# Intégration interactive de contraintes pour la réduction de dimensions et la visualisation

Lionel Martin*, Matthieu Exbrayat*, Guillaume Cleuziou*, Frédéric Moal*

* Laboratoire d'Informatique Fondamentale d'Orléans
Université d'Orléans, BP 6759 F-45067 Orléans Cedex 2
Prenom.Nom@univ-orleans.fr
http://www.univ-orleans.fr/lifo

**Résumé.** Il existe aujourd'hui de nombreuses méthodes de réduction de dimensions, que ce soit dans un cadre supervisé ou non supervisé. L'un des intérêts de ces méthodes est de pouvoir visualiser les données, avec pour objectif que les objets qui apparaissent "visuellement" proches soient similaires, dans un sens qui correspond aux connaissances d'un expert du domaine ou qui soit conforme aux informations de supervision. Nous nous plaçons ici dans un contexte semi-supervisé où des connaissances sont ajoutées de façon interactive : ces informations seront apportées sous forme de contraintes exprimant les écarts entre la représentation observée et les connaissances d'un expert. Nous pourrons par exemple spécifier que deux objets proches dans l'espace d'observation sont en fait peu similaires, ou inversement. La méthode utilisée ici dérive de l'analyse en composantes principales (ACP), à laquelle nous proposons d'intégrer deux types de contraintes. Nous présentons une méthode de résolution qui a été implémentée dans un logiciel offrant une représentation 3D des données et grâce auquel l'utilisateur peut ajouter des contraintes de manière interactive, puis visualiser les modifications induites par ces contraintes. Deux types d'expérimentation sont présentés, reposant respectivement sur un jeu de données synthétique et sur des jeux standards : ces tests montrent qu'une représentation de bonne qualité peut être obtenue avec un nombre limité de contraintes ajoutées.

## 1 Introduction

Les mécanismes d'apprentissage automatique consistent en général à établir ce qui est important, vis-à-vis d'un objectif donné, au sein d'une masse d'information disponible. C'est notamment le cas dans le cadre de la classification automatique, qu'elle soit supervisée ou non. Les objets que l'on souhaite classer sont fréquemment définis par un ensemble de descripteurs, potentiellement nombreux, qui peuvent s'avérer redondants, bruités, ou tout simplement sans objet vis-à-vis de la classification recherchée. Par ailleurs, l'abondance de descripteurs peut rendre peu lisible le mécanisme de classification.

Ce travail est soutenu par l'ANR "Masse de Données et COnnaissances" Graphem (ANR-07-MDCO-006-05)

Deux approches sont possibles pour réduire le nombre de descripteurs. La première consiste à sélectionner les descripteurs les plus significatifs (sélection d'attributs, ou *feature selection*, on pourra notamment consulter la synthèse de Guyon et Elisseeff (2003)) et la seconde à produire un ensemble restreint de descripteurs synthétiques, reflétant au mieux la répartition des objets dans l'espace d'origine (réduction de dimension, ou *dimensionality reduction*). Notons que la réduction de dimension est parfois considérée (e.g. dans Blum et Langley (1997)) comme appartenant à une catégorie de méthodes de sélection d'attributs (approche par filtre). Nous nous intéresserons ici aux méthodes de réduction de dimension, lesquelles permettent fréquemment d'obtenir, au prix d'une perte d'information raisonnable, une information synthétique limitée à quelques descripteurs, ce qui conduit à une réduction des coûts de calcul en classification automatique (recherche de voisins, estimation de mélange de lois,...), et autorise également une projection des objets dans un espace visualisable à deux ou trois dimensions.

De nombreuses approches ont été développées au cours des dernières décennies, l'une des plus connues étant l'analyse en composante principale (ACP). Rappelons qu'il s'agit d'une approche non supervisée sélectionnant les dimensions orthogonales deux à deux, issues de combinaisons linéaires des attributs d'origine, qui préservent au mieux la dispersion des objets. Elle est résolue par la recherche des plus grandes valeurs propres de la matrice de variance-covariance des objets dans l'espace d'origine. Dans un cadre supervisé, la notion de classe pourra conduire à rechercher un espace de projection qui renforce la compacité des classes, tout en les distinguant nettement les unes des autres. C'est notamment le cas de l'Analyse Discriminante Linéaire (ou ALD, Fisher (1936)), qui maximise le critère de Fisher (rapport entre variance inter-classes et variance intra-classes).

Si l'on se concentre sur la seule projection des objets, les travaux les plus récents portent sur des techniques d'apprentissage (semi-) supervisé, mettant en jeu des critères locaux ou globaux sur la proximité d'objets appartenant ou non à la même classe. L'analyse en composantes pertinentes (*Relevant Component Analysis*) proposée par Bar-Hillel et al. (2005) est une approche semi-supervisée. Le calcul de la matrice de projection repose sur un échantillon d'objets de différentes classes appelé *chunklets*.

La classification par maximisation de la marge entre plus proches voisins (*Large Margin Nearest Neighbors*, ou LMNN) proposée par Weinberger et al. (2005); Weinberger et Saul (2008) se concentre pour sa part sur le voisinage. Pour chaque objet, on répertorie les objets voisins (dans un rayon donné), puis on pose un ensemble de contraintes exprimant le fait que les objets proches de classes différentes doivent être plus éloignés que les objets proches de même classe. Cet éloignement se traduit par une marge minimum entre les deux catégories d'objets et est pris en compte dans un problème de programmation semi-défini (SDP).

Ces deux approches présentent des limitations. L'approche LMNN est adaptée à un contexte supervisé et introduit des contraintes pour l'ensemble des objets : pour chaque objet, un ensemble de contraintes spécifiant que l'objet doit être bien classé après projection est intégré. L'approche RCA utilise la seule notion de paires d'objets de même classe, alors que d'autres types de contraintes, reposant par exemple sur l'éloignement ou le rapprochement d'objets, élargiraient et assoupliraient les jeux de contraintes possibles. Ces approches, ainsi que la plupart des approches existantes, supposent en outre que toutes les contraintes soient fixées avant la phase de réduction de dimension. Il serait donc appréciable de cumuler la simplicité de l'approche RCA avec l'apport des contraintes de voisinage de l'approche LMNN, tout en proposant un mécanisme intuitif et itératif d'intégration de connaissances du domaine.

Sur le plan de la visualisation, de nombreux outils proposent une projection en 2 ou 3 dimensions des objets reposant sur diverses techniques. Les outils permettant d'agir directement sur les projections et d'observer les effets de ces actions sont toutefois plus rares. A ce titre, le *projection pursuit guided tour* proposé par Cook et al. (1995), permet de se déplacer dans l'espace des projections, tout en offrant des transitions douces entre les projections successives (lesquelles sont proches). Le parcours de l'espace des projections est guidé par l'optimisation d'un critère choisi par l'utilisateur et alterne des phases de réduction de gradient et de saut aléatoire. Un tel outil ne permet toutefois pas de définir des contraintes locales, et encore moins de les définir interactivement, au cours du processus d'exploration.

Dans cet article nous nous plaçons dans un cadre semi-supervisé, dans lequel les objets ne sont pas associés à des étiquettes de classe. Nous proposons d'utiliser deux types de contraintes de projection, à la fois naturelles et simples d'utilisation :

- positionnement de deux objets : consiste à borner (supérieurement ou inférieurement) la distance entre 2 objets,
- positionnement de deux objets $(b, c)$ par rapport à un troisième$(a)$ : consiste à borner le rapport distance($a$, $c$)/distance($a$, $b$).

L'objectif est d'intégrer ces contraintes de manière interactive : l'utilisateur observe une représentation (3D) des données (obtenue par projection linéaire), puis peut spécifier des corrections attendues en ajoutant des contraintes, par exemple dans le cas où 2 objets "semblables pour l'utilisateur" sont éloignés dans la représentation. Il est ainsi possible d'obtenir une nouvelle représentation à laquelle d'autres contraintes pourront être intégrées de manière itérative. Il s'agit donc d'un contexte semi-supervisé dans lequel la perte d'information induite par la projection est contrôlée par l'utilisateur grâce à l'ajout de contraintes, qui ont pour but de corriger les positions relatives de certains objets.

L'article est organisé comme suit : dans la section 2, nous définissons plus formellement les deux types de contraintes proposés. En section 3, nous proposons un mécanisme de résolution de contraintes dures reposant sur l'algorithme d'Uzawa (Arrow et al. (1958)). En section 4 nous présentons un ensemble de tests. Nous concluons et proposons différentes pistes en section 5.

## 2 Formalisation des différents types de contraintes

Soit un ensemble d'observations (objets) $x_1, ..., x_n$ décrits dans $\mathbb{R}^d$, où $d$ est le nombre de dimensions. Dans la suite, nous noterons $X = (x_1, ..., x_n)^T$ la matrice des observations, la ligne $i$ correspondant à la description de $x_i$. Notre objectif consiste à fournir une représentation de ces objets en k dimensions (et notamment en trois dimensions) telle que :

- la perte d'information est réduite, ce qui se traduira par la recherche d'une représentation de variance (ou inertie) maximale (principe de l'ACP),
- les contraintes spécifiées par l'utilisateur sont si possible satisfaites (ces contraintes seront nommées *contraintes utilisateur*).

La représentation sera obtenue par projection du nuage de points dans un sous-espace à $k$ dimensions ($k \ll d$). Il faut donc déterminer k vecteurs $u_1, u_2$ ... $u_k$ associés à la matrice de projection $L = (u_1, u_2, \dots u_k)$ telle que les lignes de la matrice $X.L$ correspondent aux projections des observations dans notre sous-espace de dimension $k$. Soit $h(x_i) = L^T.x_i$ la

projection de $x_i$. Dans ce contexte, la distance euclidienne entre $x_i$ et $x_j$ *après projection*, $d(h(x_i)h(x_j))$, est définie par :

$$d^2(h(x_i), h(x_j)) \quad = \quad (x_i - x_j)^T.L.L^T.(x_i - x_j) \tag{1}$$

Dans la suite, nous noterons $M = L.L^T$ la matrice caractérisant la distance. Nous chercherons ici à résoudre notre problème de réduction de dimension, préservant au mieux l'inertie, sous les contraintes introduites dans les sections suivantes.

### 2.1 Correction de position de 2 objets ($C2$)

Pour ce premier type de correction, l'utilisateur souhaite modifier la distance entre deux objets $x_a$ et $x_b$. Soit $\tilde{d}$ la distance souhaitée entre les deux objets projetés après correction. Deux cas de figure se présentent :

- on souhaite rapprocher $x_a$ et $x_b$, ce qui peut s'exprimer par : $d^2(h(x_a), h(x_b)) \leq \tilde{d}$
- on souhaite éloigner $x_a$ et $x_b$, ce qui peut s'exprimer par : $d^2(h(x_a), h(x_b)) \geq \tilde{d}$

On peut donc définir de telles contraintes par un quadruplet $(a, b, \tilde{d}, \alpha)$, avec $\alpha = 1$ pour les contraintes correspondant au premier cas et $\alpha = -1$ pour le deuxième cas. Ainsi, la contrainte associée à $(a, b, \tilde{d}, \alpha)$ s'écrit :

$$\alpha * [d^2(h(x_a), h(x_b)) - \tilde{d}] \leq 0 \tag{2}$$

Dans la suite, nous noterons $C2$ l'ensemble des contraintes de ce type.

### 2.2 Correction de position de 2 objets par rapport à un 3ème ($C3$)

Pour ce deuxième type de correction, on souhaite modifier le positionnement relatif d'un objet $x_c$ par rapport à deux objets $x_a$ et $x_b$. Deux cas sont envisageables :

- on souhaite rapprocher $x_c$ de $x_a$, de façon à avoir $d^2(h(x_a), h(x_c)) \leq d^2(h(x_a), h(x_b))$. On précise l'intensité de rapprochement grâce à une variable $\delta$. Notre contrainte devient : $d^2(h(x_a), h(x_c)) \leq \delta * d^2(h(x_a), h(x_b))$
- on souhaite éloigner $x_c$ de $x_a$, de façon à avoir $d^2(h(x_a), h(x_c)) \geq d^2(h(x_a), h(x_b))$. On précise l'intensité de cet éloignement grâce à une variable $\delta$. Notre contrainte devient : $d^2(h(x_a), h(x_c)) \geq \delta * d^2(h(x_a), h(x_b))$

Dans la suite, nous notons $C3$ l'ensemble des contraintes de ce type. Une telle contrainte sera un quintuplet $(a, b, c, \delta_{abc}, \alpha)$, avec $\alpha = 1$ pour les contraintes correspondant au premier cas, $\alpha = -1$ pour le deuxième cas. Ainsi, si $(a, b, c, \delta, \alpha) \in C3$, la contrainte associée s'écrit :

$$\alpha * [d^2(h(x_a), h(x_c)) - \delta * d^2(h(x_a), h(x_b))] \leq 0 \tag{3}$$

## 3 Résolution

Nous pouvons résumer notre problème d'optimisation sous contraintes par :

$$\begin{cases} Max_L \sum_{i,j} (x_i - x_j)^t.L.L^t.(x_i - x_j) \\ \forall i, j \quad < u_i, u_j >= 1 \; si \; i = j, \; 0 \; sinon \\ \forall (a, b, \tilde{d}, \alpha) \in C2 \;\; \alpha * [d^2(h(x_a), h(x_b)) - \tilde{d}] \leq 0 \\ \forall (a, b, c, \delta, \alpha) \in C3 \;\; \alpha * [d^2(h(x_a), h(x_c)) - \delta * d^2(h(x_a), h(x_b))] \leq 0 \end{cases}$$

Dans toutes les contraintes précédentes interviennent des termes $d^2(h(x_a), h(x_i))$ dépendant de la solution de notre problème (i.e. de la matrice $L$). Comme précédemment, on peut écrire $d^2(h(x_a), h(x_i)) = (x_a - x_i)^t.L.L^t.(x_a - x_i)$ et noter que :

$$\begin{aligned} d^2(h(x_a), h(x_i)) &= (x_a - x_i)^t.L.L^t.(x_a - x_i) \quad (4) \\ &= \sum_{j=1..k} u_j^t.(x_a - x_i).(x_a - x_i)^t.u_j \quad (5) \end{aligned}$$

où $(x_a - x_i).(x_a - x_i)^t$ est une *matrice* $n \times n$. Dans la suite nous noterons $X_{a,i}$ cette matrice, et ainsi : $d^2(h(x_a), h(x_i)) = \sum_{j=1..k} u_j^t.X_{a,i}.u_j$.
Contrairement au cas sans contraintes, on ne peut plus chercher successivement les vecteurs $u_1, u_2 \dots u_k$, puisqu'ici les contraintes utilisateurs sont *globales* : ces contraintes ne doivent pas être respectées sur chaque dimension de projection indépendamment, mais dans l'espace de projection. Nous devons donc traiter la recherche des vecteurs $u_j$ simultanément.
Nous pouvons réécrire le critère à maximiser :

$$\sum_{i,j}(x_i - x_j)^t.L.L^t.(x_i - x_j) = \sum_{j=1..k} u_j^t.X^t.X.u_j \quad (6)$$

Considérons dans un premier temps le lagrangien $\mathcal{L}$ de ce problème, en ne tenant pas compte des contraintes d'orthogonalité des vecteurs $u_j$ :

$$\begin{aligned} \mathcal{L}(L, \lambda, \mu, \psi) &= -\sum_{j=1..k} u_j^t.X^t.X.u_j + \sum_{j=1..k} \lambda_j(u_j^t.u_j - 1) \\ &+ \sum_{(a_i,b_i,\tilde{d}_i,\alpha_i)\in C2} \mu_i * ((\sum_{j=1..k} u_j^t.X_{a_i,b_i}.u_j) - \tilde{d}_i) * \alpha_i) \\ &+ \sum_{(a_i,b_i,c_i,\delta_i,\alpha_i)\in C3} \psi_i * ((\sum_{j=1..k} u_j^t.X_{a_i,c_i}.u_j) - (\delta_i * (\sum_{j=1..k} u_j^t.X_{a_i,b_i}.u_j)) * \alpha_i) \end{aligned}$$

Dérivons ce Lagrangien par rapport à $u_j$ (en multipliant par 2 pour simplifier) :

$$\begin{aligned} 2\frac{\partial \mathcal{L}(L, \lambda, \mu, \psi)}{\partial u_j} &= -X^t.X.u_j + \lambda_j.u_j + \sum_{(a_i,b_i,\tilde{d}_i,\alpha_i)\in C2} \mu_i\alpha_i X_{a_i,b_i}.u_j \\ &+ \sum_{(a_i,b_i,c_i,\delta_i,\alpha_i)\in C3} \psi_i\alpha_i * (X_{a_i,c_i}.u_j - \delta_i * (X_{a_i,b_i}.u_j)) \end{aligned}$$

Soit $\mathcal{X}_C$ la matrice définie par :

$$\mathcal{X}_C = X^t.X - \sum_{(a_i,b_i,\tilde{d}_i,\alpha_i)\in C2} \mu_i\delta_i X_{a_i,b_i} - \sum_{(a_i,b_i,c_i,\delta_i,\alpha_i)\in C3} \psi_i\delta_i * (X_{a_i,c_i} - \delta_i * (X_{a_i,b_i}))$$

la dérivée partielle du lagrangien s'écrit alors :

$$2\frac{\partial \mathcal{L}(L, \lambda, \mu, \psi)}{\partial u_j} = -\mathcal{X}_C.u_j + \lambda_j.u_j \quad (7)$$

Notons au passage que le Lagrangien s'écrit :

$$\mathcal{L}(L, \lambda, \mu, \psi) = -\sum_{j=1..k} u_j^t \mathcal{X}_C.u_k + \sum_{j=1..k} \lambda_j(u_j^t.u_j - 1) \quad (8)$$

L'annulation des dérivées partielles donne $\mathcal{X}_C.u_j = \lambda_j.u_j$. En d'autres termes, les solutions $u_j^*$ sont les vecteurs propres de la matrice $\mathcal{X}_C$, associée aux valeurs propres $\lambda_j$, ce qui a deux conséquences notables : tout d'abord, bien que n'ayant pas tenu compte de la contrainte d'orthogonalité des vecteurs $u_j$, les solutions obtenues sont orthogonales. Nous pouvons par ailleurs en déduire que la fonction duale $q(\lambda, \mu, \psi)$ du problème présente une forme assez simple :

$$q(\lambda, \mu, \psi) \quad = \quad Min_L \mathcal{L}(L, \lambda, \mu, \psi)$$

En effet, en utilisant les conditions d'optimalité précédentes :

$$\begin{aligned} q(\lambda, \mu, \psi) &= Min_L - \sum_{j=1..k} u_j^t \mathcal{X}_C.u_j + \sum_{j=1..k} \lambda_j(u_j^t.u_j - 1) \\ &= Min_L - \sum_{j=1..k} u_j^T.(\lambda_j u_j) + \sum_{j=1..k} \lambda_j(u_j^t.u_j - 1) \\ &= Min_L - \sum_{j=1..k} \lambda_j ||u_j||^2 + \sum_{j=1..k} \lambda_j(||u_j||^2 - 1) \\ &= Min_L - \sum_{j=1..k} \lambda_j \end{aligned}$$

les contraintes imposant $||u_j||^2 = 1$ pour la solution optimale.
Finalement, puisque le problème dual consiste à maximiser la fonction duale, la solution optimale correspond à maximiser la somme de k valeurs propres de la matrice $\mathcal{X}_C$, c'est-à-dire les $\lambda_j$ correspondent aux k plus grandes valeurs propres de cette matrice. Néanmoins, pour calculer les valeurs propres de $\mathcal{X}_C$, il faudrait connaître les valeurs des variables duales $\mu, \psi$, valeurs dont nous ne disposons pas. Pour cette raison, nous proposons d'utiliser l'algorithme itératif d'Uzawa, qui permet d'obtenir des approximations de ces variables. Cet algorithme est décrit dans la section suivante.

## 3.1 Mise en oeuvre de l'algorithme d'Uzawa

L'algorithme d'Uzawa, introduit par Arrow et al. (1958), repose sur l'idée de trouver un point selle du lagrangien par approximations successives. Dans la formulation classique de l'algorithme, on considère le problème d'optimisation suivant :

$$\begin{cases} Min_x J(x) \\ h(x) = 0 \\ g(x) \leq 0 \end{cases}$$

où les fonctions $h$ et $g$ désignent en réalité des familles de fonctions $h_i$ et $g_j$. Le lagrangien s'écrit dans ce cas :

$\mathcal{L}(x, \lambda, \mu) = J(x) + \sum_i \lambda_i h_i(x) + \sum_j \mu_i g_j(x)$

L'algorithme d'Uzawa consiste à fixer des valeurs initiales des coefficients de Lagrange ($\lambda^0$, $\mu^0$), chercher l'optimum du lagrangien, puis à corriger les coefficients de Lagrange en fonction de cette solution. Ce processus est itéré jusqu'à convergence (la convergence est garantie).

1 - Fixer $\lambda^0, \mu^0$,

2 - Itérer pour $n \geq 0$ ($\rho$ est un paramètre) :

2.1 Calculer la solution $x_n$ de : $Min_x L(x, \lambda^n, \mu^n)$

2.2 Mettre à jour $(\lambda^n, \mu^n)$ par :

$\lambda_i^{n+1} = \lambda_i^n + \rho * h_i(x_n)$

$\mu_j^{n+1} = max(0, \mu_j^n + \rho * g_j(x_n))$

2.3 Tester la convergence : $||x_{n+1} - x_n|| < \epsilon$

### 3.2 Application de l'algorithme d'Uzawa

Dans notre cas, l'algorithme d'Uzawa se simplifie légèrement. Il n'est pas nécessaire d'approximer les valeurs de $\lambda_i$ : les $x_n$ étant ici les vecteurs propres $u_1 \dots u_k$, les valeurs propres en découlent directement. Nous fixerons donc seulement les valeurs des coefficients $\mu^n$, c'est-à-dire, dans notre cas, des coefficients $\mu, \psi$.

Nous initialisons ces coefficients à 0. La première itération sera donc une simple ACP puisque, dans ce cas, $\mathcal{X}_C = X^t.X$,

Ensuite, les contraintes non satisfaites vont permettre de corriger la matrice $\mathcal{X}_C$ : dans l'algorithme d'Uzawa, si une contrainte ($g_j(x_n) \leq O$) n'est pas satisfaite, alors $g_j(x_n) > O$, ce qui implique que la mise à jour : $\mu_j^{n+1} = max(0, \mu_j^n + \rho * g_j(x_n))$ va rendre $\mu_j^{n+1} > 0$. La matrice $\mathcal{X}_C$ sera donc calculée en tenant compte de $g_j(x_n)$. Cette mise à jour de $\mathcal{X}_C$ est assez intuitive : prenons par exemple une contrainte de type $C2 : d^2(h(x_a), h(x_b)) \geq \tilde{d}$ ; si cette contrainte n'est pas satisfaite, dans le calcul de $\mathcal{X}_C$, il faut ajouter à $X^T.X$ la matrice $cX_{a,b}$ (où $c$ est une constante). Or $X_{a,b}$ est déjà dans $X^T.X$, cela revient donc à augmenter le "poids" de la distance $d^2(h(x_a), h(x_b))$ dans le critère à optimiser (sans contraintes). Tant que la contrainte ne sera pas satisfaite, on continuera d'augmenter ce poids...

En résumé, à chaque itération, il suffit de mettre à jour la matrice $\mathcal{X}_C$ en y ajoutant des matrices de la forme $cX_{a,b}$, puis diagonaliser la matrice $\mathcal{X}_C$.

Cette méthode permet donc de manipuler des "contraintes souples" dans le sens où la convergence peut être atteinte avant que les contraintes soient toutes satisfaites, permettant ainsi de ne pas être pénalisé par des contraintes difficiles à satisfaire.

## 4 Expérimentations

Cette méthode a été intégrée au sein de notre outil de visualisation Explorer3D (Martin et Exbrayat (2009)). Les contraintes sont saisies interactivement par sélection d'objets à l'aide de la souris dans une scène 3D, puis par saisie des bornes souhaitées en positionnant un curseur. Après recalcul des projections, les objets sont déplacés vers leur nouvelle position. L'utilisateur peut ainsi observer l'impact des contraintes sur la projection des objets.

Toutefois, l'efficacité d'une manipulation graphique est par nature difficile à évaluer objectivement. Nous proposons donc un protocole de validation permettant de mesurer s'il est possible d'atteindre une représentation satisfaisante avec un nombre limité de contraintes.

Pour mesurer ce degré de satisfaction, nous proposons de comparer la représentation obtenue (en 3D) à une organisation de référence. On peut estimer qu'un utilisateur jugera satisfaisante une organisation dans laquelle les objets de même classe sont proches et les objets de classes différentes éloignés. Nous pouvons donc utiliser le critère de Fisher $\frac{\text{variance intra}-\text{classs}}{\text{variance inter}-\text{classes}}$, critère que maximise l'analyse discriminante (ALD). En conséquence, nous utiliserons comme organisation de référence celle produite par application de l'ALD à notre jeu de données.

Nous noterons $d_{ald}$ la distance induite dans l'espace de représentation obtenu par ALD, restreint aux 3 dimensions les plus significatives (l'utilisateur manipulant les objets dans un espace 3D). Afin de pouvoir utiliser cette approche, nous utiliserons des jeux de données étiquetées, les informations de classes seront utilisées pour générer les contraintes. Par ailleurs, afin de simuler le comportement de l'utilisateur, nous utiliserons un processus automatique de choix de contraintes, choisissant en priorité la correction des déformations les plus importantes (par rapport à l'organisation proposée par l'ALD). La section suivante présente les différents types de contraintes générées.

## 4.1 Génération de contraintes

Rappelons que notre processus repose sur une projection initiale de type ACP. Nous noterons $d(a, b)$ la distance entre les objets $a$ et $b$ après projection. Implicitement, nous considérons ici que la projection se fait en trois dimensions. Nous étudions cinq types de contraintes :

$C2_{inf}$ : pour générer des contraintes du type $d(a, b) \leq \tilde{d}$, nous devons choisir 2 objets $a$ et $b$ ainsi qu'une valeur de distance seuil $\tilde{d}$. Puisque l'on souhaite ici rapprocher les projections des objets $a$ et $b$, il faut choisir 2 objets éloignés dans la projection courante, mais proches relativement à la distance $d_{ald}$. Le couple d'objets le plus mal représenté sera donc celui pour lequel le rapport $\frac{d(a,b)}{d_{ald}(a,b)}$ est maximum. La distance seuil choisie sera alors $d_{ald}(a, b)$,

$C2_{sup}$ : dans le même esprit, nous pourrons introduire la contrainte $d(a, b) \geq d_{ald}(a, b)$ en choisissant le couple $(a, b)$ qui minimise $\frac{d(a,b)}{d_{ald}(a,b)}$,

$C3_{ald}$ : les contraintes de type $C_3$ sont de la forme $d(a, c) \leq \delta d(a, b)$. Il est donc nécessaire de choisir 3 objets et une valeur seuil. Nous proposons ici de générer ce type de contraintes pour des objets $a, b, c$ tels que $a$ et $c$ soient de même classe, $b$ d'une autre classe, l'idée étant de rapprocher $c$ (projeté) de $a$ (projeté), en le positionnant par rapport à $b$. Nous choisissons les objets $a, b, c$ qui maximisent $\frac{d(a,c)}{d(a,b)}$. Le seuil choisi ici sera : $\delta = \frac{d_{ald}(a,c)}{d_{ald}(a,b)}$,

$C3_1$ : Plutôt que de fixer le seuil $\delta$ en fonction de l'ALD, nous proposons ici de le fixer simplement à 1. Ainsi, la contrainte produite est : $d(a, c) \leq d(a, b)$, i.e. l'objet $c$ doit être plus proche de l'objet $a$ que ne l'est l'objet $b$. Le choix des objets $a$, $b$ et $c$ se fait sur le même critère que précédemment.

$C3_{1/2}$ : pour ce dernier type de contraintes, nous fixons le seuil $\delta$ à $1/2$, ce qui a pour objectif de séparer plus nettement les classes.

Dans nos expérimentations, chaque test porte exclusivement sur l'un de ces types de contraintes. Les contraintes sont introduites une à une, suivant l'ordre de priorité spécifié ci-dessus. La première contrainte est déterminée en fonction de la distance initiale (ACP sans contraintes), ce qui fournit une seconde représentation (ACP avec une contrainte), induisant

une nouvelle distance qui permet de construire une seconde contrainte, et ainsi de suite. Ce modèle simule ainsi le comportement d'un utilisateur qui ajoute itérativement des contraintes en fonction de la représentation 3D courante (laquelle tient compte des contraintes précédentes).

## 4.2 Critère d'inertie

Nous considérons ici un jeu de données artificiel, doont les 75 objets sont des mots décrits par 48 attributs (14 attributs syntaxiques, 4 attributs de catégorie, 20 attributs sémantiques et 10 attributs de bruit). L'intérêt de ce jeu de données est de pouvoir choisir différentes classifications cibles (syntaxe, sémantique, catégorie). La classe cible choisie ici sera la classe thématique. Les mots sont ainsi répartis en 4 classes thématiques disjointes.

Le critère de qualité que nous avons retenu est : $\mathrm{Q} = \frac{\text{variance inter}-\text{classe}}{\text{variance totale}}$ : plus sa valeur est élevée, plus les classes sont "séparées". Ce critère est calculé sur la représentation associée aux 3 dimensions les plus significatives (laquelle correspond à une bonne organisation visuelle).

La figure 1 présente l'évolution de ce critère suivant le nombre et le type de contraintes. La ligne horizontale supérieure représente le rapport $\frac{\text{variance inter}-\text{classe}}{\text{variance totale}}$ obtenu par ALD, i.e. la valeur optimale pour ce rapport (94,2%). On peut remarquer que la plupart des types de contraintes permet d'approcher ce rapport, à l'exception des contraintes $C2_{sup}$. Ces dernières tendent à éloigner des objets, et devraient donc permettre d'améliorer le critère. Cependant, dans l'espace d'origine, la variance totale est plus grande (les objets sont en moyenne plus éloignés). Or les seuils sont choisis par rapport aux distances induites par l'ALD. Par conséquent, les seuils choisis sont vraisemblablement assez mal adaptés. De plus, ce type de contraintes est en fait redondant avec le critère optimisé dans l'ACP. Les contraintes de type $C_{2inf}$ fournissant ici les meilleurs résultats, nous les utiliserons pour la suite de nos tests.

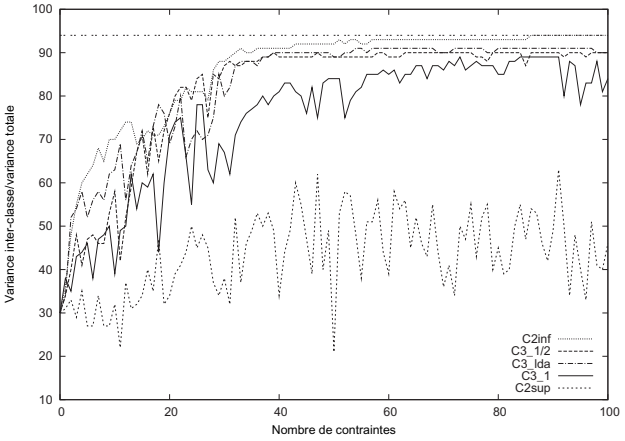


FIG. 1 – *Evolution de* $\frac{variance\ inter-classe}{variance\ totale}$ *en fonction du nombre de contraintes*

La figure 2 présente les organisations visuelles obtenues avec respectivement 0, 5, 30 et 100 contraintes de type $C2_{inf}$ générées automatiquement. Chaque classe est caractérisée par

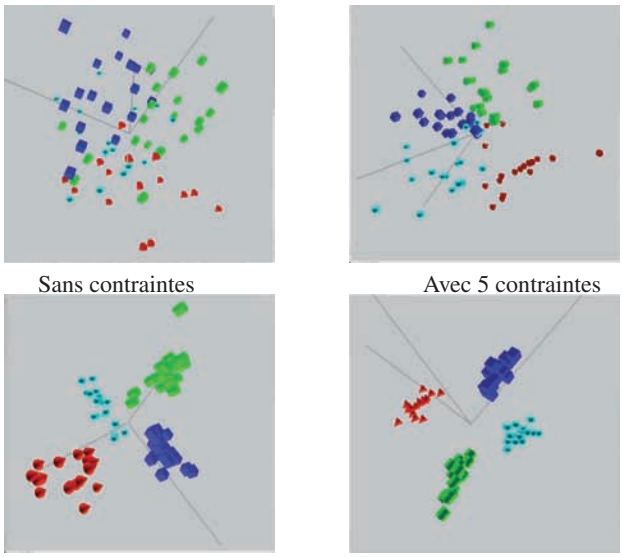


FIG. 2 – *Projections 3D obtenues avec un nombre croissant de contraintes*

une forme et une couleur. Nous pouvons constater que la séparation des classes progresse rapidement lorsque le nombre de contraintes augmente. Ces tests nous permettent de montrer que, pour ce jeu de données, avec une trentaine de contraintes, le rapport est proche de l'optimal. Cela correspond à un nombre de contraintes tout à fait acceptable si elles doivent être ajoutées par un utilisateur. De plus, les temps de calcul pour la résolution sont ici d'une seconde dans le cas avec 100 contraintes.

Nous avons mené les mêmes tests sur 6 jeux de données usuels en apprentissage : breast cancer, glass, iris, wine, yeast et zoo (de l'UCI repository). La figure 3 présente l'évolution du critère $Q$ pour les contraintes $C2_{inf}$ et confirme la tendance observée sur le jeu de données artificiel. Afin de pouvoir présenter ces courbes sur un même graphique, les valeurs représentées sont relatives à la valeur du même critère dans le cadre de l'ALD : $Q_{ALD}$. Les courbes représentent donc l'évolution du taux $Q/Q_{ALD}$, une valeur proche de 100% correspond donc à une rapport variance inter-classe/variance intra-classe proche de celle obtenue grâce à l'ALD.

## 4.3 Variation du taux de classification

Nous proposons ici un second critère consistant à mesurer le taux de classification correcte par validation croisée, qui n'est pas en lien direct avec l'objectif initial, mais qui produit des résultats intéressants. Les contraintes et la représentation sont calculées à partir d'un ensemble test. L'ensemble d'apprentissage est ensuite projeté dans l'espace de représentation obtenu où les objets sont classés en fonction de leur(s) plus proche(s) voisin(s). La figure 4 présente les

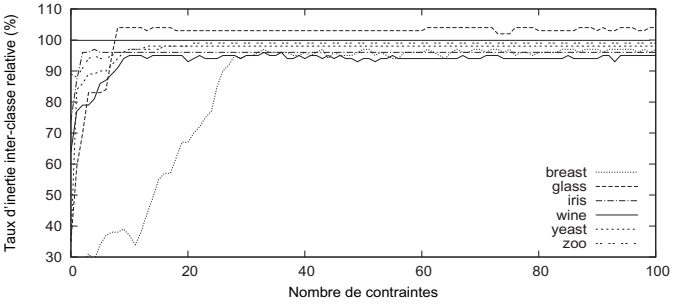


FIG. 3 – *Evolution de* $\frac{variance\ inter-classe}{variance\ totale}$ *en fonction du nombre de contraintes*

résultats obtenus sur notre jeu de données artificiel (10 exécutions de 5-fold cross-validation, 1-plus proche voisins). La ligne horizontale supérieure correspond au taux obtenu par ALD.

Les contraintes n'apportent pas ici d'amélioration du taux de classification, à l'exception des contraintes $C3_{1/2}$. Ce point est intéressant puisque ces contraintes ne népendent pas du résultat de l'ALD. Nous étudions actuellement la construction d'un classifieur basé sur cette approche.

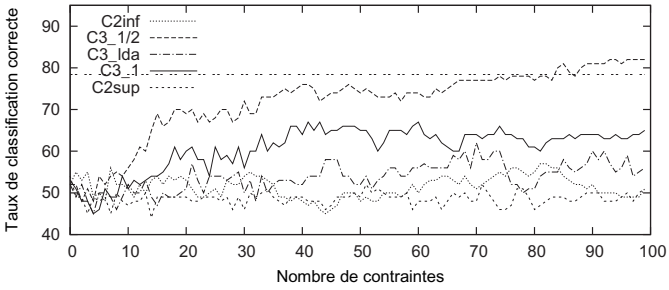


FIG. 4 – *Taux de classification correcte en fonction du nombre de contraintes*

# 5 Conclusion et perspectives

Nous avons présenté dans cet article une méthode de réduction de dimension autorisant l'ajout itératif et intuitif de contraintes de positionnement des objets dans l'espace de projection. Nous avons observé que pour différents jeux de données, l'ajout d'un nombre restreint de contraintes permet d'obtenir une solution satisfaisante. En conséquence, une telle approche ne peut que contribuer à la diffusion des méthodes de réduction de dimension auprès d'utilisateurs ne disposant pas de connaissances spécifiques sur ces techniques.

De plus, cette approche introduit le concept d'ajout progressif de contraintes, pour lequel elle propose une solution. Ce concept nous semble particulièrement prometteur, puisqu'il permet de limiter la complexité du système de contraintes sous-jacent.

Ces travaux s'inscrivent dans le cadre du projet ANR Graphem, dont l'un des objectifs est de permettre à des paléographes d'obtenir une organisation visuelle de différents styles d'écritures (non étiquetés), dont ils sont capables d'évaluer la ressemblance. Une validation de cette approche est en cours. Elle pourrait conduire à l'introduction de nouveaux types de contraintes.

## Références

Arrow, K., L. Hurwicz, et H. Uzawa (1958). *Studies in Nonlinear Programming*. Stanford, CA : Stanford University Press.

Bar-Hillel, A., T. Hertz, N. Shental, et D. Weinshall (2005). Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research 6*, 937–965.

Blum, A. L. et P. Langley (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence 97*, 245–271.

Cook, D., A. Buja, J. Cabrera, et C. Hurley (1995). Grand tour and projection pursuit. *Journal of Computational and Graphical Statistics 4*, 155–172.

Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics 7*, 179–188.

Guyon, I. et A. Elisseeff (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research 3*, 1157–1182.

Martin, L. et M. Exbrayat (2009). Explorer3d. www.univ-orleans.fr/lifo/software/Explorer3D.

Weinberger, K. Q., J. Blitzer, et L. K. Saul (2005). Distance metric learning for large margin nearest neighbor classification. In *NIPS*.

Weinberger, K. Q. et L. K. Saul (2008). Fast solvers and efficient implementations for distance metric learning. In W. W. Cohen, A. McCallum, et S. T. Roweis (Eds.), *ICML*, Volume 307 of *ACM International Conference Proceeding Series*, pp. 1160–1167. ACM.

## Summary

Many dimensionality reduction methods have been proposed both in supervised and unsupervised domains. An interesting aspect of these methods is the ability to get a visualization of data such that, if two objects are apparently close then they are similar w.r.t. a similarity notion corresponding either to some expert knowledge or additional information such as class labels. In this paper, we consider a semi-supervised context where information is introduced in an interactive way by adding constraints specifying differences between a visualization and some expert knowledge. For instance, one can specify that two objects are close in the visualization but are in fact not similar for the expert, or conversely. The method considered here is derived from the principal component analysis (PCA), where two types of constraints are added. We present here a resolution method that has been implemented in a software offering a 3D visualization were the user can interactively add constraints and then visualize the modifications induced by these latter. Experiments have been performed on both synthetic and usual datasets. They show that a representation with a high value of quality criterion can be obtained with a limited number of constraints.