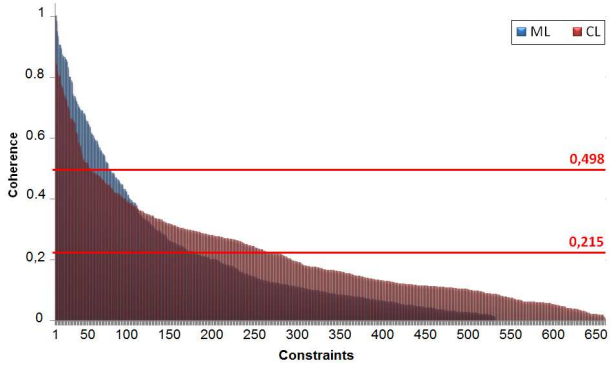


FIG. 4: Performances de S3OM en utilisant la sélection souple (Soft).

plus efficace que la sélection souple, ce qui est logique à cause de l'utilisation que des contraintes totalement cohérentes. Cependant, les $3^{me}$ et $5^{me}$ lignes du tableau montrent le contraire, surtout a cause de la grande différence entre le nombre des ML et le nombre des CL.

Deuxièmement, nous trions les 1215 contraintes informatives selon leurs cohérence. En se basant sur la cohérence globale de ce jeu de contraintes qui est 0.215, on peut souplement (Softly) sélectionner les 453 (176 ML et 277 CL) contraintes ayant une cohérence supérieur à 0.215 (les 37% premières du jeu trié), se qui correspond à la première troncature dans la figure 3. Sachant que S3OM réalise un Rand de 100% en utilisant ce jeu de contraintes, nous avons étudier le comportement de S3OM pour des pourcentages de contraintes allant de 5% à 37% (Figure 4) pour trouver le nombre minimal des contraintes (meilleure troncature) qui réalise le Rand 100%. Les différents chiffres dans la Figure 4 représentent le nombre de contraintes cor-

respondant aux différents pourcentages étudiés. Par conséquence, la meilleure résultat obtenue est que S3OM réalise un Rand de 100% avec un nombre minimal de contraintes égalant 111 (66 ML et 45 CL) et qui représente les 9.1% premières contraintes, sachant que la cohérence du $111^{me}$ contrainte est de 0.498 et qui correspond à la deuxième troncature dans la Figure 3. Ainsi, une contrainte est sélectionnée si sa cohérence est supérieur ou égale 0.498.

## 5. Conclusion

Dans ce travail, nous avons trois contributions. La première, un nouveaux algorithme appelé CrSom pour la classification topologique sous contraintes prenant en compte des contraintes instance-niveau dans la fonction objective de la version batch de l'algorirhme SOM. La deuxième, l'études des propriétés des jeux de contraintes, l'informativité et la cohérence, permettant de nous fournir une perspicacité sur les performances de notre algorithme soumis à ce jeu de contraintes (amélioration ou diminution). Ces mesures sont utilisées pour sélectionner les contraintes les plus utiles pour la classification soit avec la sélection stricte (Hard) ou avec la sélection souple (Soft). Enfin, un autre algorithme appelé S3OM pour la classification semi-supervisée permettant de sélectionner les contraintes appropriées pour une classification topologique réalisée par l'algorithme CrSom, et d'optimiser la partition résultat afin d'améliorer sa qualité.

## Références

BAR-HILLEL A., HERTZ T., SHENTAL N. & WEINSHALL D. (2005). Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, **6**, 937–965.

BASU S., DAVIDSON I. & WAGSTAFF K. (2008). *Constrained clustering : Advances in algorithms, theory and applications*. Chapman and Hall/CRC Data Mining and Knowledge Discovery Series.

BILENKO M., BASU S. & MOONEY R. (2004). Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty first international conference on machine learning*, p. 11–18.

CHENG Y. (1997). Convergence and ordering of kohonen's batch map. *Neural Computation*, **9(8)**, 1667–1676.

DASH M. & LIU H. (1997). Feature selection for classification. *Intelligent Data Analysis*, **1(3)**, 131–156.

DAVIDSON I. & RAVI S. (2005a). Agglomerative hierarchical clustering with constraints :theorical and empirical results. In *Proceedings of the ECML/PKDD*, p. 59–70.

DAVIDSON I. & RAVI S. (2005b). Clustering with constraints : feasibility issues and the k-means algorithm. In *Proceedings of the SIAM international conference on data mining*, p. 138–149.

DAVIDSON I. & RAVI S. (2007). The complexity of non-hierarchical clustering with instance and cluster level constraints. *Data mining and knowledge discovery*, **61**, 14–25.

DAVIDSON I., WAGSTAFF K. & BASU S. (2006). Measuring constraint-set utility for partitional clustering algorithms. In *Proceedings of ECML/PKDD*.

ELGHAZEL H., BENABDELSLEM K. & DUSSAUCHOY A. (2007). Constrained graph b-coloring based clustering approach. In *Proceedings of DaWaK, LNCS 4654*, p. 262–271.

FRANK A. & ASUNCION A. (2010). *UCI machine learning repository*. Rapport interne, University of California.

GOLUB T., SLONIM D., TAMAYO P., HUARD C., GAASENBEEK M., MESIROV J., COLLER H., LOH M., DOWNING L., CALIGIURI M., BLOOMFIELD C. & LANDER E. (1999). Molecular classification of cancer : Class discovery and class prediction by gene expression monitoring. *Science 15*, **286(5439)**, 531–537.

HESKES T. & KAPPEN B. (1993). Error potentials for self-organization. In *Proceedings of IEEE International Conference on Neural Networks*, p. 1219–1223.

KALYANI M. & SUSHMITA M. (2003). Clustering and its validation in a symbolic framework. *Pattern Recognition Letters*, **24(14)**, 2367–2376.

KOHONEN T. (2001). *Self organizing Map*. Berlin : Springer Verlag.

LU Z. & LEEN T. (2005). Semi-supervised learning with penalized probabilistic clustering. In *Advances in Neural information Processing Systems 17*.

RAND W. (1971). Objective criteria for the evaluation of clustering method. *Journal of the American Statistical Association*, **66**, 846–850.

WAGSTAFF K. & CARDIE C. (2000). Clustering with instance level constraints. In *Proceedings of the seventeenth international conference on machine learning*, p. 1103–1110.

WAGSTAFF K., CARDIE C., ROGERS S. & SCHROEDL S. (2001). Clustering with instance level constraints. In *Proceedings of the seventeenth international conference on machine learning*, p. 1103–1110.