Vers une approche utilisant l'apprentissage de métrique pour du clustering semi-supervisé interactif d'images

Viet Minh Vu [*] — **Hien Phuong Lai** [**] — **Muriel Visani** [**] [***]

[*] *IFI, Hanoi, Vietnam* [**] *Université de La Rochelle, Laboratoire L3i, France*
[***] *Vietnam-France ICT Lab, USTH, Hanoi, Vietnam*

RÉSUMÉ. *La problématique du clustering non supervisé et semi-supervisé est très étudiée dans le domaine de l'apprentissage automatique. En vue d'impliquer l'utilisateur dans le clustering d'images, (Lai* et al.*, 2014) a proposé un nouveau modèle de clustering semi-supervisé interactif traduisant les retours de l'utilisateur (exprimés au niveau des images) en contraintes par paires (must-link et cannot-link) entre groupes d'images constitués à l'aide d'une solution de clustering hiérarchique et de ces retours. Ces dernières années, le besoin de moyens appropriés pour mesurer la distance ou la similarité entre les données a conduit à l'émergence de l'apprentissage de métrique, qui vise à apprendre automatiquement une métrique à partir de données. Nous avons proposé une méthode intégrant un apprentissage de métrique dans le système existant de (Lai* et al.*, 2014) pour améliorer la performance et réduire le temps de calcul. Nos résultats expérimentaux, obtenus sur la base d'images Wang, montrent que les méthodes d'apprentissage de métrique s'intègrent bien dans le système existant, et améliorent ses performances et sa complexité calculatoire.*

ABSTRACT. *The problem of unsupervised and semi-supervised clustering is extensively studied in machine learning. In order to involve user in image data clustering, (Lai* et al.*, 2014) proposed a new approache for interactive semi-supervised clustering that translates the feedback of user (expressed at the level of individual images) into pairwise constraints between groups of images, these groupes being constituted thanks to the underlying hierarchical clustering solution and user feedback. Recently, the need for appropriate measures of distance or similarity between data led to the emergence of distance metric learning approaches. In this paper, we propose a method incorporating metric learning in the existing system of (Lai* et al.*, 2014) to improve performance and reduce the computational time. Our experimental results obtained on the image database Wang show that the metric learning methods fit well into the existing system improve its performances as well as its computational time.*

MOTS-CLÉS : *clustering semi-supervisé interactif, contraintes par paires, apprentissage de métrique, analyse d'images.*

KEYWORDS: *interactive semi-supervised clustering, pairwise constraints, metric learning, image analysis.*

1. Introduction

Ce travail se place dans le domaine de l'analyse d'images par le contenu, et plus précisément du *clustering* semi-supervisé interactif d'images en vue de l'utilisation d'outils de navigation dans des bases d'images ou de recherche, par exemple. (Lai *et al.*, 2014) a proposé une nouvelle méthode de *clustering* semi-supervisé interactif dans le but de combler le fossé sémantique entre les concepts de haut niveau perçus par l'utilisateur dans la collection d'images, et les signatures de bas niveau extraites à partir des images originales. Dans un contexte interactif incrémental, sa méthode implique l'utilisateur dans la phase de *clustering* pour qu'il puisse interagir avec le système afin d'améliorer les résultats fournis par le modèle de *clustering* semi-supervisé automatique. Son système convertit en contraintes entre paires de groupes d'images les informations supervisées fournies par l'utilisateur et procède itérativement au *re-clustering* semi-supervisé en pénalisant ces contraintes. Tout d'abord, son système construit un modèle de *clustering* non supervisé hiérarchique grâce à l'algorithme BIRCH pour représenter des images d'entrée dans une structure de clustering hiérarchique où les images similaires sont automatiquement regroupées dans des *clustering* compacts et représentatifs de la collection d'images. Ensuite, les résultats de ce modèle de *clustering* non supervisé sont présentés de façon visuelle à l'utilisateur pour qu'il puisse donner ses retours *via* des clics positifs et négatifs sur les images affichées ou *via* des glisser-déposer d'une image d'un cluster vers un autre. Six stratégies de déduction des contraintes entre paires de groupes d'images à partir des retours de l'utilisateur sont étudiées et expérimentées. En tenant compte des contraintes par paires générées par ce moteur de déduction, le système réorganise la structure hiérarchique des données et refait le *clustering* en utilisant une méthode de *clustering* semi-supervisé. La boucle d'interaction peut être répétée jusqu'à ce que l'utilisateur soit suffisamment satisfait du résultat.

Les mesures de similarité et de distance entre des observations jouent un rôle important dans les processus cognitifs humains et les systèmes artificiels pour la reconnaissance et la catégorisation. La question de comment mesurer de manière appropriée la distance ou la similarité est cruciale pour la performance de nombreuses méthodes d'apprentissage et de fouille de données. La tâche principale dans tous les algorithmes de *clustering* est de déterminer à quel cluster appartient une observation, c'est-à-dire que l'on a besoin d'une mesure de similarité / dissimilarité entre des points dans un ensemble de données. La distance Euclidienne est une mesure de dissimilarité qui est largement utilisée dans les tâches de *clustering*. Mais cette distance géométrique n'est pas toujours parfaite, par exemple lorsque les groupes de données sont non sphériques ou hétérogènes dans l'espace de données. Lorsque l'on travaille avec des données multidimensionnelles, la distance Euclidienne est isotrope, or, dans quelques situations, on doit considérer quelques directions en priorité, on a donc besoin d'une métrique paramétrable. L'apprentissage de la distance de Mahalanobis à partir de la distribution des observations dans l'espace des données est une solution intéressante. L'idée principale des algorithmes d'apprentissage de métrique est d'apprendre un ensemble de paramètres qui contrôle une fonction de distance particulière et, le cas échéant, de

mettre à jour incrémentalement ces paramètres en fonction de nouvelles informations. Cette idée est compatible avec le système interactif incrémental où les nouvelles informations supervisées (sous forme de retours de l'utilisateur) sont fournies à chaque itération interactive et sont utilisées pour entraîner la métrique pour rendre le résultat du modèle de *clustering* plus satisfaisant pour l'utilisateur.

Dans ce travail, nous avons proposé et expérimenté plusieurs méthodes intégrant l'apprentissage d'une distance de Mahalanobis dans le système de *clustering* semi-supervisé interactif existant de (Lai *et al.*, 2014) pour en améliorer la performance et en réduire le temps de calcul. Ces méthodes sont présentées dans la section 2. L'analyse des résultats expérimentaux et l'approche que nous privilégions dans notre contexte applicatif sont présentées dans la section 3. La section 4 conclut cet article.

2. Contexte scientifique

En général, les systèmes d'apprentissage automatique (sauf quelques systèmes d'apprentissage profond) utilisent souvent des caractéristiques de bas niveau qui sont extraites à partir des images originales *via* des algorithmes de détection de bord (*edge detection : Canny, Sobel, Prewitt, ...* - (Umbaugh, 2010)), de coin (*Corner detection : Harris, SUSAN, ...* - (Willis et Sui, 2009)), de *blob* (*blob detection : Laplacian of Gaussian (LoG), Difference of Gaussians (DoG), Determinant of Hessian (DoH), ...*), SIFT (*Scale-invariant feature transform* - (Lowe, 1999)), ...

Mais quand un utilisateur non-expert observe les résultats fournis par ces systèmes, sa perception reflète généralement des concepts sémantiques de haut niveau, comme quelles images se ressemblent ou se distinguent. On a donc un fossé sémantique entre la perception de l'utilisateur et le résultat réel fourni par le système d'apprentissage.

2.1. *Clustering semi-supervisé interactif*

La méthode de clustering *semi-supervisé interactif de (Lai* et al.*, 2014) :*

Les étapes de cette méthode sont comme suit : Les descripteurs *rgSIFT* sont extraits à partir des images originales et sont regroupés par l'algorithme K-Means pour créer un dictionnaire de mots visuels. Chaque image originale est représentée par un vecteur de fréquence des mots visuels dans le dictionnaire. L'algorithme de *clustering* non supervisé BIRCH est utilisé pour faire le *clustering* initial sur ces vecteurs, où un arbre est construit incrémentalement à partir des données initiales et ses feuilles (CF-Entrées) sont regroupées par un algorithme de *clustering* à plat (par défaut K-Means). Dans l'étape de re*clustering* interactif, on va travailler sur l'ensemble des CF-Entrées dans des nœuds feuilles de cet arbre. À chaque itération interactive, l'utilisateur visualise le résultat du *clustering* et corrige les erreurs du système *via* des clics positifs et négatifs sur les images présentées. Il peut aussi déplacer les images entre les clusters. Ces retours sont interprétés par un moteur de déduction des contraintes qui crée plusieurs nouvelles contraintes entre paires de feuilles de l'arbre CF-Tree pour modifier

la solution de *clustering* de la façon la plus proche de la satisfaction de l'utilisateur que possible. En conséquence, l'arbre CF-Tree est éventuellement modifié et le nouvel ensemble de CF-Entrées de chaque itération interactive est traité par HMRF-KMeans.

Modèle d'interaction :

L'utilisateur va intervenir à chaque itération interactive. Le système fait le *clustering* et présente le résultat dans une interface interactive (voir Figure 1). Dans le plan principal (obtenu par ACP) on représente les clusters par leurs images prototypes qui sont les images les plus représentatives de chaque cluster selon un critère choisi, par exemple le *'Silhouette Width'*. En cliquant sur une image prototype dans le plan principal, l'utilisateur peut voir plus d'informations détaillées sur le cluster correspondant : son image prototype, les 10 images les plus représentatives et les 10 images les moins représentatives qui n'ont pas encore reçu de retour. L'utilisateur peut spécifier des retours positifs (pertinents) et négatifs (non pertinents) ou déplacer une image d'un cluster vers un autre cluster. Quand une image est déplacée du cluster A vers le cluster B, elle est considérée comme un retour négatif pour le cluster A et comme un retour positif pour le cluster B. Afin de pouvoir comparer effectivement les résultats de ce système vis-à-vis de ceux des systèmes existants, un agent est utilisé pour simuler des comportements des utilisateurs quand ils donnent des retours au système. Cet agent agit comme un oracle, c'est-à-dire qu'il donne toujours la vérité terrain associée à une base annotée préalablement par un humain.

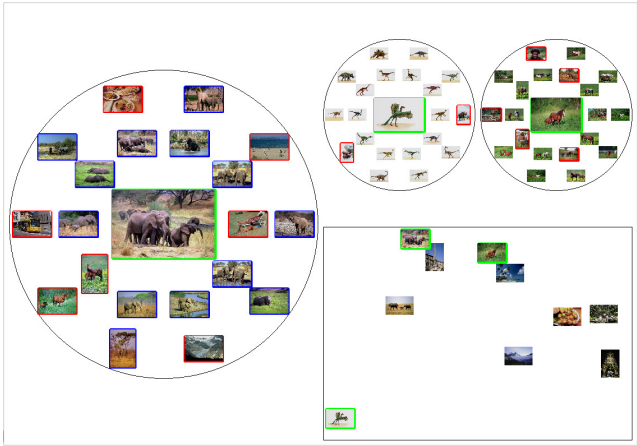


Figure 1 – L'interface interactive du système de clustering semi-supervisé interactif existant

Stratégies de déduction des contraintes entre pairs de CF-Entrées :

Dans chaque itération interactive, pour chaque cluster avec lequel l'utilisateur interagit, le système reçoit les retours sous la forme de listes d'images positives et négatives. Selon ces retours, toutes les images positives doivent rester dans leur cluster, pendant que les images négatives doivent se déplacer vers un autre cluster. Par consé-

quent, dans chaque cluster on considère que des contraintes MustLink existent entre chaque paire d'images positives, et des contraintes CannotLink existent entre chaque image négative et chaque image positive de ce cluster. À l'issue d'une itération interactive, il y a peut-être des CannotLinks entre les images d'une même CF-Entrée, ou il existe simultanément des MustLinks et CannotLinks entre les images de deux CF-Entrées CF_i et CF_j. Dans ces cas, ces CF-Entrées doivent être divisées en plusieurs CF-Entrées plus pures. Pour cela, l'algorithme s'appuie sur des regroupements intermédiaires des données : les noyaux, constitués des ensembles d'images appartenant à une même CF entrée qui sont liées par des contraintes MustLink. Puis l'algorithme de *clustering* semi-supervisé HMRF-KMeans est utilisé avec des contraintes entre CF-Entrées à la place de contraintes entre images (pour des raisons d'efficacité en termes de performances comme de temps de calcul). Une contrainte MustLink/CannotLinks est créée entre deux CF-Entrées CF_i et CF_j s'il existe au moins une contrainte MustLink/CannotLinks entre les images de CF_i et celles de CF_j. Pour plus de détails, merci de se référer à (Lai *et al.*, 2014).

2.2. *Apprentissage de métrique*

Le besoin de moyens appropriés pour mesurer la distance ou la similarité entre les données est omniprésent dans l'apprentissage automatique, la reconnaissance des formes et l'exploration de données, mais les bonnes mesures pour des problèmes spécifiques sont généralement difficiles à trouver. Cela a conduit à l'émergence de l'apprentissage de métrique, qui vise à apprendre automatiquement une métrique à partir de données et a attiré beaucoup d'intérêt dans le domaine de l'apprentissage, et en particulier de la classification et du *clustering*.

2.2.1. *Distance de Mahalanobis*

La distance de Mahalanobis permet de calculer la distance entre deux points dans un espace à d dimensions, en tenant compte de la covariance de ces d variables. En pratique, la distance de Mahalanobis entre un vecteur de plusieurs variables $x = (x_1, x_2, x_3, \ldots, x_d)$ et un ensemble de vecteurs de valeurs moyennes $\mu = (\mu_1, \mu_2, \mu_3, \ldots, \mu_d)$ est paramétrée par une matrice de covariance *définie positive* Σ comme suit :

$$D_\Sigma(x, \mu) = \sqrt{(x - \mu)^T \Sigma^{-1} (x - \mu)}$$

Le but principal de la plupart des méthodes d'apprentissage de métrique basées sur une distance de Mahalanobis est d'apprendre cette matrice de covariance Σ.

2.2.2. *Différents types d'approches d'apprentissage de métrique*

Dans le contexte de l'apprentissage semi-supervisé (dans notre contexte, c'est le *clustering* semi-supervisé interactif), les informations supervisées sont organisées sous la forme de contraintes par paires qui comprennent un ensemble de MustLinks

$\mathcal{M} = \{(x_i, x_j)\}$ où x_i, x_j devraient être similaires, et un ensemble de CannotLinks $\mathcal{C} = \{(x_i, x_j)\}$ où x_i, x_j devraient être dissimilaires.

Un algorithme d'apprentissage de métrique vise essentiellement à trouver les paramètres de la métrique qui satisfont le mieux ces contraintes. Cela est généralement formulé sous la forme du problème d'optimisation suivant :

$$\min_{\Sigma} \mathcal{J}_{obj}(\Sigma, \mathcal{M}, \mathcal{C}) + \lambda \mathcal{R}(\Sigma)$$

où $\mathcal{J}_{obj}(\Sigma, \mathcal{M}, \mathcal{C})$ est la fonction objectif (ou bien la fonction de coût) par rapport à la matrice de covariance Σ, et l'ensemble des contraintes $\mathcal{M}, \mathcal{C}$, qui est interprétée comme la pénalité des contraintes violées. $\mathcal{R}(\Sigma)$ est un terme de régularisation, pondéré par un paramètre $\lambda \geq 0$.

Approches globales

Comme on l'a vu avec la forme générale du problème d'optimisation ci-dessus, mathématiquement, l'apprentissage de métrique se concentre sur des mesures linéaires parce qu'elles sont plus faciles à optimiser (en particulier, il est plus facile de tirer des formulations convexes avec la garantie de trouver l'optimum global) et moins sujettes au sur-apprentissage. MMC (*Mahalanobis Metric Learning for Clustering with Side Information*), (Xing *et al.*, 2002) est la première approche d'apprentissage de la distance de Mahalanobis, qui vise à maximiser la somme de distances entre les points dissemblables tout en conservant la somme des distances entre les points similaires. LMNN (*Large Margin Nearest Neighbors*), (Weinberger *et al.*, 2005) est aussi une des méthodes les plus utilisées, avec une définition des contraintes très simple : pour chaque exemple d'entraînement, les k plus proches voisins de ce point devraient appartenir à une seule classe (les *"target neighbors"*), et les autres voisins de classes différentes devraient être repoussés (les *"imposteurs"*). La distance Euclidienne est utilisée pour déterminer les *"target neighbors"*.

Dans certains cas, il existe des structures non linéaires dans les données que les mesures linéaires sont incapables de capturer. Les méthodes non linéaires globales apprennent une distance de la forme $D(x_i, x_j) = \| f(x_i) - f(x_j) \|_2$ pour une certaine fonction f. En général, l'apprentissage d'une transformation non linéaire est difficile. Contrairement à la transformation linéaire qui peut être exprimée comme une matrice de paramètres, la transformation non linéaire n'est pas facilement paramétrée. Afin d'apprendre de telles transformations, il est nécessaire de limiter la forme du *mapping* non linéaire f à une classe paramétrée particulière. Les méthodes basées sur des noyaux (de kernelization) sont des méthodes ayant un *mapping* efficace. Plusieurs auteurs ont proposé des méthodes de kernelization basées sur l'Analyse en Composantes Principales avec noyau (KPCA - *Kernel Principal Component Analysis* - (Schölkopf *et al.*, 1998)), une extension non linéaire de l'ACP. En bref, KPCA projette implicitement les données dans un espace non linéaire induit par un noyau, puis exécute une réduction de dimensionnalité dans cet espace. Un algorithme d'apprentissage de métrique peut ensuite être utilisé pour apprendre une métrique dans cet espace non linéaire. (Chatpatanasiri *et al.*, 2010) ont montré que le KPCA est théoriquement robuste pour les algorithmes d'apprentissage de métrique sans contrainte.

Une autre astuce possible (impliquant un prétraitement non linéaire de l'espace de caractéristiques) est basé sur l'estimation de la densité du noyau et permet de traiter des attributs à la fois numériques et catégoriels (He *et al.*, 2013). (Wang *et al.*, 2011) abordent le problème du choix d'une fonction de noyau approprié en utilisant de multiples noyaux pour l'apprentissage de métrique.

Approches locales

Les méthodes listées ci-dessus apprennent des métriques globales (linéaires ou non linéaires). Cependant, si les données sont hétérogènes, une seule métrique peut ne pas bien représenter la complexité de la tâche et il pourrait être avantageux d'utiliser plusieurs mesures locales (par exemple, une pour chaque classe ou chaque cluster). Il a été montré que l'apprentissage de métrique locale surpasse de manière significative les méthodes globales sur certains problèmes, mais a besoin généralement de plus d'exemples annotés, de plus de temps et de mémoire. L'algorithme MPCKMeans (*Metric with Pairwise Constraints KMeans*) de (Bilenko *et al.*, 2004) est capable d'apprendre des mesures individuelles pour chaque groupe, ce qui permet d'avoir des clusters de formes différentes. Nous avons choisi cette méthode pour intégrer dans le système existant, car elle a plusieurs points communs avec l'algorithme HMRF-KMeans : elle aussi est basée sur l'algorithme EM et utilise des contraintes par paires (MustLinks et CannotLinks).

3. Méthode proposée et évaluation expérimentale

3.1. *Présentation de la méthode*

La méthode de (Lai *et al.*, 2014) (appelée désormais la méthode *Baseline*) est une adaptation de l'algorithme HMRF-KMeans (Basu *et al.*, 2004) sur des CF-Entrées obtenues par BIRCH (Zhang *et al.*, 1996) dans un contexte interactif. La distance Euclidienne est utilisée partout (dans la construction, dans la division et dans le *clustering* interactif incrémental). Or, comme nous l'avons vu précédemment, la distance Euclidienne présente plusieurs désavantages, et nous avons donc besoin d'une nouvelle métrique qui peut mieux répondre à l'exigence de l'utilisateur dans le contexte interactif et accélérer la convergence. Nous avons choisi l'algorithme d'apprentissage de métrique MPCKMeans (Metric with Pairwise Constraints KMeans) de (Bilenko *et al.*, 2004) pour remplacer l'algorithme HMRF-KMeans dans l'étape de re*clustering* interactif. La nouvelle méthode que nous proposons diffère donc de celle de (Lai *et al.*, 2014) en deux points : (1) Remplacement de l'algorithme HMRF-KMeans qui fait le *clustering* de façon interactive par MPCKMeans pour apprendre une (ou des) distance(s) de Mahalanobis plus adaptée(s) aux données. (2) Utilisation de la distance de Mahalanobis dans l'étape de construction et de division de l'arbre.

L'algorithme MPCKMeans s'adapte dans le système existant va recevoir l'ensemble des CF-Entrées comme données d'entrée. On utilise les notations : L'ensemble des CF-Entrées des feuilles de l'arbre CF-Tree $\mathcal{X} = (CF_1, ..., CF_m)$ va être regroupé selon des informations supervisées sous forme des MustLinks et des CannotLinks

entre paires de CF-Entrées : $\mathcal{M} = \{(CF_i, CF_j)\}$, $\mathcal{C} = \{(CF_i, CF_j)\}$. On utilise aussi deux constantes $\omega, \overline{\omega}$ pour pondérer les pénalités de violation des contraintes MustLinks et CannotLinks. En pratique, on utilise $\omega_{i,j} = \omega$ pour toutes les paires de contraintes MustLinks violées (CF_i, CF_j), et $\overline{\omega_{i,j}} = \overline{\omega}$ pour toutes les paires de contraintes CannotLinks violées (CF_i, CF_j).

Au niveau de chaque cluster h, on va noter son ensemble de points $\mathcal{X}_h$. On doit souvent calculer la distance entre deux points de données quelconques (CF_i, CF_j) ou bien la distance entre un point quelconque CF_i et un cluster h selon la métrique de chaque cluster qui est paramétré par la moyenne μ_h et la matrice de covariance Σ_h : $\Sigma_h = \frac{1}{|\mathcal{X}_h|}(\mathcal{X}_h - \mu_h)(\mathcal{X}_h - \mu_h)^T$. La distance de Mahalanobis est donc calculé en se basant sur l'inverse de la matrice de covariance $A_h = {\Sigma_h}^{-1}$:

$$\begin{aligned} D^2_{A_h}(CF_i, \mu_h) &= (CF_i - \mu_h)^T A_h (CF_i - \mu_h) \\ D^2_{A_h}(CF_i, CF_j) &= (CF_i - CF_j)^T A_h (CF_i - CF_j) \end{aligned} \quad [1]$$

On note l_i l'étiquette du cluster de la CF-Entrée CF_i. Le résultat attendu est K sous-ensembles disjoints $\{\mathcal{X}_h\}_{h=1}^K$ qui sont créés progressivement de façon que la fonction objectif $\mathcal{J}_{obj}$ soit minimisée localement (l'équation 2).

$$\begin{aligned} \mathcal{J}_{obj} = &\sum_{CF_i \in \mathcal{X}} (D^2_{A_{l_i}}(CF_i, \mu_{l_i}) - log(det(A_{l_i}))) \\ &+ \sum_{(CF_i, CF_j) \in \mathcal{M}} \omega_{i,j} f_M(CF_i, CF_j) \mathbb{1}[h \neq l_j] + \sum_{(CF_i, CF_j) \in \mathcal{C}} \overline{\omega_{i,j}} f_C(CF_i, CF_j) \mathbb{1}[h = l_j] \end{aligned} \quad [2]$$

Le premier terme mesure la distorsion entre un point de données (une CF-Entrée) CF_i et le centre de son cluster correspondant μ_{l_i}, suivi par un terme de logarithme de la vraisemblance qui est une constante de la normalisation d'une gaussienne avec la matrice de covariance ${A_{l_i}}^{-1}$. Les deuxième et troisième termes représentent la pénalité de la violation des contraintes par paires entre les CF-Entrées CF_i et CF_j, et sont formulés par les deux équations 3 et 4. La pénalité des contraintes MustLinks violées entre deux points qui sont proches en fonction d'une métrique devrait être plus élevée que celle de deux points éloignés. Et la pénalité des contraintes CannotLinks violées entre des points éloignés doit être supérieure à celle entre des points proches. Dans l'équation de la pénalité de CannotLink, (CF'_{l_i}, CF''_{l_i}) est la paire des deux points les plus éloignés (selon la métrique apprise) du cluster l_i. L'algorithme MPCKMeans adapté dans notre contexte est détaillé dans le pseudo-code 1.

$$f_M(CF_i, CF_j) = \frac{1}{2} D^2_{A_{l_i}}(CF_i, CF_j) + \frac{1}{2} D^2_{A_{l_j}}(CF_i, CF_j) \quad [3]$$

$$f_C(CF_i, CF_j) = D^2_{A_{l_i}}(CF'_{l_i}, CF''_{l_i}) - D^2_{A_{l_i}}(CF_i, CF_j) \quad [4]$$

Algorithme 1 : L'algorithme MPCKMeans appliqué sur l'ensemble des CF-Entrées comme données d'entrée

1. Initialisation (à l'itération $t = 0$)

1) Créer λ voisinages $\{N_p\}_{p=1}^{\lambda}$ à partir de l'ensemble des contraintes $\mathcal{M}$ et $\mathcal{C}$ entre paires de CF-Entrées

2) Initialiser les clusters :

si $\lambda \geq K$ **alors**

Initialiser $\{\mu_h^{(0)}\}_{h=1}^{K}$ en utilisant l'algorithme *Weighted Farthest-First Traversal*, en commençant à partir du voisinage le plus nombreux.

sinon si $\lambda < K$ **alors**

Initialiser $\{\mu_h^{(0)}\}_{h=1}^{\lambda}$ par les centroids des voisinages créés $\{N_p\}_{p=1}^{\lambda}$.
Pour les $K - \lambda$ clusters restants, initialiser au hasard.

2. La boucle itérative de l'algorithme *EM*

tant que *pas encore converge, à l'itération* t **faire**

1) $assign\_cluster(E\_Step)$: Assigner chaque point de données CF_i à un cluster h^* qui satisfait l'équation 5 :

$$\begin{aligned} h^* = argmin(D^2_{A_{l_i}}(CF_i, \mu_{l_i}) - log(det(A_h)) \\ + \sum_{(CF_i,CF_j)\in\mathcal{M}} \omega_{i,j} f_M(CF_i, CF_j)\mathbb{1}[h \neq l_j] \\ + \sum_{(CF_i,CF_j)\in\mathcal{C}} \overline{\omega_{i,j}} f_C(CF_i, CF_j)\mathbb{1}[h = l_j]) \end{aligned} \quad [5]$$

2) $estimate\_means(M\_Step\_A)$:

$$\{\mu_h^{(t+1)}\}_{h=1}^{K} \leftarrow \{\tfrac{1}{|\mathcal{X}_h^{(t+1)}|} \textstyle\sum_{CF_i\in\mathcal{X}_h^{(t+1)}} CF_i\}_{h=1}^{K} \quad [6]$$

3) $update\_metrics(M\_Step\_B)$:

$$\begin{aligned} A_h = |\mathcal{X}_h|(\sum_{CF_i\in\mathcal{X}_h} (CF_i - \mu_h)(CF_i - \mu_h)^T \\ + \sum_{(CF_i,CF_j)\in\mathcal{M}} \frac{1}{2}\omega_{i,j}(CF_i - CF_j)(CF_i - CF_j)^T\mathbb{1}[h \neq l_j] \\ + \sum_{(CF_i,CF_j)\in\mathcal{C}} \overline{\omega_{i,j}}((CF'_h - CF''_h)(CF'_h - CF''_h)^T - (CF_i - CF_j)(CF_i - CF_j)^T)\mathbb{1}[h = l_j])^{-1} \end{aligned} \quad [7]$$

4) $t \leftarrow (t + 1)$

fin

3.2. *Protocole expérimental*

Toutes les expérimentations sont réalisées sur la base d'images Wang qui contient 1000 images de 10 classes différentes. Chaque expérimentation est lancée 5 fois sous la même configuration, et les mesures sont calculées par la moyenne de ces 5 résultats. La machine utilisée pour toutes les expérimentations est un PC avec *Ubuntu 15.04, 64 bit, CPU Intel 2.4GHz, RAM 4GB, g++ 4.9.2*.

Les représentations de ces images sous la forme des vecteurs de fréquence de 200 mots visuels sont utilisées pour entraîner le modèle de *clustering* initial. Il va fournir l'arbre CF-Tree dont les CF-Entrées de ses nœuds feuilles sont ensuite des données d'entrée pour l'étape de *reclustering* interactif incrémental.

Le tableau 1 montre toutes les méthodes utilisées pour faire l'expérimentation. Les méthodes d'apprentissage de métrique qui utilisent la distance de Mahalanobis dépendent des paramètres de forme des matrices de covariance. On peut apprendre plusieurs matrices de covariance pour chaque cluster séparément (l'approche locale : *MPCKMEANS_LOCAL*), ou une seule matrice de covariance pour tout l'ensemble de données (l'approche globale : *MPCKMEANS_GLOBAL*). La forme de la matrice de covariance peut également différer : forme diagonale (*MPCKMEANS_DIAGONAL*) ou forme complète (*MPCKMEANS_FULL*). On combine ces configurations pour fournir 4 méthodes différentes : *MPCKMEANS_LOCAL_DIAGONAL, MPCKMEANS_LOCAL_FULL, MPCKMEANS_GLOBAL_DIAGONAL, MPCKMEANS_GLOBAL_FULL.*

<table>
<tr><td rowspan="6">La base Wang</td><td colspan="2">Méthode Baseline</td></tr>
<tr><td rowspan="4">L'apprentissage de métrique combiné avec une simple distance Euclidienne pour la construction et la division de l'arbre CF-Tree</td><td>MPCKMEANS_GLOBAL_DIAGONAL (distE)</td></tr>
<tr><td>MPCKMEANS_GLOBAL_FULL (distE)</td></tr>
<tr><td>MPCKMEANS_LOCAL_DIAGONAL (distE)</td></tr>
<tr><td>MPCKMEANS_LOCAL_FULL (distE)</td></tr>
<tr><td rowspan="2">L'apprentissage de métrique combiné avec la distance de Mahalanobis apprise pour la construction et la division de l'arbre CF-Tree</td><td>MPCKMEANS_GLOBAL_DIAGONAL (distM)</td></tr>
<tr><td>MPCKMEANS_LOCAL_DIAGONAL (distM)</td></tr>
</table>

Tableau 1 – Les méthodes utilisées pour l'expérimentation sur la base Wang

3.3. *Résultats expérimentaux*

Notre système est expérimenté dans un contexte interactif incrémental avec l'intervention de l'agent utilisateur (oracle). Chaque fois que le modèle de *clustering* fournit un résultat, il est présenté à l'utilisateur et le système utilise ses retours pour adapter le modèle de *clustering*. Le système continue à refaire l'étape de *clustering* pour se rapprocher des besoins de l'utilisateur.

On n'obtient pas de résultat pour la méthode MPCKMEANS_LOCAL_FULL à cause des matrices de covariances mal conditionnées (pas assez de données à l'inté-

rieur de chacun des clusters). On peut surmonter ce problème dans la méthode MPCKMEANS_LOCAL_DIAGONAL en bénéficiant d'un terme de *régularisation* pour les matrices de covariance diagonale. Pour plusieurs stratégies de la méthode MPCKMEANS_LOCAL_DIAGONAL, on n'obtient pas de bons résultats à cause de la convergence vers une valeur locale minimale. De plus, les temps de calcul pour la méthode MPCKMEANS_GLOBAL_FULL sont très importants (jusqu'à 4 fois supérieurs à ceux de l'apprentissage de distance avec matrice de covariance diagonale. Cela est peu adapté à notre contexte applicatif (en-ligne avec l'utilisateur). Nous focalisons donc les résultats présentés en Figure 2 (avec la distance Euclidienne pour la construction et la division de l'arbre) et en Figure 3 (avec la distance de Mahalanobis apprise pour la construction et la division de l'arbre) sur l'apprentissage d'une unique distance (globale) basée sur une matrice de covariance diagonale.

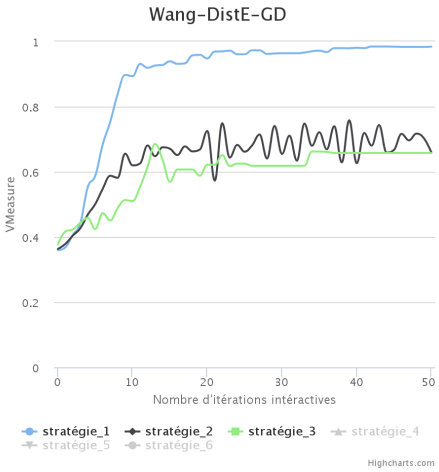
Résultats détaillés pour la méthode MPCKMEANS_GLOBAL_DIAGONAL (distE) : (voir la figure 2)

Cette méthode utilise la distance Euclidienne pour la construction et la division de l'arbre CF-Tree. Dans les figures suivantes, l'axe vertical représente la performance mesurée par la VMesure $\in [0.0, 1.0]$; l'axe horizontal représente le nombre d'itérations interactives. Le temps d'exécution des méthodes est affiché sous le format $heure : minute : seconde$

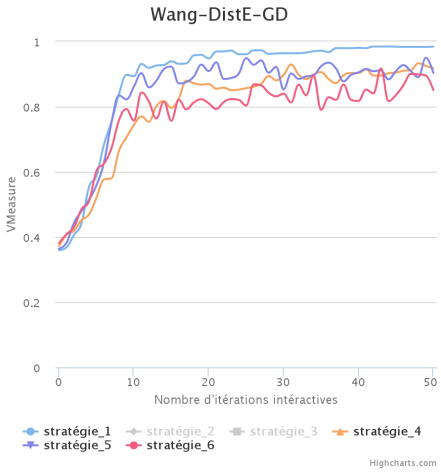
On trouve que la stratégie 1 qui déduit toutes les contraintes possibles fournit le meilleur résultat. Le seul désavantage de cette stratégie est le temps d'exécution. Les stratégies 2 et 3 possèdent un temps d'exécution acceptable mais elles ne donnent pas un bon résultat. Par contre, les stratégies 5 et 6 qui utilisent des contraintes sélectionnées donnent de meilleurs résultats que les deux stratégies précédentes. En comparaison avec la stratégie 5, la stratégie 6 optimise la déduction de contraintes et économise du temps d'exécution mais son résultat est toujours moins bon. La stratégie 4, quant à elle, donne un résultat intéressant. Elle prend seulement environ 37 secondes pour exécuter toutes les 50 boucles de re*clustering* incrémental. Le résultat de cette stratégie au début (en présence de peu de contraintes) n'est pas très bon, mais on observe sa tendance d'augmentation après chaque itération interactive, pour obtenir des résultats comparables à ceux de la stratégie 5 au bout d'environ 30 itérations.

Analyse des temps d'exécution :

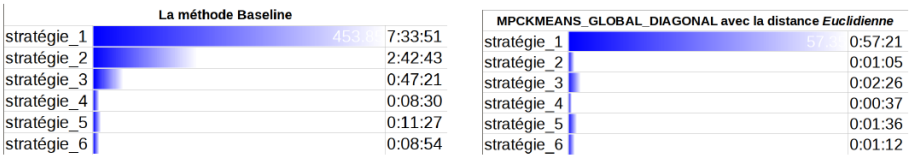
Un résultat qui peut paraître surprenant de prime abord concerne les temps d'exécution, qui sont inférieurs pour la méthode basése sur un apprentissage de métrique par rapport à la méthode Baseline. Cela peut s'expliquer comme suit : le temps pour générer et déduire des contraintes occupe la plupart du temps d'exécution dans chaque itération. Dans le système existant, quand on a une CF-Entrée incohérente qui contient en même temps un MustLink et un CannotLink entre leurs images, il faut la diviser en plusieurs CF-Entrées cohérentes. Ça donne plus de travail, autrement dit, plus de temps d'exécution pour le moteur de déduction des contraintes. Quand on applique l'apprentissage de métrique, les images dans les CF-Entrées sont plus cohérentes car il y a moins de violations au niveau des contraintes entre images. En conséquence, on



(a) Stratégies 1, 2, 3



(b) Stratégies 1, 4, 5, 6



(c) Le temps d'exécution de la méthode Baseline et MPCKMEANS_GLOBAL_DIAGONAL

Figure 2 – MPCKMEANS_GLOBAL_DIAGONAL (distE) : L'apprentissage de métrique avec une matrice de covariance globale diagonale, en utilisant la distance Euclidienne pour la construction et la division de l'arbre CF-Tree

a moins de nouvelles CF-Entrées créées après chaque itération. Ça implique qu'on a moins de contraintes générées, et donc que l'on peut économiser pas mal de temps dans cette étape par rapport à la méthode Baseline. En plus, l'implémentation de l'algorithme MPCKMeans est optimisé au niveau du compilateur et de la librairie utilisée (*Eigen* [1]). Donc on observe une très nette amélioration du temps d'exécution, qui passe par exemple de plus de sept heures à moins d'une heure pour la stratégie 1 et 50 itérations interactives.

Comparaison des méthodes selon la distance utilisée par la construction et la division de l'arbre et vis-à-vis de la méthode Baseline :

Dans la figure 3, on compare la méthode MPCKMEANS_GLOBAL_DIAGONAL avec la méthode Baseline. Pour la méthode d'apprentissage de métrique, on va aussi comparer les résultats obtenus avec les deux distances (Euclidienne et Mahalanobis)

1. http ://eigen.tuxfamily.org

pour la construction et la division des CF-Entrées. L'apprentissage de métrique donne un meilleur résultat que celui de la méthode Baseline dans la plupart des stratégies appliquées au prix cependant d'une certaine instabilité. Hormis pour la stratégie 1 qui utilise toutes les contraintes déduites de toutes les itérations interactives et donne les meilleurs résultats au prix d'un temps de calcul incompatible avec notre contexte interactif (en-ligne avec l'utilisateur), la méthode réutilisant la distance de Mahalanobis apprise pour la construction et la division de l'arbre est la meilleure.

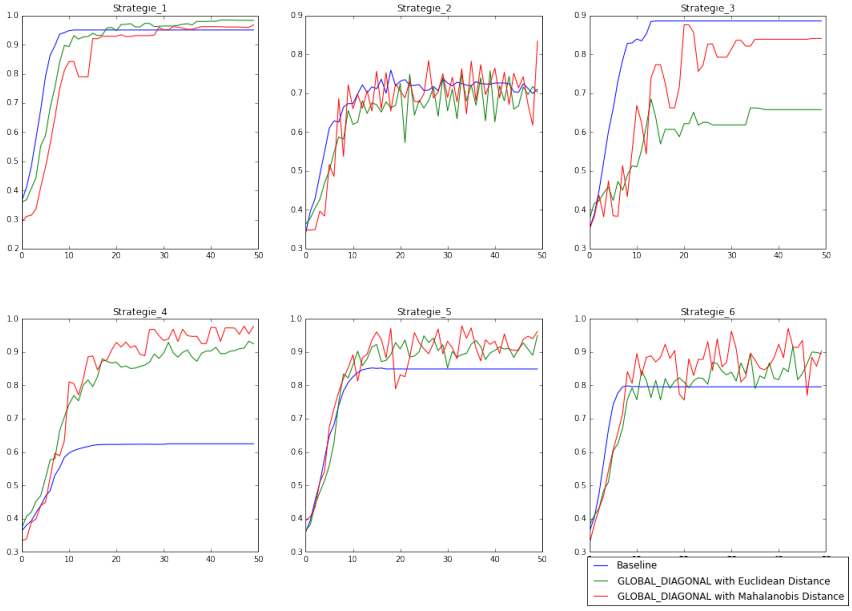


Figure 3 – L'apprentissage de métrique avec approche globale en utilisant la distance Euclidienne et la distance de Mahalanobis pour la division des CF-Entrées

Travaux en cours :

Bien que l'apprentissage de distance permette au final d'améliorer les résultats, dans les 20 premières itérations, la méthode Baseline est toujours en tête. Cela nous a ouvert une perspective qui consiste à combiner les deux méthodes : dans les premières itérations, on applique la méthode Baseline pour obtenir un départ stable, puis ensuite, on accélère la convergence vers les souhaits de l'utilisateur en utilisant une méthode d'apprentissage de métrique. Nos expérimentations préliminaires montrent que cela permet de stabiliser les résultats avec apprentissage de métrique. Reste à déterminer une méthode efficace pour fixer le nombre d'itérations de départ en fonction des caractéristiques de la base utilisée.

4. Conclusion

Dans cet article, nous avons présenté une approche de clustering semi-supervisé interactif d'images utilisant de l'apprentissage de métrique. Nous nous sommes en particulier intéressé à l'apprentissage d'une distance de Mahalanobis qui permet d'accorder une importance variable à chacune des dimensions de l'espace de représentation (ici des mots visuels). Plus précisément, nous avons utilisé l'algorithme MPCKMeans pour apprendre une distance globale, ou plusieurs distances locales (1 par cluster). Nous avons montré que, pour des raisons pratiques sauf quand le nombre d'exemples par cluster est suffisamment élevé, il est préférable d'utiliser une distance globale. De la même manière pour des raisons de temps de calcul (important dans notre contexte interactif où l'apprentissage se fait en-ligne avec l'utilisateur), nous préférons apprendre une distance de Mahalanobis avec matrice de covariance diagonale plutôt que pleine.

Nos résultats expérimentaux, menés sur la base Wang, montrent que l'apprentissage de métrique permet d'améliorer les performances de *clustering*, au prix néanmoins d'une instabilité accrue. Nous somme en train de combiner notre approche avec une approche sans apprentissage de métrique pour réduire l'instabilité tout en conservant les meilleures performances. Un autre résultat très intéressant de notre étude est que, en permettant de diminuer plus efficacement les violations des contraintes déduites des retours de l'utilisateur, l'apprentissage de métrique permet de réduire drastiquement le temps de calcul, et donc est mieux adapté à notre contexte applicatif.

5. Bibliographie

Basu S., Bilenko M., Mooney R. J., « A probabilistic framework for semi-supervised clustering », *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, p. 59-68, 2004.

Bilenko M., Basu S., Mooney R. J., « Integrating constraints and metric learning in semi-supervised clustering », *Proceedings of the twenty-first international conference on Machine learning*, ACM, p. 11, 2004.

Chatpatanasiri R., Korsrilabutr T., Tangchanachaianan P., Kijsirikul B., « A new kernelization framework for Mahalanobis distance learning algorithms », *Neurocomputing*, vol. 73, n° 10, p. 1570-1579, 2010.

He Y., Chen W., Chen Y., Mao Y., « Kernel density metric learning », *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, IEEE, p. 271-280, 2013.

Lai H. P., Visani M., Boucher A., Ogier J.-M., « A new interactive semi-supervised clustering model for large image database indexing », *Pattern Recognition Letters*, vol. 37, p. 94-106, 2014.

Lowe D. G., « Object recognition from local scale-invariant features », *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2, Ieee, p. 1150-1157, 1999.

Schölkopf B., Smola A., Müller K.-R., « Nonlinear component analysis as a kernel eigenvalue problem », *Neural computation*, vol. 10, n° 5, p. 1299-1319, 1998.

Umbaugh S. E., *Digital image processing and analysis : human and computer vision applications with CVIPtools*, CRC press, 2010.

Wang J., Do H. T., Woznica A., Kalousis A., « Metric learning with multiple kernels », *Advances in neural information processing systems*, p. 1170-1178, 2011.

Weinberger K. Q., Blitzer J., Saul L. K., « Distance metric learning for large margin nearest neighbor classification », *Advances in neural information processing systems*, p. 1473-1480, 2005.

Willis A., Sui Y., « An algebraic model for fast corner detection », *Computer Vision, 2009 IEEE 12th International Conference on*, IEEE, p. 2296-2302, 2009.

Xing E. P., Jordan M. I., Russell S., Ng A. Y., « Distance metric learning with application to clustering with side-information », *Advances in neural information processing systems*, p. 505-512, 2002.

Zhang T., Ramakrishnan R., Livny M., « BIRCH : an efficient data clustering method for very large databases », *ACM SIGMOD Record*, vol. 25, ACM, p. 103-114, 1996.