UNIVERSITE NATIONALE DU VIETNAM, HANOI
INSTITUT FRANCOPHONE INTERNATIONAL

VŨ VIẾT MINH

MISE EN PLACE D'UN APPRENTISSAGE DE METRIQUE POUR DU CLUSTERING SEMI-SUPERVISE INTERACTIF D'IMAGES

THIẾT LẬP MỘT THUẬT TOÁN HỌC TỰ ĐỘNG CÁC CHỈ SỐ PHỤC VỤ CHO PHÂN LOẠI ẢNH TỰ ĐỘNG VÀ TƯƠNG TÁC

MEMOIRE DE FIN D'ETUDES DU MASTER INFORMATIQUE

HANOI – 2015

UNIVERSITE NATIONALE DU VIETNAM, HANOI
INSTITUT FRANCOPHONE INTERNATIONAL

VŨ VIẾT MINH

MISE EN PLACE D'UN APPRENTISSAGE DE METRIQUE POUR DU CLUSTERING SEMI-SUPERVISE INTERACTIF D'IMAGES

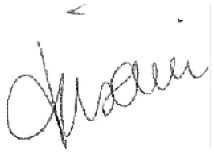
THIẾT LẬP MỘT THUẬT TOÁN HỌC TỰ ĐỘNG CÁC CHỈ SỐ PHỤC VỤ CHO PHÂN LOẠI ẢNH TỰ ĐỘNG VÀ TƯƠNG TÁC

Spécialité: Systèmes Intelligents Multimédia

Code: Programme pilote

MEMOIRE DE FIN D'ETUDES DU MASTER INFORMATIQUE

Sous la direction de: Mme Muriel Visani, Maître de Conférences HDR, Laboratoire L3i - Département Informatique, Université de La Rochelle



HANOI – 2015

ATTESTATION SUR L'HONNEUR

J'atteste sur l'honneur que ce mémoire a été réalisé par moi-même et que les données et les résultats qui y sont présentés sont exacts et n'ont jamais été publiés ailleurs. La source des informations citées dans ce mémoire a été bien précisée.

LỜI CAM ĐOAN

Tôi cam đoan đây là công trình nghiên cứu của riêng tôi.

Các số liệu, kết quả nêu trong Luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác. Các thông tin trích dẫn trong Luận văn đã được chỉ rõ nguồn gốc.

Signature de l'étudiant

Table des matières

Table des figures

Liste de Tableaux

Chapitre 1

Introduction

Ce stage en recherche d'information multimédia, se place dans la suite de la thèse de LAI Hien Phuong, qui traite 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. Son travail dans sa thèse est une étude complète sur les méthodes de clustering non-supervisé et semi-supervisé. Elle 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 reclustering 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 hiérarchique où les images similaires sont automatiquement regroupées dans des groupes compacts et représentatifs. 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 le déplacement des images entre des clusters. Beaucoup de stratégies de déduction des contraintes 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 bénéficiant d'une méthode de

clustering semi-supervisé. La boucle d'interaction peut être répétée jusqu'à la satisfaction de l'utilisateur.

1.1 Problématique et Motivation

Les mesures de la similarité et de la 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 un point de données, 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. Mais cette distance géométrique n'est pas toujours parfaite, par exemple dans l'espace de données non-sphériques ou hétérogènes. Lorsque l'on travaille avec des données multidimensionnelles, la distance Euclidienne traite toutes les dimensions de façon égale, mais dans quelques situations, on doit considérer quelques dimensions en priorité, on a donc besoin d'une métrique paramétrable. L'apprentissage de métrique qui utilise systématiquement la distance de Mahalanobis est une solution prometteuse. 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 dans chaque itération et sont utilisées pour entraîner la métrique pour rendre le résultat du modèle de clustering plus satisfaisant pour l'utilisateur.

1.2 Objectifs et Principales Contributions

L'objectif principal du stage est de mettre en place un apprentissage de métrique grâce aux informations données incrémentalement par l'utilisateur, afin d'améliorer la performance de la phase de clustering. Ce travail de stage a pour principale contribution d'enrichir une méthode existante de clustering semi-supervisé dans un contexte interactif incrémental par des méthodes d'apprentissage de métrique. Les activités réalisées dans ce stage sont les suivantes : (1)Étude de l'état de l'art et du système existant proposé dans le contexte de la thèse de LAI Hien Phuong. (2) Choix de l'algorithme d'apprentissage de métrique à mettre en œuvre, et de la manière de l'articuler avec le système

existant. Après une étude sur les méthodes de clustering non-supervisé, semi-supervisé et semi-supervisé interactif et sur différentes approches d'apprentissage de métrique, l'algorithme MPCKMeans (présenté dans la section 3.3) est choisi. (3) L'implémentation d'un prototype permettant d'intégrer l'algorithme d'apprentissage de métrique dans le système existant. L'adaptation de l'algorithme MPCKMeans sur la structure de données hiérarchique qui est disponible dans le système existant est proposée. Les résultats expérimentaux de cet algorithme avec différentes configurations sont analysés et comparés avec la méthode existante de LAI Hien Phuong.

Les autres chapitres dans ce mémoire sont organisés comme suit : Le chapitre 2 présente l'état de l'art des méthodes de clustering non-supervisé, semi-supervisé et la méthode de clustering semi-supervisé interactif récemment proposée par LAI Hien Phuong. Le chapitre 3 présente l'état de l'art des algorithmes d'apprentissage de métrique et le choix d'une méthode adaptée à notre contexte applicatif. Le chapitre 4 présente l'intégration de la méthode d'apprentissage de métrique choisie dans le système existant et les résultats expérimentaux. Le chapitre 5 termine ce travail par une conclusion.

Chapitre 2

Clustering semi-supervisé interactif incrémental

2.1 Introduction

L'apprentissage non supervisé consiste à inférer des connaissances sur les données. Car aucune information n'est fournie sur l'appartenance des données à telle ou telle classe, on souhaite trouver des groupes compacts et bien séparés et affecter à chaque observation une étiquette de classe (label). Les techniques de clustering non supervisé qui cherchent à décomposer un ensemble d'individus en plusieurs sous ensembles les plus homogènes possible sont présentées dans la section 2.2. Quand on ajoute des informations supervisées incomplètes comme les étiquettes de quelques points ou des relations explicites entre quelques points, on s'oriente vers des méthodes de clustering semi-supervisé (cf. section 2.3). Comme dans la méthode semi-supervisée on a plus de connaissances données, on souhaite améliorer le résultat du clustering non-supervisé. LAI Hien Phuong a proposé un nouveau modèle de clustering semi-supervisé interactif incrémental (cf. section 2.4). Dans son système, les connaissances fournies par l'utilisateur qui interagit avec le système sont utilisées dans les itérations suivantes pour améliorer la performance du modèle. Le dernier point que l'on doit clarifier avant d'étudier les méthodes précisées, c'est le concept de "Incrémental versus non-incrémental" : Une méthode incrémentale va être exécutée de façon continue, et va intégrer les données *au fur et à mesure de* leur arrivée dans l'algorithme. C'est-à-dire, après chaque itération interactive, si on a des nouvelles données (peut être des informations supplémentaires, ou des retours d'utilisateur, ...) elles seront utilisées dans l'itération suivante. À l'inverse, une méthode non-incrémentale va considérer un ensemble de données fournies en entrée, et sera exécutée sur cet ensemble.

Si, par la suite, une nouvelle donnée est fournie, celle-ci devrait être relancée en repartant de zéro.

2.2 Clustering non-supervisé

En général, le clustering automatique d'objets se base sur une mesure de similarité (ou distance) pour grouper les données. Le clustering non supervisé est une analyse multi-dimensionnelle qui vise à partitionner l'ensemble des objets sans besoin d'informations supervisées comme des étiquettes des objets. Une partition ou bien un cluster est une division de l'ensemble en sous-ensembles, telle que chaque objet appartienne à un seul groupe. Les principales méthodes de clustering non supervisé comprennent :

1. Méthodes par partitionnement : Construire K partitions et les corriger jusqu'à obtenir une similarité satisfaisante.
2. Méthodes hiérarchiques : Créer une décomposition hiérarchique par agglomération ou division de groupes similaires ou dissimilaires.
3. Méthodes basées sur la densité : Grouper les objets tant que la densité de voisinage excède une certaine limite.
4. Méthodes basées sur les grilles : Diviser l'espace en cellules formant une grille multi-niveaux et grouper les cellules voisines en terme de distance.

Ces méthodes sont détaillées dans la section 2.2.1 et quelques algorithmes typiques sont présentés dans la section 2.2.2.

2.2.1 Différents types de méthodes

Méthodes par partitionnement

L'idée principale de ces méthodes est de grouper les données de façon optimale pour un critère de partitionnement donné et un nombre de clusters défini par avance. Une bonne partition n'est cependant pas nécessairement la partition "optimale", on utilise donc souvent une technique heuristique pour trouver la bonne partition. Il existe trois approches :

— Première approche : chaque groupe est représenté par son centroïde. K-means (MacQueen et al. [1]) et ISODATA (Ball and Hall [2]) exploitent cette approche.

— Deuxième approche : chaque groupe est représenté par un objet dont la distance moyenne aux membres du groupe est minimale, comme K-medoids (Kaufman and Rousseeuw [3]), PAM (*Partition Around Medoids* - Kaufman and Rousseeuw [4]), CLARA (Kaufman and Rousseeuw [5]) ou CLARANS (Ng and Han [6]).

— Troisième approche : basée sur un réseau de neurones. Dans l'algorithme SOM (*Self-Organizing Map* ou *Kohonen Map* - Kohonen et al. [7]), les points similaires sont regroupés par un réseau de neurones mono-couche. La couche de sorite comprend des neurones représentant les clusters. Les neurones dans le réseau se connectent via un topologie de voisinages. SOM met en correspondance des données de hautes dimensions avec une carte de basses dimensions en cherchant pour chaque point de la couche d'entrée le nœud de la couche de sortie le plus proche.

Avantages et inconvénients :

— Ces méthodes ont l'avantage important d'avoir une complexité polynomiale (souvent linaire ou quadratique) par rapport au nombre d'objet d'entrée, certaines d'entre elles peuvent travailler avec des grandes bases de données.

— Par contre, le processus de scan de tous les objets dans la base ne permet pas à ces méthodes de bien s'adapter dans un contexte incrémental.

— Ces méthodes sont sensibles à leur initialisation et convergent vers des solutions optimales locales (c'est la caractéristique de la technique heuristique).

— Elles ne fonctionnent pas très bien avec les clusters de formes variées (par exemple les clusters non-sphériques quand la distance Euclidienne est utilisée), on a donc besoin d'un autre modèle plus riche pour la présentation des clusters, par exemple le mélange de gaussiennes. C'est la raison pour laquelle on a beaucoup de méthodes qui généralisent le clustering K-Means. Dans le cadre de ce sujet, on va aussi étudier quelques variantes semi-supervisées de K-Means.

Méthodes hiérarchiques

Le fondement du clustering hiérarchique est de créer une hiérarchie de clusters. À partir de la racine de l'arbre (qui est associée à un cluster unique), plus on descend dans l'arbre, plus les clusters sont spécifiques. Afin de former une hiérarchie comme ça, il existe deux grandes approches principales :

— Clustering hiérarchique agglomératif (Bottom-Up) : Au départ, chaque objet constitue un groupe de taille 1. Dans chaque étape, les deux groupes les plus proches sont fusionnés. (AHC : *Agglomerative Hierarchical Clustering* - Lance and Williams [8], AGNES : *AGglomerative NESting* - Kaufman and Rousseeuw [4]).

— Clustering hiérarchique par division (Top-Down) : Au départ, tous les objets sont dans un seul et unique groupe. Un algorithme de partitionnement est ensuite utilisé pour diviser ce grand groupe en deux sous-groupes. (DIANA : *DIvisive ANAlysis* - Kaufman and Rousseeuw [4], BIRCH : *Balanced Iterative Reducing and Clustering using Hierarchies* - Zhang et al. [9]).

Pour les deux approches, l'algorithme est alors appliqué récursivement jusqu'à satisfaction d'un critère d'arrêt (dans un cas extrême, par exemple, que tous les groupes soient de taille 1). Ces deux approches sont illustrées dans la figure 2.1.

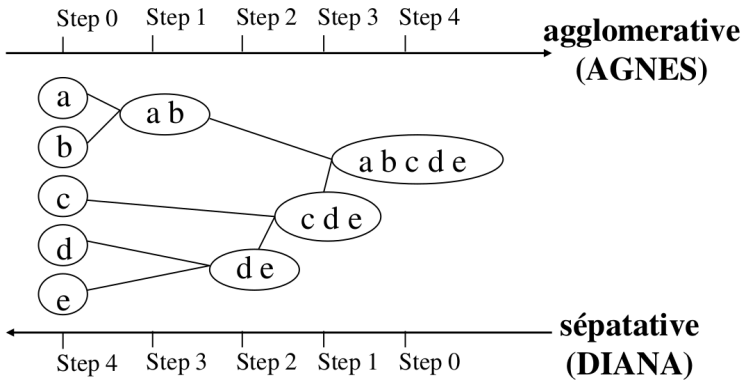


FIGURE 2.1: Illustration des méthodes de clustering non-supervisé hiérarchiques [1]

Avantages et inconvénients :

— Ce type de méthodes permet d'organiser les données dans une structure hiérarchique (un arbre ou un dendogramme). C'est peut être utile par exemple pour une visualisation interactive des données : l'utilisateur peut cliquer sur un nœud pour découvrir des sous-clusters sous ce nœud. Grâce à la représentation hiérarchique des clusters, on peut obtenir un nombre différent de clusters selon la profondeur que l'on fouille dans la hiérarchie.

— Quelques méthodes hiérarchiques traitent les objets successivement un par un, elles sont donc appropriées dans le contexte incrémental. Mais ces méthodes sont sensibles à l'ordre d'entrée des objets.

— Dans l'approche Bottom-Up les clusters les plus proches sont fusionnés itérativement, et dans l'approche Top-Down les clusters les plus éloignés sont divisés itérativement. Dans les deux cas, on a besoin d'une mesure de *dissimilarité*. Donc le choix d'une métrique est un point important qui détermine la qualité des clusters.

Méthodes basées sur la densité

Dans un espace, les zones de plus grande densité formeront les clusters. Le but des méthodes basées sur la densité est d'identifier les zones de forte densité entourées par des zones de faible densité (par exemple l'algorithme DBSCAN de Ester et al. [10]). Quand on utilise la notion de la densité, on utilise aussi des informations statistiques. L'algorithme EM de Dempster et al. [11], fait l'hypothèse que les données sont distribuées selon

1. Lien : Hierarchical Clustering Essentials - Unsupervised Machine Learning
http ://www.sthda.com/english/wiki/hierarchical-clustering-essentials-unsupervised-machine-learning

certaines lois avec une certaine probabilité. Pour découvrir la probabilité d'apparition de chaque objet, on doit alors estimer les paramètres cachés de cette distribution. Avantages et inconvénients :

— Pour certaines méthodes il n'est pas nécessaire de préciser le nombre de clusters à trouver.

— Les clusters n'ont pas pour obligation d'être linéairement séparables et certaines méthodes (par exemple DBSCAN) ne font aucune hypothèse sur la distribution de données, c'est-à-dire que l'on peut travailler avec des clusters de formes très variées (des clusters creux, des clusters entourant un autre cluster).

— Elles sont également capables de faire face au bruit qui peut exister dans les données et détecter des observations aberrantes (*outliers*).

— Par contre, la complexité est quadratique en fonction du nombre d'objets entrés.

— Ces méthodes ne sont pas adéquates dans un contexte incrémental, car quand on ajoute un nouvel objet, la distribution de tout l'ensemble va changer.

— Et bien sûr, les méthodes dans cette catégorie dépendent de paramètres (des paramètres de loi de distribution ou des paramètres pour distinguer la forte densité et la faible densité), particulièrement difficiles à estimer dans le cas où les données sont de grandes dimensions.

Méthodes basées sur les grilles

Les algorithmes dans cette catégorie (STING : *STatistical INformation Grid* - Wang et al. [12], CLIQUE : *CLustering In QUEst* - Agrawal et al. [13]) consistent en trois étapes générales :

— Diviser l'espace en cellules rectangulaires.

— Supprimer les cellules de basse densité, c'est-à-dire, si une cellule a une densité élevée, on la considère comme un cluster. Par contre, une cellule contenant peu de points est considérée comme du bruit.

— Combiner les cellules adjacentes pour former les clusters.

Les étapes pour construire une grille des cellules dans la méthode basée sur les grilles sont illustrées dans la figure 2.2.

Avantages et inconvénients :

— C'est une approche descendante : chaque cellule de niveau i est divisée en cellules plus petites au niveau $i+1$. Donc cette approche peut être utilisée dans un contexte incrémental. Les cellules formées par STING ont une structure hiérarchique.

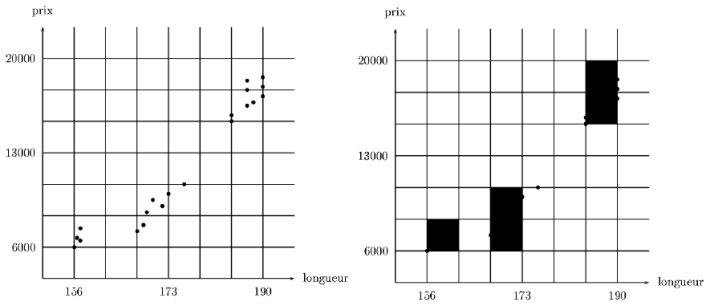


FIGURE 2.2: Illustration des méthodes basées sur les grilles

— Les informations statistiques sont calculées et stockées à chaque niveau, elles sont donc parallélisables, mises à jour progressivement à chaque niveau séparément.

— Mais à cause de la structure des grilles, les bords des clusters sont soit horizontaux soit verticaux, pas de diagonale, par exemple, ni de formes plus raffinées.

— Les données d'entrées de grandes dimensions sont toujours clairsemées (*sparse*), donc les paramètres de cellules sont difficiles à trouver. Dans ce cas, les méthodes hiérarchiques sont plus adaptées.

Bilan

Dans les travaux de Lai et al. [14], une comparaison formelle de différentes méthodes de clustering non-supervisé est présentée. La figure 2.3 résume cette comparaison. La notation est la suivante : Méthodes par partitionnement (P), Méthodes hiérarchiques (H), Méthodes basées sur les grilles (G), Méthodes basées sur la densité (D). Différents critères sont pris en compte : la complexité (*complexity*), l'adéquation avec des grandes bases de données (*adapted to large databases*), l'incrémentalité (*incrementality*), la structure hiérarchique (*hierarchical structure*), la dépendance vis à vis de l'ordre des données(*data order dependence*), la sensibilité aux valeurs aberrantes(*sensitivity to outliers*), la dépendance aux paramètres(*parameters dependence*).

On peut y voir que les méthodes les plus adaptées à un contexte interactif et incrémental, en présence de données de grandes dimensions sont : CLARA, SOM, BIRCH, R-Tree, SS-Tree, SR-Tree. Parmi ces méthodes, BIRCH (*Balanced Iterative Reducing and Clustering using Hierarchies* - Zhang et al. [9]) a la complexité la plus petite (de $O(N)$) et elle fournit une structure hiérarchique de données en sauvegardant les informations de clustering dans un arbre balancé. Elle est donc choisie pour l'étape de clustering non-supervisé dans le système de LAI Hien Phuong (cf. section 2.4).

Methods	Complexity	Adapted to large database	Incrementality	Hierarchical structure	Data order dependence	Sensitivity to outliers	Parameters Dependence
k-means (P)	$O(Nkl)$ (time) $O(N+k)$ (space)	Yes	No	No	No	Sensitive	No
k-medoids (PAM) (P)	$O(k(n-k)^2)$ (time)	No	No	No	No	Enable outliers detection	No
CLARA (P)	$O(kN_S^2+k(N-k))$ (time)	Yes	Able to add new points	No	No	Enable outliers detection	No
CLARANS (P)	$O(N^2)$ (time)	No	No	No	Yes	Enable outliers detection	Yes
ISODATA (P)	$O(Nkl)$ (time) $O(N+k)$ (space)	Yes	No	No	No	Enable outliers detection	Yes
SOM (P)	$O(Nkl)$ (time)	Yes	Yes	No	Yes	Sensitive	Yes
DIANA (H)	$O(N^2)$ (time)	No	No	Yes	No	Sensitive	No
Simple Divisitive Algorithm (MST) (H)	$O(N^2)$ (time)	No	No	Yes	No	Sensitive	No
AHC (H)	$O(N^2logN)$ (time) $O(N^2)$ (space)	No	Have incremental version	Yes	No	Sensitive	No
BIRCH (H)	$O(N)$ (time)	Yes	Yes	Yes	Yes	Enable outliers detection	Yes
CURE (H)	$O(N_S^2logN_S)$ (time)	Yes	Able to add new points	Yes	No	Enable outliers detection	Yes
R-tree (H)	$O(NlogN)$ (time)	Yes	Yes	Yes	Yes	Sensitive	Yes
SS-tree (H)	$O(NlogN)$ (time)	Yes	Yes	Yes	Yes	Sensitive	Yes
SR-tree (H)	$O(NlogN)$ (time)	Yes	Yes	Yes	Yes	Sensitive	Yes
STING (G)	$O(N)$ (time)	Yes	Yes	Yes	No	Enable outliers detection	Yes
CLIQUE (G)	$O(N+d^2)$ (time)	Yes	Able to add new points	No	No	Enable outliers detection	Yes
DBSCAN (D)	$O(NlogN)$ (time)	Yes	No	No	Yes	Enable outliers detection	Yes
OPTICS (D)	$O(NlogN)$ (time)	Yes	No	No/Yes	Yes	Enable outliers detection	Yes
EM (D)	$O(Nk^2l)$ (time)	Yes	No	No	No	Enable outliers detection	Yes

FIGURE 2.3: Comparaison des méthodes de clustering non supervisé dans [14]

2.2.2 Présentation des méthodes de clustering non-supervisé utilisées

Pour avoir une vue détaillée sur les méthodes de clustering non-supervisé, on va présenter quelques méthodes précisées dans cette partie. D'abord, on présente l'algorithme K-Means (MacQueen et al. [1]), un des algorithmes de clustering non-supervisé les plus couramment utilisés. Ensuite, on présente un autre algorithme plus généralisé, basé sur le modèle probabiliste : l'algorithme *Expectation Maximisation* - Dempster et al. [11]. Enfin, on présente l'algorithme BIRCH qui peut fournir une structure hiérarchique de données de façon compacte et représentative. L'algorithme BIRCH est utilisé dans le système de LAI Hien Phuong pour créer le clustering initial.

K-Means - MacQueen et al. [1]

K-Means est un algorithme de quantification vectorielle. Il est actuellement un des plus utilisés et des plus efficaces en analyse des données. Étant donné un ensemble de points et un entier K fixé par l'utilisateur, il faut chercher à séparer cet ensemble en K clusters *intra-homogènes* et *inter-hétérogènes*. Plus simplement dit, il faut que les points de données dans un même cluster soient similaires et que les points dans les différents clusters soient dissimilaires. Le niveau de similarité / dissimilarité est mesuré par une métrique donnée, par exemple, par la distance Euclidienne. On utilise les notations suivantes :

- L'ensemble de données $X = \{x_i\}, x_i \in \mathcal{R}^d, i \in 1, ..., N$ avec d est le nombre de dimensions, et N est le nombre de points.
- $\mu_k \in \mathcal{R}^d$ est le centre du k ème cluster.

On définit la **mesure de distorsion** ou bien la **fonction objectif** par :

$$J_{obj} = \sum_{i=1}^{N} \sum_{k=1}^{K} \| x_i - \mu_k \|^2$$

Le but de minimiser J_{obj} est réalisé dans le pseudo-code suivant.

L'arrêt de K-Means peut se faire au choix, selon deux critères d'arrêt : (1) Lorsque deux itérations successives conduisent à une même partition, c'est-à-dire que deux itérations successives donnent les mêmes représentants des clusters. En pratique, on peut relancer cette boucle tant que la différence entre l'ancienne et la nouvelle valeur de J_{obj} est inférieure à un seuil fixé (souvent très petit). (2) Lorsque le nombre maximal d'itérations est atteint. Cependant la convergence est locale, ce qui pose le problème de l'initialisation. Pour le surmonter, une méthode classique consiste à lancer K-Means plusieurs fois avec les initialisations différentes à chaque fois. Puis on compare leur mesure de distorsion J_{obj} et on choisit la répartition qui possède le coût minimal.

Algorithme 1 : L'algorithme K-Means

Initialisation On choisit K individus au hasard parmi les N points de données (il s'agit d'un tirage aléatoire simple sans remise de K individus à parti de la population de taille N).

tant que *pas encore converge* **faire**

(a) Associer chaque point x_i au centre le plus proche μ_k en utilisant, par exemple, la distance Euclidienne.

(b) Mettre à jour les nouveaux centres de chaque cluster : $\mu_k = \frac{1}{|\mathcal{X}_k|} \sum_{x_i \in \mathcal{X}_k} x_i$ avec $\mathcal{X}_k$: l'ensemble de points qui sont assignés au cluster k.
Le but de cette étape est de minimiser J_{obj} par rapport à $\{\mu_k\}$.

fin

Cette méthode est fortement liée au nombre K de clusters fixé a priori, et elle dépend du choix des centres initiaux et de la distance utilisée.

EM (*Expectation-Maximisation*) - Dempster et al. [11]

On va commencer à étudier l'algorithme EM par un modèle simple de mélange de gaussiennes. Supposons que l'on a deux gaussiennes uni-dimensionnelles différentes avec des moyennes (μ) et des écart-types (*standard deviation* σ) mais on ne sait pas exactement leurs valeurs. Ce sont des paramètres cachés que l'on veut découvrir. On a un ensemble de points dans le même espace de ces deux gaussiennes, mais on n'est par sûr de la distribution à laquelle chaque point appartient ? Donc, on a une autre variable cachée de l'appartenance à l'une ou l'autre des gaussiennes pour chaque point de données. Quelle est la relation entre cet exemple avec le problème de clustering ? Les deux gaussiennes dans cet exemple sont peut être considérées comme les clusters, mais on ne sait pas leurs informations détaillées ni leur centre de gravité (qui est la moyenne de tous les points dans sa distribution). On peut imaginer les deux hypothèses suivantes :

— Si on connaît les moyennes des deux gaussiennes, il est facile de déterminer l'étiquette de chaque point selon par exemple la distance entre le point et la moyenne. Bien sûr il existe d'autres mesures plus efficaces qui peuvent déterminer l'appartenance d'un point si on a le centre de la distribution où se trouve ce point, par exemple la vraisemblance (*likelihood*).

— Si on connaît la distribution de laquelle chaque point est venu, alors on peut estimer les moyennes des deux distributions en utilisant les moyennes des points pertinents (*sample means of the relevant points*).

On considère les deux hypothèses ci-dessus comme deux étapes que l'on peut résoudre séparément. Au lieu d'essayer d'exécuter ces deux étapes à la fois, il faut alterner entre

ces deux étapes : On va commencer avec une première estimation des deux moyennes (bien que cette estimation ne doive pas nécessairement être très précise, on peut commencer quelque part dans l'espace). Maintenant, on a assez d'information pour exécuter la première étape (que l'on appelle *'étape E'*). Et puis, compte tenu des distributions de chaque point assigné, on peut obtenir les nouvelles estimations pour les moyennes via la deuxième étape (que l'on appelle *'étape M'*). On va trouver la "discordance" entre et les moyennes hypothétiques de *l'étape E* et leurs valeurs estimées réelles dans *l'étape M*. Sans doute, on va retourner à *l'étape E* et recommencer une nouvelle boucle pour améliorer le résultat, et ce, jusqu'à ce que la discordance sont suffisamment faible. Finalement, on va découvrir toutes les variables cachées : les paramètres des distributions gaussiennes et les étiquettes de chacun des points de données, c'est-à-dire, on va résoudre le problème de clustering. Cet algorithme peut se généraliser à un problème multidimensionnel avec bien sûr un surcoût en termes de temps de calcul.

BIRCH (*Balanced Iterative Reducing and Clustering using Hierarchies*) - Zhang et al. [9]

Introduction de BIRCH

Avec les méthodes de clustering simples qui contiennent des itérations comme K-Means, il est évident que l'on doit scanner plusieurs fois l'ensemble de données. Zhang et al. [9] ont proposé une méthode d'amélioration du stockage des données : BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies). Les données fournies en entrée de cet algorithme ne sont scannées qu'une seule fois, et sont stockées dans un arbre appelé CF-Tree.

Un élément CF (*Clustering Feature*, qui est souvent appelé CF-Entrée) se compose de trois paramètres : $(N, \vec{LS}, SS)$ où N est le nombre d'objets, $\vec{LS} = \sum_{i=1}^{N} \vec{x_i}$ est la somme linéaire de ces N objets, et $SS = \sum_{i=1}^{N} \vec{x_i}^2$ est leur somme carrée. Grâce à cette représentation, la fusion de deux CFs disjoints s'effectue par :

$$CF_1 + CF_2 = (N_1 + N_2, \vec{LS_1} + \vec{LS_2}, SS_1 + SS_2) \tag{2.1}$$

Construction de l'arbre CF-Tree

L'arbre CF-Tree est contrôlé par deux paramètres qui correspondent respectivement au nombre d'enfants pour chaque nœud (B pour les nœuds internes et L pour les feuilles) et un seuil T qui indique le diamètre maximal des CF-Entrées au niveau des feuilles. Chaque nœud interne contient au maximum B entrées sous la forme $[CF_i, child_i]$, où

2. www.cs.uvm.edu/ xwu/kdd/Birch-09.ppt

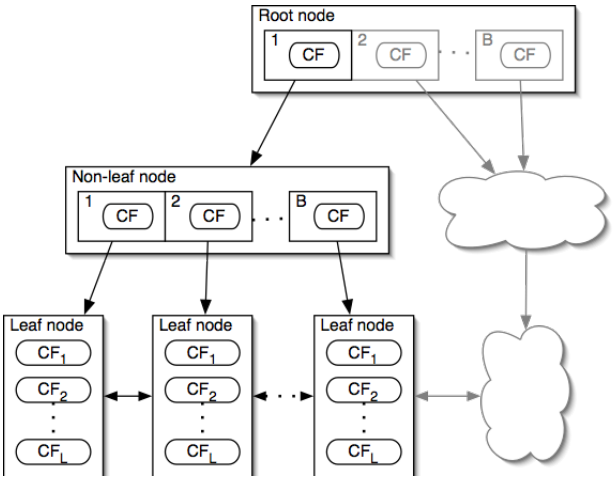


FIGURE 2.4: L'arbre CF-Tree avec au plus B CF entrées dans les nœuds non-terminaux, et avec au plus L CF entrées dans les feuilles [2]

$i \in [1, B], child_i$ est un pointeur sur le i ème nœud enfant. CF_i constitue un résumé des informations de tous ses enfants. Donc, un nœud non-terminal représente un sous-cluster.

De même, chaque feuille contient au plus L entrées dont chaque entrée est un CF. De plus, pour enchaîner toutes les feuilles ensemble pour un parcours efficace, chaque feuille est associée par deux pointeurs (qui pointent vers la feuille précédente et suivante) comme dans le *linked-list*. (Voir la figure 2.4).

L'arbre est construit dynamiquement par l'insertion des objets. Pour insérer un objet, on descend, à partir de la racine, en sélectionnant le nœud fils le plus proche à chaque niveau. Après l'insertion de l'objet, si le diamètre de la CF-Entrée de la feuille est supérieur au seuil T, la CF-Entrée est alors divisée. La division peut être propagée vers le haut dans les nœuds de la branche concernée. (Voir Figure A.1)

À la fin de la construction de l'arbre, au niveau de chaque feuille, on a un ensemble de CF entrées qui forme une nouvelle base de données de taille beaucoup plus petite et donnant un résumé structuré des données. Un algorithme de clustering non hiérarchique est ensuite utilisé sur l'ensemble des CF-Entrées des feuilles, puis chacun des objets initiaux est finalement redistribué par rapport au centroide le plus proche de la CF-Entrée correspondante.

Avantages et inconvénients

BIRCH est créé pour travailler avec un grand nombre d'objets, pour utiliser la mémoire de façon efficace et pour diminuer le temps I/O. Mais ses paramètres (B, L et T) sont difficiles à contrôler. En pratique, le facteur de branchement dépend de la mémoire physique, mais le seuil T doit être examiné de façon expérimentale. De plus, dans sa version originale, BIRCH ne s'adapte pas efficacement aux clusters de formes variés car il utilise

des paramètres tels que le diamètre et le rayon (selon une mesure de dissimilarité ou une distance donnée) pour le calcul des frontières du cluster.

Utilisation de BIRCH

Dans le système de LAI Hien Phuong, BIRCH est utilisé pour s'intégrer dans le contexte interactif où les retours de l'utilisateur sont utilisés pour réorganiser la structure de l'arbre CF-Tree. L'étape de découpage et fusion des CF-Entrées pour s'approcher de la satisfaction de l'utilisateur est présentée dans la section 2.4.

Pour conclure de cette section, on peut trouver quelques limites des méthodes de clustering non-supervisé, ce sont : la sensibilité à l'initialisation, la difficulté du choix d'une métrique (normes L1, L2, Mahalanobis, ...), la qualité de la convergence (souvent locale), l'adéquation entre la partition produite et la partition souhaitée, ... Pour réduire ces limites, on peut inclure des connaissances du domaine de données dans le processus de clustering. Pour cela on utilise l'apprentissage semi-supervisé qui est présentée dans la section suivante.

2.3 Clustering semi-supervisé

Dans le domaine d'apprentissage automatique, d'un côté, les techniques d'apprentissage supervisé permettent d'obtenir des bons résultats car on dispose d'une base annotée, mais avec un risque de sur-apprentissage auquel s'ajoute le coût important de l'annotation de la base d'apprentissage. D'un autre côté, il existe de nombreuses techniques de clustering non-supervisé (comme présenté dans la section 2.2) qui cherchent à construire la structure de données sans avoir besoin de données d'entraînement annotées. Pour améliorer la performance, on peut ajouter la connaissance sur un petit sous-ensemble d'éléments (la classes de quelques points de données ou les contraintes par paires entre quelques points). On a une nouvelle approche, le clustering semi-supervisé, qui utilise les connaissances fournies pour guider le processus de clustering des données non étiquetées. On peut noter que l'exhaustivité et/ou la qualité des connaissances fournies est trop faible pour pouvoir être utilisée dans une approche d'apprentissage supervisée.

2.3.1 Différents types de méthodes

— Clustering semi-supervisé utilisant des données étiquetées : Des données étiquetées sont utilisées pour générer des *'seeds'* qui initialisent un algorithme de clustering. Des contraintes générées à partir des données étiquetées sont aussi utilisées pour guider le processus de clustering. Dans quelques approches de clustering par partitionnement, les méthodes d'initialisation comprennent : sélectionner par hasard ou

prendre la moyenne de l'ensemble des données et de perturbation aléatoire. Dans l'algorithme Seeded-KMeans de Basu et al. [15], les informations de *'seeds'* sont utilisées pour créer les clusters initiaux. Les autres étapes de Seeded-KMeans sont exactement comme celles de K-Means.

— Clustering semi-supervisé utilisant des contraintes par paires : Cette approche maximise la qualité du partitionnement tout en réduisant les coûts d'annotation. Dans l'algorithme COP-KMeans de Wagstaff et al. [16], les points sont assignés au cluster le plus proche sans violer aucune contrainte. S'il n'y a pas de clustering satisfaisant l'ensemble de ces contraintes, l'étape de clustering va échouer. Une autre idée est d'introduire un terme de pénalité à la fonction objectif, comme présentée dans l'algorithme HMRF-KMeans (*Hidden Markov Random Fields* - Basu et al. [17]) : L'idée est de pénaliser les solutions de clustering qui mettent des exemples ayant des MustLinks entre eux dans différents groupes ou qui mettent ensemble des exemples ayant des CannotLinks entre eux. La pénalité doit être proportionnelle (ou inversement proportionnelle) à la distance entre les exemples, comme détaillé dans la section suivante.

2.3.2 Présentation de HMRF-KMeans

HMRF-KMeans (*Hidden Markov Random Fields* - Basu et al. [17]) est une méthode de clustering semi-supervisé qui utilise des contraintes par paires de MustLinks et CannotLinks entre des points de données, et utilise la distance Euclidienne comme mesure de distorsion. Elle utilise aussi l'idée de l'algorithme EM pour minimiser la fonction objective. Le modèle probabiliste de HMRF-KMeans utilise les composants suivants :

— Un ensemble de n observations $X = (x_1, \cdots, x_n)$ correspondant aux points de données.

— Un ensemble de n variables cachées $Y = (y_1, \cdots, y_n)$ correspondant aux étiquettes des points. Ce sont les variables cachées que l'on doit trouver.

— Un ensemble de variables cachées qui sont des paramètres $\Omega = \{A, M\}$, où A est la matrice des paramètres dans la fonction de distorsion, M est la présentation de K clusters $M = (\mu_1, \cdots, \mu_K)$.

— Un ensemble d'observations qui représente les contraintes par paires $C = (c_{12}, c_{13}, \cdots, c_{n-1,n})$ où $c_{i,j} = 1$ implique que $(x_i, x_j) \in MustLinks$, $c_{i,j} = -1$ implique que $(x_i, x_j) \in CannotLinks$ et $c_{i,j} = 0$ implique qu'il n'y a pas de contrainte entre x_i et x_j.

La fonction objectif est définie comme suit :

$$\begin{aligned} \mathcal{J}_{objHMRF-KMeans} &= \sum_{x_i \in X} D(x_i, \mu(x_i)) \\ &+ \sum_{(x_i,x_j) \in MustLinks, K(x_i) \neq K(x_j)} wD(x_i, x_j) \\ &+ \sum_{(x_i,x_j) \in CannotLinks, K(x_i) = K(x_j)} \overline{w}(D_{max} - D(x_i, x_j)) \end{aligned} \quad (2.2)$$

Dans l'étape E, chaque point x_i est assigné à un cluster K_j pour minimiser la fonction objectif $\mathcal{J}_{objHMRF-KMeans}$. Dans l'étape M, étant donnée l'attribution de chaque point à son cluster $K(x_i)$, le centre de chaque cluster est ré-estimé pour minimiser la fonction objectif. Et ensuite, on ré-estime les paramètres de la fonction de distorsion pour réduire la fonction objectif.

2.4 Modèle de clustering semi-supervisé interactif de LAI Hien Phuong

En général, les systèmes d'apprentissage automatique (sauf quelques systèmes d'apprentissage profond (*Deep Learning*)) utilisent souvent des caractéristiques de bas niveau qui sont extraits à partir des images originales via des algorithmes de détection de bord (*edge detection : Canny, Sobel, Prewitt, ...* - [18]), de coin (*Corner detection : Harris, SUSAN, ...* - [19]), de *blob*[3](*blob detection : Laplacian of Gaussian (LoG), Difference of Gaussians (DoG), Determinant of Hessian (DoH), ...*), SIFT (*Scale-invariant feature transform* - [20]), ...

Mais quand l'utilisateur intervient dans ces systèmes, toutes ses interactions reflètent des concepts sémantiques de haut niveau, comme quelles images se ressemblent ou se distinguent. On a donc un fossé sémantique entre le besoin de l'utilisateur et le résultat réel fourni par le système d'apprentissage. La méthode de LAI Hien Phuong [21] permet de résoudre partiellement ce problème en introduisant un nouveau modèle de clustering semi-supervisé interactif qui bénéfice des retours de l'utilisateur pour corriger les erreurs du modèle de clustering de façon interactive et incrémentale.

2.4.1 Introduction et Motivation

Selon les analyses expérimentales dans les travaux de LAI Hien Phuong, parmi les méthodes de clustering semi-supervisé, la méthode HMRF-KMeans avec l'interaction de

3. Wikipedia : https://en.wikipedia.org/wiki/Blob_detection

l'utilisateur donne le meilleur résultat. De plus, au niveau de retours de l'utilisateur, avec le même nombre de clics, les contraintes par paires donnent plus d'informations supervisées que les étiquettes. Cependant, la méthode HMRF-KMeans ne se base pas sur une hiérarchie et les contraintes utilisées sont entre des paires d'images, donc potentiellement nombreuses. Après quelques itérations interactives, le nombre de contraintes déduites et le temps d'exécution sont élevés. Ça provoque une trop grande complexité dans l'étape de reclustering.

Si les images similaires sont regroupées ensemble, puis les contraintes par paires entre des images sont remplacées pas des contraintes par paires entre des groupes d'images, on peut réduire le nombre de contraintes sans réduction de la qualité des informations supervisées. La méthode proposée par Lai et al. [21] est une méthode de clustering semi-supervisé interactif qui utilise les contraintes par paires entre des CF-Entrées au niveau des feuilles de l'arbre CF-Tree fourni par l'algorithme BIRCH. L'intégration des contraintes par paires dans l'étape de réorganisation des CF-Entrées s'inspire de l'algorithme HMRF-KMeans.

Les détails des étapes du système existant 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 (*codebook*) 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. Dans l'étape de reclustering interactif, on va travailler sur l'ensemble des CF-Entrées dans des nœuds feuilles de cet arbre. Comme l'utilisateur visualise le résultat du clustering de chaque itération, il va corriger 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 le modèle 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 leur nouvel ensemble de CF-Entrées de chaque itération interactive est traité par HMRF-KMeans.

2.4.2 Modèle d'interaction

L'utilisateur va intervenir à chaque itération interactive. Le système fait le clustering et présente le résultat sur une interface interactive. 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'*.

Une capture d'écran de l'interface interactive se trouve dans la Figure 2.5.

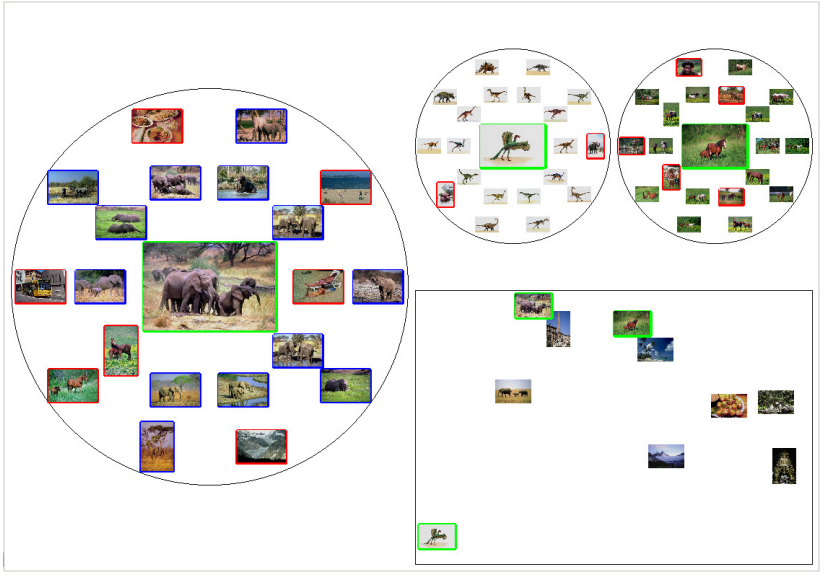


FIGURE 2.5: L'interface interactive du système de LAI Hien Phuong

En cliquant sur une image prototype dans le plan principal, l'utilisateur peut voir plus d'information détaillée sur le cluster correspondant : une 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 existantes, 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 associé à une base annotée.

2.4.3 Stratégies de déduction des contraintes

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 conséquent, 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. Il y a peut être des CannotLinks entre les images d'une même CF-Entrée, ou il existe simultanément des MustLinks et CannotLinks entres des

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. L'algorithme de clustering semi-supervisé HMRF-KMeans va utiliser des contraintes entre CF-Entrées à la place des contraintes entre images.

Un nouveau concept de *voisinage* est introduit comme un ensemble d'image qui devrait être dans le même cluster. Une matrice d'adjacence est créée pour dénoter la relation entre des voisinages. Grâce aux informations sur les voisinages, les contraintes par paires entre images sont déduites. Et ensuite, les contraintes par paires entre des CF-Entrées sont déduites à partir des contraintes par paires entre images comme suit :

— S'il y a une contrainte MustLink entre une image de CF_i et une autre image de CF_j, une nouvelle contrainte MustLink sera créée entre CF_i et CF_j.

— S'il y a une contrainte CannotLink entre une image de CF_i et une autre image de CF_j, une nouvelle CannotLink sera créée entre CF_i et CF_j.

Dans le système existant, 6 stratégies différentes sont présentées et résumées dans le tableau 2.1. Les contraintes par paires utilisées peuvent être divisées en 2 catégories : les contraintes de l'utilisateur qui sont créées directement à partir des retours de l'utilisateur dans chaque itération et les contraintes déduites qui sont créées en bénéficiant des règles de déduction. La stratégie 1, qui est la plus simple, utilise toutes les contraintes déduites possibles, et logiquement, donne plus d'information supervisée pour le reclustering. Les autres stratégies ont différentes façons de réduire le nombre de contraintes en conservant la performance de l'étape de reclustering.

	Contraintes de l'utilisateur		**Contraintes déduites**	
	Les itérations utililisées	***Les contraintes utilisées***	***Les itérations utililisées***	***Les contraintes utilisées***
Stratégie 1	Toutes	Toutes	Toutes	Toutes les contraintes possibles sont déduites.
Stratégie 2	Toutes	Toutes	Aucune	Aucune
Stratégie 3	Toutes	Toutes	Seulement l'itération courante	+ Seulement les contraintes déduites dans l'itération courante sont utilisées. + Les contraintes déduites des itérations précédentes ne sont pas prises en compte.
Stratégie 4	Toutes	Les contraintes entre les images et les prototypes positifs de chaque cluster.	Seulement l'itération courante	Seulement les contraintes entre les images et les prototypes positifs de chaque cluster dans l'itération courante sont déduites.
Stratégie 5	Toutes	+ MustLink entre les images positives qui sont les plus éloignées. + CannotLink entre les images positives et négatives qui sont les plus proches.	Toutes	+ MustLink entre les images les plus éloignées de chaque voisinage sont déduites. + CannotLink entre les images les plus proches de deux voisinages liés par un CannotLink sont déduites.
Stratégie 6	Toutes	+ MustLink entre les images positives qui sont les plus éloignées. + CannotLink entre les images positives et négatives qui sont les plus proches.	Toutes	Il en va de même que pour la stratégie 5. Mais les CannotLinks sont déduits entre les images des voisinages liés par un CannotLink en les filtrant selon la taille de ces voisinages.

TABLE 2.1: Résumé des 6 stratégies de déduction de contraintes

2.4.4 Méthode de clustering semi-supervisé interactif incrémental

Dans chaque itération interactive, après avoir déduit les contraintes par paires à partir de retours de l'utilisateur, la nouvelle méthode de clustering semi-supervisé interactif basée sur HMRF-KMeans est appliquée. L'ensemble des CF-Entrées des feuilles de l'arbre CF-Tree $S_{CF} = (CF_1, ..., CF_m)$ va être regroupé selon des informations supervisées sous forme d'ensemble de MustLinks et CannotLinks entre des CF-Entrées : $M_{CF} = \{(CF_i, CF_j)\}$, $C_{CF} = \{(CF_i, CF_j)\}$. Une contrainte MustLink $(CF_i, CF_j) \in M_{CF}$ implique que CF_i, CF_j et tous les points inclus dans ces deux CF-Entrées doivent appartenir au même cluster. De la même manière, une contrainte CannotLink $(CF_i, CF_j) \in C_{CF}$ implique que CF_i et CF_j doivent se trouver dans des différentes clusters. La fonction objectif à minimiser est comme suit :

$$\begin{aligned} J_{obj} &= \sum_{CF_i \in S_{CF}} D(CF_i, \mu_{l_i}) \\ &+ \sum_{(CF_i, CF_j) \in M_{CF}, l_i \neq l_j} w N_{CF_i} N_{CF_j} D(CF_i, CF_j) \\ &+ \sum_{(CF_i, CF_j) \in C_{CF}, l_i = l_j} \overline{w} N_{CF_i} N_{CF_j} (D_{max} - D(CF_i, CF_j)) \end{aligned} \tag{2.3}$$

— Le premier terme mesure la distorsion entre une CF-Entrée CF_i et le centre de son cluster correspondant μ_{l_i}, où l_i est l'étiquette du cluster de CF_i.

— Les deuxième et troisième termes représentent la pénalité de la violation des contraintes entre des CF-Entrées. w et $\overline{w}$ sont les constantes pondérées pour spécifier le coût de la violation. Une CF-Entrée CF_i représente des informations d'un groupe de N_{CF_i} points, une contraintes par paire entre deux CF-Entrées CF_i et CF_j correspond à $N_{CF_i} \times N_{CF_j}$ contraintes entres des points de deux CF-entrées. Le coût de la violation des contraintes par paires entre deux entrées CF_i, CF_j est donc une fonction de leur distance $D(CF_i, CF_j)$ et du nombre de points inclus dans ces deux entrées.

— D_{max} est la distance maximum entre deux CF-Entrées dans tout l'ensemble de données. Une pénalité plus élevée est assignée à la violation de MustLink entre les entrées qui sont éloignées et à la violation de CannnotLink entres les entrées qui sont proches. Le terme D_{max} peut rendre la violation des contraintes CannotLinks sensible aux observations aberrantes, et pourrait être remplacé par la valeur maximum de distance entre deux clusters.

2.4.5 Résultats expérimentaux

Mesure de la qualité de clustering

VMesure ([22]) est une mesure basée sur l'entropie qui mesure explicitement comment les critères de l'homogénéité (*homogeneity*) et de la compacité (*completeness*) ont été satisfaits. VMesure est calculée comme la moyenne harmonique (*harmonic average*) de l'homogénéité et de la compacité (tout comme la précision et le rappel sont généralement combinés en F-Mesure).

Comme la F-Mesure peut être pondérée, la VMesure peut être pondérée pour favoriser les contributions de l'homogénéité et de la compacité par le paramètre β. (Voir l'équation 2.4 et les notations utilisées dans l'annexe B)

$$\text{VMesure} = \frac{(1+\beta) * homogeneity * completeness}{\beta * homogeneity + completeness} \tag{2.4}$$

VMesure est choisie pour évaluer la performance car elle possède les avantages suivants :

— Elle évalue une solution de clustering indépendamment de l'algorithme de clustering, de la taille de l'ensemble de données, du nombre de classes et du nombre de clusters.

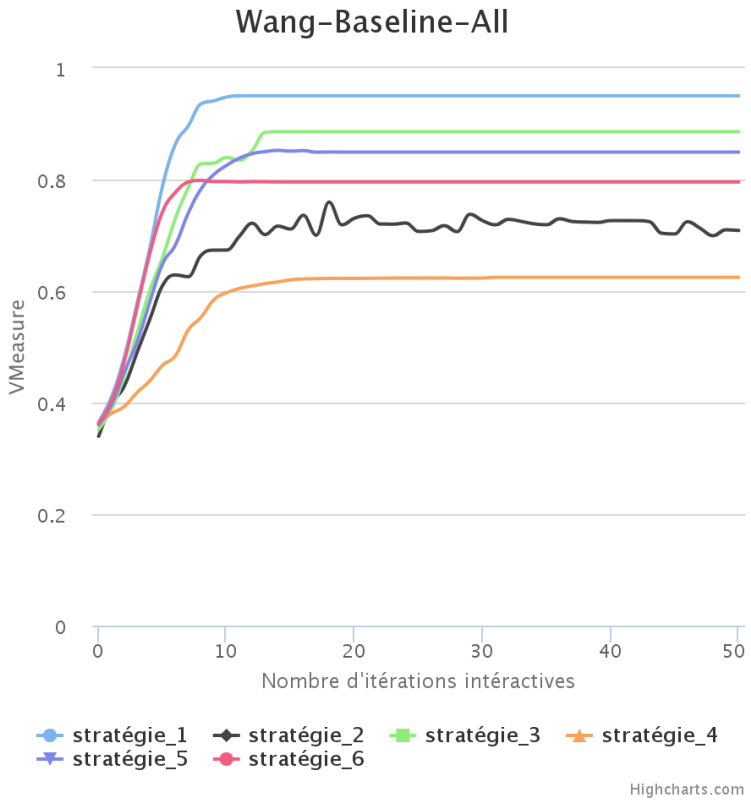
— En évaluant à la fois l'homogénéité et la compacité, la VMesure est plus complète que les mesures qui évaluent juste un seul critère.

Expérimentations et Résultats

Les expérimentations sont réalisées sur la base d'images Wang qui contient 1000 images de 10 classes séparées. Le descripteur *rgSIFT* est utilisé pour construire un dictionnaire de 200 mots. Dans chaque itération interactive, les *agents* vont interagir avec tous les 10 clusters pour générer des contraintes par paires.

Dans la figure 4.1, l'axe vertical est la performance mesurée par la VMesure $\in [0.0, 1.0]$ et l'axe horizontal est le nombre d'itérations interactives. Le temps d'exécution est affiché sous le format *heure* : *minute* : *seconde*

On trouve que la stratégie 1 qui utilise toutes les contraintes possibles donnent le meilleur résultat, mais elle prend plus de temps d'exécution. Toutes les 6 stratégies donnent des résultats assez stables. La stratégie 4, qui a une façon de déduire les contraintes très différente des autres stratégies, donne le moins bon résultat, mais avec les autres algorithmes de clustering dans les chapitres suivants, on va trouver que c'est une stratégie intéressante qui donne un bon compromis entre la performance et le temps de calcul dans un contexte de clustering interactif.



(A) Méthode de LAI Hien Phuong avec 6 stratégies

La méthode Baseline		
stratégie_1	453.6	7:33:51
stratégie_2		2:42:43
stratégie_3		0:47:21
stratégie_4		0:08:30
stratégie_5		0:11:27
stratégie_6		0:08:54

(B) Le temps d'exécution

FIGURE 2.6: Les résultats de la méthode de LAI Hien Phuong avec 6 stratégies différentes

Chapitre 3

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 d'apprentissage et les domaines connexes.

Dans la figure 3.1, on voit une vue globale d'une méthode d'apprentissage de métrique où des informations supplémentaires sont utilisées pour guider l'algorithme de clustering des visages.

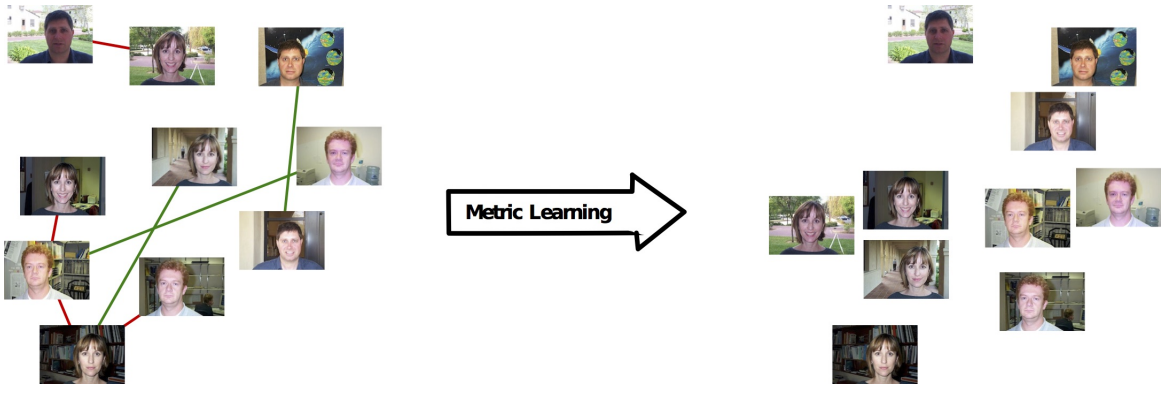


FIGURE 3.1: Une vue globale de l'apprentissage de métrique[1]
Les liens verts sont les MustLinks et ceux en rouges sont les CannotLinks

Dans ce chapitre, on porte une attention particulière à l'apprentissage de métrique de la distance de Mahalanobis, un cadre bien étudié et réussi. On va étudier l'état de l'art avec quelques approches dans le domaine d'apprentissage de métrique.

1. Lien : Aurélien Bellet - Metric learning tutorial
http : //researchers.lille.inria.fr/abellet/misc/metric_learning_tutorial.html

3.1 Introduction

3.1.1 Motivation

L'utilité de métriques et de distances est de pouvoir mesurer la ressemblance et la différence entre deux vecteurs. Il est plus probable que deux vecteurs semblables soient dans une même classe que deux vecteurs dissemblables. De fait, l'utilisation d'une métriques est une étape essentielle de l'apprentissage automatique, et en particulier de la classification et du clustering.

On peut voir un exemple de l'utilisation de la distance Euclidienne et de la distance de Mahalanobis dans la figure 3.2. On a un ensemble de données sous la forme d'une distribution gaussienne avec pour centre $(0, 0)$. On observe 4 points autour de ce centre : $point1 : (1, 1); point2 : (1, -1); point3 : (-1, 1); point4 : (-1, -1)$. On peut voir facilement que les distances Euclidiennes entre ces 4 points et le centre sont égales. Mais de façon intuitive, on trouve que $point2$ et $point4$ appartiennent à la distribution, et que $point1$ et $point3$ sont des aberrantes. La distance Euclidienne ne distingue pas ces points, mais la distance de Mahalanobis qui prend en compte les corrélations de l'ensemble de données peut fournir un résultat approprié.

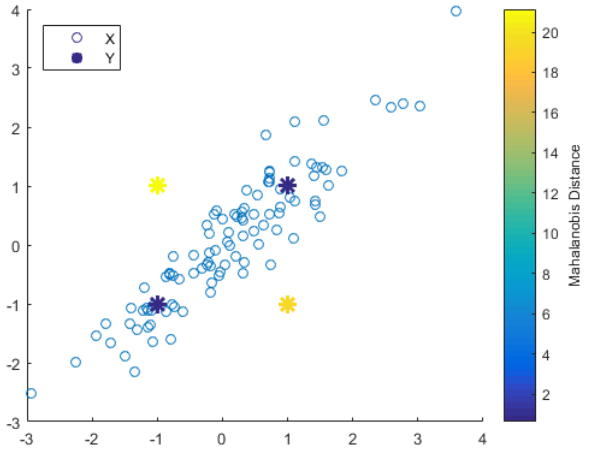


FIGURE 3.2: Un exemple de la distance de Mahalanobis. La valeur de la distance est représentée par la couleur. [2]

3.1.2 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 variance / covariance de ces d variables.

2. Lien : MathWorks Document : Mahalanobis distance
http ://fr.mathworks.com/help/stats/mahal.html ?refresh=true

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 $\mu = (\mu_1, \mu_2, \mu_3, \ldots, \mu_d)$ et est calculée en utilisant une matrice de covariance Σ comme suit :

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

On utilise la notation D_Σ pour dénoter que la distance est paramétrée par une matrice de covariance *définie positive* Σ. La distance de Mahalanobis peut aussi être définie comme la mesure de dissimilarité entre deux vecteurs aléatoires $\vec{x}$ et $\vec{y}$ de même distribution avec une matrice de covariance Σ :

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y})}$$

C'est le produit de la transposée du vecteur de différences de coordonnées de d dimensions entre les deux points, multiplié par l'inverse de la matrice de covariance et multiplié par le vecteur de différences. La distance Euclidienne correspond à la distance de Mahalanobis dans le cas où la matrice de covariance est une matrice identité, c'est-à-dire les variables sont indépendantes et normalisées par la moyenne (*normalization by mean*). Si la matrice de covariance est diagonale, on obtient la distance Euclidienne normalisée selon la formule :

$$d(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{p} \frac{(x_i - y_i)^2}{\sigma_i^2}}$$

Dans le cas normal, on travaille souvent avec deux types de la matrice de covariance : diagonale et complète.

3.2 Différents types d'approches d'apprentissage de métrique

La distance de Mahalanobis D_Σ est paramétrée par une matrice de covariance *définie positive* Σ. 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. Comme présenté dans le chapitre 2, 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 $\mathcal{M} = \{(x_i, x_j)\}$ avec x_i, x_j devraient être similaires et un ensemble de CannotLinks $\mathcal{C} = \{(x_i, x_j)\}$ avec 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. (Voir Figure 3.1 pour une illustration). Cela est généralement formulé sous forme d'un problème d'optimisation qui a la forme générale suivante :

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

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 $\lambda \geq 0$.

3.2.1 Approches globales

Comme on a vu avec la forme générale du problème d'optimisation ci-dessus, mathématiquement, le travail dans 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. Dans certains cas, il existe des structures non linéaires dans les données que les mesures linéaires sont incapables de capturer. La technique de *'kernelization'* des méthodes linéaires (astuce du noyau) peut être considérée comme une solution satisfaisante à ce problème.

Approches linéaires

MMC : *Mahalanobis Metric Learning for Clustering with Side Information*. Le travail de Xing et al. [23] est la première approche d'apprentissage de métrique de la distance de Mahalanobis. Elle repose sur une formulation convexe sans régularisation (*convex formulation with no regularization*) ; qui vise à maximiser la somme de distances entre les points dissemblables tout en gardant la somme des distances entre les points similaires.

$$\max_{\Sigma} \sum_{(x_i, x_j) \in \mathcal{M}} D_{\Sigma}(x_i, x_j)$$
$$\text{tel que} \sum_{(x_i, x_j) \in \mathcal{C}} D^2_{\Sigma}(x_i, x_j) \geq 1$$

Les auteurs ont abordé en même temps le cas diagonal (lorsque le domaine de Σ est limité à une matrice semi-définie positive diagonale), et le cas complet (lorsque le domaine de Σ est une matrice semi-définie positive complète).

LMNN : *Large Margin Nearest Neighbors*, [24] est une méthode d'apprentissage de la distance de Mahalanobis parmi les plus utilisées, et elle a nombreuses extensions (par exemple, Multi-Task LMNN de [25]). Une des raisons de sa popularité est que les contraintes sont définies d'une manière simple : pour chaque exemple d'entraînement, les k plus proches voisins de ce point devraient appartenir à une seule classe (les *"target neighbor"*), et les autres voisins de différentes classes devraient être repoussés (les *"imposteurs"*). La distance Euclidienne est utilisée pour déterminer les *"target neighbor"*.

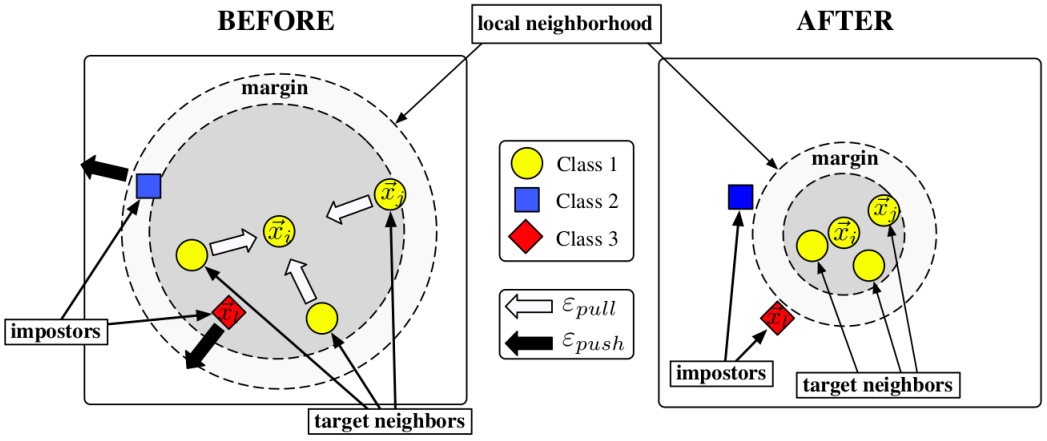


FIGURE 3.3: Illustration de la méthode LMNN [3]

L'ensemble de MustLinks et CannotLinks sont définis :

$$\mathcal{M} = \{(x_i, x_j) : \quad y_i = y_j \quad \text{et} \quad x_j \quad \text{appartient au k-voisinage de} \quad x_i\}$$
$$\mathcal{C} = \{(x_i, x_j, x_k) : \quad (x_i, x_j) \in \mathcal{M}, \quad y_i \neq y_k\}$$

La fonction objectif s'écrit comme :

$$\min_{\Sigma} \sum_{(x_i,x_j)\in\mathcal{M}} D_\Sigma(x_i, x_j) + \lambda \sum_{(x_i,x_j,x_k)\in\mathcal{C}} [1 + D_\Sigma(x_i, x_j) - D_\Sigma(x_i, x_k)]$$

L'idée de LMNN (représentée via la figure 3.3) est assez simple : le but est qu'un point de donnée devrait partager la même étiquette que ses voisins les plus proches, tandis que les points ayant des étiquettes différentes devraient être loin du point donné. Le terme *"target neighbor"* se réfère à un point qui devrait être similaire au point donné. Par contre, un *"imposteur"* est un point qui est un voisin proche, mais a une étiquette

3. Image extraite de l'article Weinberger et al. [24]
http ://papers.nips.cc/paper/2795-distance-metric-learning-for-large-margin-nearest-neighbor-classification.pdf

différente. Le but de LMNN est donc de minimiser le nombre d'imposteurs en ajustant la marge et la distance vis-à-vis des *"target neighbours"*.

Approches non linéaires

Les méthodes non linéaires globales apprennent une distances 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. La méthode de noyau (de kernelization) est une des méthodes ayant une façon de *mapping* efficace.

Récemment, plusieurs auteurs ont proposé des méthodes de kernelization basées sur l'Analyse en Composantes Principales avec le noyau (KPCA - *Kernel Principal Component Analysis* - Schölkopf et al. [26]), une extension non linéaire de PCA. En bref, KPCA projette implicitement les données dans un espace non-linéaire (potentiellement de dimension infinie) induit par un noyau et 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. [27] ont montré que le KPCA est théoriquement robuste pour les algorithmes d'apprentissage de métriques sans contrainte. Une autre astuce possible (impliquant un certain 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. [28]). Wang et al. [29] abordent le problème du choix d'une fonction de noyau approprié en utilisant de multiples noyaux pour l'apprentissage de métrique.

3.2.2 Approches locales

Les méthodes étudiées jusqu'ici apprennent des métriques globales (linéaires ou non linéaires). Cependant, si les données sont hétérogènes, une seule métrique ne peut 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). L'apprentissage de métrique locale a été montré à surpasser de manière significative les méthodes globales sur certains problèmes, mais il 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. [30] est capable d'apprendre des mesures individuelles pour chaque groupe, ce qui permet d'avoir des clusters de formes différentes. Cette méthode

est choisie pour être intégrée dans le système existant car elle a quelques points communs avec l'algorithme HMRF-KMeans : elles sont basées sur l'algorithme EM et utilisent les contraintes par paires (MustLinks et CannotLinks).

3.3 Choix d'une méthode d'apprentissage de métrique dans notre contexte

On a présenté une vue globale du système existant dans le chapitre 2, et on a vu quelques méthode d'apprentissage de métrique dans la section précédente. La méthode de LAI Hien Phuong construit d'abord un arbre CF-Tree par l'algorithme BIRCH et utilise ensuite l'algorithme HMRF-KMeans pour faire le clustering semi-supervisé interactif incrémental. On peut identifier quelques limitations de la distance Euclidienne utilisée dans sa méthode et les besoins d'une nouvelle métrique qui peut mieux répondre à l'exigence de l'utilisateur dans le contexte interactif et accélérer la convergence. L'algorithme d'apprentissage de métrique MPCKMeans (Metric with Pairwise Constraints KMeans) de Bilenko et al. [30] est donc choisie pour remplacer l'algorithme HMRF-KMeans dans l'étape de reclustering interactif.

Supposons que les données sont générées par un mélange de distributions probabilistes (qui sont représentées par des modèles mathématiques). On peut considérer un cluster comme un ensemble des points qui appartiennent à une même distribution. Le problème de clustering des données en K clusters maintenant devient un problème d'estimation des paramètres de K modèles probabilistes pour que des données d'entrée s'adaptent à ce modèle de K composantes de façon la plus parfaite possible. Donc, pour chaque composant / cluster, il faut trouver des paramètres optimaux et assigner des points aux composants correspondants pour maximiser la vraisemblance (*likelihood*) entre des données et le modèle de mélange.

Dans l'algorithme MPCKMeans, on utilise les notations suivantes :

- L'ensemble des données $\mathcal{X} = \{x_i\}_{i=1}^{N}$
- L'ensemble des contraintes *MustLinks* $\mathcal{M} = \{(x_i, x_j)\}$
- L'ensemble des contraintes *CannotLinks* $\mathcal{C} = \{(x_i, x_j)\}$
- Le nombre de clusters K, chaque i ème cluster a une étiquette l_i
- Deux constantes pondérées (*weighted constant*) des contraintes MustLinks et CannotLinks $\omega, \overline{\omega}$

Le résultat attendu : K sous-ensembles disjoints $\{\mathcal{X}_h\}_{h=1}^{K}$ sont créés progressivement de façon que la fonction objectif $\mathcal{J}_{obj}$ soit minimisée localement.

On note l_i l'étiquette du cluster de x_i. La fonction objectif $\mathcal{J}_{obj}$ est définie dans l'équation 3.1 :

$$\begin{aligned}\mathcal{J}_{obj} = & \sum_{x_i \in \mathcal{X}} (\| x_i - \mu_{l_i} \|^2_{A_{l_i}} - log(det(A_{l_i}))) \\ & + \sum_{(x_i,x_j)\in\mathcal{M}} \omega_{i,j} f_M(x_i, x_j) \mathbb{1}[h \neq l_j] \\ & + \sum_{(x_i,x_j)\in\mathcal{C}} \overline{\omega_{i,j}} f_C(x_i, x_j) \mathbb{1}[h = l_j]\end{aligned} \tag{3.1}$$

Le premier terme mesure la distorsion entre un point de donnée x_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 images x_i et x_j, et sont formulés par les deux équations 3.2 et 3.3. 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, $(x_{l_i}\prime - x_{l_i}\prime\prime)$ est la paire des deux points les plus éloignés (selon le métrique apprise) du i ème cluster.

$$f_M(x_i, x_j) = \frac{1}{2}\| x_i - x_j \|^2_{A_{l_i}} + \frac{1}{2}\| x_i - x_j \|^2_{A_{l_j}} \tag{3.2}$$

$$f_C(x_i, x_j) = \| x_{l_i}\prime - x_{l_i}\prime\prime \|^2_{A_{l_i}} + \| x_i - x_j \|^2_{A_{l_i}} \tag{3.3}$$

L'algorithme est détaillé dans le pseudo-code suivant. Le résultat expérimental de cet algorithme appliqué sur la base Wang avec des contraintes par paires entre images est présenté dans l'annexe C.

Algorithme 2 : L'algorithme MPCKMeans

1. Initialisation

(a) Créer λ voisinages $\{N_p\}_{p=1}^{\lambda}$ à partir de l'ensemble des contraintes $\mathcal{M}$ et $\mathcal{C}$

(b) 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* **faire**

(a) *assign_cluster* : Assigner chaque point de données x_i à un cluster h^* qui satisfait l'équation 3.4 :

$$\begin{aligned} h^* = argmin(\| x_i - \mu_h^{(t)} \|_{A_h}^2 - log(det(A_h)) \\ + \sum_{(x_i,x_j)\in\mathcal{M}} \omega_{i,j} f_M(x_i, x_j) \mathbb{1}[h \neq l_j] \\ + \sum_{(x_i,x_j)\in\mathcal{C}} \overline{\omega_{i,j}} f_C(x_i, x_j) \mathbb{1}[h = l_j]) \end{aligned} \tag{3.4}$$

(b) *estimate_means*(*M_Step_A*) :

$$\{\mu_h^{(t+1)}\}_{h=1}^{K} \leftarrow \{\frac{1}{|\mathcal{X}_h^{(t+1)}|} \sum_{x\in\mathcal{X}_h^{(t+1)}} x\}_{h=1}^{K} \tag{3.5}$$

(c) *update_metrics*(*M_Step_B*) :

$$\begin{aligned} A_h = |\mathcal{X}_h|(\sum_{x_i\in\mathcal{X}_h} (x_i - \mu_h)(x_i - \mu_h)^T \\ + \sum_{(x_i,x_j)\in\mathcal{M}} \frac{1}{2} \omega_{i,j} (x_i - x_j)(x_i - x_j)^T \mathbb{1}[h \neq l_j] \\ + \sum_{(x_i,x_j)\in\mathcal{C}} \overline{\omega_{i,j}} ((x_h\prime - x_h\prime\prime)(x_h\prime - x_h\prime\prime)^T - (x_i - x_j)(x_i - x_j)^T) \mathbb{1}[h = l_j])^{-1} \end{aligned} \tag{3.6}$$

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

fin

Chapitre 4

Intégration de l'apprentissage de métrique dans le système existant

La méthode de Lai et al. [21] (appelée désormais la méthode *Baseline*) est une adaptation de l'algorithme HMRF-KMeans [17] sur les CF-Entrées dans le contexte interactif. L'arbre CF-Tree est créé grâce à un algorithme de clustering hiérarchique (BIRCH [9]) en utilisant la distance Euclidienne pour la construction et la division. Ses feuilles contiennent toutes les CF-Entrées, chacune de ces dernières est considérée comme une représentation de quelques images semblables. Ces CF-Entrées sont regroupées de façon incrémentale par HMRF-KMeans. Dans chaque itération, les retours de l'utilisateur sous la forme de clics positifs et négatifs permettent de déduire des *contraintes par paires* entre des CF-Entrées, et ensuite sont fournis à HMRF-KMeans.

Dans la section 3.1.2, on a discuté de la distance de Mahalanobis et aussi de l'apprentissage de métrique. Dans le contexte du clustering semi-supervisé en utilisant des *contraintes par paires* entre images, Basu et al. [17], théoriquement et pratiquement, a prouvé que l'apprentissage de métrique (MPCK-Means : Metric with Pairwise Constraints K-Means) est meilleur que les autres approches sans apprentissage de métrique (Supervised K-Means ou PCK-Means : Pairwise Constraints K-Means). Ça a ouvert une perspective d'application de l'apprentissage de métrique dans le contexte interactif du système existant.

Dans la méthode *Baseline*, l'arbre CF-Tree est utilisé comme une représentation hiérarchique intermédiaire et les CF-Entrées dans leurs feuilles vont être divisées ou être fusionnées dans les itérations interactives selon les retours de l'utilisateur. Le CF-Tree est modifié en fonction d'une mesure de la dissimilarité, qui est contrôlée par une distance donnée. Si cette distance est susceptible d'être apprise, la mesure de la dissimilarité sera

de plus en plus proche de l'exigence de l'utilisateur et la performance de l'algorithme de clustering va être améliorée.

Donc, l'intention de ce chapitre est d'intégrer l'apprentissage de métrique dans la méthode *Baseline* avec l'intervention de l'utilisateur pour améliorer la performance en général dans le contexte incrémental. D'abord, la section 4.1 introduit la méthode proposée, la motivation et les étapes détaillées de cette méthode. L'implémentation avec le workflow, l'interprétation des formules mathématiques, les défis techniques et les solutions sont présentés dans la section 4.2. Le protocole des expérimentations, les résultats expérimentaux et les analyses se trouvent dans la section 4.3. Enfin, quelques discussions sur les résultats obtenus et la conclusion sont dans la section 4.4

4.1 Méthode proposée

4.1.1 Motivation

Dans le contexte où la mesure de dissimilarité et la distance 'physique' entre deux objets sont la même, la distance Euclidienne est dominante car sa formule est simple et assez efficace. En pratique, deux objets proches qui ont, mathématiquement, une petite valeur de distance, ne sont pas toujours similaires. On a besoin d'une autre mesure de la distance qui peut refléter la dissimilarité de façon plus proche des dissimilarités perçues par l'utilisateur humain dans la collection. La méthode Baseline a résolu jusqu'à un certain point le problème du fossé sémantique entre l'exigence de l'utilisateur et le résultat réel du modèle de clustering en utilisant les retours de l'utilisateur. En considérant l'avantage de la distance de Mahalanobis (invariante à l'échelle, prend en compte les corrélations de l'ensemble de données - McLachlan [31]), ça a ouvert une nouvelle direction d'application de cette distance dans le système existant. Dans notre contexte, on travaille sur une base de données multi-dimensionnelle, chaque vecteur contient plusieurs composantes. À la différence de la distance Euclidienne où toutes les composantes des vecteurs sont traitées indépendamment et de la même façon, la distance de Mahalanobis accorde un poids moins important aux composantes les plus dispersées. Théoriquement, on peut s'attendre à un meilleur résultat du modèle de clustering en appliquant la distance de Mahalanobis et l'apprentissage de métrique. Le but principal des algorithmes d'apprentissage de métrique est de maintenir un mécanisme de mise à jour des paramètres de la distance de Mahalanobis : le vecteur de valeurs moyennes et la matrice de covariance. Comme introduit dans la section 3.3 , l'algorithme de clustering semi-supervisé MPCK-Means (*Metric Pairwise Constraints KMeans*) [30] est une approche d'apprentissage de métrique très souple avec différentes configurations. On obtient de bons résultats quand

on applique ces algorithmes (au niveau individuel des images, sans prendre en compte de structure hiérarchique comme dans l'algorithme Baseline) sur la base choisie (Wang - cf. Annexe C). Dans le système existant, l'ensemble de CF-Entrées fourni par l'algorithme BIRCH est utilisé comme les données d'entrée du modèle de clustering. La méthode proposée va essayer d'adapter des algorithmes d'apprentissage de métrique sur une base de CF-Entrées en s'attendrant à une meilleure performance (mesurée par la VMesure). Comme la distance de Mahalanobis reflète mieux la dissimilarité, on peut espérer une convergence plus rapide. En plus, la méthode Baseline prend beaucoup de temps (pour quelques stratégies spéciales), on veut techniquement réduire un peu le temps d'exécution de l'étape de (re)clustering pour que notre système soit bien adapté dans un contexte interactif incrémental.

4.1.2 Présentation de la méthode

Le pipeline commun d'un système d'apprentissage automatique comprend deux étapes principales : (1) Présentation de données d'entrée de la façon la plus représentative pour que les algorithmes d'apprentissage puissent en bénéficier. (2) Construction des modèles, les entraîner et les exploiter selon les tâches particulières : classification, prédiction, clustering, ... Les détails des étapes du système existant sont présentés dans la section 2.4.1. Les descripteurs sont extraits à partir des images originales pour construire un dictionnaire. Les images représentées via ce dictionnaire sont regroupées de façon incrémentale par un algorithme de clustering semi-supervisé interactif. Dans la méthode Baseline, la distance Euclidienne est utilisée partout, en particulier :

— Dans la construction de l'arbre CF-Tree où un objet est inséré dans la feuille la plus proche.

— Dans l'étape de division des CF-Entrées pour diviser une CF-Entrée (qui a des contraintes incohérentes) en plusieurs CF-Entrées qui sont plus pures.

— Dans l'étape qui consiste à assigner chaque point de données à un meilleur cluster.

Dans tous les cas : la construction ou la réorganisation de l'arbre, on a toujours besoin d'une métrique efficace. On peut donc intervenir en deux points : (1)Remplacer la distance Euclidienne dans l'étape de construction et de division de l'arbre par la distance de Mahalanobis. (2) Remplacer l'algorithme HMRF-KMeans qui fait le clustering de façon interactive par des nouveaux algorithmes d'apprentissages de métrique. On garde inchangées les autres étapes de la méthodes Baseline, présentée dans la partie 2.4. La nouvelle méthode de ce chapitre utilise les mêmes stratégies qui sont présentées dans le tableau 2.1.

4.2 Implémentation de la méthode

L'algorithme MPCKMeans s'adapte dans le système existant va recevoir l'ensemble des CF-Entrées comme données d'entrée. Les étapes détaillées sont présentées dans la section 3.3. On doit redéfinir 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 : $\mathcal{M} = \{(CF_i, CF_j)\}$, $\mathcal{C} = \{(CF_i, CF_j)\}$.

Au niveau de chaque cluster k, on va noter son ensemble de points $\mathcal{X}_k$. On doit souvent calculer la distance entre deux points de données quelconques (x_i, x_j) ou bien la distance entre un point quelconque x_i et un cluster k selon la métrique de chaque cluster qui est paramétré par la moyenne μ_k et la matrice de covariance $\Sigma_k = {A_k}^{-1}$.

$$\Sigma_k = \frac{1}{|\mathcal{X}_k|}(\mathcal{X}_k - \mu_k)(\mathcal{X}_k - \mu_k)^T$$

Si on a besoin de calculer les distances entre un ensemble de points $X = \{x_1, x_2, ..., x_t\}$ et un cluster k, on doit répéter t fois le calcul de la distance pour chaque point x_i dans l'ensemble X :

$$\begin{aligned} d^2(x_1, \mu_k) &= (x_1 - \mu_k)^T(\Sigma_k)^{-1}(x_1 - \mu_k) \\ d^2(x_2, \mu_k) &= (x_2 - \mu_k)^T(\Sigma_k)^{-1}(x_2 - \mu_k) \\ &\dots \\ d^2(x_t, \mu_k) &= (x_t - \mu_k)^T(\Sigma_k)^{-1}(x_t - \mu_k) \end{aligned} \tag{4.1}$$

En appliquant la technique de vectorisation, on peut transformer les calculs *'élément par élément'* à une opération sur les vecteurs. Le code vectorisé qui apparaît plus dans les expressions mathématiques, est souvent beaucoup plus rapide que le code contenant des boucles correspondantes. On remplace les t équations du calcul de la distance de Mahalanobis entre t points x_i et un cluster par une équation du calcul de la distance entre un ensemble X et un cluster :

$$d^2(X, \mu_k) = (X - \mu_k)^T(\Sigma_k)^{-1}(X - \mu_k)$$

On est en train de travailler localement sur un cluster, on peut enlever l'indice k dans les formules. Pour éviter le calcul de la matrice inverse, on va utiliser une technique de factorisation de matrice. Les formules mathématiques suivantes sont appliquées **localement** dans chaque cluster. On utilise la décomposition de la valeur propre (*Eigen Value*

Decomposition) [32] pour décomposer la matrice de covariance symétrique :

$$\Sigma = EDE^T$$

où E est une matrice carrée dont chaque colonne est un vecteur propre (*eigen vector*) de Σ et D est une matrice diagonale dont les coefficients diagonaux sont les valeurs propres correspondantes.

On peut donc interpréter la formule de la distance de Mahalanobis entre un ensemble de points X et un cluster comme suit :

$$\begin{aligned} d^2(X,\mu) &= (X-\mu)^T(EDE^T)^{-1}(X-\mu) \\ &= (X-\mu)^T ED^{-1}E^T(X-\mu) \end{aligned} \tag{4.2}$$

Le cluster actuel qui été caractérisé par la matrice de covariance Σ, est maintenant caractérisé par la matrice des valeurs propres D et la matrice de vecteur propre E qui contient des composantes principales dans un nouvel espace orthogonal. Si on projette l'ensemble des données X dans le nouvel espace : $X_{projete} = E^T(X-\mu)$, la distance de Mahalanobis devient la distance Euclidienne dans le nouvel espace.

$$\begin{aligned} d^2(X,\mu) &= X_{projete}^T D^{-1} X_{projete} \\ d(X,\mu) &= X_{projete}^T D^{\frac{-1}{2}} X_{projete} \end{aligned} \tag{4.3}$$

D est une matrice diagonale dont les coefficients diagonaux sont les valeurs propres, donc, le calcul de la distance est maintenant plus facile et plus efficace.

4.3 Résultats expérimentaux

Dans cette section, premièrement le protocole d'expérimentation est présenté. Deuxièmement, les résultats obtenus sont présentés avec quelque analyses : d'abord, c'est une vue d'ensemble de tous les résultats expérimentaux, et ensuite, les résultats détaillés pour quelques méthodes précises sont présentés et analysés. Et enfin, on va observer le temps d'exécution de chaque expérimentation. La comparaison des méthodes se trouve dans la section de discussion.

4.3.1 Protocole d'expérimentation

Toutes les expérimentations sont réalisées sur la base d'images Wang qui contient 1000 images de 10 classes séparées. Chaque expérimentation est lancée 5 fois sous les même

configurations et la valeur finale est calculée par la moyenne de ces 5 résultats. La machine utilisée pour tous les expérimentations est un laptop avec *Ubuntu 15.04 vivid 64 bit, CPU Intel 2.4GHz, RAM 4GB, g++ 4.9.2.*

Les représentations de ces images sous la forme des vecteurs de fréquence des mots visuels sont utilisées pour entraîner le modèle de clustering dans le système existant. D'abord, ce système prend ces 1000 vecteurs pour donner à l'algorithme BIRCH. 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.

Comme présenté dans la partie 4.1.2, on peut remplacer la distance Euclidienne dans l'étape de construction et de réorganisation de l'arbre CF-Tree par la distance de Mahalanobis, et on peut remplacer HMRF-KMeans par un des algorithmes d'apprentissage de métrique.

Donc, on a trois cas pour les expérimentations :

— La distance Euclidienne est utilisée partout (dans la construction, dans la division et dans le clustering interactif incrémental). C'est la méthode Baseline.

— La distance de Mahalanobis est utilisée partout et on n'utilise plus la distance Euclidienne. C'est-à-dire, on utilise cette nouvelle distance pour construire l'arbre CF-Tree et pour diviser les CF-Entrées dans chaque itération interactive. Pour le reclustering, on utilise l'apprentissage de métrique qui bénéficie systématiquement de la distance de Mahalanobis.

— On peut simplement garder la distance Euclidienne pour la construction et la division de l'arbre CF-Tree et appliquer la distance de Mahalanobis pour le reclustering interactif incrémental via des algorithme d'apprentissage de métrique.

Le tableau 4.1 montre toutes les méthodes utilisées pour faire l'expérimentation. Les méthode d'apprentissage de métrique qui utilisent la distance de Mahalanobis dépendent des paramètres de forme des matrices de covariance. On peut utiliser 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*). En plus, la forme de la matrice de covariance est aussi variée : une forme diagonale (*MPCKMEANS_DIAGONAL*) ou une 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.*

La base Wang	Méthode Baseline	
	L'apprentissage de métrique avec la distance Euclidienne pour la construction et la division de l'arbre CF-Tree	MPCKMEANS_GLOBAL_DIAGONAL
		MPCKMEANS_GLOBAL_FULL
		MPCKMEANS_LOCAL_DIAGONAL
		MPCKMEANS_LOCAL_FULL
	L'apprentissage de métrique avec la distance de Mahalanobis pour la construction et la division de l'arbre CF-Tree	MPCKMEANS_GLOBAL_DIAGONAL
		MPCKMEANS_LOCAL_DIAGONAL

TABLE 4.1: Les méthodes pour l'expérimentation sur la base Wang

4.3.2 Analyses des résultats obtenus

Notre système est mis dans le contexte interactif incrémental avec l'intervention de l'utilisateur. Chaque fois que le modèle de clustering fournit un résultat, il est présenté à l'utilisateur et le système reçoit les retours pour guider le modèle de clustering à savoir s'adapter. Le système continue à refaire l'étape de clustering pour se rapprocher de besoins de l'utilisateur. Cette boucle est comptée comme une *itération interactive*. Pour chacune des méthodes, on va exécuter 50 itérations interactives et examiner les mesures suivantes :

— La valeur minimum de VMesure sur 50 itérations.

— La valeur maximum de VMesure sur 50 itérations.

— La valeur moyenne de VMesure sur 50 itérations.

— La valeur de la VMesure après la dixième itération (Pour voir si la tendance de la convergence est précoce ou tardive)

— Dans les 40 dernières itérations (de 11 à 50), la valeur de l'écart type (déviation standard) de la VMesure (Pour voir si la méthode est stable ou non)

— La méthode va arriver à 80 % (VMeasure = 0.8) en combien d'itérations ? (Théoriquement, l'algorithme va s'arrêter quand il satisfait les besoins de l'utilisateur, on peut donc définir un seuil de la satisfaction de l'utilisateur, par exemple quand la VMeasure est environ 0.8)

— La méthode va arriver à 90 % (VMeasure = 0.9) en combien d'itérations ?

— On arrive à la performance maximum (La valeur maximum de VMeasure) en combien d'itérations ?

La vue d'ensemble sur tous les résultats expérimentaux

Les résultats sont présentés dans les tableaux 4.2 et 4.3 Dans chaque cellule, la valeur colorée en bleu est la meilleure, celle colorée en rouge est la plus mauvaise.

Ci-après quelques commentaires préliminaire :

— En regardant le critère de la valeur maximum de la VMeasure sur 50 itérations, la méthode qui donne le meilleur résultat (la plus haute VMeasure) est MPCKMEANS_GLOBAL_FULL avec la stratégie 4 ($VMeasure = 0.9976$).

— Pour la méthode Baseline, la stratégie 1 donne toujours le meilleur résultat. Pour la méthode MPCKMEANS_GLOBAL_DIAGONAL, la stratégie 1 est aussi la meilleure. Mais la stratégie 4 est la meilleure pour la méthode MPCKMEANS_GLOBAL_FULL et la stratégie 5 est la meilleure pour la méthode MPCKMEANS_LOCAL_DIAGONAL.

— Selon la valeur de l'écart type (déviation standard) de la VMeasure, on trouve que la méthode Baseline est la plus stable. On a aussi une stabilisation chez méthode MPCKMEANS_LOCAL_FULL avec la stratégie 1. Mais en général, les algorithmes d'apprentissage de métrique sont moins stables.

— 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'intérieur 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 quelques stratégies de la méthode MPCKMEANS_LOCAL_DIAGONAL, on n'obtient pas de résultat à cause de la convergence vers une **valeur locale minimale**. C'est un phénomène dans l'étape de clustering (dans une seule étape de reclustering incrémental), après quelques itérations, la valeur de la fonction objective ne diminue pas, mais augmente même. En comparaison avec les autres méthodes qui fournissent des résultats acceptables, on peut dire que cette méthode est moins bonne que les autres.

	La méthode BASELINE	L'apprentissage de métrique avec l'approche globale		L'apprentissage de métrique avec l'approche locale	
		MPCKMEANS GLOBAL_DIAGONAL	MPCKMEANS GLOBAL_FULL	MPCKMEANS LOCAL_DIAGONAL	MPCKMEANS LOCAL_FULL
VMesure Min :	s1 : 0.37 s2 : 0.34 s3 : 0.35 s4 : 0.36 s5 : 0.36 s6 : 0.36	s1 : 0.3596 s2 : 0.3630 s3 : 0.3765 s4 : 0.3723 s5 : 0.3641 s6 : 0.3814	s1 : 0.3154 s2 : 0.3637 s3 : 0.3528 s4 : 0.3260 s5 : 0.3337 s6 : 0.3564	s1 : 0.3445 s2 : - s3 : - s4 : 0.2822 s5 : 0.3662 s6 : -	-
VMesure Max :	s1 : 0.95 s2 : 0.76 s3 : 0.89 s4 : 0.62 s5 : 0.85 s6 : 0.80	s1 : 0.9844 s2 : 0.7579 s3 : 0.6851 s4 : 0.9329 s5 : 0.9501 s6 : 0.9170	s1 : 0.9806 s2 : 0.8684 s3 : 0.8487 s4 : 0.9976 s5 : 0.9963 s6 : 0.9733	s1 : 0.8851 s2 : - s3 : - s4 : 0.6561 s5 : 0.9587 s6 : -	-
VMesure Avg :	s1 : 0.90 s2 : 0.68 s3 : 0.83 s4 : 0.59 s5 : 0.80 s6 : 0.76	s1 : 0.8897 s2 : 0.6439 s3 : 0.5951 s4 : 0.8024 s5 : 0.8400 s6 : 0.7805	s1 : 0.9175 s2 : 0.6636 s3 : 0.7575 s4 : 0.8765 s5 : 0.8905 s6 : 0.8454	s1 : 0.7830 s2 : - s3 : - s4 : 0.5263 s5 : 0.8632 s6 : -	-
À la dixième itération, le VMesure est :	s1 : 0.9480 s2 : 0.6735 s3 : 0.8395 s4 : 0.5971 s5 : 0.8255 s6 : 0.7964	s1 : 0.8935 s2 : 0.6197 s3 : 0.5104 s4 : 0.7437 s5 : 0.8612 s6 : 0.7573	s1 : 0.9648 s2 : 0.6933 s3 : 0.7559 s4 : 0.8638 s5 : 0.8602 s6 : 0.9001	s1 : 0.6406 s2 : - s3 : - s4 : 0.5607 s5 : 0.9430 s6 : -	-

TABLE 4.2: Les résultats expérimentaux sur la base Wang avec la méthode Baseline et trois méthodes d'apprentissage de métrique, en utilisant la distance Euclidienne pour la construction et la divison des CF-Entrées (1)
La valeur colorée en bleu est la meilleure, celle colorée en rouge est la plus mauvaise.

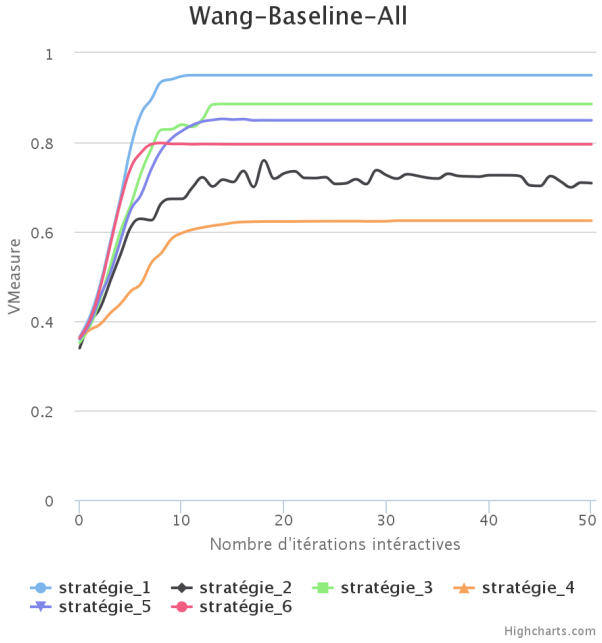
	La méthode BASELINE	L'apprentissage de métrique avec l'approche globale		L'apprentissage de métrique avec l'approche locale	
		MPCKMEANS GLOBAL_DIAGONAL	MPCKMEANS GLOBAL_FULL	MPCKMEANS LOCAL_DIAGONAL	MPCKMEANS LOCAL_FULL
Dans les 40 dernières itérations, l'écart type (déviation standard) de la VMesure est :	s1 : 0.0003 s2 : 0.0140 s3 : 0.0116 s4 : 0.0058 s5 : 0.0041 s6 : 0.0001	s1 : 0.0215 s2 : 0.0415 s3 : 0.0335 s4 : 0.0455 s5 : 0.0234 s6 : 0.0371	s1 : 0.0037 s2 : 0.1188 s3 : 0.0373 s4 : 0.0397 s5 : 0.0413 s6 : 0.0394	s1 : 0.0450 s2 : - s3 : - s4 : 0.0966 s5 : 0.0398 s6 : -	-
Arriver à 80% en :	s1 : 6 iterations s2 : n'y arrive pas s3 : 8 iterations s4 : n'y arrive pas s5 : 9 iterations s6 : n'y arrive pas	s1 : 8 iterations s2 : n'y arrive pas s3 : n'y arrive pas s4 : 13 iterations s5 : 8 iterations s6 : 11 iterations	s1 : 6 iterations s2 : 12 iterations s3 : 28 iterations s4 : 9 iterations s5 : 6 iterations s6 : 7 iterations	s1 : 12 iterations s2 : - s3 :- s4 : n'y arrive pas s5 : 5 iterations s6 : -	-
Arriver à 90% en :	s1 : 8 iterations s2 : n'y arrive pas s3 : n'y arrive pas s4 : n'y arrive pas s5 : n'y arrive pas s6 : n'y arrive pas	s1 : 11 iterations s2 : n'y arrive pas s3 : n'y arrive pas s4 : 31 iterations s5 : 11 iterations s6 : 43 iterations	s1 : 7 iterations s2 : n'y arrive pas s3 : n'y arrive pas s4 : 15 iterations s5 : 7 iterations s6 : 7 iterations	s1 : n'y arrive pas s2 : - s3 : - s4 : n'y arrive pas s5 : 10 iterations s6 : -	-
Arriver à la performance maximum en :	s1 : 11 iterations s2 : 18 iterations s3 : 14 iterations s4 : 31 iterations s5 : 14 iterations s6 : 8 iterations	s1 : 42 iterations s2 : 39 iterations s3 : 13 iterations s4 : 48 iterations s5 : 49 iterations s6 : 43 iterations	s1 : 49 iterations s2 : 46 iterations s3 : 44 iterations s4 : 48 iterations s5 : 33 iterations s6 : 26 iterations	s1 : 35 iterations s2 : - s3 : - s4 : 48 iterations s5 : 48 iterations s6 : -	-

TABLE 4.3: Les résultats expérimentaux sur la base Wang avec la méthode Baseline et trois méthodes d'apprentissage de métrique, en utilisant la distance Euclidienne pour la construction et la divison des CF-Entrées (2)
La valeur colorée en bleu est la meilleure, celle colorée en rouge est la plus mauvaise.

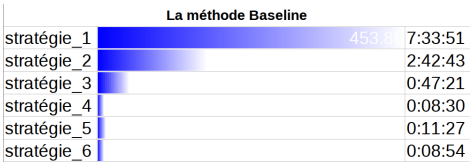
Les résultats détaillés pour quelques méthodes avec apprentissage de métrique

Les résultats expérimentaux de quelques méthodes sélectionnées avec différentes stratégies de déductions des contraintes sont introduits dans l'annexe D.1. Ici on va voir les résultats en détail de la méthode Baseline et de la méthode MPCKMEANS_GLOBAL_DIAGONAL avec les deux types de distance pour la construction et la réorganisation des CF-Entrées. 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*

Méthode Baseline : la figure 4.1



(A) La méthode Baseline avec 6 stratégies

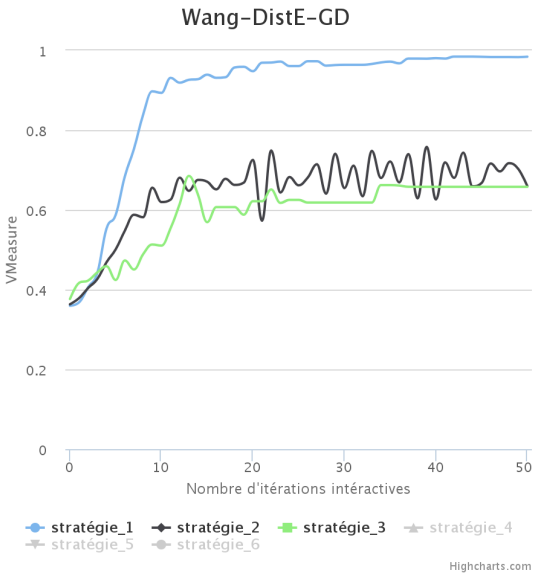


(B) Le temps d'exécution

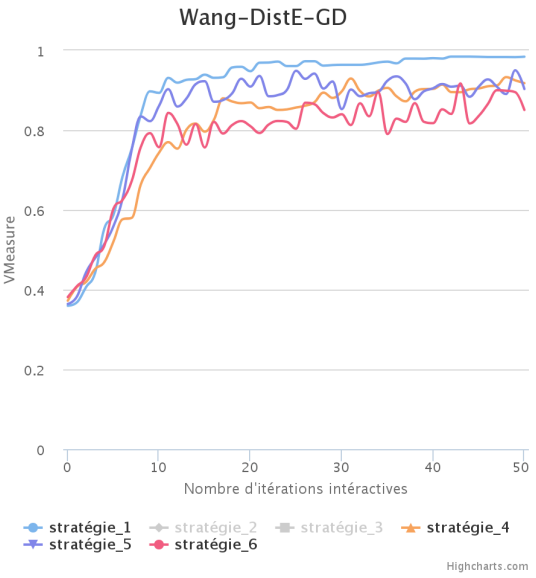
FIGURE 4.1: La méthode Baseline

*Méthode MPCKMEANS_GLOBAL_DIAGONAL (*distE*)* : la figure 4.2

Dans cette méthode, la stratégie 1 qui déduit toutes les contraintes possibles a fourni le meilleur résultat. Le seul désavantage de cette stratégie est le temps d'exécution. La stratégie 2 et 3 possèdent le temps d'exécution acceptable mais elles ne donnent pas un bon résultat. Par contre, la stratégie 5 et 6 qui utilisent des contraintes sélectionnées possèdent des résultats meilleurs que ceux de deux stratégies précédentes. En comparaison avec la stratégie 5, la stratégie 6 optimise la déduction de contraintes et économise le temps d'exécution mais son résultat est toujours moins bon. Pour le reste, la stratégie 4



(A) Stratégies 1, 2, 3



(B) Stratégies 1, 4, 5, 6

MPCKMEANS_GLOBAL_DIAGONAL avec la distance *Euclidienne*	
stratégie_1	0:57:21
stratégie_2	0:01:05
stratégie_3	0:02:26
stratégie_4	0:00:37
stratégie_5	0:01:36
stratégie_6	0:01:12

(C) Le temps d'exécution

FIGURE 4.2: MPCKMEANS_GLOBAL_DIAGONAL : 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

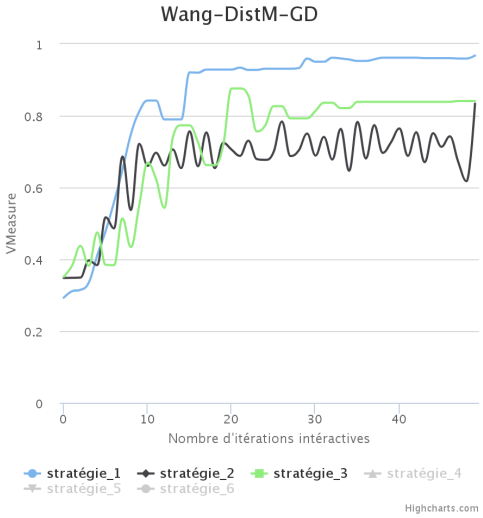
donne un résultat très remarquable. Elle prend seulement environ 37 secondes pour exécuter toutes les 50 boucles de reclustering incrémental. Le résultat de cette stratégie au début n'est pas très bon, mais on voit la tendance d'augmentation après chaque itération interactive.

*Méthode MPCKMEANS_GLOBAL_DIAGONAL (*distM*)* : la figure 4.3

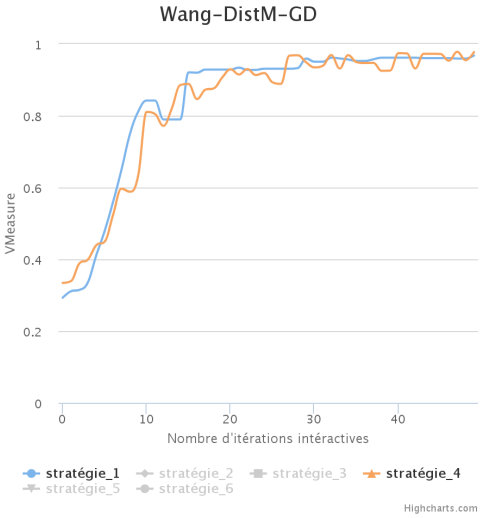
Les commentaires pour la méthode MPCKMEANS_GLOBAL_DIAGONAL en utilisant la distance Euclidienne sont encore vrais quand on applique la distance de Mahalanobis pour la construction et la réorganisation de l'arbre CF-Tree. Les stratégies 2 et 3 sont toujours moins bonnes que les stratégies 5 et 6. Dans ce cas, la stratégie 4 est la meilleure car elle surpasse la stratégie 1 au niveau de la VMesure et de son temps d'exécution.

Le temps d'exécution de chaque expérimentation

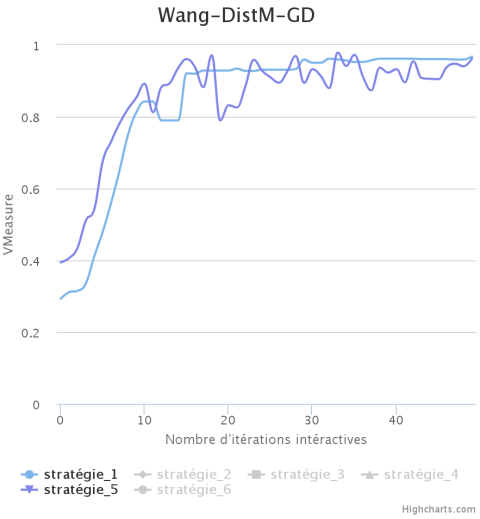
La figure 4.4a présente le temps d'exécution de la stratégie 1 de toutes les méthodes et de la méthode Baseline avec toutes les différentes stratégies. La figure 4.4b montre que



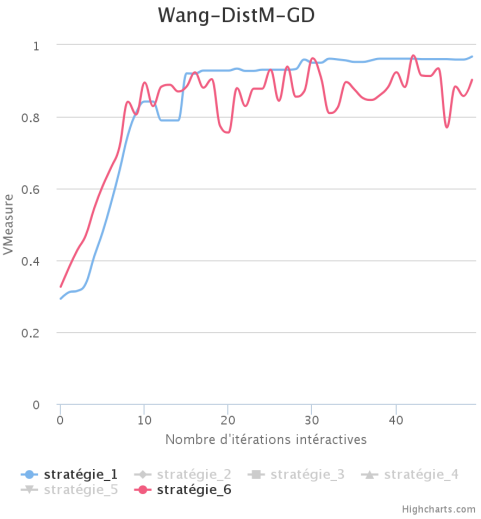
(A) Stratégies 1, 2 et 3



(B) Stratégies 1 et 4



(C) Stratégies 1 et 5



(D) Stratégies 1 et 6

MPCKMEANS_GLOBAL_DIAGONAL avec la distance de *Malahanobis*		
stratégie_1	52.7	0:52:44
stratégie_2		0:01:34
stratégie_3		0:02:53
stratégie_4		0:00:47
stratégie_5		0:01:55
stratégie_6		0:01:26

(E) Le temps d'exécution

FIGURE 4.3: MPCKMEANS_GLOBAL_DIAGONAL : L'apprentissage de métrique avec une matrice de covariance globale diagonale, en utilisant la distance de Mahalanobis pour la construction et la division de l'arbre CF-Tree

l'approche locale des algorithmes d'apprentissage de métrique prend beaucoup plus de temps que l'approche globale. Elle montre aussi que l'approche utilisant la matrice de covariance diagonale est toujours plus rapide que celle utilisant la matrice de covariance complète. Dans notre système, le temps pour générer et déduire des contraintes occupe la plupart du temps d'exécution dans chaque itération. Selon les résultats expérimentaux des algorithmes d'apprentissage de métrique avec des contraintes au niveau images individuelles (dans la section 3.3), le nombre de contraintes violées est de plus en plus faible.

Dans le système existant, quand on a un CF-Entrée incohérent qui contient en même temps un MustLink et un CannotLink entre leurs images, il faut le 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 moins de nouveaux CF-Entrées créés 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 proposée par LAI Hien Phuong. En plus, l'implémentation de l'algorithme MPCKMeans est optimisé au niveau du compilateur (contrôlé par *CMake*[1]) et de la librairie utilisée (*Eigen*[2]). Donc on observe une très nette amélioration du temps d'exécution, qui passe de plus de sept heures à moins d'une heures pour la stratégie 1 et 50 itérations interactives.

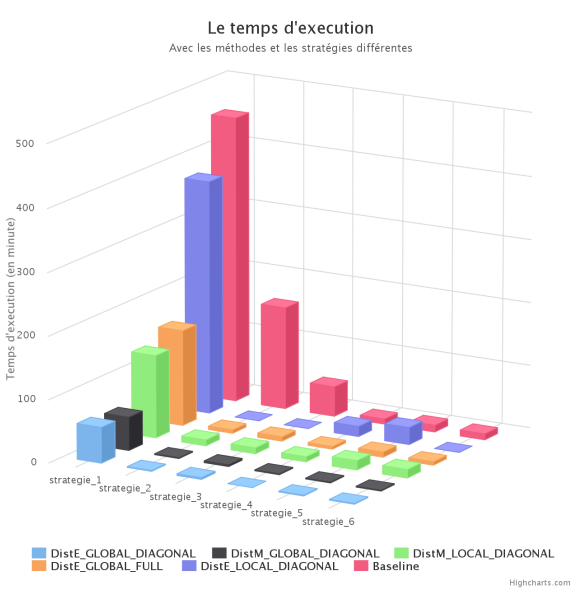
4.4 Discussion et Conclusion

Pour la discussion, on va faire la comparaison entre les méthodes présentées, donner des perspectives pour améliorer le résultat de la méthode proposée, et conclure sur la contribution que nous avons apportée avec cette méthode.

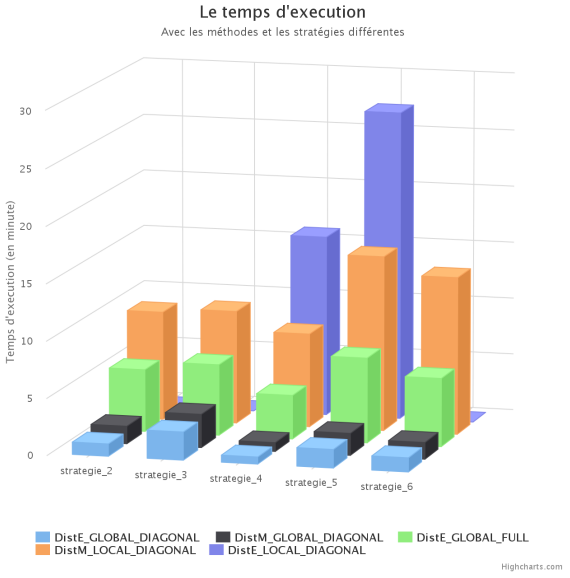
Comparaison des méthodes

Parmi les algorithmes d'apprentissage de métrique, on trouve que la méthode globale avec la matrice de covariance diagonale prend moins de temps d'exécution et fournit un bon résultat. On va comparer cette méthode (MPCKMEANS_GLOBAL_DIAGONAL) avec la méthode Baseline (dans la figure 4.5). Pour la méthode d'apprentissage de métrique, on va analyser aussi les différents résultats de deux types de distance Euclidienne et

1. https ://cmake.org/
2. http ://eigen.tuxfamily.org



(A) Le temps d'exécution de toutes les méthodes avec les stratégies différentes



(B) Le temps d'exécution des méthodes d'apprentissage de métrique avec les stratégies différentes sauf la stratégie 1

FIGURE 4.4: Comparaison du temps d'exécution de toutes les méthodes

Mahalanobis utilisés pour la construction et la division des CF-Entrées. L'apprentissage de métrique donnent un résultat meilleur que celui de la méthode Baseline dans la plupart des stratégies appliquées. Pour la stratégie 1, l'apprentissage de métrique avec la distance Euclidienne utilisée dans l'arbre CF-Tree est le meilleur. Dans les autres stratégies sauf la stratégie 3, l'apprentissage de métrique avec la distance de Mahalanobis utilisée dans l'arbre CF-Tree est le meilleur. Les autres comparaisons entre la méthode Baseline et les autres méthode d'apprentissage de métrique (avec différents types de matrices de covariance et avec les deux approches locale et globale) se trouvent dans l'annexe D.

Perspectives d'amélioration des résultats

La méthode d'apprentissage de métrique MPCKMEANS_GLOBAL_DIAGONAL est assez bon et rapide dans le contexte de clustering interactif. Si on veut obtenir un bon résultat stable, la stratégie 1 est un bon choix. Mais on peut facilement trouver que, dans les 20 premières itérations, la méthode Baseline est toujours en tête, et dans le reste, les méthode d'apprentissage de métrique commence à la surpasser. Ça ouvert une nouvelle perspective de combiner les deux méthodes : dans quelques premières itérations, on applique la méthode Baseline pour obtenir un départ stable, et ensuite, on utilise une méthode d'apprentissage de métrique pour améliorer la performance. On peut faire plus d'expérimentations pour déterminer le nombre d'itération de départ dans la combinaison des deux méthodes.

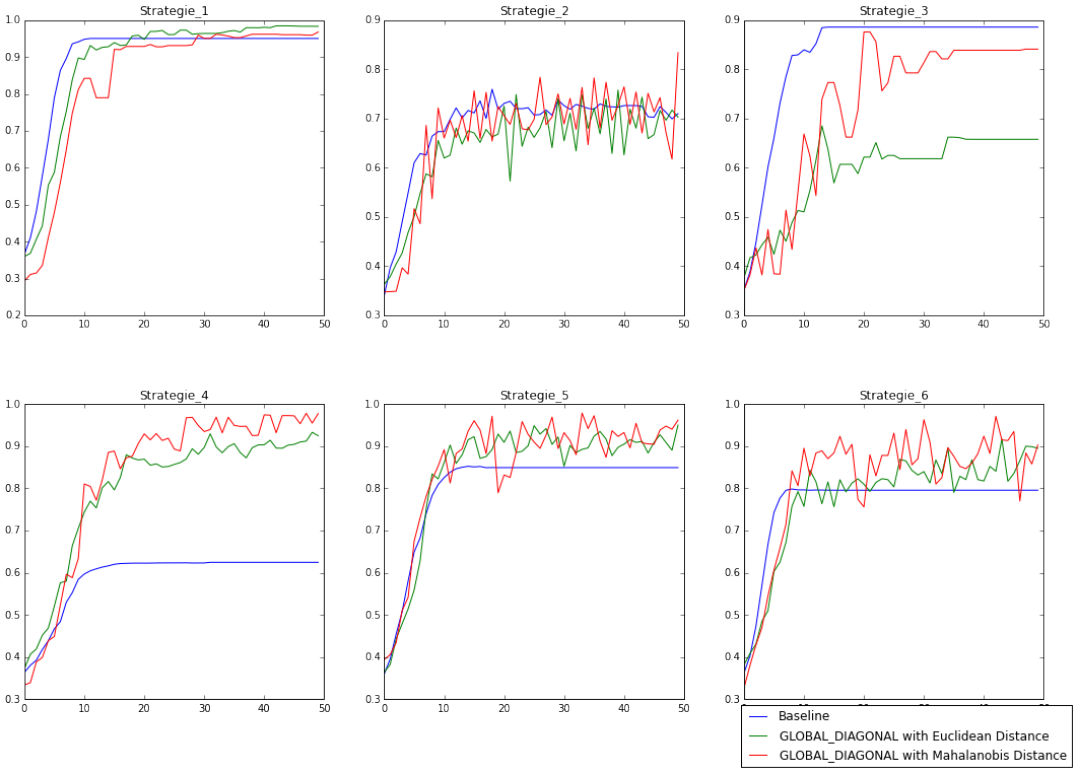


FIGURE 4.5: L'apprentissage de métrique avec l'approche globale en utilisant la distance Euclidienne et la distance de Mahalanobis pour la division des CF-Entrées

Conclusion de la méthode proposée

Nous avons comparé de nombreuses configurations pour des méthodes d'apprentissage de métrique dans la méthode de clustering semi-supervisé interactif. Il y a une tendance commune entre toutes les résultats expérimentaux : l'apprentissage de métrique est moins stable que la méthode Baseline mais il donne de meilleurs résultats dans la plupart de stratégies. Le résultat dépend en effet des stratégies de déduction de contraintes. On peut regarder la tableau 4.2 qui montre différentes mesures de différentes méthodes et la figure 4.4 qui montre le temps de calcul, pour déterminer la méthode préférée selon le besoin. Par exemple, si on veut une méthode stable, la méthode Baseline est un bon choix ; si on veut la performance la plus élevée possible, en regardant le critère de la valeur maximum de VMesure, la méthode MPCKMEANS_GLOBAL_FULL est un bon choix ; si on a besoin d'une méthode qui s'adapte au mieux au contexte en-ligne avec l'utilisateurs, on peut utiliser la stratégie 4 pour déduire des contraintes avec une méthode d'apprentissage de métrique car le temps d'exécution de cette stratégie est très faible et le résultat est assez bon.

Chapitre 5

Conclusion

Une des idées maîtresses de ce travail de stage est d'analyser l'évolution des méthodes de clustering qui cherchent à décomposer un ensemble d'individus en plusieurs sous ensembles les plus homogènes possible.

Systématiquement, on commence par les méthodes de clustering non-supervisé comme KMeans qui est assez simple et efficace pour travailler avec des grandes bases de données. Un autre algorithme plus généralisé, basé sur le modèle probabiliste, est l'algorithme *Expectation Maximisation* qui permet de traiter des données non-sphériques ou hétérogènes. On présente aussi l'algorithme BIRCH qui met des points similaires dans un même groupe pour fournir une structure hiérarchique de données plus compacte et plus représentative.

Les méthodes de clustering semi-supervisé utilisent des informations supervisées sous la forme des étiquettes de quelques points ou des contraintes par paires entres quelques points. L'algorithme HMRF-KMeans, une variante de KMeans, un framework probabiliste basé sur le modèle de Markov caché, utilise la boucle EM pour contrôler le processus de convergence, et utilise des contraintes par paires pour mettre à jour les paramètres utilisés dans le calcul de la dissimilarité. Cet algorithme ajoute aussi à la fonction objective les termes de pénalité des contraintes violées.

LAI Hien Phuong, dans sa thèse, a proposé une nouvelle méthode de clustering semi-supervisé interactif (appelée la méthode Baseline) qui bénéficie des retours de l'utilisateur dans chaque itération interactive pour guider le modèle de clustering. L'algorithme BIRCH réalise le clustering non-supervisé initial et fournit un arbre hiérarchique. L'algorithme HMRF-KMeans est utilisé dans l'étape de reclustering interactif incrémental sur l'ensemble des CF-Entrées des feuilles de l'arbre BIRCH. Avec l'intervention de l'utilisateur dans chaque itération interactive, elle a montré que avec le même nombre de clics de l'utilisateur, les contraintes par paires donnent plus d'informations supervisées que

les étiquettes. Elle a donc introduit plusieurs stratégies de déduction de contraintes par paires à partir des retours de l'utilisateur. Les résultats expérimentaux de son système montrent que sa méthode est meilleure que la méthode HMRF-KMeans dans un contexte interactif incrémental.

Une autre contribution de ce stage est une étude sur les méthodes d'apprentissage de métrique. Le besoin de moyens appropriés pour mesurer la distance ou la similarité entre les données est présent partout, mais les bonnes mesures sont 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. Dans le système existant, le but de combler le fossé sémantique entre les concepts de haut niveau de l'utilisateur et les signatures de bas niveau extraites à partir des données est accomplit à un certain degré. Le problème restant réside dans les inconvénients de la distance Euclidienne dans l'espace multidimensionnel. La distance de Mahalanobis qui prend en compte les corrélations de données peut mieux refléter une mesure sémantique de la dissimilarité. Les méthode d'apprentissage de métrique qui utilisent la distance de Mahalanobis dépendent des paramètres de forme des matrices de covariance. Si on estime plusieurs matrices de covariance pour chaque cluster séparément, on a une approche locale. En revanche, si on utilise une seule matrice de covariance pour tout l'ensemble de données, on a une approche globale. La forme de la matrice de covariance est aussi variée : une forme diagonale ou une forme complète. On a donc beaucoup de configurations de paramètres d'une méthode d'apprentissage de métrique à expérimenter.

La dernière et la plus importante contribution de ce travail est une intégration d'une méthode d'apprentissage de métrique dans le système existant. Parmi les méthodes d'apprentissage de métrique globales ou locales, linéaires ou non linéaires, la méthode MPCKMeans est choisie car elle utilise des contraintes par paires et elle a des points communs avec l'algorithme HMRF-KMeans utilisé dans le système existant. On utilise toujours l'algorithme BIRCH dans l'étape de clustering non-supervisé initiale et on utilise l'algorithme MPCKMeans dans l'étape de reclustering interactif. Selon les résultats expérimentaux de la méthode proposée, on peut conclure que les méthodes d'apprentissage de métrique s'adaptent bien dans le système existant et qu'elles donnent de meilleurs résultats dans quelques stratégies particulières. En général, la performance de la méthode d'apprentissage de métrique est toujours compétitive. Bien qu'elles ne soient pas stables comme la méthode Baseline, elles prennent moins de temps de calcul, elles sont donc appropriées dans un contexte interactif incrémental (c'est-à-dire en ligne avec l'utilisateur).

Après les études et les résultats proposés dans ce travail, on peut proposer quelques perspectives prometteuses. Pour bénéficier de la bonne initialisation de la méthode Baseline et de la bonne convergence des méthodes d'apprentissage de métrique,

la combinaison entre ces méthodes est une solution prometteuse ; par exemple dans les 10 premières itérations, la méthode Baseline est utilisée pour le reclustering, et ensuite, la méthode d'apprentissage de métrique est utilisée. De plus, il faudrait essayer d'autres méthodes d'apprentissage de métrique. En effet, le système existant utilise toujours une approche basée sur un modèle probabiliste de mélanges de gaussiennes (appliquée dans l'algorithme HMRF-KMeans et MPCKMeans). Il existe d'autres approches, par exemple l'approche basée sur le modèle de K plus proches voisins comme LMNN (*Large Margin Nearest Neighbors*) ou bien l'approche non linéaire basée sur le modèle de noyaux en utilisant KPCA (*Kernel Principal Component Analysis*).

Annexe A

Illustration des méthodes de clustering non-supervisé

Illustration de BIRCH

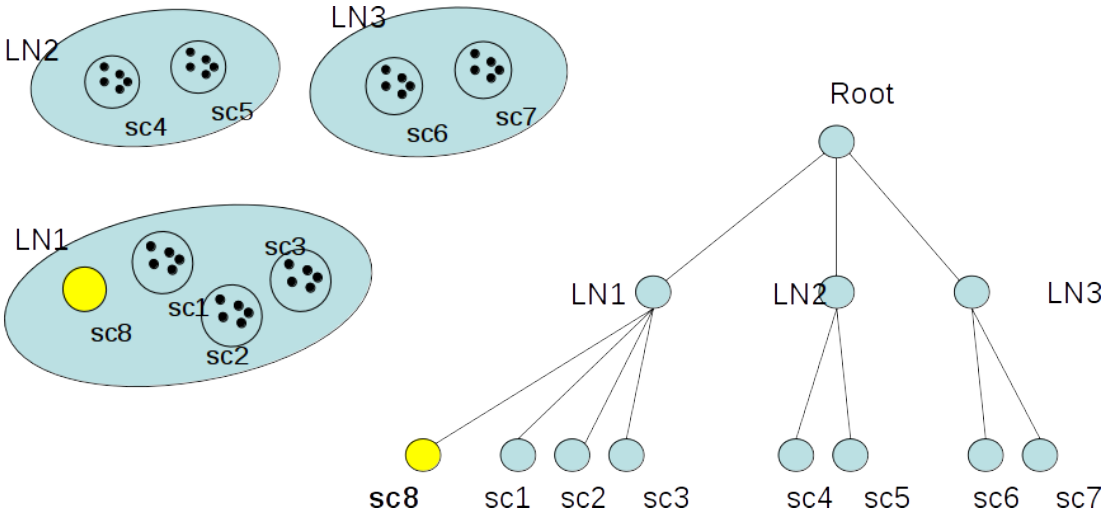
Dans les schémas suivants, les CF-Entrées sont considérées comme les sous-clusters et sont notées sc_i. Les nœuds feuilles qui contiennent les CF *entries* sont notées LN_i (Leaf Node).

Dans la figure A.1a, on a un arbre avec le facteur de branchement de nœud feuille $L = 4$, et le facteur de branchement de nœud non-feuille $B = 4$. Donc, la feuille LN_1 a 4 enfants sc_1, sc_2, sc_3, sc_8, et la racine (*Root Node*) a trois enfants LN_1, LN_2, LN_3.

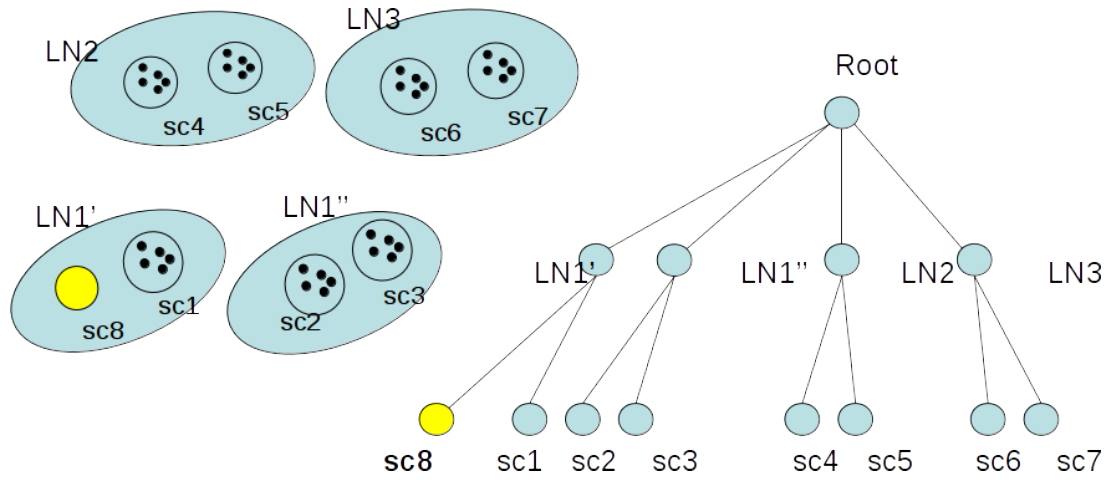
Si on change le facteur de branchement de nœud feuille $L = 3$, donc la feuille LN_1 est découpée car elle a trop d'enfants. Les nœuds feuilles LN_1' et LN_1'' sont insérées à la racine comme dans la figure A.1b. La racine maintenant a 4 enfants, et satisfait toujours la contrainte de facteur de branchement de nœud non-feuille $B = 4$.

Si on change le facteur de branchement de nœud non-feuille $B = 3$, la racine doit être découpée car elle a 4 enfants (supérieur à $B = 3$). Dans la figure A.1c, l'ancienne racine est découpée en deux, et les nœuds intermédiaires sont insérés dans la nouvelle racine. En conséquence, la hauteur de l'arbre augmente de 1.

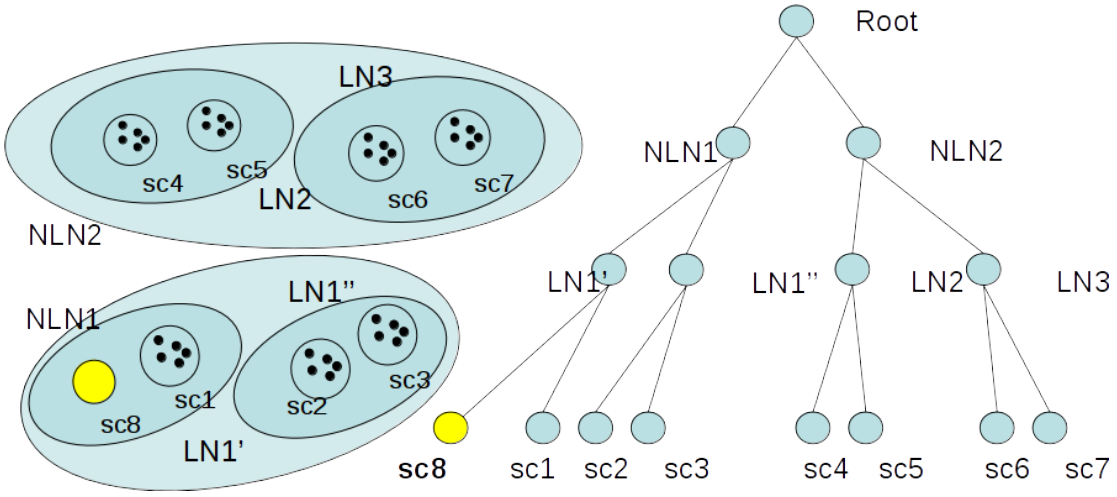
1. www.cs.uvm.edu/ xwu/kdd/Birch-09.ppt



(A) L'arbre CF-Tree avec $B = 4, L = 4$



(B) L'arbre avec le facteur de branchement de feuille $L = 3$ et le découpage à la feuille LN_1



(C) L'arbre avec le facteur de branchement de nœud non-feuille $B = 3$ et le découpage à la racine

FIGURE A.1: Illustration de l'algorithme BIRCH [1]

Annexe B

Mesures de qualité de clustering

V-measure [22]

C'est une mesure basée sur l'entropie qui mesure explicitement comment les critères de l'homogénéité (*homogeneity*) et de la compacité (*completeness*) ont été satisfaites. VMesure est calculée comme suit :

$$\text{V-measure} = \frac{(1+\beta) * homogeneity * completeness}{\beta * homogeneity + completeness} \tag{B.1}$$

Étant donné c classes avec des vérités terrains et k clusters trouvés par un algorithme de clustering. On construit d'abord une matrice de confusion (avec des éléments n_{ij}) qui comprend c lignes et k colonnes. Si un point de donnée de la classe i est mis dans le cluster j, on augment la valeur de n_{ij} par 1. Cette matrice est utilisée pour calculer les deux composantes de la VMesure.

Afin de satisfaire les critères d'homogénéité, un algorithme de clustering doit attribuer **uniquement** les points de données qui sont membres d'une seule classe à un seul cluster. On peut déterminer la proximité entre la distribution en clusters et la distribution idéale (des classes de la vérité terrain) en examinant l'entropie de la distribution conditionnelle des classes données sachant les clusters trouvés. Dans le cas parfaitement homogène, cette valeur, $H(C|K)$, est 0.

$$homogeneity = 1 - \frac{H(C|K)}{H(C)} \tag{B.2}$$

où $H(C|K)$ est l'entropie conditionnelle de la distribution de classes sachant les clusters trouvés :

$$H(C|K) = -\sum_{j=1}^{k}\sum_{i=1}^{c}\frac{n_{ij}}{N}\log\frac{n_{ij}}{\sum_{i=1}^{c} n_{ij}} \tag{B.3}$$

et $H(C)$ et l'entropie des classes :

$$H(C) = -\sum_{i=1}^{c} \frac{\sum_{j=1}^{k} n_{ij}}{N} \log \frac{\sum_{j=1}^{k} n_{ij}}{N} \tag{B.4}$$

Symétriquement, afin de satisfaire les critères de la compacité, un algorithme de clustering doit affecter **tout l'ensemble** des points qui sont membres d'une seule classe à un seul cluster. On peut évaluer ce degré en calculant l'entropie conditionnelle des clusters trouvés sachant les classes des points de données, $H(K|C)$. Dans le cas parfaitement complet, $H(K|C) = 0$.

$$completeness = 1 - \frac{H(K|C)}{H(K)} \tag{B.5}$$

où $H(K|C)$ est l'entropie conditionnelle de la distribution de clusters sachant les classes données :

$$H(K|C) = -\sum_{i=1}^{c} \sum_{j=1}^{k} \frac{n_{ij}}{N} \log \frac{n_{ij}}{\sum_{j=1}^{k} n_{ij}} \tag{B.6}$$

et $H(K)$ est l'entropie des clusters trouvés :

$$H(K) = -\sum_{j=1}^{k} \frac{\sum_{i=1}^{c} n_{ij}}{N} \log \frac{\sum_{i=1}^{c} n_{ij}}{N} \tag{B.7}$$

Annexe C

Résultat expérimental de l'algorithme MPCKMeans

La figure C.1 montre le résultat de MPCKMeans, appliqué sur la base Wang. Les contraintes par paires entre images sont générées automatiquement à partir de la vérité terrain, sans intervention de l'utilisateur. L'expérimentation est exécutée dans 95 fois indépendantes. Chaque fois, un différent ensemble de contraintes par paires entre images est utilisé. L'axe vertical est la performance mesurée par la VMesure, L'axe horizontal est le nombre de contraintes. On trouve que la performance augmente selon le nombre de contraintes données.

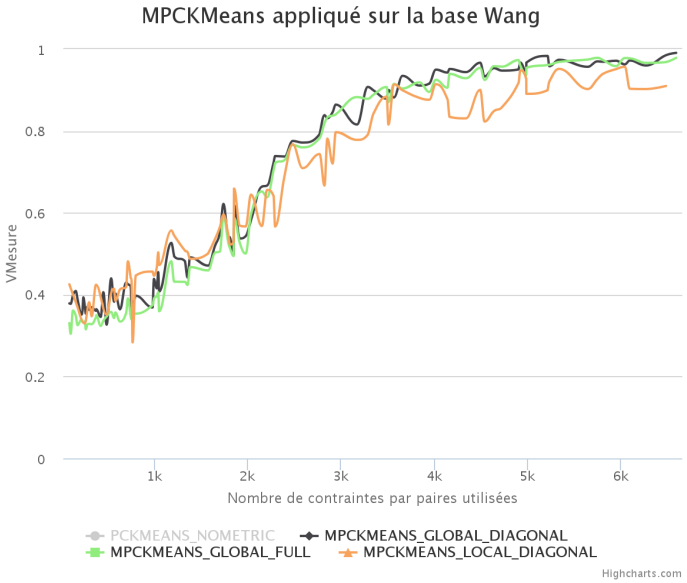


FIGURE C.1: L'algorithme MPCKMeans appliqué sur la base Wang avec des contraintes par paires entre images

Annexe D

Résultats détaillés de quelques méthodes d'apprentissage de métrique

Les résultats détaillés pour quelques méthodes d'apprentissage de métrique précisées

Voir Figure D.1

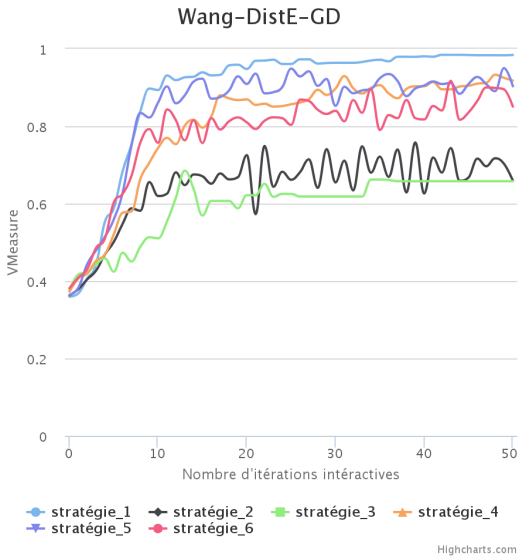
Comparaison la méthode Baseline et les méthodes d'apprentissage de métrique

L'apprentissage de métrique avec l'approche globale

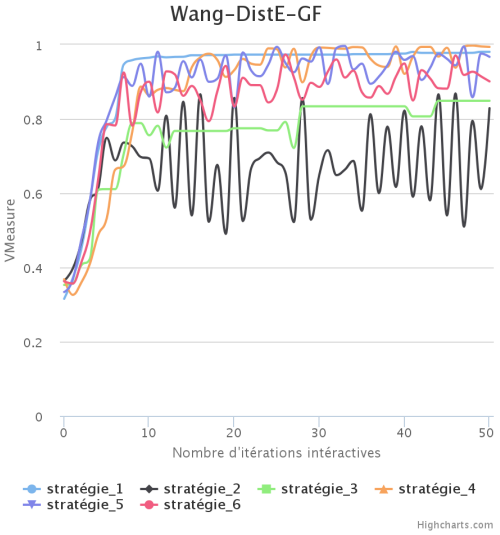
Voir Figure D.2

L'apprentissage de métrique avec l'approche locale

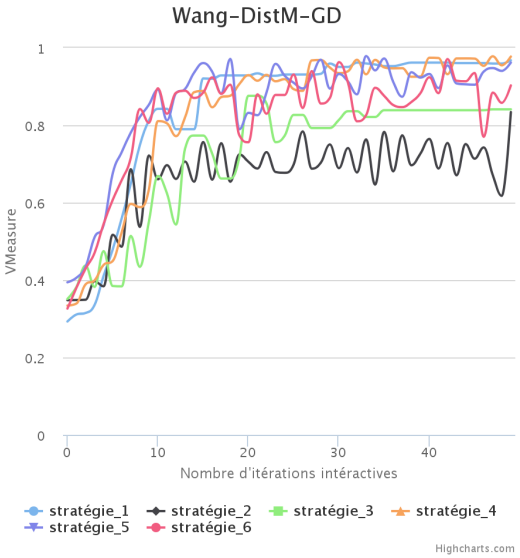
Voir Figure D.3



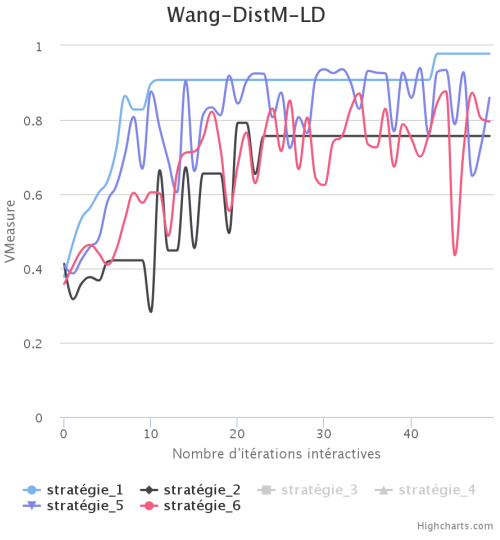
(A) MPCKMEANS_GLOBAL_DIAGONAL (*distE*)



(B) MPCKMEANS_GLOBAL_FULL (*distE*)



(C) MPCKMEANS_GLOBAL_DIAGONAL (*distM*)



(D) MPCKMEANS_LOCAL_DIAGONAL (*distM*)

FIGURE D.1: Les résultats des méthodes d'apprentissage de métrique

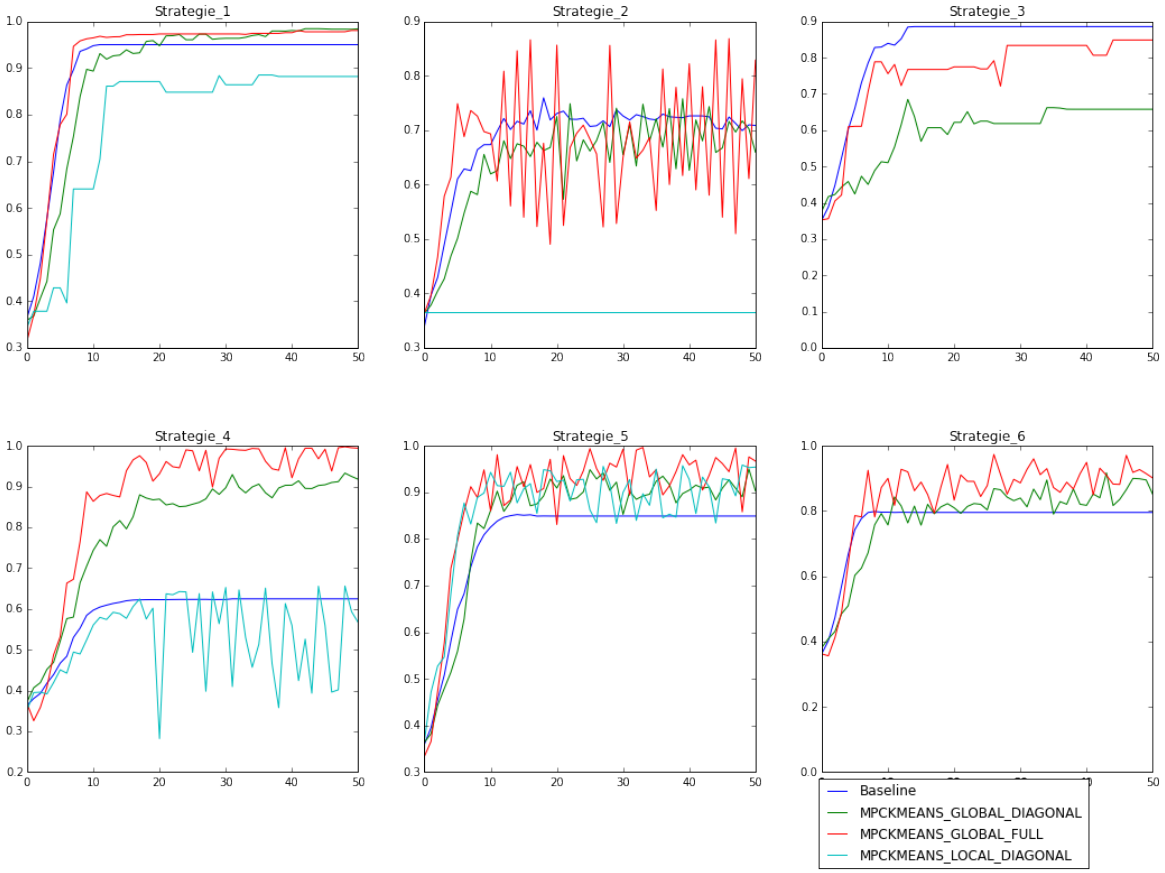


FIGURE D.2: Comparaison de la méthode Baseline et des méthodes d'apprentissage de métrique avec la distance Euclidienne pour la construction et la division des CF-Entrées

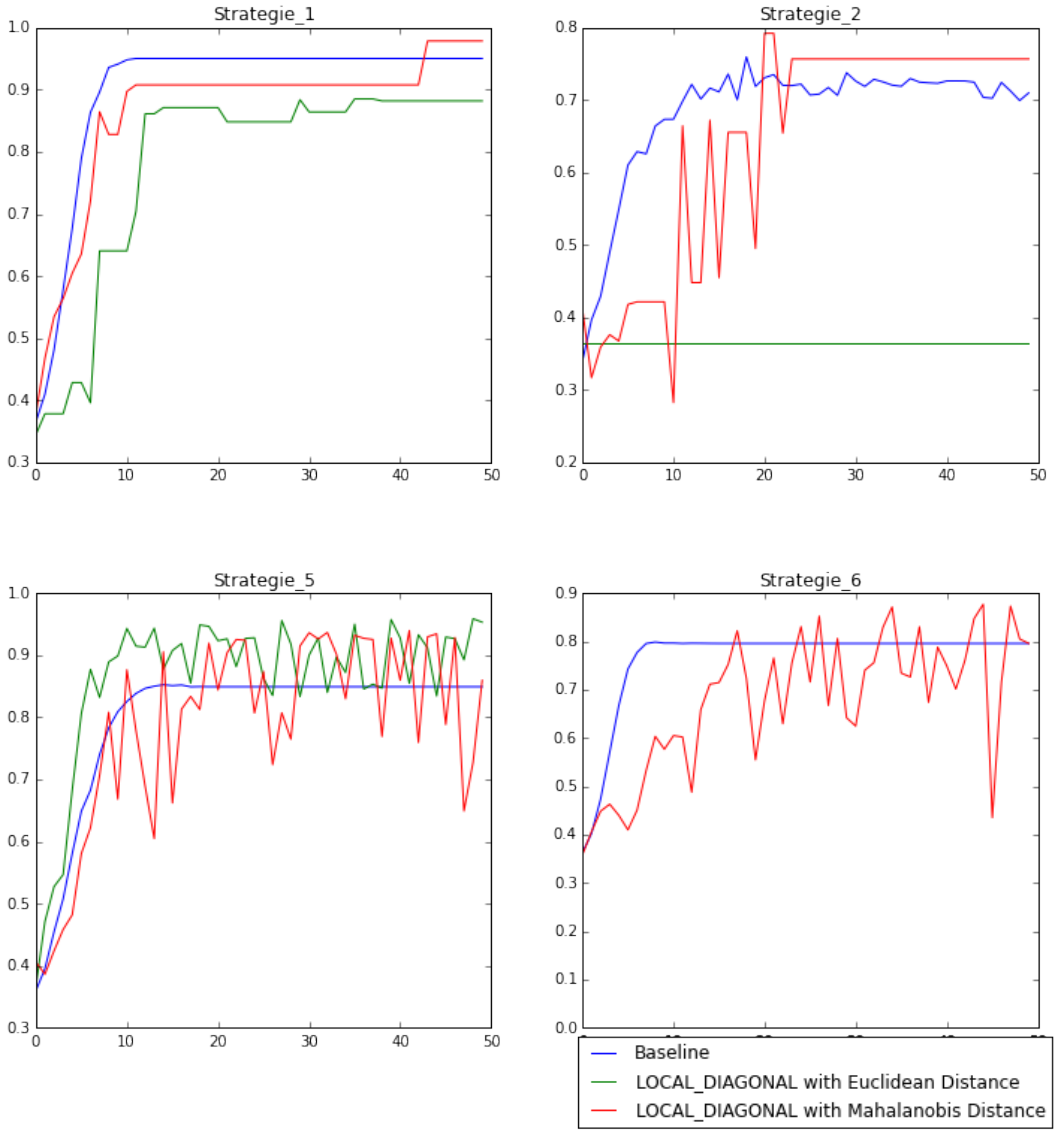


FIGURE D.3: L'apprentissage de métrique avec l'approche locale en utilisant la distance Euclidienne et la distance de Mahalanobis pour la construction et la division des CF-Entrées

Bibliographie

[1] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[2] Geoffrey H Ball and David J Hall. Isodata, a novel method of data analysis and pattern classification. Technical report, DTIC Document, 1965.

[3] L Kaufman and PJ Rousseeuw. Clustering by means of medoids. in 'y. dodge (editor) statistical data analysis based on l1 norm', 405-416, 1987.

[4] Leonard Kaufman and Peter J Rousseeuw. Partitioning around medoids (program pam). *Finding groups in data : an introduction to cluster analysis*, pages 68–125, 1990.

[5] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data : an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[6] R.T. Ng and Jiawei Han. Clarans : a method for clustering objects for spatial data mining. *Knowledge and Data Engineering, IEEE Transactions on*, 14(5) :1003–1016, Sep 2002. ISSN 1041-4347.

[7] Teuvo Kohonen, Samuel Kaski, and Harri Lappalainen. Self-organized formation of various invariant-feature filters in the adaptive-subspace som. *Neural computation*, 9(6) :1321–1344, 1997.

[8] Godfrey N Lance and William Thomas Williams. A general theory of classificatory sorting strategies ii. clustering systems. *The computer journal*, 10(3) :271–277, 1967.

[9] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch : an efficient data clustering method for very large databases. In *ACM SIGMOD Record*, volume 25, pages 103–114. ACM, 1996.

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[11] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[12] Wei Wang, Jiong Yang, Richard Muntz, et al. Sting : A statistical information grid approach to spatial data mining. In *VLDB*, volume 97, pages 186–195, 1997.

[13] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998.

[14] Hien Phuong Lai, Muriel Visani, Alain Boucher, and Jean-Marc Ogier. An experimental comparison of clustering methods for content-based indexing of large image databases. *Pattern Analysis and Applications*, 15(4) :345–366, 2012.

[15] Sugato Basu, Arindam Banerjee, and Raymond Mooney. Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. Citeseer, 2002.

[16] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584, 2001.

[17] Sugato Basu, Mikhail Bilenko, and Raymond J Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68. ACM, 2004.

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

[19] Andrew Willis and Yunfeng Sui. An algebraic model for fast corner detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2296–2302. IEEE, 2009.

[20] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[21] Hien Phuong Lai, Muriel Visani, Alain Boucher, and Jean-Marc Ogier. A new interactive semi-supervised clustering model for large image database indexing. *Pattern Recognition Letters*, 37 :94–106, 2014.

[22] Andrew Rosenberg and Julia Hirschberg. V-measure : A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, volume 7, pages 410–420, 2007.

[23] Eric P Xing, Michael I Jordan, Stuart Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512, 2002.

[24] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2005.

[25] Shibin Parameswaran and Kilian Q Weinberger. Large margin multi-task metric learning. In *Advances in neural information processing systems*, pages 1867–1875, 2010.

[26] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5) :1299–1319, 1998.

[27] Ratthachat Chatpatanasiri, Teesid Korsrilabutr, Pasakorn Tangchanachaianan, and Boonserm Kijsirikul. A new kernelization framework for mahalanobis distance learning algorithms. *Neurocomputing*, 73(10) :1570–1579, 2010.

[28] Yujie He, Wenlin Chen, Yixin Chen, and Yi Mao. Kernel density metric learning. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 271–280. IEEE, 2013.

[29] Jun Wang, Huyen T Do, Adam Woznica, and Alexandros Kalousis. Metric learning with multiple kernels. In *Advances in neural information processing systems*, pages 1170–1178, 2011.

[30] Mikhail Bilenko, Sugato Basu, and Raymond J Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 11. ACM, 2004.

[31] G.J. McLachlan. Mahalanobis distance. *Resonance*, 4(6) :20–26, 1999. URL http://dx.doi.org/10.1007/BF02834632.

[32] Eric W Weisstein. Eigen decomposition. 2002.