Prise en compte de connaissances pour la visualisation par l'intégration interactive de contraintes

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. La projection et la visualisation d'objets dans un espace à deux ou trois dimensions constituent une tâche courante de l'analyse de données. Cette opération induit des pertes dans le sens où certains objets peuvent se retrouver très proches alors qu'ils sont à l'origine assez éloignés. A partir de cette visualisation, il semble intéressant d'offrir à l'utilisateur la possibilité d'apporter de la connaissance sous forme de contraintes spécifiant les similarités attendues entre divers objets, lorsque ceux ci sont, dans l'espace d'observation, visuellement trop proches, ou au contraire trop éloignés.

Nous proposons ici trois types de contraintes et présentons une méthode de résolution de celles-ci dérivée de l'analyse en composantes principales (ACP). Deux types d'expérimentation sont présentées, 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 automatiques 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.

Deux approches sont envisageables afin de réduire le nombre de descripteurs. La première consiste à ne retenir que 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 appelée 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 permet de réduire les 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. La dualité entre visualisation et classification s'avère remarquable dans les méthodes de réduction de dimension. La visualisation fournit un outil intuitif, accessible au non spécialiste. La classification automatique permet de quantifier objectivement l'efficacité de la méthode.

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, 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 en variance inter-classes et variance intra-classes).

Lorsque la répartition des données présente certaines caractéristiques topologiques (objets répartis, à grande échelle, à proximité d'une surface donnée), il est possible d'appliquer des méthodes reposant sur le concept de *variété*. Une partie de ces méthodes procède en deux temps : tout d'abord, on recense les paires d'objets proches dans l'espace d'origine. Ensuite, on cherche à optimiser un critère de projection (par exemple la variance globale) tout en limitant les déformations des distances entre paires d'objets proches, c'est à dire en limitant les déformations locales de l'espace. Les projections sont calculées soit par recherche de valeurs propres (Tenenbaum et al. (2000)), soit par résolution d'un système de contraintes (Roweis et Saul (2000); Weinberger et Saul (2006)). Il existe également des approches à une phase, telle l'analyse en composantes curvilignes (Demartines et Hérault (1997)) qui cherche à minimiser la somme des différences entre distances d'origine et distance projetées, une pondération autorisant alors de plus grandes distorsions pour des paires d'objets distants.

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 ici sur un échantillon d'objets de différentes classes : l'utilisateur déclare un ensemble de paires d'objets de même classe. Par clôture transitive, l'algorithme reconstitue des groupes d'objets de même classe, appelés *chunklets* (lesquels ne constituent qu'un échantillon de l'ensemble X des objets disponibles). Une matrice de covariance intra-chunklet $\hat{C}$ est produite, laquelle sert ensuite de base à la projection des objets ($X_{new} = \hat{C}^{-1/2}X$) et/ou au calcul d'une distance de Mahanalobis entre eux : $d(x_1, x_2) = (x_1 - x_2)^t \hat{C}^{-1}(x_1 - x_2)$. Il est également possible d'insérer une étape intermédiaire de réduction de dimension à partir de $\hat{C}$.

La classification par maximisation de la marge entre plus proches voisins (*Large Margin Nearest Neighbors*, ou LMNN) proposée par Weinberger et al. (2005) et Weinberger et Saul (2008) se concentre pour sa part sur la notion de 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. Les auteurs proposent une formulation du problème sous forme d'un programme semi défini, dont la résolution produit une matrice M servant de support au calcul d'une distance de Mahanalobis entre les objets. Cette résolution repose sur un solveur spécifique. De cette matrice peut se déduire une matrice de projection en faible dimension L, telle que $M = L^t L$.

Ces deux approches présentent des limitations. L'approche LMNN introduit des contraintes pour l'ensemble des objets. La résolution peut donc s'avérer lourde, notamment en cas de zones très hétérogènes. 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 MLNN, tout en proposant un mécanisme intuitif et itératif d'intégration de connaissances du domaine.

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 trois 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 deux objets,
- positionnement de deux objets (b, c) par rapport à un troisième(a) : consiste à borner le rapport distance(a, c)/distance(a, b).
- voisinage d'un objet : il s'agit d'une généralisation du cas précédent. Un objet donné doit se retrouver dans le voisinage d'un groupe d'objets donnés.

Nous souhaitons pouvoir intégrer ces contraintes de manière interactive : l'utilisateur observe une représentation en deux ou trois dimensions des données, puis peut spécifier des corrections attendues en ajoutant des contraintes, par exemple dans le cas où deux objets "semblables pour l'utilisateur" se retrouvent 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 est important de noter que cette approche a pour objectif d'intégrer des contraintes de distances dans un processus de réduction de dimension pouvant conduire à de la visualisation. Il diffère donc des deux approches présentées précédemment (RCA et LMNN), non seulement dans les types de contraintes définies, mais également dans le sens qu'il offre clairement la possibilité d'interactions graphiques. A l'inverse, il diffère des autres outils d'exploration visuelle de données par projection, tels Da Costa et Venturini (2007), dans le sens où il ne s'agit pas de déplacer le point de vue de l'utilisateur (afin d'identifier des groupes d'objets). Il s'agit ici d'agir au sein d'une visualisation globale, "uniforme" dans le sens ou elle reproduit au mieux les distances entre tous les objets dans un espace euclidien.

L'article est organisé comme suit : dans la section 2, nous définissons plus formellement les trois types de contraintes proposés. En section 3, nous proposons un mécanisme de résolution de contraintes dures reposant sur l'algorithme d'Uzawa. En section 4, nous présentons l'inter-

face graphique d'ajout interactif de contraintes. En section 5 nous présentons un ensemble de tests. Nous concluons et proposons différentes pistes en section 6.

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 :

$$\begin{aligned} d^2(h(x_i), h(x_j)) &= < h(x_i) - h(x_j), h(x_i) - h(x_j) > \quad (1) \\ &= (h(x_i) - h(x_j))^T.(h(x_i) - h(x_j)) \\ &= (L^T.x_i - L^T.x_j)^T.(L^T.x_i - L^T.x_j) \\ &= (L^T.(x_i - x_j))^T.(L^T.(x_i - x_j)) \\ &= (x_i - x_j)^T.L.L^T.(x_i - x_j) \quad (2) \end{aligned}$$

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 :

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

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 δ. 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 δ. 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 :

$$\alpha * [d^2(h(x_a), h(x_c)) - \delta * d^2(h(x_a), h(x_b))] \leq 0 \tag{4}$$

2.3 Modification de voisinage (Cv)

Ce troisième type de contrainte a pour objectif de repositionner un objet x_a dans le voisinage d'un groupe d'objets. Soit Vs_a le voisinage source de x_a et Vc_a son voisinage cible. On souhaite que x_a soit à la fois rapproché des objets de Vc_a et éloigné des objets de Vs_a. Cela peut se traduire par un jeu de contraintes portant respectivement sur les objets de ces deux voisinages.

Concernant le voisinage cible, la liste des k_a objets composant Vc_a est déterminée par l'utilisateur. Concernant le voisinage source, on retient les k plus proches voisins de x_a. La valeur de k peut être fixée arbitrairement (par exemple $k = 3$) ou être égale à k_a. Les contraintes de modification de voisinage peuvent s'exprimer comme suit :

- Pour tout $x_i \in Vs_a$, notons d_{ai} la distance observée, dans l'espace de projection courant, entre x_a et x_i. L'utilisateur souhaite que la distance observée après correction soit plus grande, ce que l'on peut traduire par la contrainte suivante :

  $\forall x_i \in Vs_a \quad d^2(h(x_a), h(x_i)) \geq \beta * d_{ai}$

  β étant un paramètre à fixer (par exemple, $\beta = 5$).

  Notons cependant que $d(x_a, x_i)$, la distance entre x_a et x_i dans l'*espace d'origine* (avant projection), constitue par nature une borne supérieure de $d^2(h(x_a), h(x_i))$. Nous pouvons donc modifier notre contrainte pour exprimer que $d^2(h(x_a), h(x_i))$ doit être assez proche de $d^2(x_a, x_i)$:

  $\forall x_i \in Vs_a \quad d^2(h(x_a), h(x_i)) \geq \gamma * d^2(x_a, x_i)$

  où $\gamma \in [0, 1]$ est un paramètre à fixer (e.g. 0,75). Nous retenons cette formulation.
- Concernant les objets de Vc_a, nous pourrions essayer de minimiser $d^2(h(x_a), h(x_i))$. Nous pouvons également ajouter une contrainte spécifiant que $d^2(h(x_a), h(x_i))$ est bornée supérieurement par une valeur à fixer. Nous pouvons considérer comme borne une valeur dépendant de la distance moyenne $\overline{d_{Vc_a}}$ entre les objets de Vc_a :

  $\forall x_i \in V_a \quad d^2(h(x_a), h(x_i)) \leq \epsilon * \overline{d_{Va}}$

  où ϵ est un paramètre à fixer (e.g. 1,5). Nous retenons cette formulation proche de celle obtenue pour les objets de Vs_a.

Dans la suite nous noterons Cv l'ensemble des contraintes de ce type. Une contrainte appartenant à Cv est ainsi un quintuplet $(a, i, d, \theta, \alpha)$ tel que :

$$\alpha * [(d(h(x_a), h(x_i)) - \theta * d)] \leq 0 \tag{5}$$

avec :

- $\theta = \gamma$, $\alpha = -1$ et $d = d(x_a, x_i)$ pour les contraintes issues des voisinages sources, et
- $\theta = \epsilon$, $\alpha = 1$ et $d = \overline{d_{Vc_a}}$ pour les contraintes issues des voisinages cibles.

3 Résolution

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

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

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 :

$$\begin{aligned} d^2(h(x_a), h(x_i)) &= (x_a - x_i)^t.L.L^t.(x_a - x_i) && (6) \\ &= \sum_{j=1..k} u_j^t.(x_a - x_i).(x_a - x_i)^t.u_j && (7) \end{aligned}$$

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 $u_1, u_2 \ldots u_k$, puisqu'ici les contraintes utilisateurs sont *globales* : ces contraintes ne doivent pas être respectées sur chaque dimension de projection indépendamment, mais globalement dans l'espace de projection. Nous devons donc traiter la recherche des k vecteurs u_j simultanément.

Nous pouvons réécrire le critère à maximiser :

$$\sum_{i,j} (x_i - x_j)^t.L.L^t.(x_i - x_j) = \sum_{j=1..k} u_j^t.X^t.X.u_j \tag{8}$$

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 :

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

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

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

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

$$\begin{aligned}\mathcal{X}_C &= X^t.X-\sum_{(a_i,b_i,\tilde{d}_i,\alpha_i)\in C2}\mu_i\delta_i X_{a_i,b_i}\\ &- \sum_{(a_i,b_i,c_i,\delta_i,\alpha_i)\in C3}\psi_i\delta_i*(X_{a_i,c_i}-\delta_i*(X_{a_i,b_i}))\\ &- \sum_{(a_i,i_i,d_i,\theta_i,\alpha_i)\in Cv}\rho_i\alpha_i*(X_{a_i,i_i})\end{aligned}$$

la dérivée partielle du lagrangien s'écrit alors :

$$2\frac{\partial\mathcal{L}(L,\lambda,\mu,\psi,\rho)}{\partial u_j} = -\mathcal{X}_C.u_j+\lambda_j.u_j \tag{9}$$

Notons au passage que le Lagrangien s'écrit :

$$\mathcal{L}(L,\lambda,\mu,\psi,\rho) = -\sum_{j=1..k}u_j^t\mathcal{X}_C.u_k+\sum_{j=1..k}\lambda_j(u_j^t.u_j-1) \tag{10}$$

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 λ_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 $q(\lambda,\mu,\psi,\rho)$ du problème présente une forme assez simple :

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

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

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

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 λ_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 μ, ψ et ρ, 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 :

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

L'algorithme d'Uzawa consiste à fixer des valeurs initiales des coefficients de Lagrange (λ^0, μ^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 λ^0, μ^0,

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

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

2.2 Mettre à jour (λ^n, μ^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 λ_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 μ^n, c'est-à-dire, dans notre cas, des coefficients μ, ψ.

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 de diagonaliser la matrice $\mathcal{X}_C$.

Notons que cette méthode permet 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 Mise en œuvre graphique

L'ajout interactif des contraintes a été implanté dans le cadre d'un logiciel de visualisation 3D. L'utilisateur procède ainsi : il définit le type de contrainte à appliquer puis sélectionne les objets concernés. Il dispose alors d'un "ascenceur" horizontal afin de spécifier à la fois le sens de la contrainte (contraintes de type C2 et C3) et l'intensité de celle-ci.

4.1 Ajout de contraintes de type C2

La figure 1 représente la fenêtre de saisie de contrainte pour un couple d'objets donné. Ces deux objets sont sélectionnés dans la fenêtre 3D, et sont visualisés respectivement par des cibles rouge et verte (fig. 2). Nous pouvons observer la présence d'un ascenceur, dont le curseur est placé par défaut sur la valeur "65", ce qui signifie que dans la projection courante, la distance entre les deux points (5,62) correspond à 65% de leur distance dans l'espace d'origine (8,58).

Si l'on déplace l'ascenceur vers la droite, la contrainte reviendra à éloigner les deux objets (avec, comme borne supérieure, leur distance dans l'espace d'origine). Si on le déplace vers la gauche, la contrainte indiquera un rapprochement des deux objets (avec, comme borne inférieure, une distance nulle). Ici, nous choisissons de rapprocher les deux objets : l'ascenceur est placé sur 20.

La figure 4 présente l'espace de projection recalculé avec application de cette seule contrainte. Nous pouvons observer que les deux objets se retrouvent visuellement plus proches.

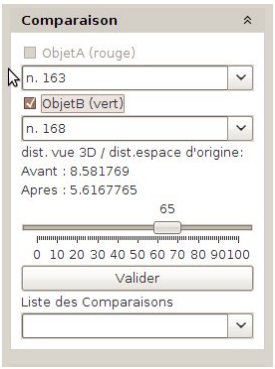

FIG. 1 – *Saisie de contraintes de type C2*

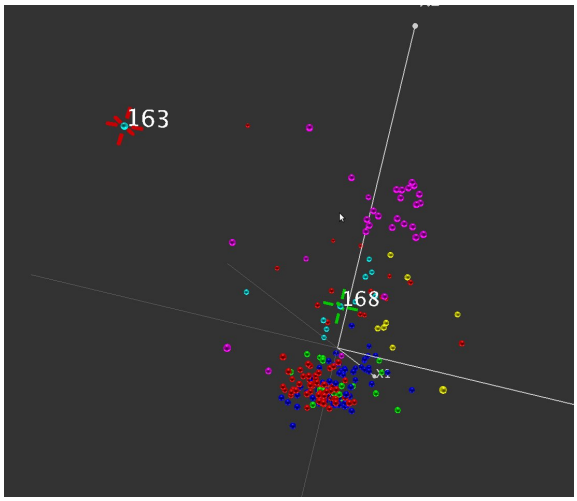

FIG. 2 – *Les objets contraints*

Si nous sélectionnons à nouveau ces deux objets, nous pouvons vérifier, grâce à la fenêtre de saisie de contraintes, que leur nouvelle distance représente 20% de leur distance dans l'espace d'origine (fig. 3).

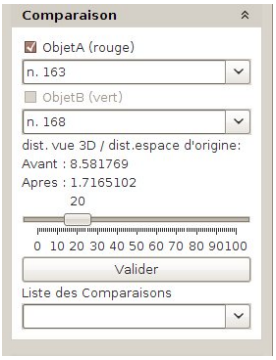

FIG. 3 – *Distance observée après recalcul*

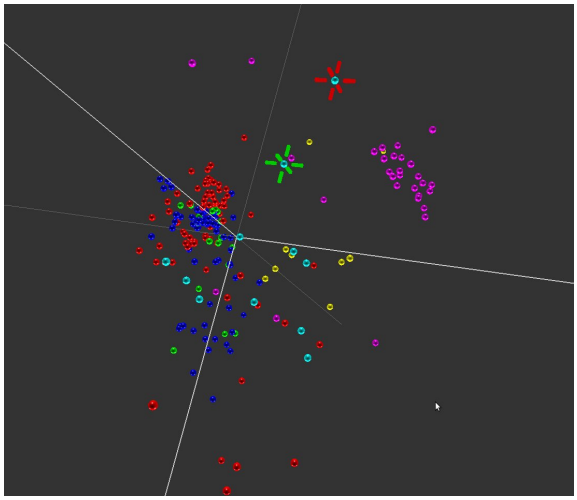

FIG. 4 – *Les objets contraints après recalcul*

4.2 Ajout de contraintes de type C3

La figure 5 représente la fenêtre de saisie de contrainte pour un triplet d'objets donné. Les trois objets sont sélectionnés dans la fenêtre 3D, et sont visualisés respectivement par des cibles rouge, verte et bleue (fig. 6). Il est possible de choisir un attribut utilisé pour l'affichage d'un label ce qui améliore l'utilisation de cette interface (ici les données représentées correspondent au jeu de données "zoo" de l'UCI repository).

Sur cette figure, l'ascenceur indique le rapport entre les distances AC et AB (seasnake-tuatara et seasnake-carp). Il est initialement de 162 (la distance d(seasnake, tuatara) vaut 162% de la distance d(seasnake, carp)). Nous choisissons de rendre "tuatara" plus proche de "seasnake" que ne l'est "carp" (le rapport est diminué, grâce à l'ascenceur, à 50).

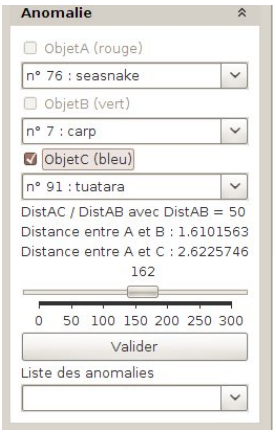

FIG. 5 – *Saisie de contraintes de type C3*

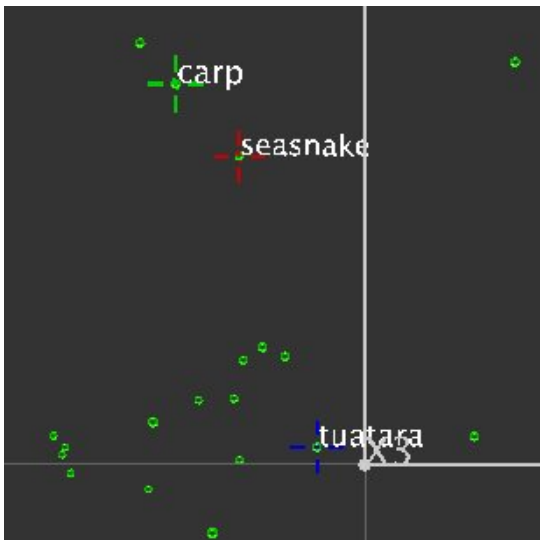

FIG. 6 – *Les objets contraints*

La figure 8 présente l'espace de projection recalculé avec application de cette seule contrainte. Nous pouvons observer que "tuatara" est désormais plus proche de "seasnake" que ne l'est "carp". Si nous sélectionnons à nouveau ces trois objets, nous pouvons vérifier, grâce à la fenêtre de saisie de contraintes, que le nouveau rapport de distance est proche de 50 (fig. 7).

5 Expérimentations

L'un des principaux objectifs de la méthode proposée ici consiste à fournir un outil interactif de manipulation graphique, dans lequel des utilisateurs peuvent poser des contraintes et observer leur impact sur la projection des objets. Néanmoins, 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.

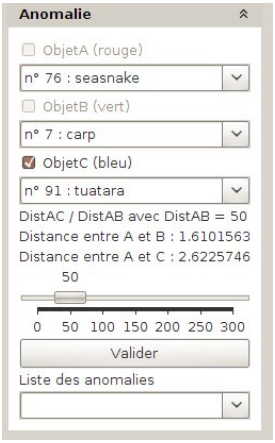

FIG. 7 – *Distance observée après recalcul*

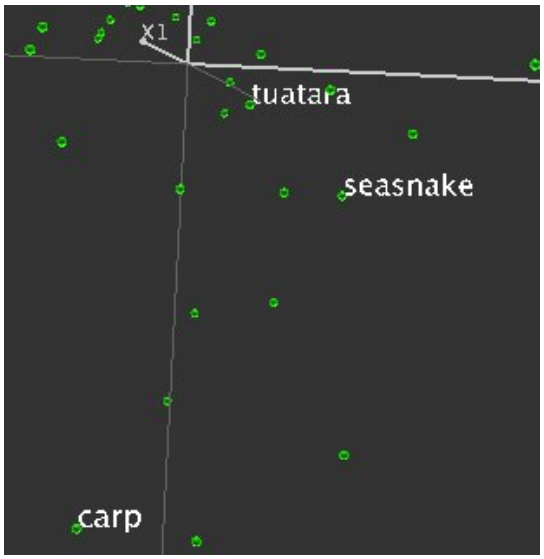

FIG. 8 – *Les objets contraints après recalcul*

Pour mesurer ce degré de satisfaction, nous proposons de comparer la représentation obtenue (en trois dimensions) 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. Or il s'agit là de la définition du 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. Naturellement, dans une utilisation réelle, non supervisée, l'utilisateur ne dispose pas, a priori, d'information de classe. Nous allons donc utiliser pour nos tests des jeux de données étiquetés. L'étiquette des objets sera utilisée afin de guider un utilisateur simulé dans ces choix. En d'autre termes, elle permettra la mise en place d'une "connaissance d'expert", et servira donc de manière indirecte. L'étiquette des objets n'intervient jamais directement dans le processus de calcul de projection.

Nous noterons d_{ald} la distance induite dans l'espace de représentation obtenu par ALD, restreint aux trois 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. 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.

5.1 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 deux 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 deux 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 trois 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 δ 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 δ 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).

Les contraintes de voisinage Cv étant des ensembles de contraintes de types $C2$, nous n'avons pas inclu ces contraintes dans nos expérimentations

5.2 Critère d'inertie sur un jeu de données artificiel

Nous considérons ici un jeu de données artificiel, où les 75 objets sont des mots décrits par 48 attributs :

- 14 attributs syntaxiques incluant : le nombre de caractères, de voyelles, de consonnes, le nombre d'occurrence des 10 lettres les plus fréquentes dans la langue française,
- 4 attributs binaires représentant la catégorie syntaxique (adjectif, nom, verbe, nom propre), certains mots peuvent appartenir à plusieurs catégories (ex. orange),
- 20 attributs sémantiques représentant des nombres d'occurrences de ces mots dans des collections de documents,
- 10 attributs de bruit, générés aléatoirement.

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, ces classes sont essentiellement induites par les 20 attributs sémantiques. Par ailleurs, l'un des objectifs est de tester si cette méthode permet facilement à l'utilisateur de mettre l'accent sur les caractéristiques importantes et de réduire l'impact des attributs de bruit.

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 trois dimensions les plus significatives (laquelle correspond à une bonne organisation visuelle).

La figure 9 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'améliorer 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.

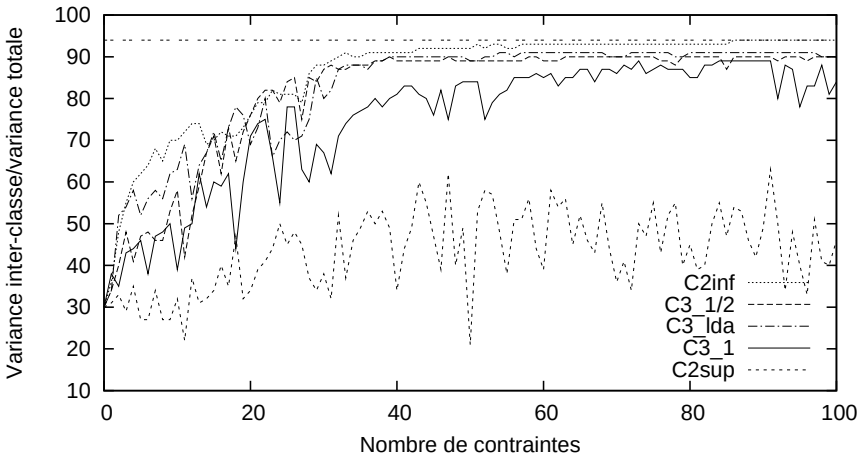

FIG. 9 – *Evolution de* $\frac{variance\ inter-classe}{variance\ totale}$ *(données synthétiques)*

5.3 Visualisation du jeu de données artificiel

La figure 10 présente les organisations visuelles obtenues avec respectivement 0, 5, 30 et 100 contraintes de type $C2_{inf}$ générées automatiquement, pour le même jeu de données que dans la section précédente. A titre de comparaison, nous présentons également la projection obtenue par ALD. Chaque classe est caractérisée par une forme et une couleur. Nous pouvons

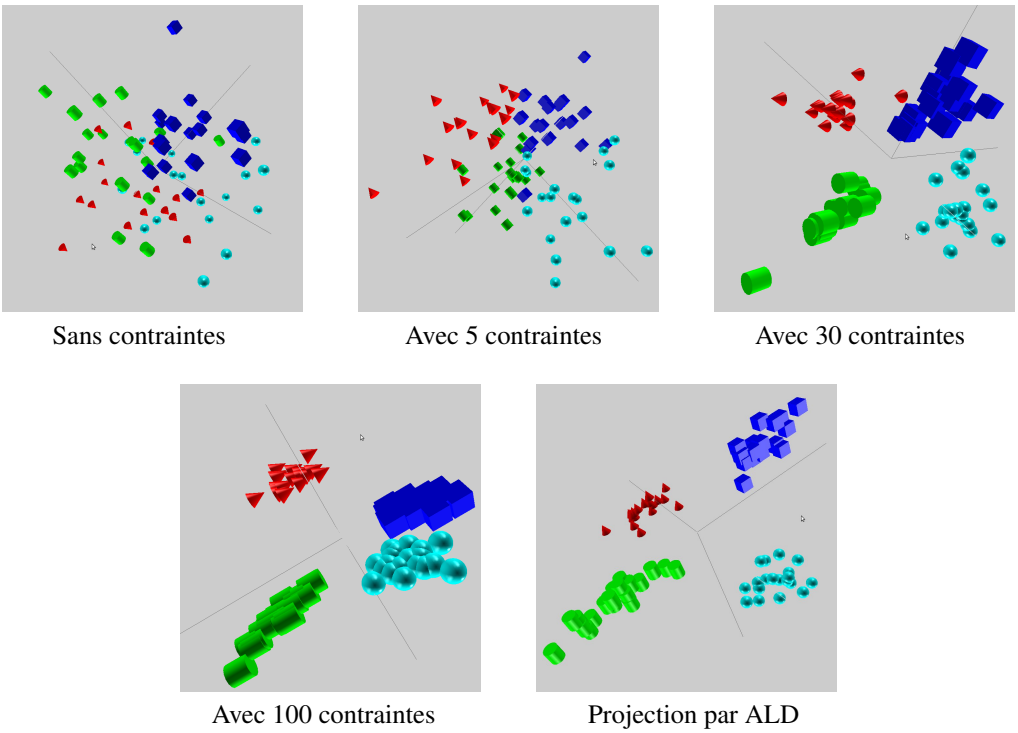

FIG. 10 – *Projections 3D obtenues avec un nombre croissant de contraintes C2inf*

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'environ une seconde dans le cas avec 100 contraintes.

5.4 Critère d'inertie sur des jeux de données usuels

Nous avons réalisés les mêmes tests que ceux que la section 5.2 sur 6 jeux de données usuels en apprentissage : breast cancer, glass, iris, wine, yeast et zoo (de l'UCI repository). La figure 11 présente l'évolution du critère Q pour les contraintes $C2_{inf}$ et confirment la tendance observée sur le jeu de données artificiel.

Afin de pouvoir présenter ces courbes sur un même graphique, les valeurs représentées sont relative 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 répartition variance inter-classe/variance intra-classe proche de celle obtenue grâce à l'ALD.

Les variations observées ici sont proches des celles obtenues sur le jeu de données artificiel. Nous pouvons remarquer que pour la plupart des jeux de données, le taux optimal est obtenu

avec un nombre très réduit de contraintes, à l'exception de "breast cancer" pour lequel une trentaine de contraintes est nécessaire.

Une anomalie semble se produire sur le jeu "glass" puisque le taux obtenu par notre approche dépasse celui obtenu par ALD alors que ce dernier est optimal, il n'est donc pas possible de le dépasser théoriquement. Une explication possible pour ce phénomène concerne la variance totale qui est très faible pour l'ALD (0.18 contre 1.89 pour notre approche). La configuration obtenue par ALD correspond donc à une très faible dispersion globale, il est très possible que la solution effectivement obtenue par ALD ne soit pas exacte : en effet, l'ALD nécessite l'inversion d'une matrice, cette opération est connue comme étant assez sensible aux variations, par exemple dues aux approximations liées aux calculs sur les flottants.

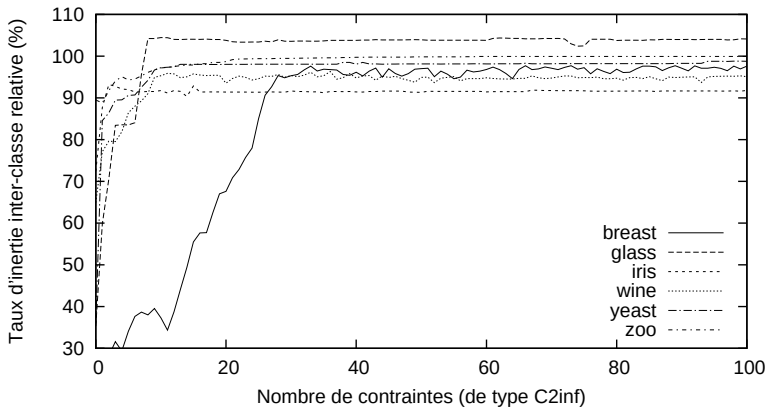

FIG. 11 – *Evolution de* $\frac{variance\ inter-classe}{variance\ totale}$ *(données UCI)*

5.5 Variation du taux de classification

Nous proposons ici un second critère consistant à mesurer le taux de classification correcte par validation croisée. 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 12 présente les 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 $C3_1$ et $C3_{1/2}$. Nous expliquons ce résultat par le fait que ces deux types de contraintes ne dépendent pas du résultat de l'ALD. Or ce dernier est assez sensible au phénomène de sur-apprentissage sur ce jeu de données (100% de classification correcte sur l'ensemble d'apprentissage, 78.5% sur l'ensemble test). De même, pour les autres types de contraintes, les mauvaises performances obtenues sont sans doute liées à l'utilisation de l'ALD. Notons que l'on retrouve des résultats analogues sur les jeux de données de l'UCI.

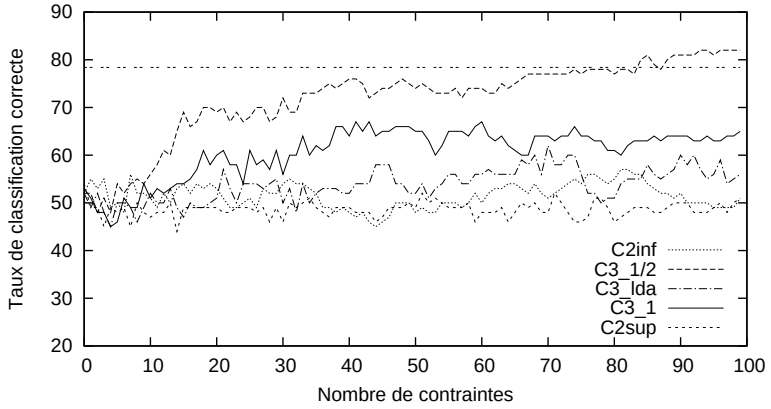

FIG. 12 – *Taux de classification correcte (données synthétiques)*

5.6 Souplesse de la méthode

Dans la simulation proposée, l'utilisateur choisit systématiquement d'agir sur la plus forte distorsion observée vis-à-vis de la projection par analyse discriminante. Afin de relativiser l'importance du choix de la distorsion, nous proposons quelques mesures complémentaires. La figure 13 présente l'évolution du rapport $\frac{variance\ inter-classe}{variance\ totale}$ dans un cadre assoupli : l'utilisateur ne choisit plus à chaque étape la distorsion la plus importante, mais la centième distorsion par ordre d'importance décroissante. Nous pouvons observer que les résultats sont très similaires à ceux de la figure 11. En conséquence, un utilisateur réel disposera d'une certaine marge de manoeuvre dans le choix des distorsions à corriger.

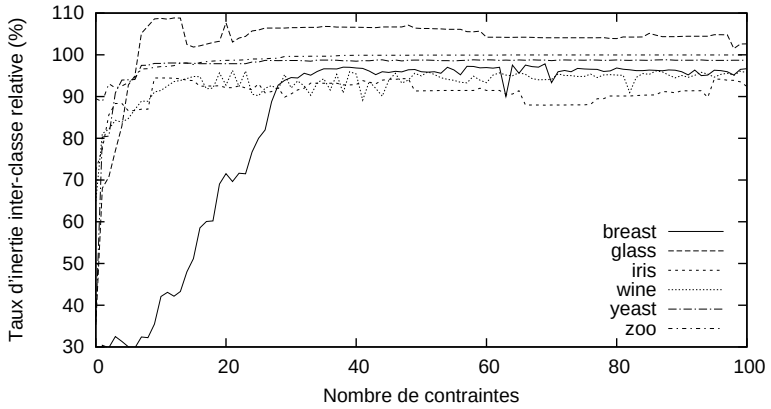

FIG. 13 – *Evolution de* $\frac{variance\ inter-classe}{variance\ totale}$ *(choix assoupli)*

5.7 Mesures complémentaires

Nous proposons ici deux mesures complémentaires portant sur les jeux de test de l'UCI, concernant l'évolution des distorsions lors de l'ajout de contraintes. La figure 14 présente l'évolution de la somme des distorsions observées. Etant donnée la diversité des jeux, nous avons retenu une échelle logarithmique en ordonnée. Dans l'ensemble, on observe que cette somme décroit rapidement puis se stabilise après un nombre de contraintes en rapport avec la somme initiale des distorsions. Cette dernière donne donc une indication du nombre de contraintes nécessaires. Il convient toutefois de souligner qu'en utilisation réelle, semi-supervisée, l'utilisateur ne dispose d'aucun moyen d'estimer automatiquement les distorsions et donc leur somme. Le principal enseignement de cette figure est qu'un choix judicieux des contraindre aura tendance à réduire les distorsions. Notons que les calculs effectués en mode assoupli (voir section précédente) conduisent à des courbes similaires.

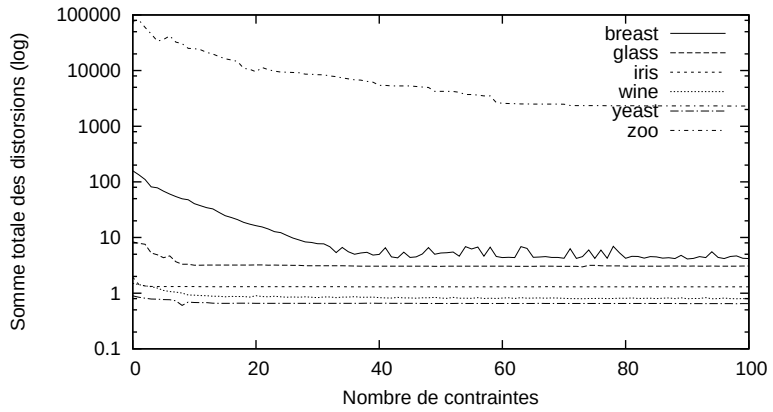

FIG. 14 – *Evolution de la somme des distorsions en fonction du nombre de contraintes*

La figure 15 montre la décroissance rapide des distorsions lors de l'ajout de contraintes. Pour cette figure, nous avons tout d'abord calculé les distorsions avant ajout de contraintes, et nous avons noté la valeur de la millième distorsion par ordre décroissant. Nous avons ensuite noté, après chaque ajout de contrainte, le nombre de distorsions présentant une valeur supérieure à ce seuil. Nous pouvons observer que dans l'ensemble ce nombre chute rapidement. Nous pouvons en déduire que l'ajout de chaque contrainte a une forte influence sur l'ensemble des distorsions. A nouveau, les calculs effectués en mode assoupli conduisent à des résultats similaires.

6 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

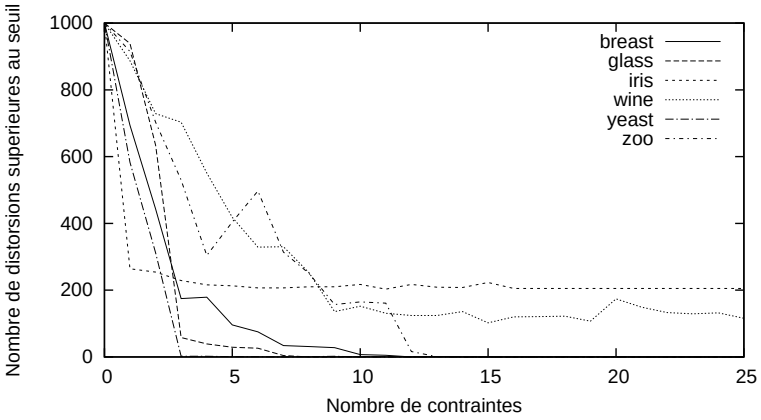

FIG. 15 – *Evolution du nombre des distorsions les plus fortes*

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.

Dans le cadre d'un tel outil, il est important que les temps de calcul restent suffisament faibles pour permettre l'interaction avec un utilisateur. Il convient donc de souligner que lors des tests menés, les temps de calcul sont toujours restés faibles, en général en deça d'une seconde, alors que nous utilisions une machine standard, de type ordinateur portable.

Dans notre approche, nous utilisons un calcul de projection sous contraintes, à partir des dimensions d'origine. Une autre méthode envisageable aurait consisté à effectuer une projection initiale, puis à ne travailler ensuite que sur la matrice de distances / dissimilarités sous-jacente, par le biais, par exemple, d'une technique de positionnement multidimensionnel comme le *least square scaling* proposé par Sammon (1977). Cela pourra faire l'objet de réflexions ultérieures.

Ces travaux s'inscrivent dans le cadre d'un projet ANR, dont l'un des objectifs est de permettre à des utilisateurs d'obtenir une organisation (visuelle) de différents styles d'écritures. Dans ce cadre, chaque objet représente une image (associée à un style d'écriture) décrite par un ensemble de caractéristiques extraites automatiquement à partir de l'image seule. Les styles d'écriture ne sont pas étiquetés, mais les paléographes sont capables de quantifier la ressemblance entre deux styles d'écriture (images). Une validation de cette approche par des experts du domaine 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.

Da Costa, D. et G. Venturini (2007). A visual and interactive data exploration method for large data sets and custering. In Springer (Ed.), *ADMA2007*, Number 4632 in LNAI, Harbin, China, pp. 553–561.

Demartines, P. et J. Hérault (1997). Curvilinear component analysis : A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transaction on Neural Networks 8*(1), 148–154.

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.

Roweis, S. T. et L. K. Saul (2000). Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science 290*(5500), 2323–2326.

Sammon, P. H. (1977). A discrete least squares method. *Mathematics of Computation 31*(137), 60–65.

Tenenbaum, J. B., V. de Silva, et J. C. Langford (2000). A global geometric framework for nonlinear dimensionality reduction. *Science 290*(5500), 2319–2323.

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 (2006). Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision 70*(1), 77–90.

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

Projecting and visualizing objects in a two- or tree-dimension space is a current data analysis task. From this visualization it might be of interest to offer to the user the ability to add knowledge in the form of (di)similarity constraints among objects, when those appear either to close or to far in the observation space. In this paper we propose three kinds of constraints and present a resolution method that derives from PCA. Experiments have been performed with both synthetic and usual datasets. They show that a representation with a high value of quality criterion can be obtained with a limited set of constraints.