N° d'Ordre :
EDSPIC :

Université Paris 13

THESIS

by

Parisa RASTIN

DOCTORAT DÉLIVRÉ PAR L'UNIVERSITÉ SORBONNE PARIS CITÉ
ET PRÉPARÉ À L'UNIVERSITÉ PARIS 13
(Arrêté du 25 mai 2016)

École Doctorale Sciences, Technologie, Santé, Galilée (ED 146)

SPÉCIALITÉ : Informatique

Automatic and Adaptive Learning for Relational Data Stream Clustering

Defended on 26th June 2018 before the following jury :

Thesis supervisor :		
B.	Matei	Maitre de Conferences HDR , Université Paris 13
Rapporteurs :		
P.	Gançarski	Professeur, Université de Strasbourg
F. de A.T.	de Carvalho	Professeur, Universidade Federal de Pernambuco
G.	Cleuziou	Maitre de Conferences HDR, Université d'Orléans
Examinateurs :		
R.	Verde	Professeur, Seconda Universita di Napoli
Y.	Bennani	Professeur, Université Paris 13
G.	Cabanes	Maitre de Conferences, Université Paris 13
T.	Couronne	Data Scientist PhD, Mindlytix

Résumé

Le travail de recherche exposé dans cette thèse concerne le développement d'approches d'apprentissage non-supervisé adaptés aux grands jeux de données relationnelles et dynamiques. La combinaison de ces trois caractéristiques (taille, complexité et évolution) constitue un défi majeur dans le domaine de l'exploration de données et peu de solutions satisfaisantes existent pour le moment, malgré les besoins de plus en plus manifestes des entreprises. C'est un véritable challenge, car les approches adaptées aux données relationnelles ont une complexité quadratique inadaptée à l'analyse de données dynamiques. Nous proposons ici deux approches complémentaires pour l'analyse de ce type de données. La première approche est capable de détecter des clusters bien séparés à partir d'un signal créé lors d'un réordonnancement incrémental de la matrice de dissimilarité, sans paramètre à choisir (par ex. le nombre de clusters). La seconde propose d'utiliser des points de support parmi les objets afin de construire un espace de représentation permettant de définir des prototypes représentatifs des clusters. Enfin, nous appliquons les approches proposées au profilage en temps réel d'utilisateurs connectés. Les tâches de profilage visent à reconnaître "l'état d'esprit" des utilisateurs à travers leurs navigations sur différents sites.

Mots-Clés : Apprentissage Non-supervisé, Données Relationnelles et Dynamiques, Coordonnées Barycentriques, Marketing en Ligne

Abstract

The research work presented in this thesis concerns the development of unsupervised learning approaches adapted to large relational and dynamic data-sets. The combination of these three characteristics (size, complexity and evolution) is a major challenge in the field of data mining and few satisfactory solutions exist at the moment, despite the obvious needs of companies. This is a real challenge, because the approaches adapted to relational data have a quadratic complexity, unsuited to the analysis of dynamic data. We propose here two complementary approaches for the analysis of this type of data. The first approach is able to detect well-separated clusters from a signal created during an incremental reordering of the dissimilarity matrix, with no parameter to choose (e.g., the number of clusters). The second proposes to use support points among the objects in order to build a representation space to define representative prototypes of the clusters. Finally, we apply the proposed approaches to real-time profiling of connected users. Profiling tasks are designed to recognize the "state of mind" of users through their navigations on different web-sites.

Keywords: Clustering, Unsupervised Learning, Relational and Dynamic Data, Barycentric Coordinates, Online Marketing

Avant-propos

Cet avant-propos en français présente de façon synthétique les enjeux et objectifs de ce travail de thèse, ainsi que les principales contributions développées dans la suite du manuscrit.

1 Introduction

L'apprentissage non supervisé, ou clustering, est une tâche importante dans le processus d'extraction des connaissances à partir des données. La complexité de cette tâche a considérablement augmenté au cours des dernières décennies avec une explosion de la taille des données disponibles. Le traitement de ce type de données nécessite le développement de méthodes très peu complexes ou fortement parallélisées. Malgré de nets progrès dans ce domaine, de nombreux défis subsistent, en particulier pour le traitement de données complexes et dynamiques.

En effet, la production croissante de données avec une structure complexe nécessite des outils d'analyse efficaces et appropriés. L'analyse de données complexes en général est un domaine en pleine croissance. La plupart des approches non supervisées sont adaptées à des vecteurs dans un espace euclidien, les clusters étant calculés sur la base de la distance euclidienne. Cependant, dans de nombreux cas, les objets ne peuvent pas être facilement définis dans un espace euclidien sans perte d'information et/ou un pré-traitement coûteux. Les solutions de transformation de ces informations en données vectorielles ont en effet montré leurs limites, et les chercheurs s'intéressent maintenant à des solutions directement adaptées à chaque type de données, avec des métriques de similarité adaptées à la complexité des données. Les algorithmes spécifiques étant évidemment limités à un type de données particulier, une solution consiste à utiliser une approche de clustering "relationnel" basé sur des noyaux ou des matrices de dissimilarité. Les algorithmes de clustering relationnel forment une famille de méthodes adaptées aux données relationnelles. Certains algorithmes de clustering sont naturellement adaptés aux matrices de dissimilarité et peuvent être utilisés pour analyser des ensembles de données relationnels. Cependant, ils ont tous une complexité en temps de calcul non linéaire.

De plus, les jeux de données sont souvent en perpétuelle évolution, caractérisées par une structure variable dans le temps, de nouvelles informations apparaissant constamment. Le

clustering de données dynamiques ou de flux de données est un problème difficile en raison des coûts de calcul et de stockage associés aux volumes impliqués. Des algorithmes efficaces doivent pouvoir travailler avec une occupation de mémoire constante malgré l'évolution des données. En outre, la distribution de probabilité associée aux données peut changer avec le temps ("dérive de concepts"). En conséquence, la segmentation des données (c'est-à-dire la structure du flux) est également en constante évolution.

La combinaison de ces trois caractéristiques (taille, complexité et évolution) constitue un défi majeur dans le domaine de l'exploration de données et peu de solutions satisfaisantes existent pour le moment, malgré les besoins de plus en plus manifestes des entreprises. En effet, malgré les progrès récents de l'analyse de données évolutive, il existe actuellement très peu de travaux sur l'apprentissage évolutif adaptés à des données complexes ou hétérogènes, qui constituent néanmoins la majorité des données produites par les entreprises. Cette thèse porte sur le développement de nouvelles méthodes de clustering adaptées aux données dynamiques et relationnelles. C'est un véritable challenge car les approches habituellement adaptées aux données relationnelles ont une complexité non linéaire par rapport au nombre d'objets dans l'ensemble de données, ce qui n'est pas adapté à l'analyse de données dynamiques à évolution rapide.

Ce travail est financé par l'ANRT (Association Nationale de la Recherche et de la Technologie) dans le cadre d'une convention CIFRE (« Conventions Industrielles de Formation par la Recherche ») avec la société Mindlytix.

2 Approches à base de traitement de signal

Bien que de nombreuses méthodes de clustering aient été proposées, cette tâche reste un problème difficile. En particulier, beaucoup d'algorithmes ont besoin du nombre de clusters à définir en tant que paramètre, malgré le fait que ce nombre soit rarement connu.

Nous proposons dans ce chapitre un algorithme de réordonnancement incrémental pour flux de données relationnelles, qui permet de mettre à jours et de visualiser la matrice de dissimilarité au fur et à mesure de l'évolution de la structure des données au cours du temps. Puis nous présentons un algorithme de clustering basé sur le traitement de signal, qui est capable de détecter des clusters à partir d'un signal créé lors du réordonnancement. Ces algorithmes ont l'avantage de ne pas avoir de paramètre à choisir. En particulier, le nombre de cluster est découvert automatiquement.

2.1 Réordonnancement incrémental de matrice

Les données relationnelles, dans lesquelles les objets sont définis par leurs similitudes les unes avec les autres, peuvent être représentées sous la forme d'une matrice de dissimilarité. Il est possible de produire une image de la matrice qui reflète la structure des données. Cependant, les objets doivent être organisés dans un ordre qui reflète cette structure :

pour que les clusters soient visibles, les objets appartenant au même cluster doivent être positionnés à proximité les uns des autres dans la matrice.

Plusieurs algorithmes de réordonnancement ont été proposés pour trouver l'ordre optimal des objets dans une matrice de similarité, mais aucun ne permet une construction incrémentale. Nous proposons donc ici une approche incrémentale permettant de visualiser la structure dynamique d'un flux de données. Dans cette approche, chaque objet est présenté une fois et comparé uniquement aux objets déjà présentés. L'idée principale est de construire de manière incrémentale la matrice de similarité d'une manière ordonnée à partir des objets présentés dans une séquence définie dans le temps. La matrice est contrainte de conserver une taille fixe ou de représenter des données correspondant à une fenêtre temporelle de taille définie, afin de supprimer les informations obsolètes et d'assurer un coût de mémoire et de calcul raisonnable.

Le principe de cette approche est d'optimiser de manière incrémentale la longueur du chemin hamiltonien des données ordonnées. Nous considérons que l'ordre des objets dans une matrice de dissimilarité correspond à un chemin à travers un graphe où chaque nœud représente un objet et est visité exactement une fois : un chemin de Hamilton. En minimisant la longueur de ce chemin, nous minimisons globalement les similarités entre les objets adjacents. En fonction de nos besoins et de notre limitation de vitesse ou de mémoire, nous définissons la taille maximale de la matrice ou de la fenêtre temporelle : lorsqu'un nouvel objet est présenté au système, si la matrice (ou la fenêtre) est supérieure à cette taille, la ligne et la colonne correspondants à l'objet le plus ancien sont supprimées. Cet algorithme est incrémental par construction et est adapté pour visualiser les variations temporelles de la structure de données.

Nous avons testé la qualité de la matrice réordonnée sur un ensemble de données statiques artificielles et réelles et comparé les résultats avec la qualité des algorithmes existants. L'objectif était de démontrer que la performance de la méthode proposée est au moins similaire à celle de ses concurrents pour réordonnancer une matrice de similarité. L'algorithme fonctionne très bien sur les ensembles de données statiques par rapport à l'état de l'art : sa qualité reste comparable ou supérieure, malgré le fait de travailler de manière incrémentale à partir d'une présentation aléatoire des données, tandis que les autres algorithmes peuvent utiliser toute information sur la matrice de similarité à tout moment. L'algorithme proposé est la meilleure méthode pour minimiser l'indice de longueur du chemin hamiltonien, ce qui n'est pas surprenant car il a été conçu pour optimiser ce critère. Cependant, il est légèrement moins efficace pour les ensembles de données avec des clusters en contact ou mal définis.

En outre, l'algorithme assure un coût de mémoire et de calcul raisonnable pour les ensembles de données dynamiques, et les visualisations montrent que notre approche est bien adaptée pour détecter les variations temporelles de la structure de données, telles que les changements de densité, les apparitions ou disparitions de clusters, ainsi que les changements dans les similitudes entre les clusters. De plus, la division progressive d'un

cluster en deux nouveaux clusters ou la fusion de deux clusters en un seul sont aussi bien représentés. La figure 1 illustre les résultats d'un réordonnancement dynamique appliqué à un flux de donnée artificiel.

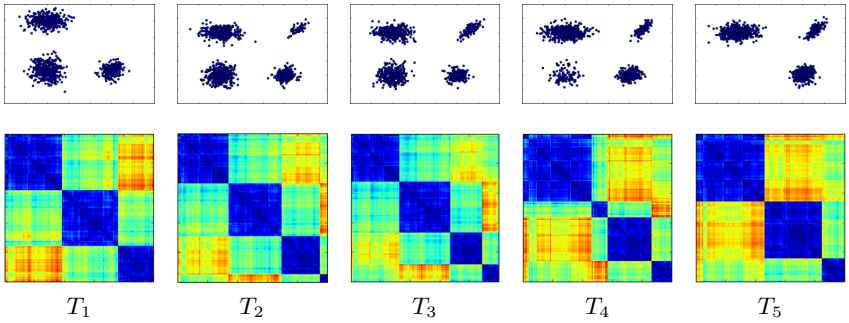


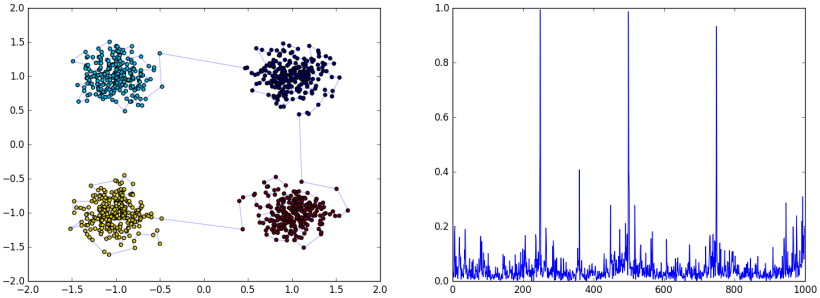
Figure 1: Exemple de visualisation obtenue à partir de l'approche incrémentale proposée.

2.2 Clustering non-paramétrique par traitement de signal

Les approches de réordonnancement permettent une visualisation efficace de la structure des données, mais ne fournisse pas une segmentation automatique en différents clusters. Nous proposons donc un nouvel algorithme de clustering autonome pour les données basées sur la similarité. L'idée est d'utiliser les propriétés du réordonnancement afin de produire un signal unidimensionnel de paires de distances. Puisque les objets appartenant aux mêmes clusters sont voisins dans la matrice réordonnée, il sera possible de détecter des "sauts" de distances d'un cluster à un autre, si les deux clusters sont suffisamment éloignés. Pour détecter ces sauts, nous proposons d'utiliser une procédure de débruitage multi-échelle de signaux. Le principal avantage de cette procédure est qu'il n'y a pas de paramètre à modifier pour obtenir un clustering raisonnable. L'approche s'adapte automatiquement à la structure des données et trouve le nombre correct de clusters. Il est adapté à toute forme de clusters ou de variation de densité parmi les observations. La seule exigence est que les clusters ne doivent pas se chevaucher.

En résumé, la stratégie comporte deux étapes : (A) on réordonne la matrice de dissimilarité, puis on construit un signal contenant les distances paire-à-paire entre les données ordonnées et (B) on détecte les pics de distance dans le signal précédemment construit afin de définir les frontières entre clusters. À partir des données observées, nous calculons d'abord un réordonnancement des données minimisant la somme des distances le long du chemin hamiltonien (Figure 2a). Le signal des dissimilarités par paires de nos observations ordonnées est sa première diagonale principale non nulle de la matrice réordonnée (Figure 2b). Au sein d'un cluster, on s'attend à avoir de petites oscillations de la dissimilarité, alors qu'entre deux clusters différents il y a un pic de grande amplitude si les clusters ne se chevauchent

pas. Par lissage successif du signal, il est possible de détecter les pics de distance qui restent stable à chaque itération. Ces pics définissent des frontières entre clusters.



a. Données et chemin hamiltonien b. Signal 1D Signal de dissimilarités

Figure 2: Exemple de signal créé sur un jeu de données artificiel. Les trois pics qui définissent les frontières entre clusters sont détectés par lissage successif du signal afin d'éliminer le bruit.

La qualité de cet algorithme a été comparé à d'autre approches de clustering adaptés aux matrices de dissimilarité, sur un ensemble de jeux de données réels et artificiels. Il ressort de ces résultats que l'algorithme proposé se comporte très bien par rapport à l'état de l'art, la qualité du clustering obtenu est toujours au moins aussi bonne que celle des concurrents. La principale différence est que notre algorithme est autonome, il n'y a pas de paramètres à définir, alors que la plupart des autres algorithmes doivent définir le nombre de clusters, ou d'autres valeurs de paramètres qui doivent être adaptés à chaque ensemble de données. Être capable de proposer un bon clustering de manière autonome est un grand avantage pour un algorithme de clustering, puisque dans la plupart des applications réelles il n'existe aucune connaissance préalable sur le nombre de clusters à trouver ou d'autres valeurs des paramètres.

Les approches décrites dans ce chapitre montrent des résultats intéressants, avec l'avantage de pouvoir traiter des données dynamiques et une détection automatique du nombre de clusters. Le seul paramètre réellement choisi par l'utilisateur est la taille de la fenêtres temporelles (ou de la matrice), qui dépend essentiellement de l'échelle de temps que l'on souhaite considérer, ou de limitations en mémoire ou en vitesse de calcul. Le principal inconvénient de l'approche de clustering proposée réside dans sa difficulté à détecter les clusters qui se chevauchent. Pour y remédier, une idée serait de calculer des prototypes représentant les clusters, d'une manière adaptée aux données relationnelles ayant une structure dynamique. C'est le sujet du chapitre suivant.

3 Approches à base de points de support

Les approches basées sur les prototypes sont très populaires en apprentissage non supervisé, en raison de la compacité du modèle résultant (les prototypes), de la puissance descriptive de ces prototypes et de la faible complexité de calcul du modèle (chaque objet est comparé à un petit nombre de prototypes). Habituellement, le meilleur choix de prototype est le barycentre du cluster. Le prototype est alors défini comme l'objet minimisant la somme des distances carrées avec tous les objets du cluster. Si les objets sont décrits comme des vecteurs numériques dans un espace euclidien, la définition des prototypes de clusters est simple. Cependant, dans de nombreux cas, les objets ne peuvent pas être facilement définis dans un espace euclidien sans perte d'information et/ou un pré-traitement coûteux (par exemple des images, des réseaux, des séquences, des textes). La similarité entre ces objets n'est généralement pas une distance euclidienne et le calcul habituel des prototypes n'est plus valide.

Peu de travaux ont encore été réalisés sur le clustering relationnel basée sur des prototypes, mais certains auteurs ont travaillé sur l'adaptation des K-moyennes aux données relationnelles. Le problème principal dans ce cas est la définition des prototypes basé uniquement sur les distances entre les objets. Certains auteurs ont proposé de représenter les prototypes comme une combinaison linéaire des objets d'entrée. Mais, en ce qui concerne la puissance de traitement et l'utilisation de la mémoire, cette implémentation est très coûteuse ($\mathcal{O}(N^2)$ pour N objets), la rendant inutilisable pour de grands ensembles de données. L'utilisation de "points de support" pour définir les prototypes permet de réduire cette complexité, à condition que cet ensemble d'objets soit fixe au cours de l'apprentissage.

Dans ce chapitre, nous proposons une approche de K-moyennes relationnelle utilisant un ensemble unique de points de support à travers le processus d'apprentissage, puis nous introduisons le formalisme des Coordonnées Barycentriques, afin d'unifier la représentation des objets et des prototypes et permettant un processus d'apprentissage incrémental simple pour le clustering relationnel.

3.1 K-moyennes relationnel à points de supports fixes

Soit un ensemble de points de supports O_S, choisis parmi les données et associés à une représentation dans un espace pseudo-euclidien inconnu X. Soit x^i la représentation (inconnue) de l'objet o^i dans X, la représentation des points de supports sont notés s^p, $s^p \in X$. Nous voulons représenter chaque cluster k par un prototype m^k associé à un vecteur μ^k inconnu, défini comme une combinaison linéaire normalisée des points de supports :

$$\mu^k = \sum_{p=1}^{P} \beta_p^k s^p, \text{ avec } \sum_{p=1}^{P} \beta_p^k = 1 \tag{1}$$

Puisque la représentation s^p des points de support est inconnue, les prototypes sont définis pas les coefficients β^k. Soit $d(o^i, o^j)$ la distance carrée entre deux objets et D_s la matrice de dissimilarité entre les points de support calculée selon d, la dissimilarité entre un objet et un prototype peut être calculée comme suit:

$$d(o^{(i)}, w^{(k)}) = \sum_{p=1}^{P} \beta_p^{(k)} d(o^{(i)}, s^{(p)}) - \frac{1}{2} \beta^{(k)T} D_s \beta^{(k)} \tag{2}$$

On cherche pour chaque cluster un prototype minimisant la somme des distances carrées. Les minima locaux sont obtenus en appliquant la méthode des multiplicateurs de Lagrange. En choisissant un seul ensemble de points de support pour représenter toutes les données, le problème de minimisation est accéléré. En particulier, les distances entre objets et points de supports ne sont calculées qu'une seule fois.

Les performances de l'algorithme ont été comparées aux approches existantes sur un ensemble de jeux de données réels et artificiels. En termes de qualité interne et externe, l'algorithme est comparable aux approches existantes. En ce qui concerne le coût de traitement, notre approche permet d'obtenir de très bonnes performances en comparaison avec les autres algorithmes relationnels. Pour finir, en ce qui concerne l'utilisation de la mémoire, l'algorithme proposé est nettement moins gourmand : nous observons un gain de mémoire notable lorsque nous utilisons des points de support communs par rapport aux approches existantes.

L'algorithme proposé est donc un bon candidat pour optimiser l'utilisation de la mémoire et le temps de traitement des données relationnelles. Cependant, il n'est pas adapté aux données incrémentales et dynamiques. Nous introduisons donc dans la section suivante le formalisme des Coordonnées Barycentriques, afin d'unifier la représentation des objets et des prototypes et de permettre un processus d'apprentissage incrémental simple pour le clustering relationnel.

3.2 Coordonnées barycentriques pour le clustering relationnel

Dans cette section, nous présentons une approche de clustering relationnel basée sur le système de Coordonnées Barycentriques pour homogénéiser la représentation des objets et des prototypes et traiter de grands ensembles de données.

Dans le système de Coordonnées Barycentriques, l'espace de représentation est défini par un ensemble unique de points de support P choisis parmi les objets O. La définition d'un prototype dans l'algorithme précédant correspond au calcul d'un objet dans l'espace barycentrique. En d'autres termes, β^k sont les coordonnées barycentriques de w^k par rapport au système de points de support X_S. Cependant, contrairement aux algorithmes précédents, tout objet o^i dans le jeu de données peut également être défini en utilisant les Coordonnées Barycentriques. Les Coordonnées Barycentriques des objets se calculent facilement, elles

dépendent uniquement de la dissimilarité entre objets et points de support. Dès lors, la distance carrée entre un objet o^i et un prototype w^k peut s'écrire en fonction des coordonnées de l'objet et du prototype :

$$d(o^i, w^k) = -\frac{1}{2}(\beta^i - \beta^k)^T \cdot D_S \cdot (\beta^i - \beta^k), \tag{3}$$

avec D_S la matrice de similarité entre les points de supports.

Nous proposons deux algorithmes pour calculer les coordonnées des prototypes minimisant la somme des distances carrées : une version batch, où l'ensemble de données est conservé en mémoire pendant tout le processus d'apprentissage et une version stochastique où les objets sont présentés un par un. Bien que les approches stochastiques soient généralement légèrement plus lentes que les versions batch, elles ont l'avantage de permettre une meilleure gestion de l'utilisation de la mémoire car il n'est pas nécessaire de stocker l'ensemble de données en mémoire pour chaque étape du processus. Les processus stochastiques sont également à la base des approches de clustering en ligne et dynamique.

La version batch suppose que tout l'ensemble de données peut être stocké dans la mémoire, ce qui permet de calculer et de stocker les coordonnées barycentriques de tous les objets. La mise à jour des coordonnées des prototypes devient :

$$\beta^k = \frac{1}{|C_k|} \sum_{i|o^i \in C_k} \beta^i \tag{4}$$

Dans la version stochastique de l'algorithme, les objets de l'ensemble de données sont présentés un par un de manière aléatoire. La mise à jour des prototypes est calculée de façon incrémentale pour chaque objet présenté :

$${\beta^k}_{t+1} = \beta^k_t - \gamma(\beta^i - \beta^k_t). \tag{5}$$

avec γ le pas d'apprentissage.

Ces algorithmes ont été comparés à d'autres approches de l'état de l'art adaptés aux données relationnelles. Les approches ont été testées sur un ensemble de données réelles et artificielles de différents types. Les approches proposées ont une complexité en temps de calcul en $\mathcal{O}(N)$, significativement plus faible que l'état de l'art, indépendamment de la structure des données. En particulier, la matrice de dissimilarité n'a pas besoin d'être calculée dans son ensemble. La version batch doit avoir les objets en mémoire ($\mathcal{O}(N)$), alors que la consommation de mémoire de la version stochastique est indépendante du nombre d'objets, puisqu'ils peuvent être stockés et libérés un par un pendant le processus. Comme prévu, le temps de calcul expérimental des algorithmes proposés augmente beaucoup plus lentement que pour les autres approches (Figure 3).

Bien que la faible complexité de notre approche soit un grand avantage pour les applications réelles, il est important de vérifier si la qualité du clustering résultant n'est pas

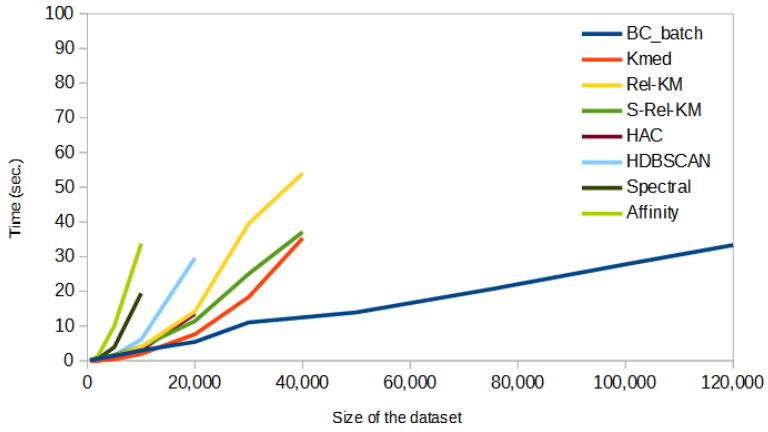


Figure 3: Effet du nombre d'objets sur le temps de calcul. L'algorithme proposé est noté BC_batch. Pour chaque algorithme la simulation s'arrête lorsque la mémoire (16Go) est saturée.

significativement inférieure à la qualité de ses concurrents. Les résultats expérimentaux obtenus démontrent la qualité des approches proposées par rapport aux algorithmes de l'état de l'art. Les qualités internes et externes de nos algorithmes sont, la plupart du temps, au moins aussi bonnes que celles des concurrents sur les jeux de données expérimentaux. Les versions batch et stochastique sont de qualité très similaire. Les algorithmes proposés sont donc aussi bons que d'autres approches lorsque la taille de l'ensemble de données est suffisamment petite pour être traitée par toutes les approches dans un temps et une utilisation de la mémoire raisonnable. Cependant, lorsque les ensembles de données à analyser commencent à être trop grands pour les approches traditionnelles, les algorithmes proposés semblent être les meilleurs candidats.

Pour finir, nous avons étudié l'effet du nombre de points de support sur la qualité de l'algorithme proposé. L'utilisation de points de support dans l'espace barycentrique est équivalent à la méthode des projections aléatoires dans un espace vectoriel. Il a été montré que cette approche est plus efficace que d'autre algorithmes de projections plus coûteux. Un choix aléatoire des points de support, en évitant la colinéarité, semble donc judicieux. Concernant le choix optimal du nombre de points de support, il devrait être égal à la dimension+1 de l'espace de représentation des données. Cependant, cette dimension est rarement connue pour les données relationnelles. Il existe pourtant un résultat théorique intéressant associé aux projection aléatoire : on peut estimer une borne d'erreur qui ne dépend que du nombre d'observations et du nombre de points de support. Ce résultat est indépendant de la dimension de l'espace et est donc un applicable dans notre cas. Cependant, les résultats des tests expérimentaux semblent indiquer qu'un nombre relativement petit de points de support donne de très bons résultats en pratique.

Les algorithmes proposés ici ont une complexité en temps de calcul linéaire et une faible utilisation de la mémoire. Ils sont ainsi adaptés aux grands ensembles de données.

En outre, les clusters qui se chevauchent ne posent aucun problème dans les approches basées sur des prototypes. Cependant, en comparaison avec les algorithmes proposés au chapitre précédent, le nombre de clusters est maintenant un paramètre à choisir, bien qu'il soit rarement connu dans les cas réels. Enfin, bien qu'étant incrémentaux, ces algorithmes à base de prototypes ne sont pas encore adaptés aux flux de données. Nous présentons dans le chapitre suivant un algorithme qui est adapté au clustering de flux de données et qui met à jours automatiquement le nombre de clusters au cours du temps en fonction de la dynamique de la structure de données.

4 Application à la catégorisation et au suivit de comportements à partir de données de navigation en ligne

Dans ce chapitre, nous appliquons les approches proposées dans cette thèse aux besoins de la société Mindlytix. Notre motivation pratique est de réaliser un profilage en temps réel des utilisateurs connectés. Les tâches de profilage visent à reconnaître l'"état d'esprit" des utilisateurs à travers leurs navigations sur différents sites. C'est une tâche très importante sur le marché international de la publicité en ligne. En effet, en comprenant mieux les intérêts des utilisateurs se connectant à un site, la publicité affichée correspondra au mieux aux besoins de ces utilisateurs. Ces informations sont calculées à partir d'une très grande base de données de navigation Internet qui répertorie les séquences d'URLs visités par un grand nombre de personnes. Chaque URL est caractérisée par des informations contextuelles et sémantiques.

Compte tenu de la nature des données sous la forme d'un flux continu et très volumineux, nous souhaitons mettre en œuvre une analyse automatique et adaptative des variations d'intérêt des utilisateurs connectés. Pour cela, nous avons effectué une catégorisation des URLs visités au cours du suivit, ainsi qu'une catégorisation des utilisateurs en fonction de leurs navigations. Pour cela, nous proposons une extension des algorithmes barycentriques proposés dans cette thèse, afin de les adapter aux flux de données, permettant un suivit dynamique des tendances d'intérêt des utilisateurs au cours du temps. Cette étude est complétée par un suivit individuel des utilisateurs, afin de détecter des changements de comportements à la fois dans les catégories d'URLs visités et dans les déplacements réels de l'utilisateur (grâce à des informations de géolocalisation). Pour cela, nous proposons une adaptation de l'algorithme de clustering basé sur l'analyse de signaux afin qu'il détecte des variations comportementales.

4.1 Classification et suivit des tendances d'intérêts sur Internet

Afin de montrer l'adéquation de nos approches pour l'analyse de l'évolution des tendances d'intérêt en ligne, nous avons analysé une base de données de suivit de navigation

d'internautes français sur deux semaines en août 2017. Environ 6 millions de connexions ont été enregistrées au cours de cette période, correspondant à 142794 utilisateurs distincts.

Pour traiter ces données, nous avons développé une adaptation de l'approche basée sur les coordonnées barycentriques capable de traiter des flux de données complexes. Dans cette approche, chaque observation est projetée dans l'espace Barycentrique lorsqu'elle est traitée, puis elle est assignée au prototype le plus proche si la distance est plus petite qu'un rayon maximum et le prototype est mis à jour afin de réduire sa distance à l'objet dans l'espace barycentrique. Dans le cas contraire, l'algorithme crée un nouveau prototype ayant les mêmes coordonnées barycentriques que cet objet. Enfin, l'âge de tous les autres prototypes est augmenté et chaque prototype d'âge supérieur à un seuil fixé est supprimé. Ainsi, le nombre de prototypes varie en fonction des variations temporels du flux.

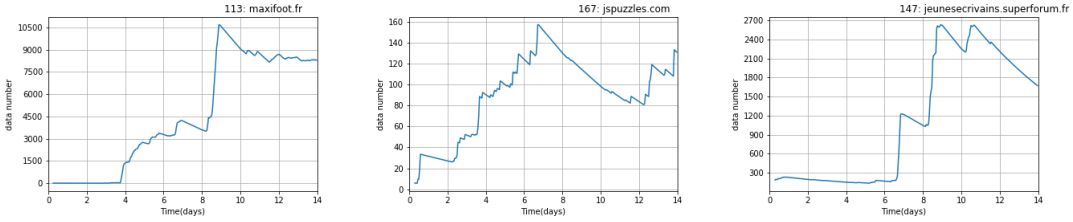
Table 1: Exemple de clusters obtenus à partir de la similarité sémantique.

# de cluster	**Exemples d'URLs associés**	**Label**	**Concept**
2	cuisinevg.fr/endives recettes.de/pomme-de-terre auvertaveclili.fr/soupe-crue-aux-carottes-noix-de-cajou grands-meres.net/crepes-au-saumon larecette.net/pommes-de-terre-tornade-parmesan recettes.de/saute-de-veau-aux-carottes	Recipe Seasoning Condiment	Recettes
3	cheveux-naturels.fr/meches-bresiliennes.php beautiful-boucles.com/coiffure-produits beautiful-boucles.com/gel-pour-definir-ses-boucles perruquescheveuxnaturels.net beautiful-boucles.com	Hair Hair dressing Hair style	Coiffure

Pour calculer les similarités entre URLs, Mindlytix a fourni deux mesures adaptées à leurs besoins. La première mesure utilise des informations sémantiques associées à chaque URL. Cette mesure permet également de comparer des URL avec un seul ou un groupe de mots. L'autre mesure est basée sur des informations contextuelles (les URL souvent visitées pendant une courte période par les mêmes utilisateurs sont similaires). Les résultats sont convaincants, ils ont été validés par les experts de Mindlytix. Les clusters sont homogènes et correspondent à des concepts clairs. La Table 1 présente deux exemples de clusters obtenus avec la similarité sémantique. Les labels ont été obtenu en prenant les titres des trois pages Wikipédia les plus similaires au prototype. Les concepts associés ont été fournis par Mindlytix.

De plus, l'évolution de la structure du flux a été enregistrée et analysée afin de mettre en évidence les tendances et les variations dans les comportements des utilisateurs au fil du temps. Certaines des variations observées peuvent être associés à des événements précis ou à des cycles dans l'intérêt des utilisateurs. Par exemple, dans la Figure 4 on peut détecter un pic d'intérêt pour le football qui correspond à un match de classification de l'équipe

de France ; une augmentation de l'intérêt pour des sites de jeux lors des weekends, ou un intérêt soudain des jeunes internautes pour la littérature à la fin des vacances scolaires.



a. maxifoot.fr b. jspuzzles.fr c. jeunesecrivans.fr

Figure 4: Évolution du nombre de visites dans chaque cluster en fonction du temps (jours). Le domaine le plus similaire au prototype est donnée pour chaque cluster.

4.2 Détection de variations dans le comportement individuel des internautes

Le suivi du comportement individuel des utilisateurs est un outil utile dans le domaine du marketing. Il existe différentes techniques pour détecter les changements dans le mode de vie des utilisateurs ou leurs comportements. L'une des solutions consiste à détecter par géolocalisation le changement d'habitude géographique des utilisateurs. En particulier, les personnes qui décident de déménager sont des cibles très intéressantes pour les fournisseurs de publicité en ligne. Une autre solution est de suivre les intérêts d'un utilisateur lors de ses navigations sur Internet. En suivant les utilisateurs, les agences de publicité ont plus de chance de vendre leurs offres. Nous proposons ici une approche basée un algorithme de détection de changement dans les signaux afin de détecter des variations dans le comportement d'un utilisateur. Nous avons testé l'approche sur un ensemble de données simulé, puis nous l'avons appliqué à un ensemble de données réelles fournies par la société Mindlytix pour plus de 140000 personnes.

L'idée principale de l'approche proposée consiste à représenter le comportement d'un utilisateur par la distribution des lieux fréquentés, en fonction du code postal du lieu de connexion, ou par la distribution des catégories des URLs visités, les catégories étant définie par les clusters d'URLs calculés précédemment. Pour chaque utilisateur, une distribution de référence est calculée sur une fenêtre temporelle de taille fixe : 10 jours pour les données de géolocalisation et 5 jours pour les données de navigation. La distribution de cette fenêtre de référence est comparée avec les distributions calculées sur un fenêtre glissante de même taille qui suit les variations avec un pas de un jour. La similarité entre deux distributions (fenêtre de référence et fenêtres glissantes) est calculée par la divergence de Jensen-Shannon

(JSD) basée sur la divergence de Kullback-Leibler (D_{KL}) :

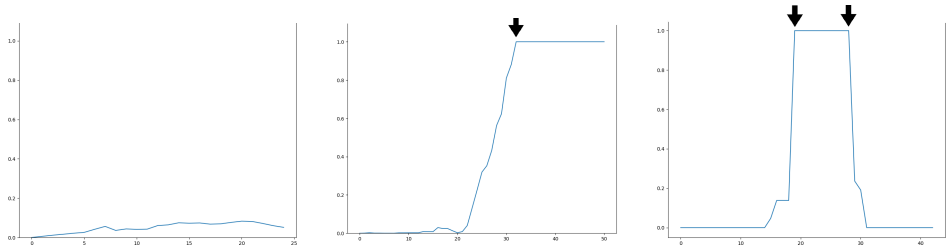
$$JSD(P \parallel Q) = \frac{1}{2} D_{KL}(P \parallel M) + \frac{1}{2} D_{KL}(Q \parallel M), \text{ avec } M = \frac{1}{2}(P + Q) \tag{6}$$

Pour deux distributions de probabilité discrètes P et Q, la divergence de Kullback-Leibler s'écrit :

$$D_{KL}(P \parallel Q) = -\sum_i P(i) \log \frac{Q(i)}{P(i)} \tag{7}$$

Le signal ainsi obtenu représente l'évolution des différences par rapport à la fenêtre de référence, et permet de détecter des changements importants de distributions : un déménagement ou un changement d'intérêt. Cette détection est automatique, grâce à une version légèrement modifiée de l'algorithme de clustering par traitement de signal décrit précédemment. Cette fois, l'algorithme détecte des "sauts", dans le signal, qui caractérisent des variations de comportement. Un processus de lissage itératif permet à nouveau d'éliminer les fluctuations aléatoires du signal.

Les tests effectués sur les signaux simulés montrent que l'algorithme est performant pour cette tâche. Seuls 1.1% des changements ne sont pas correctement détectés, aucun faux positif n'a été produits et l'erreur moyenne pour la prédiction de la date de changement est inférieure à 2%.



a. Pas de changement b. Changement majeur c. Changement temporaire

Figure 5: Exemples de signaux obtenus sur les données de l'entreprise et changements détectés.

La Figure 5 présentes trois exemples de signaux caractéristiques d'utilisateur qui ne changent pas de comportements, qui changent de comportement ou qui changent de comportement puis reviennent au comportement de référence. Les changements détectés automatiquement par l'algorithme semblent correspondre aux données et ont été validés par les experts.

5 Conclusions et perspectives

En résumé, cette thèse présente des algorithmes adaptés aux données relationnelles statiques et dynamiques. Les approches basées sur la réorganisation de matrice fournissent une visualisation utile et ne nécessitent pas de paramètre, mais peuvent être coûteuses en termes de mémoire et ont des difficultés à détecter correctement les clusters en contact, qui ne sont généralement pas correctement représentés par le processus de réorganisation. Les approches basées sur des prototypes sont adaptées au grand volume de données et à l'analyse des flux de données, mais reposent sur des paramètres définis par l'utilisateur (en particulier le nombre de clusters).

Afin de produire des approches capables à la fois de détecter le nombre de cluster et la présence de clusters en contact, une première idée serait d'utiliser une estimation locale de densité, dans laquelle une densité élevée représente un centre de cluster alors que de faibles densités sont caractéristiques des frontières entre clusters. De plus, puisque nous sommes en mesure de traiter des données relationnelles massives basées sur une mesure de similarité, l'étape suivante consisterait à trouver automatiquement cette mesure de similarité. Une idée consisterait à entrainer un algorithme d'apprentissage profond capable de produire, pour chaque type de données, une mesure de similarité définie comme la quantité d'informations communes partagée par les différents objets.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Publications

International peer-reviewed Journal:

Rastin, P., Matei, B., Cabanes G., Bennani, Y., Marty, J.M. (2018): *A new sparse representation of complex data: application to dynamic clustering of web navigation.* (under review in Pattern Recognition).

Rastin, P., Matei, B., Cabanes G., Grozavu, N., Bennani, Y. (2018): *Impact of Learners' Quality and Diversity in Collaborative Clustering.* Accepted (Invited)

International refereed conference papers in proceedings:

Rastin, P., Matei, B. (2018): *Prototype-based Clustering for Relational Data using Barycentric Coordinates.* International Joint Conference on Neural Networks (IJCNN).

Rastin, P., Matei, B. (2017): *Incremental Matrix Reordering for Similarity-Based Dynamic Data sets.* International Conference on Neural Information Processing (ICONIP).

Rastin, P., Matei, B., Cabanes, G. (2017): *Visualizing changes in the distribution of relational data using dynamic reordering.* 61st World Statistics Congress (WSC).

Rastin, P., Matei, B., Cabanes, G., El Baghdadi, I. (2017): *Signal-Based Autonomous Clustering for Relational Data.* International Joint Conference on Neural Networks (IJCNN).

Cherki, S., **Rastin, P.**, Cabanes, G., Matei, B. (2016): *Improved Sparse Prototyping for Relational K-means.* IEEE Symposium Series on Computational Intelligence (SSCI).

Rastin, P., Zhang, T., Cabanes, G. (2016): *A new clustering algorithm for dynamic data.* International Conference on Neural Information Processing (ICONIP).

Rastin, P., Cabanes, G., Grozavu, N., Bennani, Y. (2015): *Collaborative Clustering: How to Select the Optimal Collaborators*? IEEE Symposium on Computational Intelligence and Data Mining (CIDM).

Rastin, P., Kanawati, R. (2015): *A multiplex-network based approach for clustering ensemble selection.* IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM).

Chapter 1

Introduction

Table of Contents

1.1 Problematic

Unsupervised learning, or clustering, is an important task in the process of extracting knowledge from data [28]. Clustering is the technique of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other clusters [13]. The complexity of this task has increased dramatically in recent decades with an explosion in the size of available data-sets. The processing of this type of data requires the development of very low complexity or highly parallelized approaches. This has been the subject of numerous studies recently, with the popularization of the methods of "Big Data" based on highly parallelized architectures [55]. Despite clear progress in this area, many challenges remain, particularly for the processing of complex and dynamic data.

Indeed, the growing production of data with a complex structure (e.g. images, networks, texts, sequences) requires efficient and appropriate analytic tools. The analysis of complex data in general is a growing field, with the proliferation of textual data on the Internet, data from social networks, symbolic data (intervals, distribution) coming for example from sensor networks, etc. Most unsupervised approaches are adapted to deal with vectors in a Euclidean space, clusters being computed based on the Euclidean distance. However, in many case, the objects cannot be easily defined in a Euclidean space without a loss of information and/or a costly pre-processing. The solutions of transforming this information into vector data have indeed shown their limits, and researchers are now interested in solutions that are directly adapted to each type of data, with similarity metrics adapted to the data complexity: in [99, 193, 192] the authors propose a method of clustering for symbolic data (histogram), [57] proposes a clustering approach for textual data-sets, in [35] a clustering approach in proposed for interval data-set, etc. Specific algorithms are obviously limited to specific data type. For example, an algorithm created for textual data-sets cannot deal with images data-sets. One solution is to use a Relational Clustering approach based on kernel or dissimilarity matrices (the dissimilarity setting is more general than the kernel setting, because it is always possible to construct a dissimilarity matrix from a kernel [168]). Indeed, most data-sets can be represented by their relations or their similarities. When it is the case, they are sometimes called Relational Data. Relational clustering algorithms form a family of algorithms adapted to relational data. Some clustering algorithms are naturally adapted to deal with dissimilarity matrices and can be used to analyze relational data-sets, such as the DBSCAN family [81, 37], Spectral Clustering [174], Affinity Propagation [65] or Hierarchical Clustering [133]. None of these algorithms use prototypes and they do not benefit from the associated advantages. In particular, they all have a non-linear computational complexity. If the objects of a data-set are described in vectorial forms, the definition of cluster's prototypes is straightforward. In that case, a prototype is a vector defined in the same vectorial space, usually defined as the vectorial barycenter of the objects belonging to its cluster. Few works have been done yet in the domain of prototype-based

relational clustering, but some authors have worked on using K-means with relational data [152, 45, 90, 169]. However, as far as processing power and memory usage are concerned, the implementations are still very expensive ($\mathcal{O}(N^2)$ for N objects), making it unusable for large data-sets.

Moreover, in many cases, these data are in perpetual evolution, characterized by a variable structure over time, new information constantly appearing. Dynamic clustering processes have been the subject of many works in recent years due to the large number of possible applications in many fields [123]. However, this is a difficult problem because of the calculation and storage costs associated with the volumes involved. Efficient algorithms must be capable of working with constant memory occupancy despite the evolution of the data, the entire database being unable to be stored in memory. Indeed, the stream of information represents usually an enormous mass of data (these data are often called data stream) [146]. In addition, the probability distribution associated with the data may change over time ("concept drift"). As a result, the segmentation of the data (i.e. the structure of the stream) is also constantly evolving [188]. If one wants to keep a history of the structure of the data over time, it is necessary to be able to describe this structure in a highly condensed form (in particular without having to memorize each observation). Furthermore, the algorithms must be able to detect and manage any change in the data structure, by comparing a new data with a model of the data perceived up to now. The whole learning process will have to adapt in real time to the evolution of the data, which is the main challenge of this field. Most methods adapted to data streams assume that the segmentation must be computed over the entire stream (see [150]). However, the stream of data can also be seen as an infinite process consisting of data that evolve constantly over time [2, 1]. In the context of supervised learning (each data is associated with a class, which the algorithm must learn to predict), several solutions have been proposed for classifying data streams in the presence of concept drift. These solutions are generally based on adaptive maintenance of a discriminating structure. Binary rule set methods [198], decision trees [96] and classifier sets [181, 113] can be mentioned. In the field of unsupervised learning, the detection of novelty in the structure of a database has been the subject of many works in recent years (see, for example, [120, 108]) because of the large number of possible applications in many fields [77, 15, 35]. The main stakes for the study of data streams are the condensed description of the properties of a stream [70, 135, 34], but also the detection of variation or change in the stream structure [39, 6].

The combination of these three characteristics (size, complexity and evolution) presents a major challenge in the field of data mining, and few satisfactory solutions exist at the moment, despite increasingly evident needs from companies. Indeed, in spite of recent advances in scalable data analysis, there is currently very little works on evolutionary learning adapted to complex or heterogeneous data, which nevertheless form the majority of the data produced by companies [175]. This thesis focuses on the development of new clus-

tering methods adapted to dynamic and relational data. This is a real challenge because usually approaches adapted to relational data have a non-linear complexity regarding the number of objects in the data-set, which is not adapted to the analysis of fast-changing dynamic data. This work is funded by the ANRT ("Association Nationale de la Recherche et de la Technologie") in the context of a CIFRE convention ("Conventions Industrielles de Formation par la REcherche") with the company Mindlytix.

1.2 Industrial Needs

Mindlytix is a company that offers an innovative real-time profiling solution for connected users. The goal of Mindlytix is to put on the market a platform that recognizes the "mindset" of people through their navigation on the various websites or their interaction with digital "touch points" (the varying ways that a brand interacts and displays information to current and prospective customers). It intervenes in the international market for "programmatic advertising" (computer-based individualized advertising in real time) by assigning a profile to each user connecting to a site that can offer advertising, so that the displayed advertising corresponds best to the needs of the user. These profiles are computed from a very large database of Internet browsing which lists URL sequences or touch points visited by a large number of people.

Each URL or touch point is characterized by semantic information close to the natural language. It is necessary to have an "actionable" representation of these opportunities in order to be able to select them according to different criteria (linguistic, conceptual, proximity, etc.). Given the nature of the data in the form of a continuous and very voluminous stream, Mindlytix wishes to be able to implement (for example, on a national scale) an automatic and adaptive analysis of the frequent changes and the fluctuating behaviors of the connected users using an adapted representation of the data structure. The interest of the company is to develop optimal clustering algorithms on these very large, complex and dynamic databases, in order to detect, on the one hand, informative "concepts" describing the URLs and touch points visited in function of the semantic information associated with them, on the other hand to categorize the URLs and touch points according to their similarity, and finally to detect "mindset" of a user (profiles) from the sequences of concepts encountered during its recent navigation.

1.3 Challenges

The aim of this thesis is to design innovative methods of scalable unsupervised learning for complex data, adapted to the problems of the company. Indeed, the proposed methods must be adapted to the different stages of the Mindlytix data analysis process (definition of concepts, grouping of touch points and creation of mindset). Each of these steps requires an automatic clustering of dynamic objects: grouping of URLs and touch points, grouping

of semantic information for the definition of concepts and grouping of navigation traces from the concepts. Each step will be based on the results obtained in the previous stage, in order to work on increasingly synthetic levels of abstraction allowing a fine analysis and interpretation of the data and their evolution. The proposed approaches will therefore be adapted to both the large volume of data and the rapid evolution of the structure of these data. To achieve this objective, we must compute a condensed representation of the data distribution, so as to provide a simplified representation of the data while reducing the dimensions of the problem. This representation can be based, for example, on the learning of sets of referents. The approaches used for this must be very fast (low complexity) and have a low memory usage in order to be able to process large volumes of data, while offering a suitable representation of the data structure, minimizing the loss of information. This combination presents an important challenge.

The clustering algorithms will have to be adapted to the chosen representation model, in order to use all the available information, but the problem of the speed of execution is slightly less important, which offers more possibilities for obtaining a result of good quality. An important step of the work will be to define similarity measures adapted to each type of data (semantics, navigation trace, etc.). These measurements will be used by the algorithms during the representation phase of the data structure and/or for the clustering. The new approaches of analysis proposed during this thesis will be tested on simulated data at first and then on the actual data of the company. The performances of the proposed methods will be compared with those of existing methods in order to validate the innovative nature of our approaches. Performance will be evaluated according to criteria defined jointly by the laboratory and the company, which will include the quality of the results obtained, the execution time and the simplicity of implementation. We will pay particular attention to scale-up tests to verify the algorithm's capabilities on very large volumes of fast-moving data.

The optimization sought concern both the speed and efficiency of the algorithms and the real cost reductions. In practical industrial life, large volumes of data must be processed in a very short time on reduced material resources (the daily volume to be treated by Mindlytix can exceed several terabytes). Moreover, the databases to be processed are dynamic and the proposed methods must be able to be updated quickly to detect without delay the emergence of new concepts or the emergence of new user profiles. In particular, the Mindlytix solution delivers mindset in a few milliseconds, several hundred thousand times per second, and the status of people can change every second. Finally, the data to be processed are complex because they are characterized by semantic, incomplete and noisy information. These data are very hollow, which requires an adapted approach for the creation of concepts, classes, topological space of representation with metrics not necessarily Euclidean. The solution therefore requires a very compact representation space for all the information.

1.4 Proposed Approaches

To process relational dynamic data-sets and fulfill the need of the Mindlytix company, we propose in this thesis new incremental clustering methods based on similarity measurements. Once a similarity measure is defined as a distance or a kernel function, the algorithm uses only the similarities between objects to create a model of the data structure. These methods are very powerful because they can adapt to any type of data (as long as it is possible to define a similarity between them). However, they are currently not adapted to large or dynamic data, because it is necessary to calculate the similarity between all the data and to store these values in memory, which is impossible when the bases are large or dynamic.

We will thus explore first an incremental construction of the similarity matrix restricted to a limited time window. In that way the time and memory complexities stay linear regarding the number of objects. The method also must treat the objects "on the fly", in an unpredictable order. We present an algorithm adapted to dynamic data-sets, allowing a visualization of the current structure of data-sets. The proposed approach is able to forget the outdated information. We will also propose a new clustering algorithm able to discover the structure of such matrices without any user-chosen parameters. This approach is based on signal theory. By using the characteristics of reordering methods to produce a one-dimensional signal of pairwise distances. The aim is to detect "jumps" of distances between two clusters if two clusters are distant enough.

In addition, we propose new prototype-based algorithms for relational data. These algorithms improve existing relational K-means approaches by reducing the computational complexity and the memory usage of the clustering tasks. The idea is to use a limited number of support points, chosen from the data, as support for the construction of the prototypes representing the clusters, which will then be defined as a weighted combination of support points based on the Barycentric Coordinates formalism. In particular, this formalism allows the construction of incremental clustering algorithms adapted to big data-sets and data streams.

In order to analyze the dynamics of users' behavior in the data provided by Mindlytix, we propose an algorithm adapted to relational data streams. The number of prototypes is updated in real time, so as to obtain an optimal model of the distribution of the data over time. In particular, mechanisms for detecting changes in the structure of the data stream are implemented, in order to modulate the plasticity of the system in real time and find a good compromise between stability and plasticity. We also propose to analyze the temporal evolution of the inter-structure and the intra-structure of the groups, which will allow to identify proximities or differences between the subsets of data, and to explain the structure identified thanks to a fine analysis of similarities or differences and characterize groups explicitly and conveniently for the users.

1.5 Thesis Organization

The rest of the manuscript is organized as follows:

- Chapter 2: Review of Clustering Approaches for Relation Data and Data Streams
  In this chapter we review some unsupervised algorithm which can be adapted to relational data-sets. These algorithms are described in different categories: distance-based algorithms, graph-based algorithm and density-based algorithm.

  Then, we provide an overview of the clustering approaches adapted to data streams. We present the general model, the use of time windows and the different families of clustering approaches.

  Finally, we present some classical indexes able to assess the quality of a clustering.

- Chapter 3: Presentation of the Experimental Data-sets
  In this chapter we introduce the data-sets used in our experimental protocols in the following chapters. These data-sets are divided to four categories: vector data-sets, sequential data-sets, distributional data-sets and textual data-sets. The real data-sets come from UCI Machine Learning Repository [124], UniProt [50] and Wikipedia [199]. In addition, we generated several artificial data-sets.

  For each type of data, we present some of the most used measures of similarity. We also provide visualizations of some of these data-sets.

- Chapter 4: Clustering Approaches Based on Incremental Matrix Reordering
  This chapter introduces a new incremental algorithm for the reordering of similarity matrix. This matrix is constructed in real time so as to follow the dynamics of the data (This work is published in ICONIP 2017 [161]).

  We then present a new autonomous clustering algorithm for similarity-based data using reordering matrix. The idea is to introduce a method of clustering which does not need the number of clusters as a parameter to give (This work is published in IJCNN 2017 [163]).

- Chapter 5: Prototype-based Clustering for Relational Data
  After reviewing the relational K-Means approaches, this chapter presents a new algorithm of sparse relational K-Means optimizing the number of support points (This work is published in SSCI 2016 [45]).

  Then, we propose a new incremental relational prototype-based algorithm using the Barycentric Coordinate formalism (This work is accepted in IJCNN 2018 [162] and submitted in Pattern Recognition).

  We finish this chapter with a discussion on *how* and *how many* support points we must choose.

- Chapter 6: Application to the Analysis of Internet User's Behavior
  This chapter describes a relational data-stream clustering algorithm and presents the results of the analysis from the data provided by the company Mindlytix. We used two ways of computing similarities between URLs and we tested several labeling approaches. The dynamics of the users' interest over time is also monitored (This work is submitted in Pattern Recognition).

  In addition, we propose a new application to follow and detect the individual change of users' behavior based on geolocation and navigation logs (This work is in preparation for an international conference).

- Chapter 7: General Conclusion and Perspectives
  This chapter concludes the manuscript and presents some ideas and projects as perspectives of the work presented in this thesis. In particular, we describe a two-level algorithm to improve the clustering results of dynamic neural networks models (This work is published in ICONIP 2016 [164]).

1.6 Definitions and Notations

In this document some technical terms will be frequently used. We give here a definition of the most important notions, as well as, if necessary, the mathematical notations used in this manuscript:

- **Object**: An object o is the elementary description of a phenomenon to be studied. The set of objects available for the analysis of the phenomenon is called in this manuscript "data-set" and is denoted by $O = \{o^1 \ldots, o^N\}$, where N is the size of the data-set.

- **Vectors**: object are often represented by vectors (set of variables with known values). A vector of $\mathbb{R}^d$ is noted $x = (x_1, \ldots, x_d)$, d being the number of numerical variables associated with each object. d is also called the "dimension" of the data representation space. When the objects are represented by vectors, they are sometimes referred to as "data points".

- **Similarity**: The similarity between two objects i and j is denoted $s(i, j)$. Similarity between the objects to be studied is the main (and often unique) information allowing the clustering algorithm to partition the data-set. In most cases, this similarity is calculated according to a "distance" or "dissimilarity" measure adapted to the type of data and the problem to be solved. This distance, generally denoted $d(i, j)$, is a measure of dissimilarity that satisfies certain mathematical properties. A small similarity corresponds of a big value of s and a small value of d. When the data are described in a vector space, the distance most used is the Euclidean distance, then denoted $\|i - j\|$.

- **Prototypes**: A prototype w is a representative of a set of data. A prototype is often of the same type as the object (e.g. in the case of vector data the prototype is a vector), and it is always possible to define a similarity between a prototype and an object. When the prototype is a vector in $\mathbb{R}^d$, it is here noted μ. Some algorithms proposed in this thesis are based on the learning of a set of prototypes $W = \{w^1, ..., w^K\}$, where K is the number of prototypes. This process is called learning because the prototypes are iteratively adjusted to the object during the execution of the algorithm. At the end of the prototypes learning step, we can associate each object o with its most representative prototype (i.e. the most similar). We note this prototype $w^*(o)$. Sometimes, we also define the second most representative prototype of an object, which we denote by $w^{**}(o)$.

- **Data structure**: When we talk about data "structure" in this thesis, we refer to the underlying distribution of the data-set. In particular, a clustering algorithm proposes a model of this distribution in the form of a partition of clustered data. In each cluster the objects are more similar than objects in different clusters. We call "natural" clusters of the data the clusters which represent distinct modes in the distribution of the data: the density of "intermediate" objects should be lower than the density of objects in each cluster.

Chapter 2

Review of Clustering Approaches for Relation Data and Data Streams

Table of Contents

2.1 Introduction

Unsupervised learning, or clustering, is a very important tool in exploratory analysis of unlabeled data. It is used for cluster detection, when there is no *a priori* information about the internal structure of these data. A clustering problem can be defined as the partitioning of a set of elements into several relevant subgroups (clusters). The objects grouped in the same cluster must be similar to each other (internal homogeneity), unlike objects belonging to different groups (external separation). Unsupervised learning approaches play a very important role in understanding varied phenomena described in data-sets (see, for example, [102, 201, 22, 48, 179]). The most important applications are speech recognition [56], image segmentation [27], text mining [7], categorization of customers [159], etc. Clustering algorithms are also widely used in geography [63], astronomy [42] or genetics [14].

A large number of algorithms have been proposed in the literature [101]. These different algorithms have been grouped according to different taxonomies according to the characteristics taken into account:

1. **The representation of the objects**. Each algorithm is generally adapted to a single type of data. The object can be represented in different forms, for example:
   - Digital or categorical vectors.
   - Sequences, trees, graphs.
   - Similarity matrix.
2. **The grouping methods implemented:**
   (a) *Agglomerative methods*: The objects are iteratively added to the more relevant cluster.
   (b) *Divisive methods*: The objects are all cut into a single cluster, then the algorithm iteratively divides this cluster into smaller clusters for better representation of the data structure.
3. **The form of the partition obtained:**
   (a) *Partitive methods*: The algorithm attempts to produce a disjoint partition of data.
   (b) *Fuzzy methods*: The partition is not necessarily disjointed; each object belongs to several clusters.
   (c) *Hierarchical methods*: The data are grouped hierarchically under a tree (or dendrogram). The nodes belonging the same cluster have the common parents.
4. **The criteria for defining a partition.** All algorithms are based on a measure of similarity between objects, the choice of which is very important for the quality of the results obtained. There are different ways of using this similarity, for example:

(a) Partition according to *distance* or *dissimilarity*: Clusters must be dissimilar from one another. The similarity between two clusters can be defined in different ways. Most often, we measure the similarity between the barycenters or the medoids (the most representative element) of the data belonging to each cluster, or between the two most similar objects belonging to one of the clusters.

(b) Partitioning according to *connectivity*: These methods construct a graph from of the measure of similarity between objects. Clusters are then searched by minimizing connectivity between different clusters and maximizing this connectivity within each cluster, using graph analysis tools.

(c) Partitioning according to *density*: These methods are based on an estimate of the *density* of the data in the representation space as a function of their similarity. The algorithm looks for boundaries between clusters, these boundaries are characterized by a low density with respect to the density of each cluster. Thus, low density areas define the boundaries of clusters.

In the remainder of this chapter we first present the main clustering algorithms adapted to relational data (i.e. objects described by their dissimilarities). Then, we describe the various existing approaches for data stream clustering. At the end of this chapter, we describe the quality indexes used during this thesis.

2.2 Clustering Approaches Adapted to Relational Data

2.2.1 Relational Data

A numerical object data $x = (x_1, \ldots, x_d)$ is in general a vector in a finite d-dimensional real space, where each vector component x_i is a feature value of the associated data x. But not all objects are or can be described by a vector. Another way of representing objects of a data-set is by the relations between them. Let's consider a set of objects $O = \{o^1, \ldots, o^N\}$. These objects can represent virtually anything: Tweets, cars, sequences of protein or music scores. They are commonly represented by a relational matrix $R = [relation(o^i, o^j)]$ with $1 \leq i, j \leq n$. The relational matrix often takes the form of a dissimilarity matrix D, where the values can be interpreted as a dissimilarity or a distance d between objects. Small values represent very similar data and *vice versa* (note that the opposite can be used in the form of a similarity measure s). The minimal constraints on a dissimilarity measure $d : (i, j) \longrightarrow d(o^i, o^j)$ are [168]:

- Non negativity: $d(x, y) \geq 0$ for all x and y
- Symmetry: $d(x, y) = d(y, x)$ for all x and y
- Reflexivity: $d(x, x) = 0$ for all x

Thus, the input data-set for such data is a dissimilarity matrix.Consequently, the dissimilarity matrix D for an N elements data-set, is:

- Square: $N \times N$ matrix
- Hollow: $d(i,i) = 0$ for all i
- Symmetric: $d(i,j) = d(j,i)$ for all i and j
- Non-negative: $d(i,j) \geq 0$ for all i and j

We have chosen here to classify these algorithms according to their criterion of definition of a partition.

2.2.2 Distance-Based Approaches

Most clustering algorithms are based on a measure of the distance between the objects. These algorithms generally minimize a cost function that favors the discovery of compact and well separated clusters: the objects of the same cluster are close to one another and are far from the objects of the other cluster. The most classical approaches are hierarchical methods and partitive methods. In Hierarchical Agglomerative Classification (HAC) approaches, a hierarchical tree (dendrogram) is constructed from the sets to be classified, the nodes of the tree coming from the same parent forming a homogeneous group [197]. On the contrary, the partitive methods (K-Medoid for example) gather data without using a hierarchical structure. Generally, in this case, each cluster is represented by a "center" or "prototype" defined in the data space.

2.2.2.1 K-Medoids Algorithm

The main difficulty of prototype-based approaches is that they require the data points to be the elements of a Euclidean space, since we need to average the data points somehow. To deal with this issue, K-medoids [109] is an alternative version of K-means which uses objects of the data-set as prototypes, i.e. using medoids instead of centroids. In that way, it is not necessary to define prototypes as a separate set of objects (or an "average" object). The most used implementation of the K-medoids approach is Partitioning Around Medoids (PAM) [109]. PAM is known to be more robust to noise and outliers than regular K-means, mainly because it works directly with pairwise dissimilarities. On the other side, PAM update step can cause cluster overlapping that cannot be untangled in the following iterations, because of the limited number of potentials centers compared with K-means.

The idea of K-medoids is to find the prototypes among the objects o of the data-set $O = \{o^1, \ldots, o^N\}$ in order to minimize a cost function $\mathcal{J}$: the sum of distances between each data and its closest medoid (see algorithm 1).

$$\mathcal{J} = \sum_{k=1}^{K} \sum_{i|o^i \in C_k} d(o^i, m^k) \tag{2.1}$$

with K the number of clusters (i.e. the number of medoids), o^i an object of the data-set, m^k the medoid for the cluster k and C_k the set of objects in cluster k.

Algorithm 1 K-medoids

Input: D, K.
Output: The K medoids m^k.

```
1: Choose randomly K medoids.
2: Associate each data point to the closest medoid.
3: while the cost of the configuration decreases do
4:     for each pair of object o and medoid m do
5:         Swap m and o.
6:         Recompute the cost using equation (2.1).
7:         if the cost increased then
8:             undo the swap.
```

K-Medoids algorithm is a fast algorithm in compared to the others algorithms but this algorithm uses a greedy search which may fail to find the optimum solution.

2.2.2.2 Hierarchical Agglomerative Clustering Algorithms

As said before, Hierarchical Agglomerative Clustering (HAC) methods use a hierarchical tree (dendrogram) constructed from the sets to be classified. The nodes of the tree from the same parent form a homogeneous group [197, 178, 81, 104]. There are many methods for creating the dendrogram, all based on a simple general algorithm.

Algorithm 2 Hierarchical Agglomerative Clustering

Input: D, k, a linkage criterion.
Output: A dendrogram and a cuttoff level.

```
1: Assign each object o^i its own cluster number.
2: Compute the similarities between clusters according to a previously selected measure.
3: Merge the two most similar clusters and update the hierarchical tree (Figure 2.1).
4: while all objects are grouped into a single cluster do
5:     Return in 2.
6: Select a cutoff level of the tree to get k clusters.
```

The main source of variation between the different algorithms proposed in the literature is the choice of the linkage criterion: the measure of similarity between two clusters making it possible to select the clusters to be merged at each step of the process. The main measures proposed are:

1. **Simple Link**: This is a measure of the minimum distance between an object in a cluster and an object in the other cluster. This measure is very popular. It makes it possible to detect clusters of any form, potentially non-hyperspherical, which is often

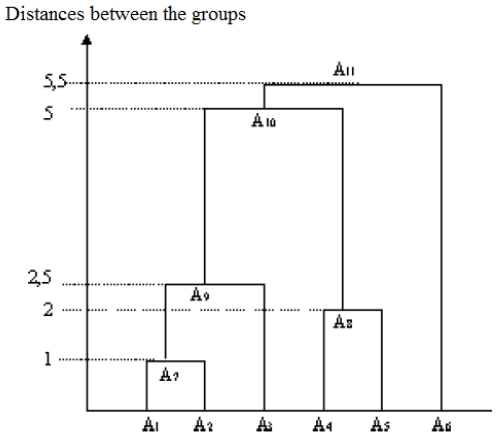


Figure 2.1: Tree constructed by Hierarchical Classification.

necessary for the search for natural clusters in the data. In contrast, this measure is very sensitive to noise and is not able to detect clusters in contact.

2. **Full Link**: This is a measure of the maximum distance between an object of a cluster and an object of the other cluster. This measurement is very sensitive to noise and extreme values.

3. **Medium Link**: is the average distance between an object in a cluster and an object of the other cluster. This measure is not sensitive to noise but tends to favor clusters of hyperspheric shapes.

There are two main limitations to the use of this type of algorithm for analysis of actual data. First, once the dendrogram is obtained, it is necessary to choose a cut-off level to obtain the clusters. The choice of this level of cut remains a difficult problem despite many proposed methods (see [104]). Second, all of these algorithms have a minimum complexity proportional to the square of the number of objects, making them unusable for large database analysis.

2.2.3 Graph-Based Approaches

Another approach proposed by many authors is to consider relationships of distance or similarity between objects as a graph [139, 200, 81, 88]. This representation is independent of the type of objects (vectors, texts, images, etc.) since only the similarity between objects is used. In this type of graph, each node represents an object and each weight associated with an arc represents the distance or the similarity between the two connected objects. A cluster is then defined as a set of strongly connected objects (the values associated with the edges are large) and weakly connected to the other objects (low values associated with

the edges). Such a representation allows the use of tools derived from the theory of graphs. We present in this section two algorithms based on a representation in the form of a graph and which present good performances: the Spectral Clustering [174, 149] and the Affinity Propagation [65].

2.2.3.1 Create a Graph from the Data

Let a set of objects $O = \{o^1, \ldots, o^N\}$ and a similarity measure $s(i,j) \geq 0$ between each pair of objects o^i and o^j. If we have no information other than the similarity between the data, a similarity graph $G = (V, E)$ is an interesting representation of the relations between data. V is the set of vertices of the graph, each vertex represents a data. E is the set of edges of G connecting the data to each other if their similarity is not zero, each edge between two objects o^i and o^j is weighted by the value of $s(i,j)$.

To create such a graph on the basis of $s(i,j)$ similarities, several methods have been proposed (See [194]):

1. *The complete graph*: Each object is connected with all others if the similarity is not zero. All arcs are weighted by the values of $S = (s(i,j))_{i,j=1,\ldots,N}$. In general, this method is used only if the similarity measure already translates the local neighborhood between data. We often use a Gaussian similarity function based on a dissimilarity measure d: $s(i,j) = \exp\left(\frac{-d(o^i,o^j)^2}{2\sigma^2}\right)$. The parameter σ controls the size of the neighborhood.

2. *The ϵ neighborhood graph*: In this case, only objects with dissimilarity (or distance) less than ϵ are connected in the graph. In general, the edges of the graph are not weighted, which returns to creating a complete graph on a binary discretization of the similarity matrix.

3. *The graph of k nearest neighbors*: In this type of graph an object o^i is connected to another object o^j, if one of the two data forms part of the k closest neighbors of the other. Unlike the previous method, this type of graph correctly represents clusters of different densities. A variant of this method is the graph of k closest neighbors, which gathers two objects only if each is one of the k closest neighbors of the other. In this case, the graph tends to connect objects in regions of homogeneous density and not to connect regions of different densities.

2.2.3.2 Spectral Algorithm

The similarity matrix between data $S = (s(i,j))_{i,j=1,\ldots,N}$ is also the matrix of weight of the edges, called adjacency matrix of G. The degree of a node of G is defined such that:

$$d_i = \sum_{j=1}^{N} s_{ij}$$

The matrix of degrees Deg is then the diagonal matrix with the degrees of $d_1, ..., d_N$ on the diagonal. The main tool of Spectral Clustering is the Laplacian of a Graph, which can be defined as follows:

$$L = Deg - S$$

It is often preferable to use a standardized Laplacian [174, 149], there are two types:

$$L_{sym} = Deg^{-1/2} L Deg^{-1/2}$$

$$L_{rw} = D^{-1} L$$

A very interesting property of the Laplacian is that the number of zero eigenvalues of L is equal to the number of subsets of the graph not connected to the other vertices. Each subset is a well-separated cluster of the data-set, also called a connected component for a graph. Moreover, any eigenvector of L associated with a zero eigenvalue is a vector of size N (the number of objects) taking a non-zero constant value for each data belonging to one of the related components and a zero value for the other object. Each of these eigenvectors is thus a representation of a connected component, the set of eigenvectors associated with a zero eigenvalue represents the set of connected components of the graph.

In the same way, if the graph is divided into k weakly connected subsets, the k least eigenvalues of L will correspond to eigenvectors with stronger values for the vertices of one of the subsets than for the others highs. These vectors are called the first k eigenvectors of L. The idea of the spectral analysis is therefore to choose the number k of desired groups and then to represent each object in a space with k dimensions where the coordinates correspond to the values of the first k vectors. In this space, the strongly connected objects will be very close (or similar if the set is a connected component) and the weakly connected sets will be very far apart. A simple k-mean is enough to discriminate the different clusters. The algorithm is described in Algorithm 3.

If the Laplacian L_{sym} is used, it is necessary to normalize M such that the sum of each line is equal to 1 (see [149]). The main advantage of spectral analysis is that it is possible to detect clusters of arbitrary shapes. Indeed, such clusters are represented in the graph as strongly connected components. These components will be projected by the algorithm in a hyper-spherical form easily detected by an algorithm like K-Means. There are two

Algorithm 3 Spectral Clustering

Input: A similarity matrix S and a number k of clusters.

Output: Data segmentation.

1: Construct the graph from S.
2: Compute the Laplacian L of the graph.
3: Compute the first k eigenvectors $v_1, ..., v_k$ of L..
4: Let $M \in \mathbb{R}^{N \times k}$ be the matrix containing the vectors $v_1, ..., v_k$ in column. For all $i = 1, ..., N$, define a vector $y_i \in \mathbb{R}^k$ corresponding to the ith line of M..
5: Segment the points $(y_i)_i = 1, ..., N$ into $\mathbb{R}^k$ with k-means to obtain a segmentation of the data.

major disadvantages to this approach. First, we can note a high sensitivity to noise and more generally a difficulty in detecting clusters in contact. Secondly, the complexity of the calculations involved does not make it possible to use spectral analysis for large databases. Indeed, the main step of the process is the computation of the vectors and eigenvalues of a matrix of size $N \times N$, N being the number of objects. On large matrices, this step can be extremely costly in computing time [194].

2.2.3.3 Affinity Propagation Algorithm

Affinity Propagation (Algorithm 4) is another very popular algorithm based on a graph representation [65]. The main idea is to start from a (often complete) graph of the data and to transmit messages between the data along the edges as a function of the value of these edges (it means the similarity between the objects, here in negative values). These exchanges should help to determine which object are good local representatives and which representatives best model each of the other object. The object exchanges two types of messages along the graph. The "responsibility" $r(i, k)$ sent from i to k represents the quality of k as the representative of i with respect to the other available representatives. The "availability" $a(i, k)$ sent from k to i represents the quality of k as a representative of i with respect to the existence of other objects well represented by k and the absence of a good representative of k. The total quality of the representation of i by k is given by $r(i, k) + a(i, k)$. The values for each of the two types of messages depend on the values of the other type. Thus $r(i, k)$ depends on the similarity between k and i, as well as on the similarity between i and other potential representatives $k\prime$ according to their availability $a(i, k\prime)$:

$$r(i, k) = s(i, k) - \max_{k\prime \neq k} \{a(i, k\prime) + s(i, k\prime)\}$$

Similarly, $a(i, k)$ depends on the self-responsibility of k, $r(k, k)$, which is all the higher as k has no good representatives. It also depends on the quality of the representation of k for

other object $i\prime$, $r(i\prime, k)$:

$$a(i,k) = min\{0, r(k,k) + \sum_{i\prime \notin \{i,k\}} max\{0, r(i\prime, k)\}$$

Algorithm 4 Affinity Propagation

Input: A similarity matrix $S = (s(i,j))_{i,j=1,\ldots,N}$ and a value $s(k,k)$ associated with each object o^k.

Output: A segmentation of the data and a list of cluster representatives.

Construct the graph from S.
for For any pair of neighboring points (i,k) on the graph **do**
    Initialize the availability $a(i,k)$ to 0 and the responsibilities $r(i,k)$ such that:

$$r(i,k) = s(i,k) - \max_{k\prime \neq k}\{s(i,k\prime)\}$$

while Until convergence **do**
    Update availability with a damping factor $\lambda \in [0,1]$:

$$a_t(i,k) = \lambda \cdot a_{t-1}(i,k) + (1-\lambda) \cdot \left[min\{0, r(k,k) + \sum_{i\prime \notin \{i,k\}} max\{0, r(i\prime, k)\}\right]$$

$$a_t(k,k) = \lambda \cdot a_{t-1}(k,k) + (1-\lambda) \cdot \left[\sum_{i\prime \notin \{i,k\}} max\{0, r(i\prime, k)\}\right]$$

    Update responsibilities according to λ:

$$r(i,k) = \lambda \cdot r_{t-1}(i,k) + (1-\lambda) \cdot \left[s(i,k) - \max_{k\prime \neq k}\{a(i,k\prime) + s(i,k\prime)\}\right]$$

$$r(k,k) = \lambda \cdot r_{t-1}(k,k) + (1-\lambda) \cdot \left[s(k,k) - \max_{k\prime \neq k}\{a(k,k\prime) + s(k,k\prime)\}\right]$$

for each data i **do**
    find the best representative k, the one that maximizes $a(i,k) + r(i,k)$.
    if a data k^* is its own best representative **then**
        it defined a cluster.
    Associate each data i with a representative defining a cluster by maximizing $a(i,k^*) + r(i,k^*)$, so as to obtain a segmentation of the data.

Only the positive portion of $r(i\prime, k)$ is preserved in such a way that the existence of objects weakly represented by k does not penalize the fact that k represents strongly a certain number of local objects. $a(i,k)$ always remains negative to limit the influence of strong positive responsibilities. The values $s(k,k)$ correspond to an a *priori* information of the potentiality of each given to be a good local representative. These values influence the

total number of clusters obtained, the smaller they are overall, the smaller the number of clusters, and *vice versa*.

The main advantage of Affinity Propagation is that it is not necessary to define a *priori* the number of clusters that one wishes to obtain. This parameter is obtained automatically during the process as a function of the values $s(k, k)$ initially associated with the data. However, the main disadvantage of this algorithm is its slowness. Indeed, the number of messages transmitted is proportional to the number of edges of the graph, or at worst a complexity in N^2. Such complexity limits analysis to small data-sets and is not suited to large databases.

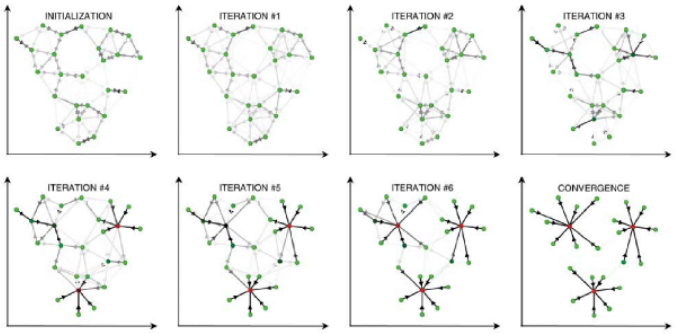


Figure 2.2: Example of an execution of the Affinity Propagation algorithm [65]. The more a point is red, the more it is candidate to be a representative. The color of the edges is proportional to the strength of the messages transmitted.

2.2.4 Density-Based and Probabilistic Approaches

Another family of clustering approaches is no longer based on the distances between objects but on the density fluctuations of the data representation space. There are many clustering approaches based on density [62, 202, 190, 19, 151, 153]. The general idea is that the groups to be discovered consist of a set of points of high density which forms the center of the group, surrounded by points of lower density which form the periphery. From this point of view, areas with low local densities (in relation to neighboring areas) define the boundaries between the groups.

2.2.4.1 DBSCAN Algorithm

DBSCAN [62, 202] (Algorithm 5) is a clustering algorithm based on density. It does not rely on a set of prototypes and applies directly to the data. It only needs the distance between objects. Two parameters Eps and $MinPt$ are defined. Eps is the radius which will be used to calculate the density around a point and $MinPt$ is the minimum number of points located in a radius of Eps around a data so that this data is considered to be part

of the center of a group. A point P is directly reachable from a point Q if there are more than $MinPt$ points in a radius of Eps around P and Q is part of these points. A point P_1 is reachable from P_n if there exists a sequence of points $P_2, ..., P_{n-1}$ such that for all i, P_i is directly reachable from P_{i+1}. If two points are reachable from the same data, they belong to the same cluster (see Figure 2.3).

Figure 2.3: The DBSCAN approach: (a) p is reachable from q. (b) p and q belong to Same group.

Algorithm 5 DBSCAN

Input: I, $MinPt$, Eps
Output: Data segmentation.

1: **Initialization:** Let I be the set of objects.
2: Choose an object o^i at random from I.
3: **if** There is more than $MinPt$ data within a radius of Eps around o^i **then**.
4: create a New cluster C_i, otherwise remove o^i from I and go to step 3.
5: Find all available data from o^i and assign this data to the C_i cluster.
6: Remove from I the data assigned to C_i.
7: **while** I is empty **do**
8: Repeat step 2.

This algorithm shows good performance on two dimensional databases with noise. On the other hand, there is no reduction in dimensions, which causes visualization problems in dimensions greater than three. Moreover, the choice of parameters is difficult without prior knowledge of the data structure, especially when there are large variations in density between groups (see [203] et [153]).

2.2.4.2 Extensions of DBSCAN Algorithm

Various extensions to the DBSCAN algorithm have been proposed, including methods for parallelization, parameter estimation, and support for uncertain data. The basic idea has been extended to hierarchical clustering by the OPTICS algorithm. DBSCAN is also used as part of subspace clustering algorithms like PreDeCon [29] and SUBCLU. OPTICS [11] can be seen as a generalization of DBSCAN that replaces the Eps parameter with

a maximum value that mostly affects performance. $MinPts$ then essentially becomes the minimum cluster size to find. While the algorithm is much easier to parameterize than DBSCAN, the results are a bit more difficult to use, as it will usually produce a hierarchical clustering instead of the simple data partitioning that DBSCAN produces.

Generalized DBSCAN (GDBSCAN) [171, 170] is a generalization by the same authors to arbitrary "neighborhood" and "dense" predicates. The Eps and $Minpts$ parameters are removed from the original algorithm and moved to the predicates. For example, on polygon data, the "neighborhood" could be any intersecting polygon, whereas the density predicate uses the polygon areas instead of just the object count.

Recently, one of the original authors of DBSCAN has revisited DBSCAN and OPTICS, and published a refined version of hierarchical DBSCAN (HDBSCAN)[37] which no longer has the notion of border points. HDBSCAN or Hierarchical Density-Based Spatial Clustering of Applications with Noise [37] [38] is a hierarchical version of DBSCAN which is also faster than OPTICS, from which a flat partition consisting of the most prominent clusters can be extracted from the hierarchy. HDBSCAN has theoretically and practically improved density-based and hierarchical clustering approaches, providing a clustering hierarchy from which a simplified tree of significant clusters can be constructed. For obtaining a "flat" partition consisting of only the most significant clusters (possibly corresponding to different density thresholds), the authors proposed a novel cluster stability measure, formalize the problem of maximizing the overall stability of selected clusters, and formulate an algorithm that computes an optimal solution to this problem. The approach computes DBSCAN over varying epsilon values and integrates the result to find a clustering that gives the best stability over epsilon. This allows HDBSCAN to find clusters of varying densities (unlike DBSCAN) and be more robust to parameter selection.

Bohm et al. extended the DBSCAN algorithm using a preference distance measure to capture the subspace of each cluster [29]. In this way, PreDeCon works with high-dimensional data. The preference distance is simply a weighted Euclidean distance that ignores attributes having variant values greater than a threshold value. Moreover, definitions in DBSCAN (such as $\epsilon - neighborhood$, core point, density-reachable, etc) are adapted using the preference distance. Similar to DBSCAN, PreDeCon constructs clusters by extending and merging its core points. PreDeCon shows better performance than DBSCAN in high-dimensional data-sets such as biological data-sets.

DENCLUE [93] is a combination of density based and grid-based approaches. It divides the high-dimensional data space into grid cells, each of which contains information of data points within the cell. DENCLUE generalizes the definition of density by using a mathematical influence function, such as a square wave influence function or a Gaussian influence function. The density of an object is defined as the sum of influence functions from surrounding data objects. The local maximum of density function is called density attractor;

arbitrary shape clusters are constructed by locating density attractors and their attracted neighbors.

2.2.5 Summary

This synthetic panorama of the various clustering approaches adapted to relational data shows the variety of algorithms proposed, but also the difficulty of the clustering problem. Indeed, the search for natural clusters in real data-sets, potentially large in size, imposes a number of constraints that the algorithm used must satisfy:

1. Discovery of clusters of arbitrary forms.
2. Low complexity in computing time.
3. Resistance to noise.
4. Automatic selection of parameters, in particular the number of clusters.
5. Visualization or simple interpretation of the results obtained.

The algorithms we presented in this chapter have a high complexity and memory usage for a big data-sets. K-Medoids [109], HBDSCAN [37] and Affinity propagation [65] have usually a time complexity of $\mathcal{O}(N^2)$, with N the number of objects. Hierarchical Ascendant Clustering [133] is known to have a complexity of $\mathcal{O}(N^2 Log(N))$, whereas the spectral clustering approach [174] is in $\mathcal{O}(N^3)$. Some optimization and approximations have been proposed to reduce the complexity of these algorithms (down to $\mathcal{O}(Nlog(N))$ depending on the data structure).

In our case, we are interested in dynamic relational unsupervised algorithms. Several approaches exist for dynamic algorithms, but these algorithms work for a specific type of data: vectors, histogram, graphs, images, etc... One major challenge of this thesis is to develop algorithms able to deal with any kind of dynamic data-sets and data streams.

2.3 Data Stream Clustering

Nowadays, with the advance of technology, many applications generate huge amounts of data streams at very high speed. Examples includes network traffic, web click streams, video surveillance, and sensor networks. Data stream mining has become a hot research topic. Its goal is to extract hidden knowledge/patterns from continuous data streams. Unlike traditional data mining where the data-set is static and can be repeatedly read many times, data stream mining algorithms face many challenges and have to satisfy constraints such as bounded memory, single-pass, real-time response, and concept-drift detection [175].

As most approaches for data stream are only adapted to vector data, we define a data stream DS as a sequence of vectors: $DS = (x^1, x^2, ..., x^i, ...)$, where x^i is the $i-th$ ob-

servation. Each data object x^i has a label $y^i \in C = \{c_1, c_2, ..., c_m\}$ when classifying data stream, and there is no label when clustering data stream. Data streams have intrinsic characteristics, such as possibly infinite volume, chronological order and dynamical changes. For example, Google processes more than 100 million searches daily, each of which is attached with a time stamp; and these searches are changed according to different hot topics at different times.

	Traditional Data Mining	Data Stream Mining
Number of passes	multiple	single
Time	unlimited	real-time
Memory	unlimited	bounded
Number of concepts	one	multiple
Result	accurate	approximate

Figure 2.4: Comparisons between traditional data mining and data stream mining.

Table 2.4 shows comparisons between traditional data mining and data stream mining. Traditional data mining is able to scan data-sets many times; execute with unlimited time and memory; has only one concept; and needs to produce fairly accurate results. On the other hand, data stream mining may produce approximate results and has to satisfy constraints, such as single-pass, real-time response, bounded memory, and concept-drift detection:

1. Single-pass
   Unlike traditional data mining that may read static data-sets repetitively many times, each sample in a data stream is examined at most once and cannot be backtracked.

2. Real-time response
   Many data stream applications such as stock market prediction require real-time response. The amount of time for processing the data and providing decision must be fast.

3. Bounded memory
   The amount of arriving data is extremely large or potentially infinite. As we may only compute and store a small summary of the data streams and possibly throw away the rest of the data; approximate results are acceptable.

4. Concept-drift detection
   In data streams, concept drifts refer to the situation when the discovered patterns (or the underlying data distribution) change over time. A formal definition of concept-drifts was introduced by Kelly et al. [110]. Here, a concept at time instant t is defined as a set of probabilities of the classes and class-conditionals:

$$CD = (P(c_1), P(x^t \mid c_1)); (P(c_2), P(x^t \mid c_2)); ...; (P(c_m), P(x^t \mid c_m)).$$

By monitoring changes in this set of probabilities CD, a data stream model is able to detect and adapt itself according to concept drifts.

A general model of data stream algorithms is illustrated in Figure 2.5. When a data stream comes, a buffer is used to store the most recent data. The stream mining engine reads the buffer to create a synopsis of the data in memory. In order to maintain the synopsis, the system may apply different time-window and computational approaches.

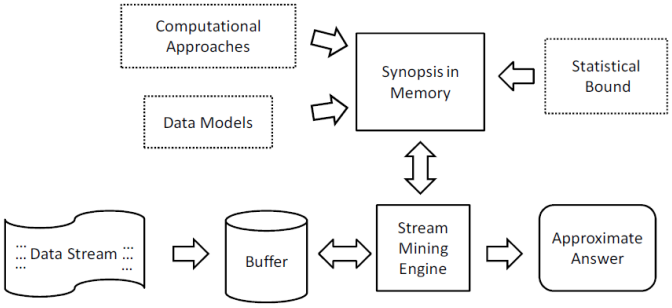


Figure 2.5: A general model for data stream mining.

When certain criteria are triggered; for example, a user's request or after a certain time lapse; the stream mining engine will process the synopsis and output approximate results. In general, most data stream algorithms are derived and adapted from traditional mining algorithms.

2.3.1 Time Window Models

As data streams are potentially infinite, it is possible to only able to process a portion of the entire data streams. This interesting portion is defined as a time-window of data objects. $W_d[i, j] = (x^i, x^{i+1}, ..., x^j)$, where $i < j$. There are different types of time-windows: landmark window, sliding window, fading window and tilted-time window (see Figure 2.6).

1. Landmark window

   In the landmark window, we are interested in the entire data stream from starting time instant 1 to the current time instant t_c; the window is $W_d[1, t_c]$. Using the landmark window, all transactions in the window are equally important; there is no difference between past and present data. However, as data stream evolves continuously, the model built with old data objects may become inconsistent with the new ones. In

order to emphasize recent data, one may apply the sliding window, tilted window or fading window variants.

2. Sliding window
   In the sliding window variant $W_d[t_c - w_d + 1, t_c]$, we are only interested in the w_d most recent transactions; the others are eliminated. The mining result is dependent on the size of the window w_d. If w_d is too large and there is a concept drift, the window possibly contains outdated information and the accuracy of the model decreases. If w_d is small, the window may have deficient data, the model over-fits and suffers from large variances. Previous work considers a fixed value for the size of the sliding window specified by users or an experimental value. Recently, there are proposals for flexible sliding windows where the size of the window changes according to the accuracy of the model [25, 118]. When the accuracy is high, the window extends; and when the accuracy is low, the window shrinks.

3. Fading window
   In the fading window variant, each data object is assigned a different weight according to its arrival time so that new transactions receive higher weights than old ones [39, 44, 85]. Using the fading window, we reduce the effect (importance) of old and outdated transactions on the mining results. A decreasing exponential function $f(\Delta t) = \lambda^{\Delta t} (0 < \lambda < 1)$ is usually used in the fading model. In this function, Δt is the age of a data object that is equal to time difference between the current time and its arrival time. The fading window needs to choose a suitable fading parameter λ, which is typically set in the range [0.99,1] in real applications.

4. Tilted-time window
   The tilted-time window variant is somewhere between the fading window and sliding window variants [2, 85]. One is more interested in recent data at fine scale than long-term data from the past at coarse scale. Tilted-time window provides a nice tradeoff between storage requirements and accuracy. It approximately stores the entire data-set and considers all the transactions at the same. However, the model may become unstable after running for a long time. For example, the tree structure in FP-Stream [72] will become very large over time, and the process of updating and scanning over the tree may degrade its performance. Similarly, the micro-structures in On-Demand Classification [4] will become larger and larger that may give rise to the problem of low-purity clustering with large micro-clusters [205]. For fading window, λ is set to 0.9; the weights of data objects decrease. For tilted-time window, we store the 4 most recent quarters of an hour, then the last 24 hours, and last 31 days.

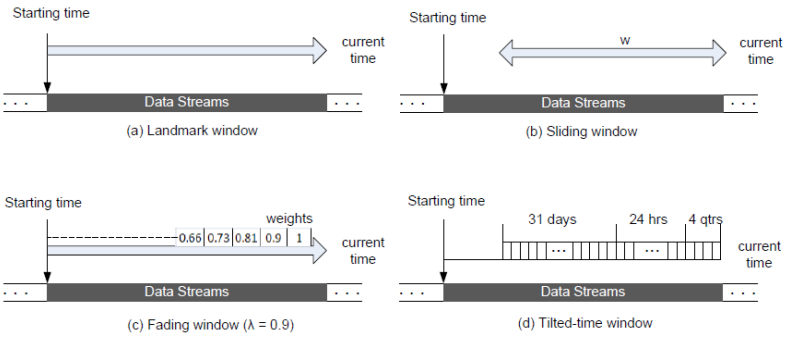


Figure 2.6: Examples of time-windows.

2.3.2 Computational Strategies

Besides various time-window variants, there are computational approaches to process the data streams.

1. Incremental Learning

   Incremental learning is a popular computational strategy for data streams [53, 78, 79, 96, 121, 127, 89, 177, 181]. In this approach, the model incrementally evolves to adapt to changes in incoming data. There are two ways to update the model: window and data instance. Street et al. deployed an ensemble of classifiers for data stream [181]. It evaluated a window of incoming data and adapted the model by adjusting the weight of each classifier or replacing an old classifier with an updated one. The incremental approach has the advantage of providing mining results instantly, but it requires much computational resources.

2. Two-phase Learning

   Two-phase learning, also known as online-offline learning, is a common computational strategy in data streams [2, 4, 39, 44, 114, 154, 155, 158, 172, 195]. The basic idea is to divide the mining process into two phases. In the first phase (online phase), a synthetic model of the data is updated in a real-time manner. In the second phase (offline phase), the mining process is performed on the stored model whenever a user sends a request. The two-phase learning approach is able to process data streams at very high speed. However, its limitation is that users must wait until the mining results are available.

2.3.3 Clustering Algorithms

As mentioned in the general model (Figure 2.5), data stream algorithms typically maintain synopses of the data using different time-window and computational approaches, the clustering process being applied on these synopses. Data stream clustering algorithms gen-

erally extend traditional algorithms to work for data streams with the goal to satisfy constraints, such as bounded memory, single-pass, real-time processing, and concept drifts. Table 2.1 summarizes the main capabilities of the state-of-the-art approaches described in this section.

Table 2.1: Capabilities of data stream clustering algorithms.

Algorithm	**STREAM[80]**	**CluStream[2]**	**HPStream[3]**	**SWClustering[205]**	**E-Stream[189]**	**RepStream[127]**	**OpticsStream[]**	**Den-Stream[39]**	**incPreDeCon[115]**	**D-Stream[44]**	**MR-Stream[195]**	**Cell-Tree[154]**	**Cell*Tree[154]**	**XWAVE[78]**	**SWEM[53]**	**GCPSOM[177]**
Bounded Memory		✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Single-Pass	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Real-Time Response	✓	✓	✓	✓	✓		✓	✓		✓	✓		✓	✓	✓	
Concept-Drift Adaptation		✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Concept-Drift Classification					✓											
High-Dimensional Data			✓						✓							

2.3.3.1 Partitioning Approaches

1. STREAM [80]: STREAM is one of the first data stream algorithms. It addresses the bounded memory and single-pass constraints. It uses a divide-and-conquer strategy to perform clustering incrementally and hierarchically. The STREAM algorithm works in a window mode. When an amount of data objects fits into main memory, it is clustered using LSEARCH, an advanced k-medoids algorithm. Intermediate medians with its weight representing result clusters are stored. The process is repeated for subsequent windows. The LSEARCH is recursively applied to the representative medians. Together with a sampling method, STREAM is able to perform clustering with limited time and memory. However, it fails to detect concept drifts and discover non-spherical clusters and is sensitive to parameter k due to the intrinsic properties of k-medoids.

2.3.3.2 Hierarchical Approaches

1. CluStream [2]:
   CluStream extends the traditional clustering approach BIRCH [204] for data streams. CluStream uses micro-clusters to store the summary of data streams. A micro-cluster is an extension of the Clustering Feature in BIRCH with two more dimensions: time and square of time. It applies the tilted time window to optimize the number of stored snapshots (the status of micro-clusters in the data stream). CluStream follows

the online-offline approach, which is similar to the multi-phase clustering technique in BIRCH. In the online phase, it continually maintains a set of q micro-clusters in the data stream. In the offline phase, it performs k-means to cluster the stored q micro clusters. CluStream analyzes the evolution of clusters by using additional property to extract information of micro-clusters during a specific time range.

Based on CluStream's framework, many extensions have been proposed. HPStream [3] addresses the problem of high dimensional data streams by deploying a projection technique to select the best attribute set for each cluster (subspace clustering). Similar to CluStream, HPStream maintains micro cluster structures as synopses of data streams. Furthermore, each micro-cluster consists of a set of relevant attributes, which can be considered its subspace. When a new data instance arrives, the average Manhattan distance between the new instance and each cluster is computed. Only relevant attributes of the clusters are utilized in the distance computation. Then, the new instance is assigned to the closest cluster if their distance does not exceed a limiting range, a multiple of the cluster's radius. Moreover, the statistical properties of the closest cluster are also updated. HPStream only maintains a fix number of micro-clusters. When the number of clusters reaches a threshold value, it removes the oldest cluster to give space for a new one.

SWClustering [205] identifies a problem that the clustering results of CluStream may degrade after running for a long time. For example, when the center of a micro-cluster gradually shifts, CluStream maintains the micro-cluster with growing radius, instead of splitting it into many micro-clusters. To summarize data streams, SWClustering creates a Temporal Cluster Feature (TCF) for a sliding window. The TCF is similar to a micro-cluster; the only difference is that TCF stores the latest timestamp, while micro-cluster stores the sum of timestamps. An Exponential Histogram of Cluster Features (EHCF), a collection of TCFs, is used to capture the evolution of individual clusters. SWClustering not only produces more qualified clustering due to the fine granularity of EHCFs, it also has better performance than CluStream in terms of running time and memory usage. E-Stream [189] classifies cluster evolution into five categories: appearance, disappearance, self-evolution, merging, and splitting. It utilizes the fading model and cluster histograms to identify the type of cluster evolution.

2. REPSTREAM [127]:
   Inspired by CHAMELEON [107], REPSTREAM is another graph-based hierarchical clustering approach for data streams. To identify clusters, REPSTREAM updates two sparse graphs that are formed by connecting each vertex to its k-nearest vertices. The first graph captures the connectivity relationship among coming data points and is used to select a set of representative vertices. The second graph of representative vertices helps to make clustering decisions at a higher level. REPSTREAM applies

the fading window to diminish the effect of old data. REPSTREAM keeps track of the connectivity between the representative vertices and performs merging or splitting according to their connectivity.

2.3.3.3 Density-based Approaches

1. DenStream [39]:
   DenStream is a density-based stream clustering algorithm that extends the DBSCAN algorithm. Similar to CluStream, DenStream uses micro-clusters to capture synopsis information of data streams; its online component continually updates the micro-clusters collection. Each micro-cluster has a center and a radius that are derived from its clustering feature vector. DenStream applies the fading model where elements of its feature vector decrease over time. Given threshold values for the weight and radius, there are three types of micro clusters: a core micro cluster, a potential core micro cluster, and an outlier micro cluster. For the offline components, it applies DBSCAN on these kinds of micro clusters; a cluster is created as a group of micro clusters that are dense and close to another.

2. OPTICS-Stream [184]:
   OPTICS-Stream is an extension of the OPTICS algorithm for data streams. Similar to DenStream, it uses micro-clusters and the fading model to construct the synopsis. The offline component performs clustering by using the definitions of core-distance and reachability distance in OPTICS [11]. The reachability plot, which becomes a 3-D plot given the time dimension, can be used to visualize the changes of the cluster structures in the data stream over time.

3. incPreDecon [115]:
   incPreDecon is an incremental version of the PreDeCon [29] algorithm, which is designed to work for dynamic data. The algorithm supports two types of updates, a single instance update and batch update. Both updating methods share the same strategy. They first identify a group of affected objects, whose properties may be changed into one of the following cases: (i) core $\rightarrow$ non-core, (ii) non-core $\rightarrow$ core, and (iii) core $\rightarrow$ core but under different preferences. Given new arriving data p, the group of affected objects consists of reachable objects from p. Properties of these objects, as well as their clusters, are updated according. The ability of incPreDecon to work for data streams is still unclear as only experiments with relatively small data-sets have been performed.

4. D-Stream [44]:
   D-Stream is a density-based clustering approach for data streams. It can be considered as an extension of the DENCLUE algorithm [92]. In D-Stream, each dimension is

divided into p segments. With this, a grid of equal size hyper-rectangular cells is created. Similar to a microcluster, a grid cell in D-Stream is used to store synopsis information of data objects falling into it. As D-Stream uses the fading model to decrease the weight of cell over time, it periodically removes sparse grid cells to save memory and accelerate the mining processes. D-Stream performs clustering upon a user request. A cluster is defined as a group of adjacent dense grid cells.

2.3.3.4 Grid-based Approaches

1. MR-Stream [195]:
   MR-Stream is a multi-resolution density-based clustering approach for data stream. MR-Stream takes advantages of both D-Stream [44] and STING [196]. MR-Stream proposes a tree of grid cells to capture the hierarchical structure of the data space. A deeper level tree node has higher granularity level. Similar to D-Stream, MR-Stream applies the fading model and periodically prunes sparse grid cells to save memory. Additionally, MR-Stream significantly reduces the number of tree nodes by merging sibling nodes of a parent node if they are all dense or sparse. Thus, MR-Stream preserves more memory than D-Stream and accelerates the clustering process. Moreover, MR-Stream provides a better cluster result by extending the neighborhood range concept and supports a memory sampling approach that helps users to detect when concept drifts occur.

2. CellTree [154]:
   CellTree is another grid-based algorithm for data streams. It starts by partitioning the data space into a set of mutually exclusive equal size cells. When a weight of a cell is greater than a threshold value, the cell is dynamically divided into two intermediate cells using a hybrid-partition method that selects a better method between *μ-partition* and *σ-partition*. The *μ-partition* divides a dimension with the largest standard deviation, while the *σ-partition* method chooses to split a dimension with the smallest standard deviation. To save memory, CellTree prunes sparse cells with density less than the threshold value. CellTree has been extended to a better version $Cell^*Tree$ [155] that uses a B+Tree to store the synopses of data streams. The hybrid-partition method has a drawback that CellTree is not able to employ any indexing structure to access a specific grid cell immediately. In Cell*Tree, a dense grid cell is divided into a fixed number of equal size grid cells, which are indexed easily based on its order. Moreover, Cell*Tree applies the fading model to emphasize the latest change of information in a data stream on the clusters.

2.3.3.5 Model-based Approaches

1. SWEM [53]:
   SWEM is an EM-based clustering algorithm for data streams using a sliding window. In SWEM, each micro-component is represented by a tuple consisting of a weight, a mean, and a covariance matrix. For the first data window, SWEM applies the EM algorithm to obtain the converged parameters. Then, in the incremental phase, SWEM utilizes the converged parameters in the previous window of data object as the initial values for the mixture models' parameters. If the two sets of parameters are significantly different; SWEM re distributes components in the entire data space by splitting those micro-components with large variance and merging neighbor micro-components. SWEM also deploys the fading model to expire the statistic summarization of the micro components. In short, SWEM may be considered as an EM clustering using Mahalanobis distance with fading window.

2. GCPSOM [177]:
   There are two important extensions of SOM: Growing self-organizing map (GSOM)[8] and cellular probabilistic Self-Organizing Map (CPSOM) [46]. In GSOM, there is no need to pre-specify the size of the output map; it dynamically grows nodes at the boundary of the map whenever its accumulated error exceeds a threshold.
   CPSOM is an online algorithm and is suitable for large datasets. CPSOM uses a fading window to reduce the weight of the neuron state. Thus, CPSOM may forget old patterns and adapt to new patterns as they appear. GCPSOM is a hybrid algorithm that aggregates the advantages of both the GSOM and CPSOM. Therefore, GCPSOM dynamically grows the feature map for clustering data streams and keeps track clusters as they evolve.

2.3.4 Summary

To summarize the above ideas and give a coherent view of data stream clustering approaches, we show in Figure 2.7 an intuitive diagram to outline the relationship between traditional clustering algorithms and data stream clustering algorithms [175]. Traditional clustering approaches are shown on the top of the diagram, and data stream clustering methods are on the bottom. In the middle, the two computational strategies and the four time-windows categories are associated to each algorithm.

Based on this diagram, we observe that most data stream clustering approaches are adapted from traditional clustering algorithms but apply different computational strategies and time windows. For example, STREAM extends the k-means algorithms with incremental computational strategy and landmark window. The key idea of STREAM is to apply the LSEARCH technique to perform K-means incrementally and hierarchically. CluStream ex-

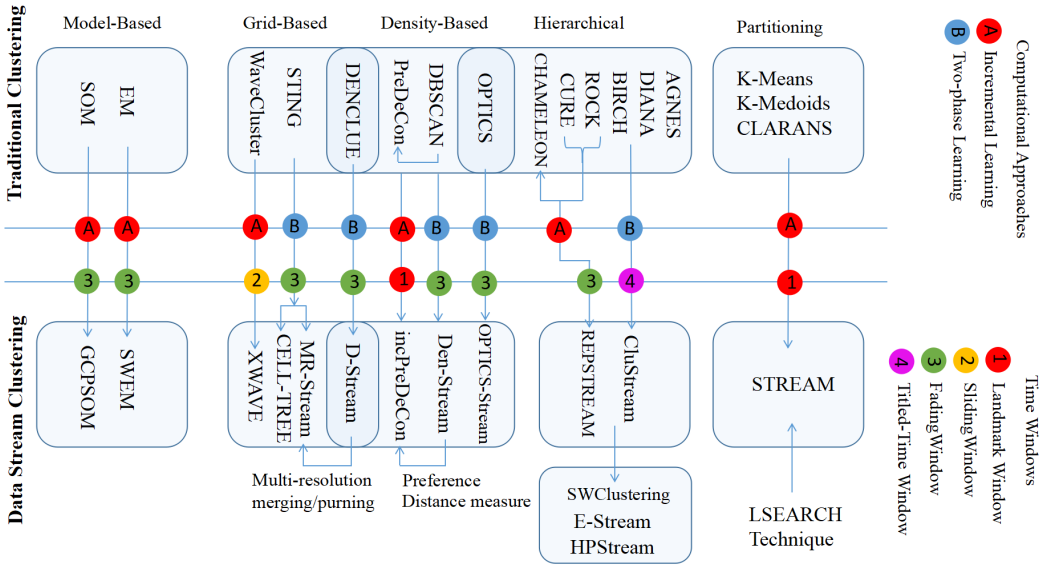


Figure 2.7: The relationships between traditional clustering methods and stream clustering methods.

tends the BIRCH algorithm and applies the two-phase learning strategy and tilted-time window. HPStream improves CluStream to work for high dimensional data streams; SWClustering enhances CluStream on long-term running, and E-Stream extends CluStream to classify different types of concept drifts. REPSTREAM extends CHAMELEON algorithm with the incremental learning strategy and fading window. Similarly, DenStream is an extension of DB-SCAN with the two-phase learning strategy and fading window. incPreDeCon combines it with the preference distance measure to work with data streams. D-Stream inherits from DENCLUE; it has been extended into a multi-resolution approach with merging and splitting operations in MRStream. Moreover, CELL-TREE, XWAVE, SWEM, and GCPSOM are extensions of STING, WaveCluster, EM, and SOM respectively.

The advantages and limitations of the different categories of algorithm are summarized in Table 2.2.

Table 2.2: Advantages and limitations of clustering approaches.

Algorithm	**Advantages**	**Limitations**
Partitioning	Simple and relatively efficient. Terminate at a local optimum.	Need to specify the number of clusters. Unable to discover non-spherical clusters.
Hierarchical	Derive more meaningful cluster structures.	High complexity. Sensitive to the order of the data records.
Density-based	Can find arbitrary-shape clusters. Robust to noises.	Need many parameters: density and noise thresholds. Difficult to detect clusters with different densities.
Grid-based	Fast and can discover arbitrary shape clusters. Robust to noises.	The clustering quality depends on the grid granularity. Unsuitable to high-dimensional data.
Model-based	Simple and can include domain knowledge.	Depends strongly on the assumed models.

2.4 Indexes for the Evaluation of the Algorithms' Quality

This section introduces the external and internal quality indexes used during this thesis to evaluate the quality of a data clustering. Evaluating the performance of a clustering algorithm is not as trivial as counting the number of errors or the precision and recall of a supervised classification algorithm. A popular strategy is to evaluate if the obtained clustering defines separations of the data similar to some ground truth set of classes. If the ground truth is unknown, it is common to checks structural assumptions such as: members of the same cluster should be more similar than members of different clusters, according to a similarity index [156]. External indexes provide a measure of similarity between the cluster assignment proposed by the algorithms and the "true" labels, when they are known [59]. Internal indexes are used to measure the goodness of a clustering structure without external information [185].

The description of the chosen quality indexes are as follow:

- **"Adjusted Rand Index"** [94]:

  The Adjusted Rand Index (ARI) is an agreement between two partitions. It measures the similarity between two data clustering, by considering all pairs of objects and counting pairs that are assigned in the same or different clusters in the predicted and true clustering. The adjusted Rand index have a value close to 0.0 for random labeling and exactly 1.0 when the clusters are identical. It is the corrected-for-chance version of the Rand Index and can be computed as follow:

$$ARI(h_a, h_b) = \frac{\sum_{i=1}^{k_a}\sum_{j=1}^{k_b}\binom{N_{ij}}{2} - t_3}{1/2(t_1 + t_2) - t_3} \tag{2.2}$$

  where $t_1 = \sum_{i=1}^{k_a}\binom{N_{ia}}{2}$, $t_2 = \sum_{j=1}^{k_b}\binom{N_{bj}}{2}$, $t_3 = \frac{2t_1t_2}{N(N-1)}$, and $h_a = \{C_1^a, C_2^a, ..., C_{k_a}^a\}$ and $h_b = \{C_1^b, C_2^b, ..., C_{k_b}^b\}$ with k_a and k_b clusters, respectively, are both clustering on data-set D with N samples; N_{ij} signifies the number of common objects in cluster c_i in clustering h_a and in cluster c_j in clustering h_b; N_{ia} denotes the number of objects in cluster c_i in clustering h_a; and N_{bj} stands for the number of cluster objects in clusters in cluster c_j in clustering h_b.

- **"Normalized Mutual Information"** [182]:

  NMI is an external clustering quality index, that quantify the mutual information between the obtained clustering and the labels associated to each object. It is a measure of the similarity between the clustering C and the ground truth L. The values are normalized to scale between 0 (no mutual information) and 1:

$$NMI(C, L) = \frac{I(C, L)}{\sqrt{H(C) \cdot H(L)}},$$

where $I(C, L)$ is the mutual information between the k clusters and the r labels:

$$I(X, Y) = \sum_{i=1}^{k} \sum_{j=1}^{r} P(C_i \cap L_j) \log \frac{P(Ci \cap Lj)}{P(C_i)P(L_j)};$$

where $P(C_i)$ is the probability for an object to belong to cluster i, $P(L_j)$ is the probability for an object to be associated to label j and $P(C_i \cap L_j)$ is the probability for an object to both belong to cluster i and be associated to label j. $H(C)$ and $H(L)$, respectively, are the entropy of C and L:

$$H(C) = -\sum_{i=1}^{k} P(C_i) \log(P(C_i)) \text{ and } H(L) = -\sum_{j=1}^{r} P(L_j) \log(P(L_j)).$$

- **"Adjusted Mutual Information"** [182]:

  Adjusted Mutual Information (AMI) is an adjustment of the Mutual Information (MI) score to account for chance. It accounts for the fact that the MI is generally higher for two clustering with a larger number of clusters, regardless of whether there is actually more information shared. For two clustering U and V, the AMI is given as:

$$AMI(U, V) = \frac{[MI(U, V) - E(MI(U, V))]}{[max(H(U), H(V)) - E(MI(U, V))]} \tag{2.3}$$

  Where $H(U)$ is the entropy associated with the partitioning U and $H(V)$ is the entropy associated with the partitioning V and E is the expected mutual information between two clustering U and V if we applied random permutation to the labels.

- **"V-measure"** [166]:

  The V-measure is the harmonic mean between homogeneity and completeness:

$$v = 2 * \frac{homogeneity * completeness}{homogeneity + completeness} \tag{2.4}$$

  A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class and a clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster.

- **"Fowlkes-Mallows Index"** [64]:

  The Fowlkes-Mallows Index (FMI) is defined as the geometric mean between precision and recall. It can be expressed as:

$$FMI = \frac{TP}{\sqrt{(TP + FP) * (TP + FN)}} \tag{2.5}$$

Where TP is the number of True Positive (i.e. the number of pairs of points that belong to the same clusters in both "true" and proposed clustering), FP is the number of False Positive (i.e. the number of pairs of points that belong to the same clusters in the "true" clustering, but not in the proposed one) and FN is the number of False Negative (i.e. the number of pairs of points that belong to the same clusters in the proposed clustering but not in the "true" one). The score ranges from 0 to 1. A high value indicates a good similarity between two clusters.

- **"Silhouette"** [103]:

  Silhouette is an internal method of interpretation and validation of consistency within clusters of data. Silhouette coefficients are measures of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The Silhouette index for an object i is:

  $$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

  Where $a(i)$ is the average dissimilarity between i and the other objects in the same cluster, $b(i)$ is the average dissimilarity between i and the objects in the nearest neighbor cluster. The value of $S(i)$ ranges over the interval $[-1, 1]$. The value closer to 1 demonstrates that the object i is clustered more appropriately within the partition. The average value over all objects in the data-set is a measure of how appropriately the data have been clustered.

2.5 Conclusion

Clustering algorithms for relation data do not need coordinates for the objects and work on the dissimilarity between the objects. The input for relational clustering algorithms is therefore a similarity or a dissimilarity matrix. The complexity in terms of processing time and memory cost is expensive (at least $\mathcal{O}(N^2)$). However, in data stream clustering, because of the speed of new data arriving over the time and the impossibility to store the data, algorithms need to be fast and memory efficient. So, a similarity matrix for input in size of $(N \times N)$, where N is the size of the data-set, is not possible. For these reasons, we introduce in this thesis new approaches of clustering for relational data with a much lower complexity and memory need, which can be used for big data-sets and data stream clustering.

Chapter 3

Presentation of the Experimental Data-sets

Table of Contents

3.1 Introduction

In this chapter, we present the data-sets we used to validate the algorithms proposed in the following chapters. In addition, for each type of data, we give some examples of adapted similarity measures [75]. Most of our approaches have been tested and compared on the following real and artificial data-sets of various types. Table 3.1 presents a summary of the experimental data.

Table 3.1: Data-sets descriptions.

Data-set	# Objects	Type	# Classes
Static_Gauss_2	10000	Vector (Gaussian)	6
Static_Gauss_10	10000	Vector (Gaussian)	9
Art1	1000	Vector (Gaussian)	4
Art2	1000	Vector (Gaussian)	4
Artificial convex	8750	Vector (Convex)	5
Art3	800	Vector (Convex)	5
Art4	1000	Vector (Non Convex)	8
Artificial non-convex	10000	Vector (Non Convex)	5
Static_Noconv_2	10000	Vector (Non Convex)	7
Dyn_Noconv_2	10000	Vector (Non Convex, Dynamic)	3
Dyn_Gauss_2	10000	Vector (Non Convex, Dynamic)	4
Dyn_Gauss_10	10000	Vector (Non Convex, Dynamic)	9
Iris	150	Vector (real)	3
Glass	214	Vector (real)	6
Digits	1797	Vector (real)	10
Wine	178	Vector (real)	3
Prot1	115	Sequence	3
Prot2	64	Sequence	2
Prot3	50	Sequence	3
Prot4	129	Sequence	5
Prot5	98	Sequence	3
Prot6	78	Sequence	4
Hist-mean	1000	Distribution	5
Hist-shape	1000	Distribution	5
Hist-std	1000	Distribution	5
People	300	Concept	3

3.2 Experimental Data-sets and Dissimilarity Measures for each Type of Data

Dissimilarities or distances between two objects plays an important role in many data mining tasks like classification and clustering which involves distance computation. A distance d is a metric if it satisfies the following properties:

1. Limited Range: $d(o_1, o_2) \leq C$, for some arbitrarily large number C.
2. Identity: $d(o_1, o_2) = 0$ if and only if $o_1 = o_2$.
3. Symmetry: $d(o_1, o_2) = d(o_2, o_1)$.
4. Triangle Inequality: $d(o_1, o_2) + d(o_2, o_3) \geq d(o_1, o_3)$.

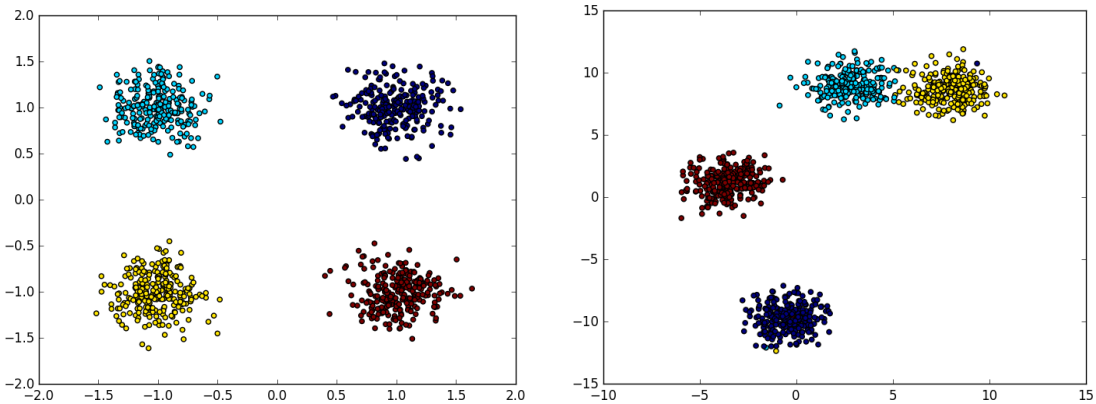
A typical clustering algorithm uses a dissimilarity measure for comparing the objects. This measure must be defined for each data-set based on the type of data to analyze. The challenge is to determine which similarity measure is suitable for which data type. Various similarity/dissimilarity measures have been formulated throughout the years, each with its own strengths and weaknesses. Some measures use raw data, some normalize the numerical values before using them, some use the ranks of these values, and some use joint probabilities of corresponding values.

3.2.1 Vectorial Data

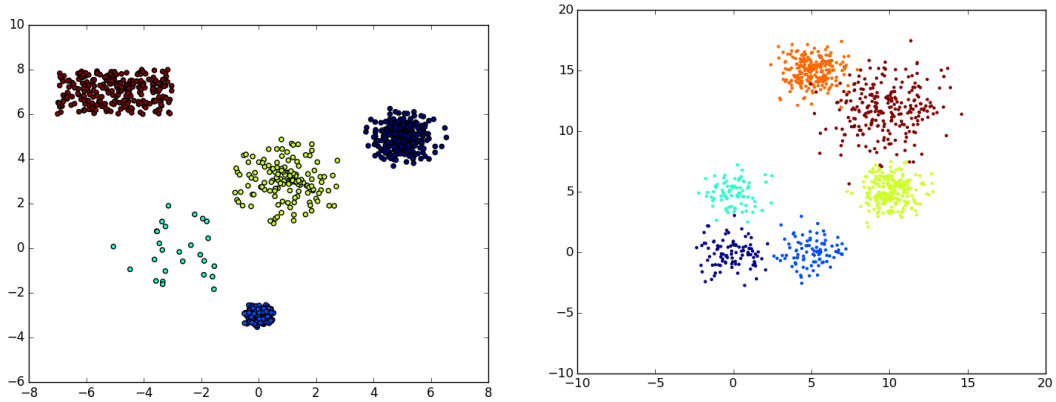
We tested the algorithms on a set of vectorial data-sets, either real or simulated [163, 124]. Some of these data-sets were artificially generated with a different number of clusters of various shapes and densities. Here, each object is defined as a vector and the similarity matrix is obtained using the *Euclidean distance*. The data-set generation and the computation of the similarity matrix were made using the "scikitlearn" [156] python libraries.

3.2.1.1 Description of the Data-sets

- Artificial convex data-sets:
  We generated a series of data-sets of various sizes, dimensions and number of cluster: "Static_Gauss_2", "Static_Gauss_10", "ART1", "ART2", "ART3" and "Artificial convex". The clusters in these data-sets are all convex, usually following a Gaussian distribution (see Figure 3.1). Table 3.1 presents a description of the size and number of cluster in each data-set.



(a) ART1 (b) ART2

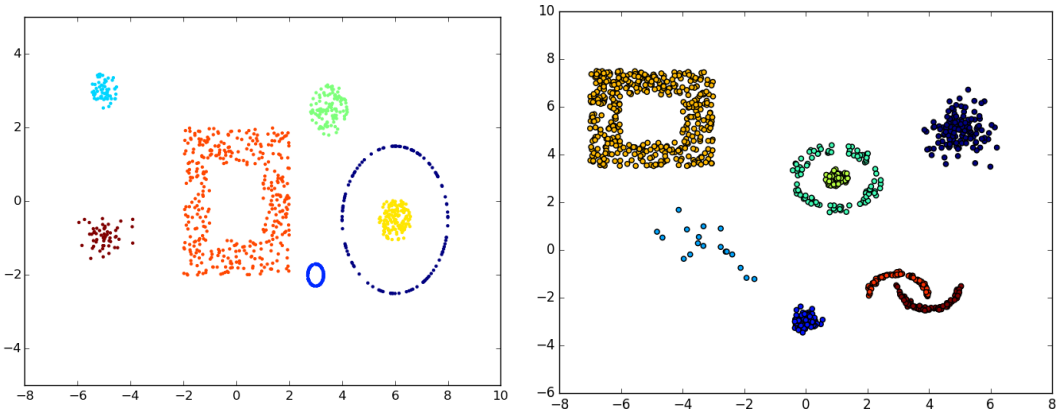


(c) ART3 (d) Static_Gauss_2

Figure 3.1: Visualization of the artificial convex data-sets (two first dimensions). Different colors are used to represent the different clusters. ART1 have well separated clusters, whereas ART2 have overlapping clusters. ART 3 have well separated clusters with different size and density. Static_Gauss_2 have overlapping clusters with different size and density.

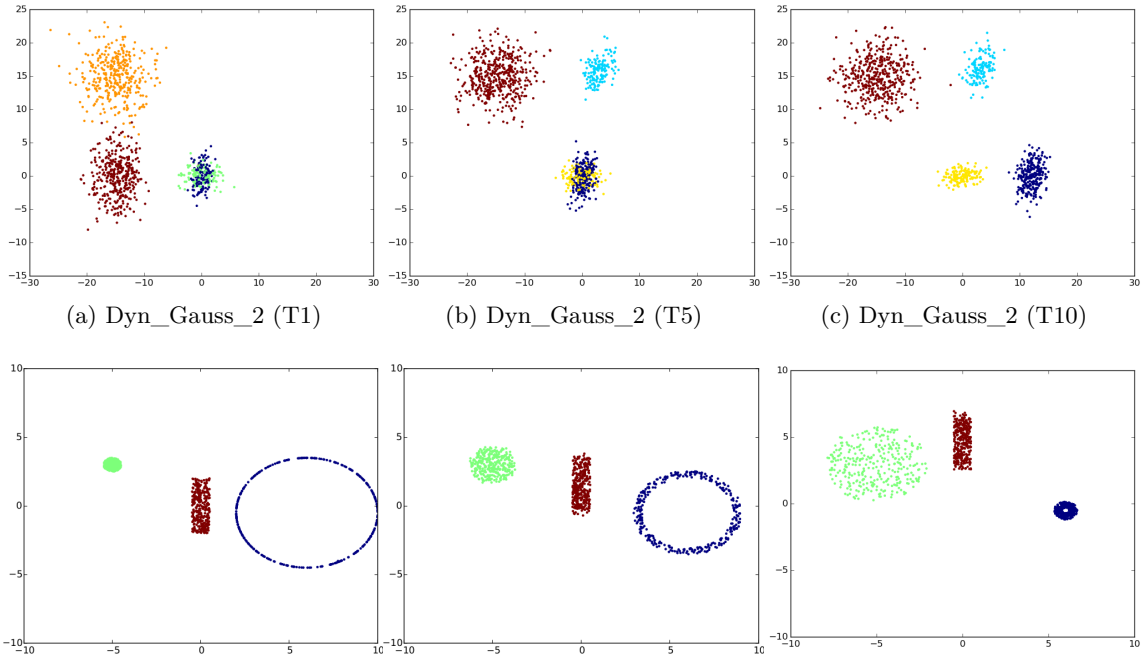
- Artificial non-convex data-sets:
  We also generated three non-convex data-sets of various sizes, dimensions and number of cluster: "ART4", "Static_Noconv_2" and "Artificial non-convex". The clusters in these data-sets have various shapes, convex or non-convex (Figure 3.2). Table 3.1 presents a description of the size and number of cluster in each data-set.

- Artificial dynamic data-sets:
  Finally, we generated three dynamic data-sets: "Dyn_Noconv_2", "Dyn_Gauss_2" and "Dyn_Gauss_10". The structure of these data-sets changes over time, either because the shape or size of the clusters vary or because of new clusters appearing and old clusters disappearing (see Figure 3.3 and Table 3.1).



(a) Static_Noconv_2 (b) ART4

Figure 3.2: Visualization of the artificial non-convex data-sets.



(a) Dyn_Gauss_2 (T1) (b) Dyn_Gauss_2 (T5) (c) Dyn_Gauss_2 (T10)

(d) Dyn_Noconv_2 (T1) (e) Dyn_Noconv_2 (T5) (f) Dyn_Noconv_2 (T10)

Figure 3.3: Visualization of the artificial dynamic data-sets. Clusters can change in shape and position, as well as appearing or disappearing. T1 represents the first 1000 objects, T5 is the 5th set of 1000 objects and T10 presents the 10th objects

- Iris data-set:

  This real data-set contains 3 classes of 50 objects each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are overlapped and are not linearly distinct. The predicted attribute is the class of iris plants (Figure 3.4).

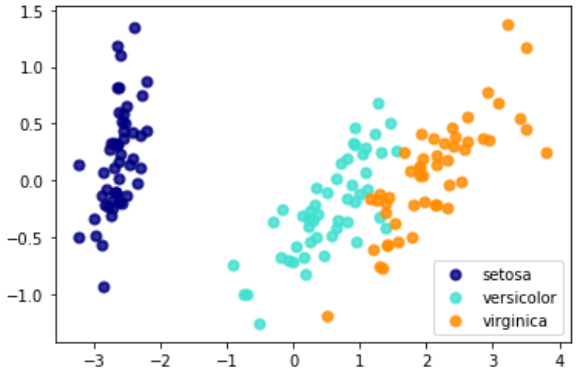


Figure 3.4: Visualization of the real data-set Iris (two first dimensions).

- Glass data-set:
  This real data-set contains 6 classes for a total of 214 objects, where each class correspond to a type of glass. An object is described by 10 attributes [124].

- Digits data-set:
  Each object of this real data-set is a rectangular 8×8 box in a gray scale of 16 values representing of a handwritten digit. There is 1797 objects, with 64 vectorial features, separated into 10 classes.

- Wine data-set:
  These data are the results of a chemical analysis of 178 wines (samples) grown in the same region in Italy but derived from 3 different categories. The attributes describe the quantities of 13 constituents found in each of the three types of wines.

3.2.1.2 Similarity Measures Between Vectors

1. Euclidean distance
   The Euclidean distance or Euclidean metric is the "ordinary" straight-line distance between two points in a Euclidean space. The Euclidean distance between points x and y is the length of the line segment connecting them ($\overline{\mathbf{xy}}$).

   In Cartesian coordinates, if $x = (x_1, x_2, ..., x_d)$ and $y = (y_1, y_2, ..., y_d)$ are two points in d-dimensional Euclidean space, then the distance $d(x, y)$ from x to y, or from x to y is given by the Pythagorean formula:

$$d(x, y) = d(y, x) = \sqrt{(x_1, y_1)^2 + (x_2, y_2)^2 + ... + (x_d, y_d)^2}$$

$$d(x, y) = \sqrt{\sum_{i=1}^{d} (x_i, y_i)^2}$$

2. Jaccard distance

   If $x = (x_1, x_2, \ldots, x_d)$ and $y = (y_1, y_2, \ldots, y_d)$ are two vectors with all real $x_i, y_i \geq 0$, then their Jaccard similarity coefficient is defined as

   $$J(x, y) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)},$$

   and the Jaccard distance can be written as:

   $$d_J(x, y) = 1 - J(x, y).$$

3. Minkowski distance

   The Minkowski distance between two vectors or points x and y, with standard coordinates x_i and y_i, respectively, is

   $$d^p_{\text{Minkowski}}(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p}.$$

4. Chebyshev distance

   The Chebyshev distance between two vectors or points x and y, with standard coordinates x_i and y_i, respectively, is

   $$d_{\text{Chebyshev}}(x, y) = \max_i(|x_i - y_i|).$$

   This equals the limit of the L_p metrics:

   $$d_{\text{Chebyshev}}(x, y) = \lim_{p \to \infty} \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p},$$

   hence it is also known as the L_∞ metric. Mathematically, the Chebyshev distance is a metric induced by the supremum norm or uniform norm. It is an example of an injective metric. The computational complexity of L_p norm is close to that of L_1 norm. After finding the difference of corresponding measures in x and y , L_1 norm finds the absolute of the differences while L_p norm take p power of the differences. Therefore, the absolute-value operation in L_1 norm is replaced with p multiplication in L_p norm.

3.2.2 Sequential Data

The first set of non-vectorial data we used in the experiments is based on sequences of proteins with similar or different functions in several animal species (see [163]).

These data-sets were constructed from a set of protein sequences coding for various proteins as found in several different animal species. All these distance matrices have been acquired by a alignment of several sequences of randomly taken proteins in different organisms. These protein sequences and their alignment is extracted from the protein database Uniprot [50]. The similarity of each pair of sequence is then computed by *edit distance* using scores of mismatch and gap (insertions or deletions) in the two sequences of amino acid [145]. The score of two identical amino acids is 0, it is 1 when there is a mismatch and 2 for a gap. The similarity is the mean value of the scores of all pairs of amino acids or gaps in the aligned sequence. The construction of the matrices for these data-sets have been made using the "biopython" package [41].

3.2.2.1 Description of the Data-sets

- Prot1 data-set:
  This data-set consists of 115 sequences of three proteins, including 32 sequences of the RAS protein as found in 32 animal species, 38 sequences of RAF protein from 38 species and 45 sequences of cytochrome b. The variability of each protein sequence amongst the species is lower than the variability amongst different proteins, we therefore expect a structure of three clusters for this data-set.

- Prot2 data-set:
  This data-set consists of 64 sequences of two proteins, including 31 sequences of the tyrosinase-related protein 1 and 33 sequences of the PKC protein. The variability of each protein sequence amongst the species is lower than the variability amongst different proteins, we therefore expect a structure of two clusters for this data-set.

- Prot3 data-set:
  This data-set consists of 50 sequences of three proteins, including 19 sequences of the kappa casein protein as found in 19 animal species, 17 sequences of ferritin from 17 species and 14 sequences of alpha-lactalbumin. For the same reason as above, we expect a structure of three clusters for this data-set.

- Prot4 data-set:
  This data-set consists of 129 sequences of five proteins, including 34 sequences of the histone $H4$ protein as found in 34 animal species, 16 sequences of erythropoietin, 19 sequences of kappa casein, 16 of somatotropin and 45 sequences of cytochrome c. We expect a structure of five clusters for this data-set.

- Prot5 data-set:
  This data-set consists of 98 sequences of three proteins, including 31 sequences of the tyrosinase-related protein 1, 32 sequences of the PKC protein and 35 sequences of the H4 protein. Again, we expect a structure of three clusters for this data-set.

- Prot6 data-set:
  The last data-set consists in 76 sequences of three proteins, including 31 sequences of PKC, 16 sequences of insulin and 29 sequences of the glucagon protein, divided into two families (20 and 9 sequences respectively). We expect a structure of four clusters for this data-set.

3.2.2.2 Similarity Measures Between Sequences

1. Edit distance
   Edit distance is a way of quantifying how dissimilar two strings (e.g., words or sequence of proteins) are to one another by counting the minimum number of operations required to transform one string into the other. Edit distances find applications in natural language processing, where automatic spelling correction can determine candidate corrections for a misspelled word by selecting words from a dictionary that have a low distance to the word in question. In bioinformatics, it can be used to quantify the similarity of DNA sequences, which can be viewed as strings of the letters A, C, G and T. Given two strings a and b on an alphabet (e.g., the set of ASCII characters, the set of bytes [0..255], etc.), the edit distance $d(a, b)$ is the minimum-weight series of edit operations that transforms a into b. One of the simplest sets of edit operations is that defined by Levenshtein in 1966 [122].

   (a) **Insertion** of a single symbol. If $a = uv$, then inserting the symbol x produces uxv. This can also be denoted $\varepsilon \to x$, using ε to denote the empty string.

   (b) **Deletion** of a single symbol changes uxv to $uv(x \to \varepsilon)$.

   (c) **Substitution** of a single symbol x for a symbol $y \neq x$ changes uxv to $uyv(x \to y)$.

   In information theory, linguistics and computer science, the Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. In Levenshtein's original definition, each of these operations has unit cost (except that substitution of a character by itself has zero cost), so the Levenshtein distance is equal to the minimum number of operations required to transform a to b. Mathematically, the Levenshtein distance between two strings a and b (of length $|a|$ and $|b|$ respectively)

is given by $\text{lev}_{a,b}(|a|,|b|)$ where:

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & \text{if } min(i,j) = 0, \\ \min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \quad \text{Otherwise.} \\ lev_{a,b}(i-1,j-1)+1_{(a_j \neq b_j)} \end{cases} \end{cases} \tag{3.1}$$

where $1_{(a_i \neq b_j)}$ is the indicator function equal to 0 when $a_i = b_j$ and equal to 1 otherwise, and $\text{lev}_{a,b}(i,j)$ is the distance between the first i characters of a and the first j characters of b. Note that the first element in the minimum corresponds to deletion (from a to b), the second to insertion and the third to match or mismatch, depending on whether the respective symbols are the same. Other variants of edit distance are obtained by restricting the set of operations. Longest common subsequence (LCS) distance is edit distance with insertion and deletion as the only two edit operations, both at unit cost [147]. Similarly, by only allowing substitutions (again at unit cost), Hamming distance is obtained; this must be restricted to equal-length strings [147]. Jaro-Winkler distance can be obtained from an edit distance where only transpositions are allowed.

2. Rank Distance

   This measure is defined as the L_1 norm of rank ordered values in two numerical sequences. Given sequences $x = \{x_i : i = 1, ..., n\}$ and $y = \{y_i : i = 1, ..., n\}$, measure x_i is replaced with its rank $R(x_i)$ and measure y_i is replaced with its rank $R(y_i)$. To reduce or eliminate ties among ranks in a sequence, the sequence is smoothed with a Gaussian of a small standard deviation. The rank distance between sequences x and y is defined by:

$$D_r = \frac{1}{n}\sum_{i=1}^{n} |R(x_i) - R(y_i)|. \tag{3.2}$$

   Since $0 \leq |R(x_i) - R(y_i)| \leq n$, D_r will be between 0 and 1. The smaller is the rank distance between two sequences, the less dissimilar the sequences will be. Rank distance works quite well in sequences that are corrupted with impulse noise. In addition, rank distance is insensitive to white noise if noise magnitude is small enough not to change the rank of values in an sequence. Furthermore, rank distance is insensitive to bias and gain differences between values in sequences just like other ordinal measures.

   Rank distance is one of the fastest ordinal measures as it requires only a subtraction and a sign check at each pixel once ranks of the values are determined. The major portion of the computation time is spent on ranking the values in each sequence, which is on the order of $n \log_2 n$ comparisons for a sequence of size n. Therefore, the computational complexity of rank distance is on the order of $n \log_2 n$.

3. Shannon Mutual Information

The Mutual information was introduced by Shannon [173] and later generalized by Gel'fand and Yaglom [71]. The generalized Shannon mutual information is defined by [60, 131]:

$$S_{MI}(x,y) = \sum_{i=0}^{S_0}\sum_{j=0}^{S_0} p_{ij}\log_2\frac{p_{ij}}{p_i p_j}. \tag{3.3}$$

where p_{ij} is the probability that corresponding values in the sequences x and y are i and j, respectively, and shows the value at entry ijth in the joint probability distribution (JPD) of the sequences; p_i is the probability of value i appearing in sequence x and is equal to the sum of entries in the ith column in the JDP of the sequences; and p_j is the probability of value j appearing in sequence y and is equal to the sum of entries in the jth row of the JPD of the sequences. S_0 is the size of the alphabet used to produce the sequences. Equation (3.3) can also be written as follows:

$$S_{MI}(x,y) = \sum_{i=0}^{S_0}\sum_{j=0}^{S_0} p_{ij}\log_2 p_{ij} - \sum_{i=0}^{S_0} p_i\log_2 p_i - \sum_{j=0}^{S_0} p_j\log_2 p_j, \tag{3.4}$$

where

$$p_i = \sum_{j=0}^{S_0} p_{ij} \text{ and } p_j = \sum_{i=0}^{S_0} p_{ij}. \tag{3.5}$$

Therefore, letting

$$E_i = \sum_{j=0}^{S_0} p_j\log_2 p_j,\ E_j = \sum_{i=0}^{S_0} p_i\log_2 p_i \text{ and } E_{ij} = \sum_{i=0}^{S_0}\sum_{j=0}^{S_0} p_{ij}\log_2 p_{ij}, \tag{3.6}$$

we have

$$S_{MI} = E_{ij} - E_i - E_j, \tag{3.7}$$

which defines mutual information as the difference between the sum of Shannon marginal entropies and the joint entropy. Shannon's mutual information is a powerful measure for determining the similarity between multimodality sequences, but it is sensitive to noise. As noise in one or both sequences increases, dispersion in the JDP of the sequences increases, reducing the mutual information between perfectly matching sequences, causing mismatches. When calculating the mutual information of sequences x and y, the implied assumption is that the sequences represent random and independent samples from two distributions. This condition of independency is often violated because x_i and x_{i+1} depend on each other, and y_i and y_{i+1} depend on each other. As a result, calculated mutual information is not accurate and not reflective of the dependency between x and y. Since mutual information between two sequences varies with the content and size of sequences, Studholme et al. [183] pro-

vided a means to normalize mutual information with respect to the size and content of the sequences. This normalization enables effective localization of one sequence with respect to another by sliding one sequence over the other and determining the similarity between their overlap area. Shannon mutual information is one of the most widely used similarity measures in sequence registration.

The computational complexity of Shannon mutual information is proportional to $S_0^2 + n$ because creation of the JPD of two sequences of size n takes on the order of n additions and calculation of E_{ij} takes on the order of S_0^2 multiplications and logarithmic evaluations. Its computational complexity, therefore, is a linear function of n but with larger coefficients than those of the energy of JPD.

3.2.3 Distributional Data

We also generated 3 data-sets constituted by different configurations of histogram data.

3.2.3.1 Description of the Data-sets

Each object is a set of 10 histograms and each data-set includes 1000 objects in 5 clusters defined according to the different distributions parameters of the distribution: mean, variation and shape. The histograms are generated, each one by a thousand random values, using a Gamma distribution with three parameters: the mean value, the standard deviation and a shape parameter, controlling the skewness of the distribution. The three parameters m, s and h of the Gamma distribution are generated by a Normal distribution $\mathcal{N}(\mu, \sigma^2)$ with μ in [0,5] and $\sigma = 0.1$.

The sequence of thousand values is distributed in 10 continuous intervals corresponding to the bins of the histogram. We have chosen to build equi-depth histograms, so the bounds of the intervals correspond to the deciles of the distribution of the values. Each bin has a weight $\pi = 0.1$.

For each data-set, the data are generated for different sub-populations (clusters) by a distribution that differs for one parameter (m, s, h) while the other two parameters have the same Normal distribution in different cluster. We used the L_2 *Wasserstein metric* [116] (also known as Mallow's distance [134]) to compute the distances between histogram data.

We produced 3 data-sets according to the selected parameters:

- Hist_mean data-set:
  The clusters are defined by a difference in the mean m value of the distributions describing the objects.

- Hist_std data-set:
  The clusters are defined by a difference in the standard deviation s of the distributions describing the objects.

- Hist_shape data-set:
  The clusters are defined by a difference in the shape h of the distributions describing the objects.

3.2.3.2 Similarity Measures Between Distributions

1. Wasserstein distance

   The Wasserstein metric is a distance function defined between two probability distributions on a given metric space. Let x^k be an empirical distribution and it is represented by a sequence of continuous and no-overlapped intervals (or bins) I_v^k with associated a weight or relative frequency π_v (for $v = 1, \ldots, h$) such that $\sum_{v=1}^{h} \pi_v = 1$:

   $$x^k = [(I_1^k, \pi_1), \ldots, (I_v^k, \pi_v), \ldots, (I_h^k, \pi_h)] \tag{3.8}$$

   Assuming a uniform density for each I_v, with each histogram is associated an empirical distribution function $F(x)$, namely, a cumulated relative frequency function. Its inverse, denoted by $F^{-1}(t)$, is the quantile function, a piecewise linear function with domain in $[0, 1]$.

   The L_2 Wasserstein metric [116] (also known as Mallow's distance [134]) is defined as follows:

   $$d_W^2(x^k, x^{l)}) = \int_0^1 \left(F_k^{-1}(s) - F_l^{-1}(s)\right)^2 ds. \tag{3.9}$$

   where $F_k^{-1}(s)$ and $F_l^{-1}(s)$ are respectively the quantile functions associated with the x^k and x^l histogram data.

   Since the quantile functions are piecewise linear, the integral is not solved globally but trough a sum of simple integrals defined on the bounds of each pair of corresponding pieces of the quantile functions. That is:

   $$d_W^2(x^k, x^{l)}) = \sum_{i=1}^{h} \int_{q_{v-1}}^{q_v} \left(F_k^{-1}(s) - F_l^{-1}(s)\right)^2 ds. \tag{3.10}$$

   where: $q_v = \sum_{i=1}^{v} \pi_i$ (for $i = 1, \ldots, h$) are the cumulated relative frequencies. In that case, all the histogram data must have been homogenized to have the same values π_v on the corresponding interval I_v^k. In an easier case, we can consider equi-frequency or equi-depth histograms.

   For each I_v bin, we assume that the values are uniformly distributed such that it can be represented as a function of its center and radius as follows: $c + r(2t - 1)$ for $0 \leq t \leq 1$. According to this formulation, Irpino et al. [98] proved that the L_2

Wasserstein distance $d_W^2(x^k, x^l)$ can be rewritten in terms of centers and radii of two histogram data, as follows:

$$d_W^2(x^k, x^l) = \sum_{i=1}^{h} \pi_i \left[\left(c_v^k - c_v^l \right)^2 + \frac{1}{3} \left(r_v^k - r_v^l \right)^2 \right]. \tag{3.11}$$

This definition simplifies the computational aspects and it gives an interesting interpretation of the L_2 Wasserstein distance in terms of weighted sum of the squared Euclidean distances between the centers and between the radii of the corresponding bins of the histograms.

2. Jensen-Shannon divergence
   The Jensen-Shannon divergence [136, 52] between two distributions is based on the Kullback-Leibler divergence, with some notable (and useful) differences, including that it is symmetric and it is always a finite value. The Jensen-Shannon divergence (JS) is a symmetrized and smoothed version of the Kullback-Leibler divergence $D(P \parallel Q)$ between two discrete distributions. It is defined by

$$JS(P \parallel Q) = \frac{1}{2} D(P \parallel M) + \frac{1}{2} D(Q \parallel M)$$

   Where $M = \frac{1}{2}(P + Q)$. For discrete probability distributions P and Q, the Kullback-Leibler divergence from P to Q is defined [130] to be

$$D_{KL}(P \parallel Q) = -\sum_i P(i) \log \frac{Q(i)}{P(i)} = \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$

3.2.4 Textual Data

The last type of data is information about conceptual objects (entities) described by textual documents.

3.2.4.1 Description of the Data-sets

Each entity is described with a bag of words extracted from the corresponding Wikipedia page summary. Then we transformed each bag of words to a vector using the Fast-text algorithm [30], and we computed the distances between each vector of bag of words with the *Cosine distance* [176]. Finally, to obtain a proper distance metric, we used the angular similarity coefficient [74]. We generated one data-set with this protocol.

- People data-set:
  This data-set consists of 300 entities in 3 different categories including the 100 best footballers in the world in 2016, the 100 most famous Jazz musicians and the top 100 actors of cinema of all time [47].

3.2.4.2 Similarity Measures Between Documents

String similarity measures operate on string sequences and character composition. A string metric is a metric that measures similarity or dissimilarity between two text strings for approximate string matching or comparison. There are many different similarity measures for strings which can be Character-Based or Term-based, like: LCS, Jaro [105], N-gram [18], Cosine similarity [186], Jaccard [100] etc. Moreover, there are different similarity measure for Corpus-Based Similarity. Corpus-Based similarity is a semantic similarity measure that determines the similarity between words according to information gained from large corpora. A Corpus is a large collection of written or spoken texts that is used for language research. Some examples for this kind of similarity measure are: Hyperspace Analogue to Language(HAL)[129][128], Latent Semantic Analysis(LSA) [117], Generalized Latent Semantic Analysis(GLSA)[140], Explicit Semantic Analysis(ESA)[68] etc. In follow there are some document similarity measure defined.

1. Jaccard similarity
   Paul Jaccard in 1901 presented Jaccard similarity which is a simple but intuitive measure of similarity between two sets.

   $$J(doc_1, doc_2) = \frac{doc_1 \cap doc_2}{doc_1 \cup doc_2}$$

   For documents we measure it as proportion of number of common words to number of unique words in both documents. In the field of NLP Jaccard similarity can be particularly useful for duplicates detection. text2vec [143] however provides generic efficient realization which can be used in many other applications.

2. Latent Semantic Analysis (LSA)
   Is the most popular technique of Corpus-Based similarity. LSA assumes that words that are close in meaning will occur in similar pieces of text. A matrix containing word counts per paragraph (rows represent unique words and columns represent each paragraph) is constructed from a large piece of text and a mathematical technique which called singular value decomposition (SVD) is used to reduce the number of columns while preserving the similarity structure among rows. Words are then compared by taking the cosine of the angle between the two vectors formed by any two rows.

3. Cosine distance

   Classical approach from computational linguistics is to measure similarity based on the content overlap between documents. For this we will represent documents as bag-of-words, so each document will be a sparse vector. And define measure of overlap as angle between vectors:

$$Similarity(doc_1, doc_2) = \cos(\theta) = \frac{doc_1, doc_2}{|doc_1||doc_2|}$$

   By cosine distance/dissimilarity we assume following:

$$Distance(doc_1, doc_2) = 1 - Similarity(doc_1, doc_2)$$

Another category for document similarity measure is knowledge-based similarity which is one of semantic similarity measures that bases on identifying the degree of similarity between words using information derived from semantic networks [142]. WordNet [144] is the most popular semantic network in the area of measuring the Knowledge-Based similarity between words. WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual, semantic and lexical relations.

3.3 Discussion

In this chapter we introduced the data-sets we used in our experimental validations. The different data-sets belong to different type of data, and for each data type we reviewed some common similarity measures and we pointed out which one is used in our experiences. The purpose of describing the similarity measures in this chapter is to explain that the choice of such similarity is not obvious. Different categories of similarity measures can be used for each type of data, but the accuracy of the results depends on different factors, like the structure of the data-sets, its distribution and the chosen clustering algorithm.

As an illustration, from the data-set shown in Figure 3.5, a similarity matrix is computed by using different similarity functions: Euclidean distance, Cosine distance, Jaccard and Correlation. Then, similarity graphs were constructed based on the 7 nearest neighbors criteria. The similarity graphs generated from the different similarity measures are shown in figure 3.6.

As you can see, different similarity measures gives different graph of similarity. It means that, based on the data structure and the measure of similarity, the results of an analysis can be very different. Some family of similarity measures are suitable for some data type and not suitable for others. On the other hands, some similarity measure can be adapted to different types of data.

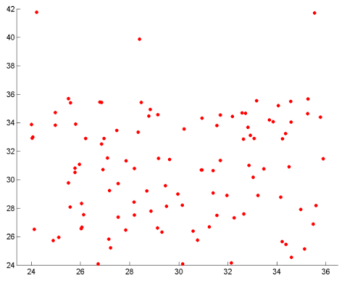
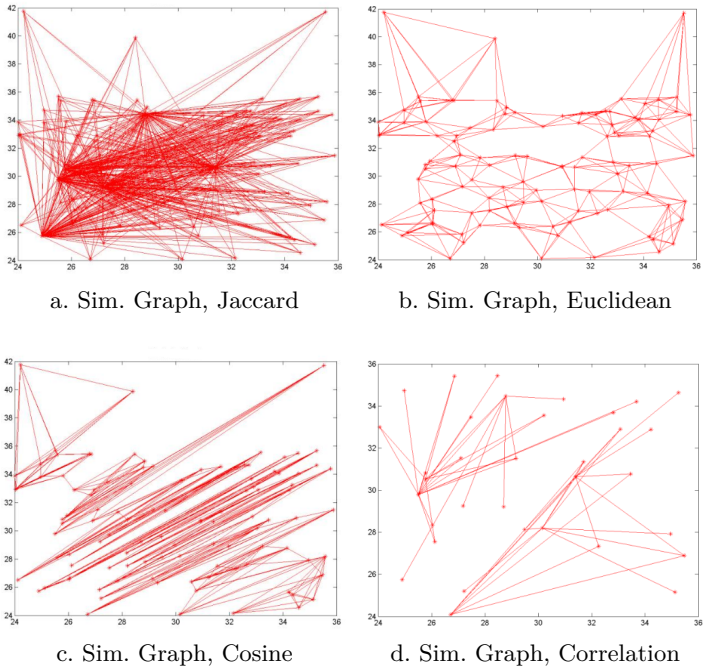


Figure 3.5: Visualization of the generated 2D data-set.

In any case, the choice of a similarity measure is one of the most important step in the clustering process, because it affects directly the outcome. Being able to have unsupervised algorithms working from any type of similarity measures, as these proposed in this thesis, allows a greater flexibility and gives the opportunity of choosing the most adequate measure regarding the data and the context of the analysis.



a. Sim. Graph, Jaccard

b. Sim. Graph, Euclidean

c. Sim. Graph, Cosine

d. Sim. Graph, Correlation

Figure 3.6: Similarity graphs generated by different similarity measures.

Chapter 4

Clustering Approaches Based on Incremental Matrix Reordering

Table of Contents

4.1 Introduction

Although numerous approaches of clustering have been proposed for clusters detection, this task remains a difficult problem. Indeed, most algorithms need the number of clusters to define as a parameter, despite the fact that this number is rarely known. In addition, there is no consensus on the criterion to optimize the definition of the result's quality [111]. For example, some algorithms propose only convex clusters based on a distance criterion, although others propose non-convex clusters based on density criteria. Different approaches will lead to different clustering of the same database, and there is no simple way to define and choose the "best" solution. To address this problem, several research tracks have been proposed, including works on a uniform definition of cluster quality [21, 10] or works on ensemble clustering [182, 187], where several approaches are applied to a data-set and a consensus is computed based on all the results.

However, some authors focus instead on the construction of useful visualization of the data-set's structure. Data visualization is the presentation of data in a graphical format. It is an important step for exploratory data analysis, as it enables analysts to grasp difficult concepts or identify new patterns, and greatly helps the choice of data mining tools to apply [157]. In this way, the user can directly "see" the clusters, their shapes and their relative organization. From an optimal visualization, it is easier to decide the optimal number of clusters and detect sub-clusters. Such visualization gives a lot of information about the data-set and can be used "as it" or be the basis of a choice of parameter values for an automatic analysis, especially for unsupervised analysis (see [43] for a review).

Most clustering algorithms are not adapted to the analysis of relational data, in which objects are defined by their similarities with each other [84], in the form of a similarity or a dissimilarity matrix, for example a distance matrix. It is possible to produce an image of the matrix that reflect the structure of the data. However, the objects in the matrix must be organized in an order than reflect this structure: for the clusters to be seen, objects belonging to the same cluster (i.e. objects that are similar to each other) should be positioned close to one another in the matrix [16]. Several reordering algorithms have been proposed to find the optimal order of objects in a similarity matrix (see [125] for a review).

In this chapter, we wish to propose an incremental reordering algorithm adapted to dynamic relational data-sets. In a dynamic data-set, new objects are added constantly, increasing the size of the data-set without a fixed limit [175]. This type of data-set is increasingly frequent with the development of Internet and other communication networks or with the development of sensor networks. The analysis of a dynamic data stream is challenging because its structure can change over time ("concept drift") and it is not possible to keep all of the information in memory [69]. The algorithm must treat the objects "on the fly", in an unpredictable order. In addition, the objects composing the stream of information in such data-sets are often complex (for example tweets, videos, images, etc.). Here we propose the

first approaches able to create incrementally, from a dynamic relational data-set, a similarity matrix in an ordered way that allows the visualization of the data underlying structure. We propose first a "static" version of the algorithm adapted to non-dynamic data-sets, in order to be able to compare our proposal with the state-of-the-art reordering algorithms. Then, we present a version fully adapted to dynamic data-sets, allowing a visualization of the "current" structure of the data-set and being able to forget "outdated" information in a reasonable computation time and memory consumption.

In the second part of the chapter, we present a new autonomous clustering algorithm for similarity-based data. We propose to use the properties of reordering methods in order to produce a one-dimensional signal of pairwise distances. As objects belonging to the same clusters are neighbors in the reordered matrix, it will be possible to detect "jumps" of distances from one cluster to another, if the two clusters are distant enough. To detect these jumps, we propose to adapt a procedure introduced in [138] for multi-scale denoising of piece wise smooth signals. The main advantage of this procedure is that there is no parameter to tweak in order to obtain a reasonable clustering. The approach adapts automatically to the data structure and finds the correct number of clusters. It is adapted to any shape of clusters or variation of density among the observations. The only requirement is that the clusters must not overlap.

4.2 Incremental Matrix Reordering for Relational Dynamic Data-sets

4.2.1 Related Works

This section presents various optimization criteria and existing reordering algorithms adapted to relational data.

4.2.1.1 Cost Function and Quality

As there is no consensus on the criterion associated to this problem to optimize, different algorithms often optimize different cost functions and potentially produce different results. To illustrate this point, we present here several criteria that are often used as cost functions to optimize, and/or as indexes to evaluate the quality of the obtained reordering:

1. **"Weighted Gradient" [95]:**
   A symmetric dissimilarity matrix D, where the values in all rows and columns only increase when moving away from the main diagonal is called a perfect anti-Robinson matrix [165]. A suitable measure which quantifies the divergence of a reordered matrix

from the anti-Robinson form is the Weighted Gradient:

$$L(D) = \sum_{i=1}^{N}(\sum_{i<k<j} d(o^i, o^j) - d(o^i, o^k) + \sum_{i<k<j} d(o^i, o^j) - d(o^k, o^j))$$

With N number of objects and $d(o^i, o^j)$ distance between object i and object j as expressed in row i and column j in matrix D.

2. **"BAR" Banded Anti-Robinson [61]:**
   BAR is a simplified measure for closeness to the Anti-Robinson form in a band of size b with $1 \leq b < N$ around the diagonal. By using $b = N \times 20\%$, as in [61] we have:

$$L(D) = \sum_{\|i-j\| \leq b} (b + 1 - \|i - j\|) d(o^i, o^j)$$

3. **"Path Length" Hamiltonian path length [40]:**
   The order of the objects in a dissimilarity matrix corresponds to a path through a graph where each node represents an object and is visited exactly once: a Hamilton path. The length of the path is defined as the sum of the edge weights, i.e., dissimilarities.

$$L(D) = \sum_{i=1}^{N-1} d(o^i, o^{i+1})$$

4. **"Inertia" Inertia criterion [40]:**
   Another way to look at the reordering problem is not to focus on placing small dissimilarity values close to the diagonal, but to push large values away from it. A function to quantify this is the moment of inertia of dissimilarity values around the diagonal. By using $\|i - j\|^2$ as a measure for the distance to the diagonal and with the weight d_{ij} one has:

$$L(D) = \sum_{i=1}^{N} \sum_{j=1}^{N} d(o^i, o^j) \|i - j\|^2$$

5. **"2SUM" 2-Sum Criterion [17]:**
   The $2-Sum$ loss criterion multiplies the similarity between objects with squared rank differences. With $s(i, j) = 1/(1 + d(o^i, o^j))$ the similarity between o^i and o^j we get:

$$L(D) = \sum_{i,j=1}^{N} 1/(1 + d(o^i, o^j))(i - j)^2$$

6. **"Least squares criterion" [40]:**
   Another natural loss function is to quantify the deviations between the dissimilarities in D and the rank differences of the objects. Such deviations can be measured, e.g, by the sum of squares of deviations, by considering $\|i - j\|$ the rank difference or gap

between o^i and o^j we define:

$$L(D) = \sum_{i=1}^{N} \sum_{j=1}^{N} (d(o^i, o^j) - \|i - j\|)^2$$

7. **"Moore Stress" [20]:**
   This index measures the conciseness of the presentation of a matrix and can be seen as a purity function which compares the values in a matrix with their neighbors. The stress measures used here are computed as the sum of squared distances of each matrix entry from its adjacent entries.

$$L(D) = \sum_{i,j=1}^{N} \sigma_{ij},\ \sigma_{ij} = \sum_{k=max(1,i-j)}^{min(N,i+1)} \sum_{l=max(1,j-1)}^{min(N,j+1)} (d(o^i, o^j) - d(o^k, o^l))^2$$

4.2.1.2 Existing Reordering Algorithms

Several approaches have been proposed to solve the reordering problem. In this chapter, the proposed algorithm is compared to the following existing algorithms:

- Optimal Leaf Ordering algorithm ("OLO") [16]:
  "OLO" propose a hierarchical clustering (average-link) with additional leaf-node reordering to minimize the Hamiltonian Path Length.

- Spectral seriation ("Spect") [17]:
  Spectral seriation uses a relaxation to minimize the 2-Sum Problem. It uses the order of the Fiedler vector of the similarity matrix's Laplacian (normalized) and then sort the component of a specified eigenvector of the Laplacian. The permutation vector computed by the spectral algorithm is a closest permutation vector to the specified Laplacian eigenvector.

- Visual Assessment of (clustering) Tendency ("VAT") [24]:
  VAT creates an order based on Prim's algorithm for finding a Minimum Spanning Tree (MST) in a weighted connected graph representing the distance matrix. The order is given by the order in which the nodes (objects) are added to the MST. The approach is able to signal the presence of well-separated clusters on the main diagonal of the ordered matrix.

- Multidimensional scaling ("MDS") [32]:
  MDS acts as a (often non-linear) dimension-reduction technique techniques and can be used to find a linear order from a dissimilarity matrix. Usually, ordering along the first component of MDS provides good results, minimizing the Least Square Criterion.

- Grauvaeus and Wainer algorithm ("GW") [76]:
  In this algorithm, the clusters are ordered at each level so that the objects at the edge

of each cluster are adjacent to that object outside the cluster to which it is nearest. The method produces an unique order, minimizing the Hamiltonian Path Length.

- Hierarchical Clustering ("HC") [83]:
  Using the order of the leaf nodes in a dendrogram obtained by hierarchical clustering can be used as a very simple seriation technique in order to minimize the dissimilarity of adjacent objects.

4.2.2 Proposed Algorithms

In this section we present the proposed incremental algorithm for matrix reordering. We describe the approach in two different versions. The first version is adapted to static data-sets: the reordered matrix must describe the full structure of the data. In our approach, each object is presented once and compared only with the objects already presented. The matrix is then reordered incrementally. The second version is adapted to dynamic data-sets: in that case the reordered matrix must represent the current structure of the data-set and must be able to "forget" outdated information. The main idea is to incrementally build the similarity matrix in an ordered way from the objects presented in a time-defined sequence. The matrix is constrained to keep a fixed size, or to represent data from a size-defined time window, in order to remove outdated information and to assure a reasonable memory and computation cost.

4.2.2.1 Algorithm for Static Data-sets

The main idea of this approach is to incrementally optimize the Hamiltonian Path Length [40] of the ordered data. We consider that the order of the objects in a dissimilarity matrix corresponds to a path through a graph where each node represents an object and is visited exactly once: a Hamilton path. By minimizing this path length, we globally minimize the similarities between adjacent objects. The same criterion is used in algorithms such as GW [76] or OLO [16] (see section 4.2.1). However, we propose here to incrementally optimize this criterion.

Let's introduce some notations. We call D the $N \times N$ dissimilarity matrix to reorder, or to construct in an ordered way, n being the total number of objects. In our approach, we do not need a dissimilarity matrix *a priori*. A set of objects $\mathcal{O} = \{o^{(1)}, \ldots, o^{(N)}\}$ and a dissimilarity measure d are sufficient, as the objects are treated one by one to construct the ordered matrix. Let k be an integer $1 \leq k \leq N$ representing the number of objects already processed. Also let $L(k) = \{o^1, \ldots, o^k\}$ be the list of objects already treated and $D(k)$ be the matrix under construction.

The first object, which is arbitrarily fixed, forms the initial list $L(1)$. Then, for each new object o^k, the algorithm adds o^k to the list $L(k)$ and compares the best position of o^k in the matrix $D(k-1)$, by minimizing the Hamilton path length. At each step, the path

length is increased by a value λ_p, which depends on the chosen position p of the new object and the dissimilarities $d(o^k, o^p)$ and $d(o^k, o^{p+1})$. For each possible position p, we compute λ_p according to Eq. 4.1 and we choose $p_{opt} = argmin_p(\lambda_p)$. This solution is inspired by a solution from the travelling salesman problem proposed in [167]. The matrix $D(k-1)$ is then updated with the addition of the dissimilarities between o^k and the object in $L(k-1)$, at the p_{opt} position. At the end of the process, the algorithm produces an ordered matrix with a minimized Hamiltonian Path Length. The full algorithm is described in Algorithm 6.

$$\lambda_p^k = \begin{cases} d(o^k, o^p), \text{ for } p = 1 \\ \\ d(o^k, o^{p-1}) + d(o^k, o^p) - d(o^{p-1}, o^p), \\ \quad \text{for } 2 \leq p \leq k-1 \\ \\ d(o^k, o^{p-1}), \text{ for } p = k \end{cases} \tag{4.1}$$

Algorithm 6 Incremental Matrix Reordering

Require: O, d
1: $D = [0]$
2: $L = [o^1]$
3: **for** each new object o^k **do**
4: Compute the similarities $d_{kp} = d(o^k, o^p), o^p \in L$
5: Compute λ_p^k by using Eq. 4.1
6: Compute $p_{opt} = argmin_p(\lambda_p)$
7: Update L by inserting o^k
8: Update D by inserting d_{kp} at the position p_{opt}

4.2.2.2 Algorithm for Dynamic Data-sets

In this section, we aim to present an algorithm to reorder dynamic data (Algorithm 7). The difference here is that new objects are dynamically presented to the system, so each object o^k is now associated with a time stamp t^k, $O = \{(o^1, t^1), \ldots, (o^n, t^n)\}$. The total number of objects in the stream is potentially infinite, and we wish to propose a visualization of the current data's structure at any time. To this end, the constructed matrix is constrained to keep a fixed size, in order to remove outdated information and to assure a reasonable memory and computation cost. In this case we define a parameter θ which indicates the maximum size of the matrix. When a new object is presented to the system, if the updated matrix is bigger than θ_{max}, the row and column corresponding to the older object is removed. An alternative possibility could be to keep a fixed time window instead of a fixed size. In that case, we define a minimum time stamp t_{min} and, at the end of each

step, objects older than t_{min} are removed from the matrix. This algorithm is incremental by construction and is adapted to visualize the temporal variations of the data structure.

Algorithm 7 Incremental Matrix Reordering for Dynamic Data-set

Require: θ_{max} or t_{min}, $\mathcal{O}$, d

$D = [0]$
$L = [(o^1, t^1)]$
for each new object (o^k, t^k) **do**
    Compute the similarities $d_{kp} = d(o^k, o^p), o^p \in L$
    Compute λ_p^k by using Eq. 4.1
    Compute $p_{opt} = argmin_p(\lambda_p)$
    Update L by inserting (o^k, t^k)
    Update D by inserting d_{kp} at the position p_{opt}
    Remove from L and D any information about the object o^i with $t^i < t_{min}$ or the oldest object if $size(D) > \theta_{max}$

4.2.3 Experimental Validation

In this section we will present the experimental protocol we used to validate the proposed algorithm. We first tested the quality of the reordered matrix on a set of artificial and real static data-sets and compared the results with the quality of the state-of-the-art algorithms. The aim was to demonstrate that the performance of the proposed approach is at least similar to its competitors, to reorder a similarity matrix.

Then, we present the results of a dynamic reordering applied to two data-sets with dynamic structures. We show that the proposed algorithm is able to produce a dynamic visualization of the data structure, including the detection of emerging and disappearing clusters of data as well as changes in size between different clusters.

4.2.3.1 Databases Description

Table 4.1: Description of the experimental data-sets.

Data-sets	Size	Dim	Clusters	Type
Art1	1000	5	4	Vector (artificial)
Art2	1000	2	6	Vector (artificial)
Art3	1000	2	7	Vector (artificial)
Iris	150	4	3	Vector (Real)
Digits	537	64	3	Vector (Real)
Glass	214	10	3	Vector (Real)
Wine	178	13	3	Vector (Real)
Prot2	64	-	2	Sequence
Prot3	50	-	3	Sequence
Prot4	129	-	5	Sequence

To validate the effectiveness of the proposed reordering approach, the performances of the proposed algorithm were tested on different static data-sets. The data-set generation and the computation of the similarity matrix were made using biopython [41] and scikilearn [156] Python 2.7 libraries. The description of the experimental data-sets are listed in Chapter 3 and summarized in table 4.1.

4.2.3.2 Evaluation of the Proposed Algorithm Quality

To demonstrate the effectiveness of the proposed approach, we evaluated its performances in terms of reordering quality in comparison to a set of state-of-the-art algorithms. The quality of the proposed ("Prop") algorithm was tested using seven quality indexes (BAR, Moore Stress, Least Square, Weighted Gradient, Path Length, Inertia and 2SUM, see section 4.2.1). Its quality is compared with the quality obtained by random permutations of the matrix ("Rand") and the quality of six existing algorithms: "OLO" [16], "GW" [76], "HC" [83], "VAT" [24], "Spectral" [58], "MDS" [32], see section 4.2.1. The state-of-the-art algorithms and the quality indexes were computed using the "seriation" package [82] in R 3.3.2. We restricted our experiments to algorithms with a complexity in time no more than quadratic ($\mathcal{O}(N^2)$), for practical reasons.

The results are presented in Table 4.2 to 4.8 and summarized in Figure 4.1. Some visual examples are shown in Figures 4.2 and 4.3. To improve the readability of the tables, the indexes' values were normalized using a Min-Max normalization [86] to fit in the $[0, 1]$ interval. In this manner, for each data-set and each index, the algorithm with the "best" quality has a value of 1, whereas the algorithm with the "worst" quality has a value of 0. This has been done regardless of whether the "best" value is the smaller ("BAR", "Path Length", "2SUM", "Least Square" and "Moore Stress") or the higher one. All the algorithms we tested performed better than the random permutations for all indexes, therefore the normalized values for the random permutations is always 0.

Table 4.2: Weighted Gradient values (normalized).

	Art1	Art2	Art3	Iris	Digits	Glass	Wine	Prot1	Prot2	Prot3
Prop	0.96	0.95	0.94	0.96	0.70	0.87	1.00	1.00	0.99	0.86
OLO	0.94	0.89	0.93	0.94	0.77	0.92	1.00	1.00	0.99	0.81
GW	0.94	0.82	0.80	0.93	0.80	0.83	1.00	1.00	0.99	0.78
HC	0.95	0.65	0.59	0.49	0.75	0.70	0.44	0.99	0.97	0.81
VAT	0.86	0.91	0.82	0.82	0.72	0.71	0.94	1.00	0.97	0.87
Spect	0.99	0.96	1.00	1.00	0.71	0.79	1.00	1.00	1.00	0.76
MDS	0.99	0.95	0.99	1.00	0.85	0.88	1.00	1.00	0.96	0.79
Rand	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 4.3: BAR values (normalized).

	Art1	Art2	Art3	Iris	Digits	Glass	Wine	Prot1	Prot2	Prot3
Prop	0.98	0.99	0.99	0.99	0.87	0.96	1.00	1.00	1.00	0.97
OLO	0.97	0.97	1.00	0.98	0.90	0.98	1.00	1.00	1.00	0.95
GW	0.97	0.92	0.91	0.93	0.93	0.91	0.99	1.00	1.00	0.93
HC	0.98	0.89	0.88	0.78	0.87	0.81	0.73	0.99	0.99	0.90
VAT	0.95	0.90	0.93	0.84	0.86	0.81	0.95	1.00	0.99	0.97
Spect	0.99	0.95	0.96	1.00	0.60	0.72	1.00	0.99	0.99	0.70
MDS	0.99	0.92	0.92	1.00	0.72	0.89	1.00	0.99	0.95	0.83
Rand	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 4.4: Path Length values (normalized).

	Art1	Art2	Art3	Iris	Digits	Glass	Wine	Prot1	Prot2	Prot3
Prop	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.00
OLO	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.99
GW	0.99	1.00	1.00	0.98	0.98	0.95	1.00	1.00	1.00	0.94
HC	0.98	0.99	0.98	0.95	0.92	0.88	0.94	0.99	1.00	0.90
VAT	0.94	0.98	0.99	0.90	0.81	0.83	0.98	0.99	1.00	0.97
Spect	0.89	0.90	0.85	0.88	0.32	0.45	0.99	0.98	0.99	0.52
MDS	0.89	0.86	0.78	0.88	0.38	0.66	0.99	0.96	0.94	0.58
Rand	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 4.5: 2SUM values (normalized).

	Art1	Art2	Art3	Iris	Digits	Glass	Wine	Prot1	Prot2	Prot3
Prop	0.98	0.97	0.95	0.97	0.73	0.82	1.00	1.00	0.99	0.85
OLO	0.98	0.97	0.95	0.96	0.84	0.95	1.00	1.00	0.99	0.80
GW	0.98	0.95	0.89	0.94	0.84	0.88	1.00	1.00	0.99	0.78
HC	0.98	0.87	0.76	0.67	0.79	0.77	0.76	1.00	0.98	0.81
VAT	0.95	0.96	0.89	0.88	0.79	0.80	0.96	1.00	0.97	0.86
Spect	1.00	0.98	0.99	1.00	0.81	0.83	1.00	1.00	1.00	0.80
MDS	1.00	0.97	0.98	1.00	0.93	0.93	1.00	1.00	0.99	0.81
Rand	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure 4.1 show the mean values (± standard error) of the different quality indexes over the ten data-sets for each tested algorithm. In order to test the statistical significance of the differences between the proposed approach and the other algorithms, we performed a Friedman test with post-hoc analysis [49], this test was computed using the "pgirmess" package in R 3.3.2.

It is clear from these results that the proposed algorithm performs very well in comparison to the state-of-the-art algorithms. The main difference is that our algorithm works in an incremental way from a random presentation of the data, while the other algorithms have access to all information about the similarity matrix at any time.

Table 4.6: Inertia values (normalized).

	Art1	Art2	Art3	Iris	Digits	Glass	Wine	Prot1	Prot2	Prot3
Prop	0.94	0.91	0.88	0.95	0.62	0.87	1.00	1.00	0.99	0.83
OLO	0.93	0.85	0.86	0.93	0.74	0.91	1.00	1.00	0.99	0.76
GW	0.92	0.76	0.75	0.93	0.76	0.81	1.00	1.00	0.99	0.74
HC	0.94	0.54	0.45	0.40	0.70	0.66	0.33	0.99	0.96	0.78
VAT	0.81	0.92	0.76	0.82	0.67	0.67	0.94	1.00	0.95	0.83
Spect	0.99	0.96	0.98	1.00	0.78	0.83	1.00	1.00	1.00	0.78
MDS	1.00	0.96	0.97	1.00	0.92	0.88	1.00	1.00	0.98	0.78
Rand	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 4.7: Moore Stress values (normalized).

	Art1	Art2	Art3	Iris	Digits	Glass	Wine	Prot1	Prot2	Prot3
Prop	1.00	1.00	1.00	1.00	0.99	0.99	1.00	1.00	1.00	0.97
OLO	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00
GW	1.00	1.00	1.00	0.99	0.99	0.99	1.00	1.00	1.00	0.99
HC	1.00	0.99	0.99	0.96	0.96	0.97	0.93	1.00	0.99	0.98
VAT	0.99	0.99	0.99	0.93	0.92	0.94	0.98	1.00	0.99	1.00
Spect	0.99	0.95	0.97	1.00	0.41	0.73	1.00	0.99	1.00	0.66
MDS	0.99	0.92	0.90	1.00	0.68	0.85	1.00	0.99	0.96	0.84
Rand	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 4.8: Least Square values (normalized).

	Art1	Art2	Art3	Iris	Digits	Glass	Wine	Prot1	Prot2	Prot3
Prop	0.96	0.99	0.89	0.95	0.70	0.87	1.00	1.00	0.99	0.86
OLO	0.94	0.95	0.88	0.94	0.77	0.92	1.00	1.00	0.99	0.81
GW	0.94	0.93	0.76	0.93	0.80	0.83	1.00	1.00	0.99	0.87
HC	0.95	0.59	0.56	0.48	0.75	0.70	0.44	0.99	0.97	0.78
VAT	0.86	0.86	0.78	0.81	0.72	0.71	0.94	1.00	0.97	0.81
Spect	0.99	0.99	0.95	1.00	0.71	0.79	1.00	1.00	1.00	0.76
MDS	0.99	0.98	0.94	1.00	0.85	0.88	1.00	1.00	0.96	0.79
Rand	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Despite this limitation, although the obtained results are not always better than the competitors depending on the indexes and data-sets, its quality remains comparatively very high. The statistical comparisons, based on our experimental data-sets, show that the quality of the proposed approach is not significantly lower than the existing algorithms for the seven indexes tested. It is actually significantly better than "HC" for five of the seven indexes and significantly better than "VAT", "Spectral" or "MDS" for a few indexes. The proposed algorithm is the best approach to minimize the Path Length index, which is not surprising as it was conceived to optimize this criterion. It is, however, slightly less efficient for data-sets with highly overlapping or ill-defined clusters (data-set "Digits" is a

good example), which also affects the quality of algorithms such as "VAT", "Spectral" or "MDS".

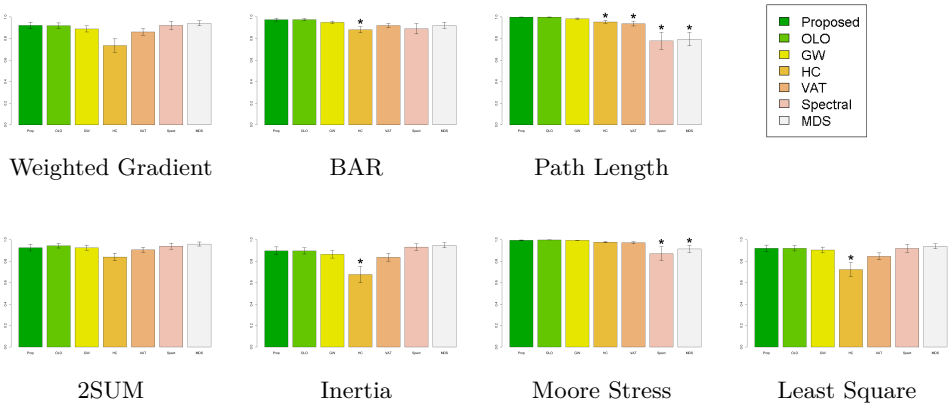


Figure 4.1: Mean values (± standard error) of the different quality indexes over the ten data-sets for each tested algorithm. "*" denotes a significant difference between an existing approach and the proposed algorithm (Friedman test with post-hoc analysis).

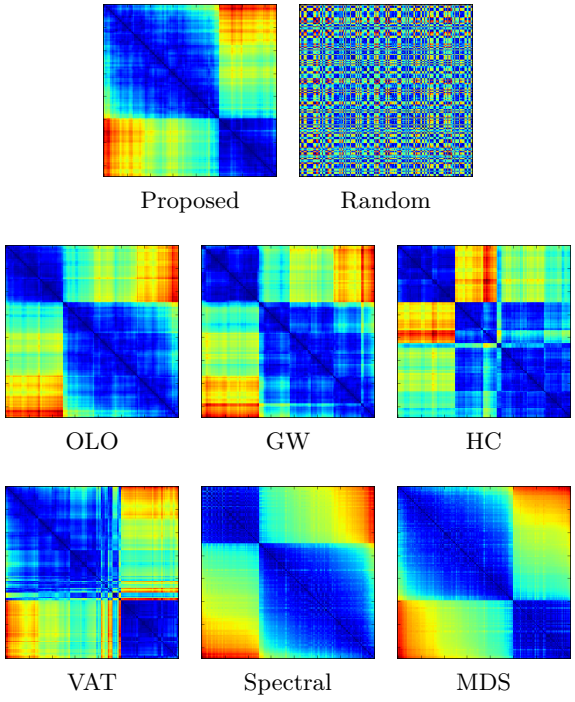


Figure 4.2: Example of visualization of the reordered similarity matrix obtained with different algorithm for the "Iris" data-set.

The visual inspection of the resulting matrices (Figures 4.2 and 4.3) confirms the adequacy of the proposed approach for the reordering task.

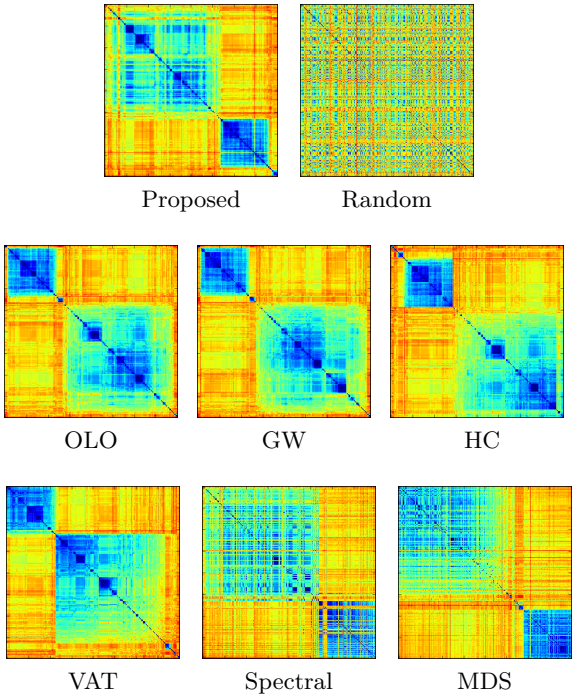


Figure 4.3: Example of visualization of the reordered similarity matrix obtained with different algorithm for the "Prot3" data-set.

4.2.3.3 Application to Dynamic Data-sets

In the previous section we showed that our algorithm performs at least as well as the existing approaches despite computing the reordering in an incremental way from a random presentation of the data. In this section we will show the capability of our algorithm to visualize the dynamic structure of an evolving data-set. To this end we applied the algorithm on two generated data-sets presenting a dynamic structure:

"Dyn1" is a data-set of $10,000$ two-dimensional vectors with a predefined temporal organization. The first data points are organized in three clusters of different shapes and dimensions, but with a similar data distribution in each cluster (i.e. a similar number of data point in each cluster). However, the distribution of the clusters changes over time and the probability of a data point to be generated in the first cluster increases whereas the probability of belonging to the third cluster decreases. The size of these clusters in the representation space also changes over time. In addition, the data from the second cluster's "shift" in the representation space and are increasingly generated away from the two other clusters. Figure 4.4 shows a visualization of the data-set's dynamics and the corresponding visualization of the data structure with the proposed algorithm. All similarities are com-

puted using the Euclidean distance. We have chosen here to keep the matrix size to 1000 data points, in order to have a representation of the "current" structure of the data for ten periods of time. Note that the ten matrix visualizations presented in Figure 4.4 are "snapshots" of the same matrix at different moment of the process. Each of the 10000 data points is presented once to the algorithm, in an order randomly chosen to follow the data-set's distributions dynamic (i.e. from distributions that vary over time). The 3 clusters are perfectly represented in the matrix. The change in the probability belonging to the different clusters is clearly visible as the size of the first and the last cluster vary in the matrix over time. The increasing distance of the second cluster is also clearly visible as the areas between the clusters in the matrix become increasingly red (i.e. the clusters become small and smaller).

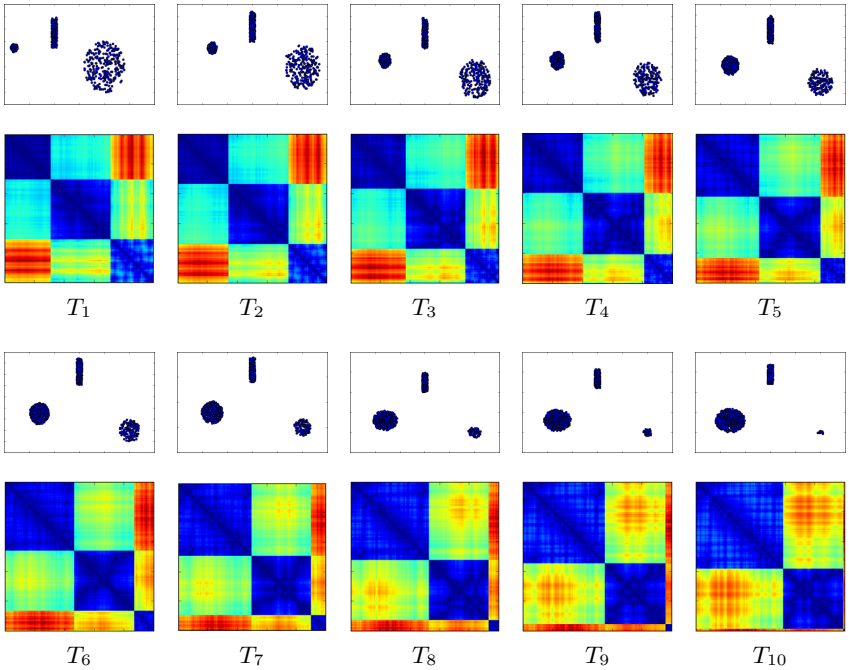


Figure 4.4: Example of visualization obtained using the proposed incremental reordering approach to the "Dyn1" data-set. The size of the matrix is kept at 1000×1000, allowing a description of the structure of the last 1000 observations.

The second dynamic data-set, "Dyn2", is also a data-set of $10,000$ two-dimensional vectors with a predefined temporal organization (Figure 4.5). This time, the number of clusters varies over time. The first data points (T_1) are organized in three Gaussian clusters of different densities. Then a new cluster appears and its density increases over time (T_2 to T_5), until the second cluster's density starts to decrease and finally completely disappears (T_3 to T_5). Then one cluster starts to split into two new clusters (T_6 to T_{10}). Figure 4.5 shows a visualization of the data-set's dynamic and the corresponding visualization of the data structure with the proposed algorithm.

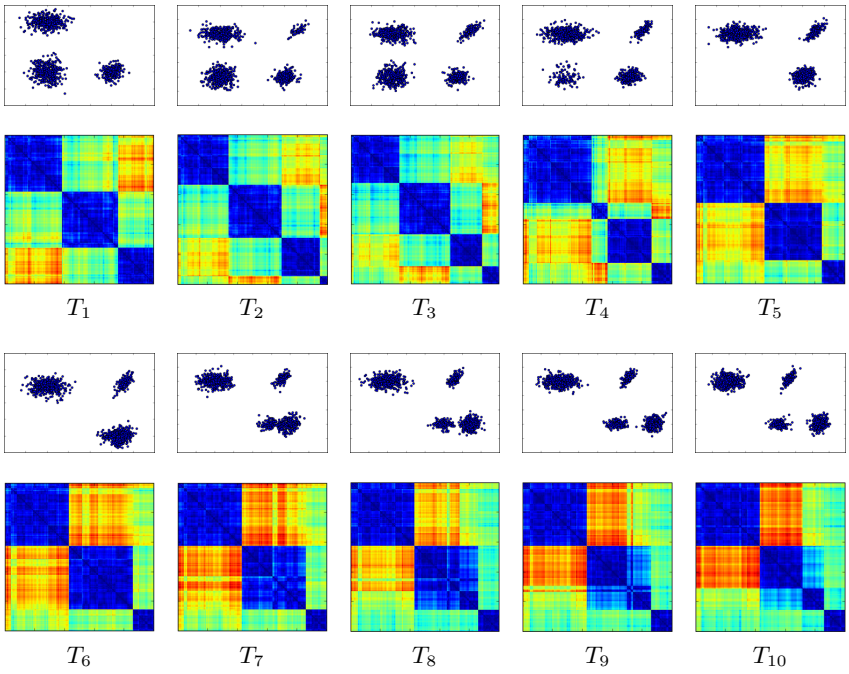


Figure 4.5: Example of visualization obtained using the proposed incremental reordering approach to the "Dyn2" data-set.

One can observe that the clusters are perfectly represented in the matrix. The emergence of new clusters is clearly visible, as well as the disappearance of the second cluster. In addition, the progressive split of one cluster into two new clusters can be seen in the matrix from one big cluster up to two well separated smaller clusters. This example is more complex than the first one, and the quality of the resulting visualization demonstrates the effectiveness of our approach.

4.2.4 Summary

In this section, we proposed an incremental approach to matrix reordering to visualize the structure of static and dynamic data-sets. Our algorithm performs very well on static data-sets in comparison to the state-of-the-art algorithms: its quality remains comparable or higher, despite working in an incremental way from a random presentation of the data, while the other algorithms can use any information about the similarity matrix at any time. In addition, the algorithm assures a reasonable memory and computation cost for dynamic data-sets, and the visualizations show that our approach is perfectly suitable to detecting temporal variations in the data structure, such as changes in density, appearance and disappearance of clusters of data or changes in similarities between clusters.

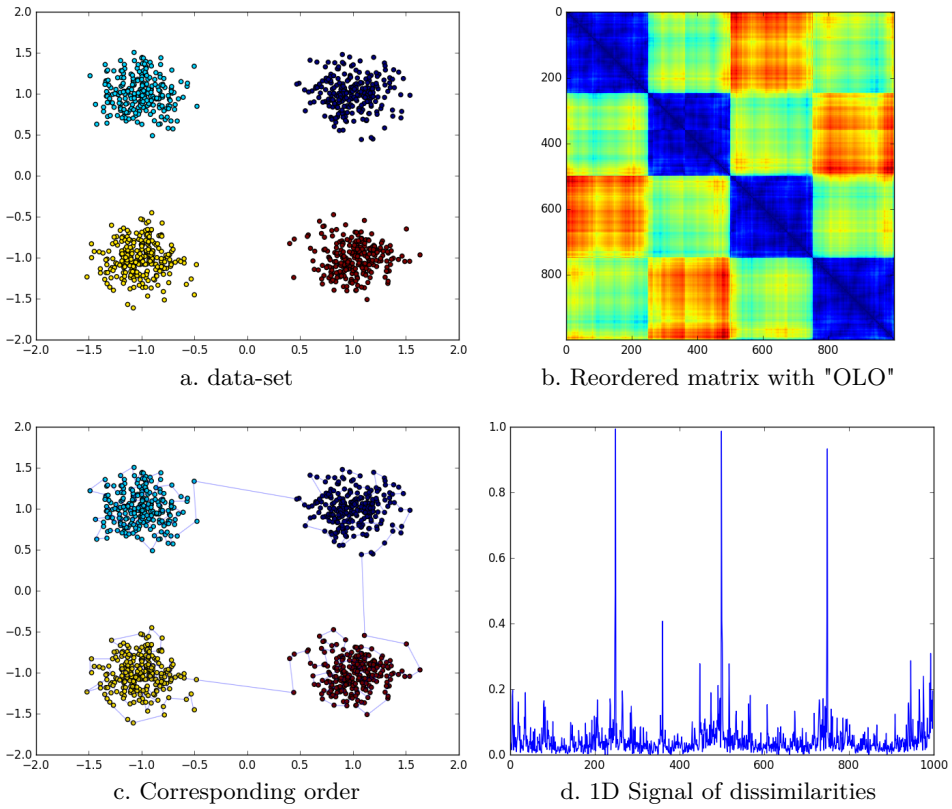
4.3 Signal-based Autonomous Clustering for Relational Data

In this section we propose a new clustering approach adapted to relational data-sets. This approach uses a reordering matrix to transform the relation between data into a one-dimension signal. Then we apply a peaks detection algorithm to this signal in order to find the distance boundaries between clusters.

4.3.1 Peaks Detection for Relational Data Clustering

In a nutshell, our strategy has two steps: (A) reorder the dissimilarity matrix and construct a vector (signal) containing the pairwise distance between the ordered data and (B) detect the peaks of distance in the previously constructed vector to detect the boundaries between clusters.

4.3.1.1 Reordering and Construction of the Signal



a. data-set b. Reordered matrix with "OLO"

c. Corresponding order d. 1D Signal of dissimilarities

Figure 4.6: The proposed approach is based on the reordering of the dissimilarity matrix in order to produce a 1D signal of pairwise dissimilarities. A non-parametric peaks detection algorithm is applied on the signal to detect clusters boundaries.

From the observed data, we first compute a reordered dissimilarity matrix minimizing the sum of the distances along the Hamiltonian path connecting the observation in the given order (see an example on Figure 4.6b and 4.6c). In this way, observations from the same clusters are close to each other in the order. Once the reordered matrix is obtained, the signal of pairwise dissimilarities of our ordered observations is its first non-zero main diagonal. This signal can be used to automatically detect the clusters' boundaries. Indeed, within a cluster we expect to have small oscillations of the dissimilarity, while between two different clusters there is a peak of large amplitude if the clusters are not overlapping (see Figure 4.6d).

4.3.1.2 Peaks Detection Algorithm

Let v be the vector representing the first non-zero main diagonal of the dissimilarity matrix. In order to detect the peaks of large amplitude in v, we propose to adapt a procedure introduced in [138] for jumps detection in noisy signals. There are two important differences between our procedure and the procedure in [138]: the first one is the choice of the discretization defined in Equation (4.2) and the second one is the definition of the cost function in Equation (4.4). The detection method is of particular interest for us as it determines the significant peaks in the signal vector v without knowing their number and location, nor a specific threshold to decide the significance of a peak.

We consider that the elements $\{v_i\}_{i=1}^N$ of v are the local weighted averages of an underlying piece-wise continuous function $v : I = [0, 1[\subset \mathbb{R} \to \mathbb{R}$ on the intervals $I_i^N = [i/N, (i+1)/N[$ defined by the uniform subdivision of step $1/N$ of the interval $[0, 1[$. With these precisions, the observed values are given by:

$$v_i^N = N \int_{I_i^N} v(t) w(n \cdot t - i) dt, i = 0, \ldots, N-1. \tag{4.2}$$

In other words, in our procedure we consider the weighted averages of the signal v against the hat function $w(t) = \max(1 - |t|, 0)$. Our model function v is a piece-wise regular function containing a finite number of peak singularities located in $L_p = \{y_1, y_2, \ldots, y_p\} \subset I$ meaning that

$$v'(y_i^+) := \lim_{x \to x_i, y > y_i} v'(y) \neq v'(y_i^-) := \lim_{x \to x_i, y < y_i} v'(y), \tag{4.3}$$

where v' remains for the first derivative. Each fixed peak in the set L_p belongs to a fixed interval of our subdivision I_i^N. We aim to detect all elements of the list L_p. To this end we will assume that $M_d = \min_{i \neq j} |y_i - y_j| > 3$, d stays here for detection. In other words, we assume that two different significant peaks in L_p are well separated by at least three intervals.

Algorithm 8 Signal-based Clustering using Peaks Detection

Input: v^n = first non-null diagonal vector of D of length N.

Output: Peaks List: $L_{peaks} = L_1 \cap L_2 \cap ... \cap L_l$.

1: Initialization: $l = [\log_2(N)], v^l = v^N, M_d = [\log_2(N)]/4$

2: **while** $l \geq 1$ **do**

3: Compute L_1 list of local maxima of v^l by using the local maxima of the second order finite difference:

$$F(i) = \underset{i_M d \leq k \leq i+M_d}{\operatorname{argmax}} \mid v^l_{k+1} - 2 \cdot v^l_k + v^l_{k-1} \mid$$

4: Compute for each i the local maxima of F within the interval $[i - M_d, i + M_d]$. By construction:

$$L_l = \{i^*; i^* = F(i), 1 \leq i \leq N\}$$

5: Apply the smoothing operator:

$$v^{l/2}_i = \frac{1}{4}(v^l_{2i-1} + 2 \cdot v^l 2i + v^l_{2i+1}), i = 1, ..., l/2 - 1$$

6: $l = l - 1$

The detection strategy is based on the idea introduced in [87]. To this end we define the stencil $\mathcal{S}_i$ by $\{i + l,\ -M_d \leq l \leq M_d - 1\}$ which is a set of indexes of intervals containing I^N_i. Of course, there are $2 \cdot M_d - 1$ such stencils. For each interval i we consider the least oscillatory stencil containing the i interval. This choice is carried out so as to minimize the effect of the presence of a potential singularity. We define m^*_i:

$$m^*_i = \underset{-M_d \leq m \leq M_d}{\operatorname{argmin}} F(I^N_{i+m}) = \underset{-M_d \leq m \leq M_d}{\operatorname{argmin}} \sum_{l \in \mathcal{S}_{i+m}} |\Delta^2 v^N_l|, \tag{4.4}$$

where $\Delta^2 v_l = v_{l+1} - 2 \cdot v_l + v_{l-1}$. A small $F(I^N_i)$ is associated with a locally smooth stencil and a free peak interval.

In that context, the singularities at level n are associated with the indexes i such that $m^*_{i-1} = -M_d$ and $m^*_{i+1} = M_d$: the least oscillatory stencils for $i - 1$ and $i + 1$ avoid the interval indexed by i. One defines intervals I^N_i potentially containing a peak singularity as those satisfying:

$$F(I^N_i) > F(I^N_{i \pm r}), \text{ for all } r = 1, \cdots, 2M_d + 1. \tag{4.5}$$

This definition means that the singularity is associated with the most oscillatory stencil involving the interval I_i^N. Therefore we get the list of the suspicious interval to contain a peak for the subdivision of order n as follows: $L^N = \{i^*; i^* = \text{argmax}_i\, F(I_i^N)\}$.

In practice, too many peaks singularities are detected on a fine scale n and of course not all correspond to peaks of interest. To avoid this situation, one needs to adopt a rough to a more refined strategy and keep only the most relevant ones by chaining across the different levels according to a certain propagation law. The coarse version of v is obtained by weighted averaging:

$$v_i^{N/2} = \frac{1}{4}(v_{2i-1}^N + 2 \cdot v_{2i}^N + v_{2i+1}^N), i = 1, \ldots, N/2 - 1. \tag{4.6}$$

We could summarize our procedure as follows: we first find L^N the list of suspicious intervals associated to the N order subdivision, and secondly, by considering the coarse version of v, we obtain a second list $L^{n/2}$ of suspicious intervals as before. After that, these two lists merge in the list L_{peaks} as follows: a peak will be considered in the interval I_{2i}^N or I_{2i+1}^N if the interval $I_i^{N/2}$ is also detected as suspicious at the coarse scale. This two-step procedure is iterated $[\log_2(N/2)] + 1 = [\log_2(N)]$ times with $[\log_2(N)]$ the integer part of $\log_2(N)$. A peak is observed if a chain of detection exists from refined to coarse scales. Note that for the smaller N, by using the averaging defined in 4.6, only the peaks of interest with large amplitude are detected since the others with small amplitude have mainly been smoothed out.

Finally, the number of clusters is obtained by $p = |L_{\text{peaks}}| + 1$. As each L_{peaks} is a boundary between with clusters, it is easy to associate the observations to the different clusters. Indeed, thanks to the reordering properties, all observations between two peaks belong to the same clusters.

4.3.1.3 Complexity

Most clustering approach for relational data have a complexity of at least $\mathcal{O}(N^2)$, such as DBSCAN, Affinity Propagation and Hierarchical Clustering, up to $\mathcal{O}(N^3)$ for Spectral clustering. The proposed peak detection algorithm has a low computational complexity of $\mathcal{O}(N)$. The overall complexity of the proposed approach depends on the reordering algorithm. The approach proposed in the previous section has a complexity of $\mathcal{O}(N)$.

4.3.2 Experimental Validation

In this section we present the experimental protocol we used to validate the proposed algorithm. The algorithm is compared to four state-of-the art clustering algorithms adapted to dissimilarity matrices. The different approaches have been tested on four artificial and six real data-sets. The quality of each algorithm has been tested with four quality indexes.

4.3.2.1 Databases Description

To validate the effectiveness of the proposed clustering approach, we tested its performances on different relational data-sets. We summarized the data-set description in table 4.9 (for a description of these data-sets please see Chapter 3).

Table 4.9: Description of the experimental data-sets.

Data-sets	Size	Dim	# Classes	Type
Art1	1000	2	4	Vector (artificial)
Art2	1000	5	4	Vector (artificial)
Art3	800	2	5	Vector (artificial)
Art4	1000	2	8	Vector (artificial)
Prot1	115	-	3	Sequence
Prot2	64	-	2	Sequence
Prot3	50	-	3	Sequence
Prot4	129	-	3	Sequence
Prot5	98	-	5	Sequence
Prot6	76	-	4	Sequence

4.3.2.2 Evaluation of the Proposed Algorithm Quality

The quality of the proposed algorithm was tested using four external quality indexes (see previous section) and compared with the quality of four existing approaches adapted to dissimilarity matrices (DBSCAN [62], Spectral Clustering [149], Affinity Propagation [65] and Single Link Hierarchical Clustering [104]. These indexes and algorithms were implemented using the *scikitlearn* package in python [156].

The proposed approach is based on the reordering of the dissimilarity matrix using the approach presented in this chapter. The peaks detection algorithm presented in Section 4.3.2 was then applied to the signal obtained with the pairwise distances between the ordered observations. No parameter is needed and the number of clusters is detected automatically. For the Affinity Propagation algorithm, we chose a fixed damping factor value of 0.5. The min sample parameter of the DBSCAN algorithm has been fixed at 5 and the eps value has been adjusted experimentally for each data-set in order to obtain the best possible result. The Spectral Clustering and the Single Link Hierarchical Clustering algorithms were given the true number of clusters to find for each data-set.

The results are presented in Table 4.10 to 4.13, and some visual examples are shown in Figures 4.7 and 4.8. It is clear from these results that the proposed algorithm performs very well in comparison to the state-of-the-art algorithms. The main difference is that our algorithm is autonomous, there are no parameters to tweak in order to improve the results. However, the quality of the obtained clustering is always as least as good as the competitors.

Table 4.10: Values of the Adjusted Rand Index for each algorithm and each data-set.

Data-sets	Proposed	DBSCAN	Spectral	Affinity	Single
Art1	1.00	1.00	1.00	0.90	1.00
Art2	1.00	1.00	1.00	0.32	1.00
Art3	1.00	0.99	0.86	0.31	1.00
Art4	0.99	0.96	0.55	0.25	0.94
Prot1	0.98	0.98	0.98	0.22	0.98
Prot2	0.94	0.94	0.94	-0.01	0.94
Prot3	0.88	0.88	0.88	0.57	0.88
Prot4	0.86	0.86	0.86	0.75	0.86
Prot5	0.94	0.94	0.94	0.33	0.94
Prot6	0.80	0.79	0.78	0.45	0.79

Table 4.11: Values of the Adjusted Mutual Information Index for each algorithm and each data-set.

Data-sets	Proposed	DBSCAN	Spectral	Affinity	Hierarchical
Art1	1.00	1.00	1.00	0.87	1.00
Art2	1.00	1.00	1.00	0.37	1.00
Art3	1.00	0.97	0.86	0.43	1.00
Art4	0.99	0.93	0.74	0.38	0.88
Prot1	0.96	0.96	0.96	0.11	0.96
Prot2	0.90	0.90	0.90	-0.05	0.90
Prot3	0.85	0.85	0.85	0.53	0.85
Prot4	0.84	0.84	0.84	0.72	0.84
Prot5	0.91	0.91	0.91	0.11	0.91
Prot6	0.75	0.73	0.73	0.73	0.73

Table 4.12: Values of the V-measure for each algorithm and each data-set.

Data-sets	Proposed	DBSCAN	Spectral	Affinity	Hierarchical
Art1	1.00	1.00	1.00	0.92	1.00
Art2	1.00	1.00	1.00	0.53	1.00
Art3	1.00	0.98	0.90	0.54	1.00
Art4	0.99	0.95	0.80	0.49	0.94
Prot1	0.96	0.96	0.96	0.42	0.96
Prot2	0.90	0.90	0.90	0.10	0.90
Prot3	0.86	0.86	0.86	0.67	0.86
Prot4	0.86	0.86	0.86	0.77	0.86
Prot5	0.92	0.92	0.92	0.34	0.92
Prot6	0.81	0.80	0.81	0.54	0.80

Other algorithms need either the true number of clusters, for example Single Link and Spectral Clustering, or a "scale factor" for DBSCAN. In our experiments, only Affinity Propagation doesn't change its parameters' values depending on the data-set to analyze.

Table 4.13: Values of the Fowlkes-Mallows score for each algorithm and each data-set.

Data-sets	Proposed	DBSCAN	Spectral	Affinity	Hierarchical
Art1	1.00	1.00	1.00	0.93	1.00
Art2	1.00	1.00	1.00	0.62	1.00
Art3	1.00	0.99	0.90	0.58	1.00
Art4	1.00	0.97	0.65	0.49	0.95
Prot1	0.98	0.98	0.98	0.42	0.98
Prot2	0.97	0.97	0.97	0.09	0.97
Prot3	0.92	0.92	0.92	0.76	0.92
Prot4	0.90	0.90	0.90	0.81	0.90
Prot5	0.96	0.96	0.96	0.51	0.96
Prot6	0.87	0.86	0.87	0.69	0.87

However, the results show clearly that Affinity Propagation achieves a clustering of lesser quality than the other approaches.

Being able to propose a good clustering in an autonomous way is a great advantage for a clustering algorithm. Indeed, in most real applications there is no prior knowledge of the number of clusters to find or other parameter values.

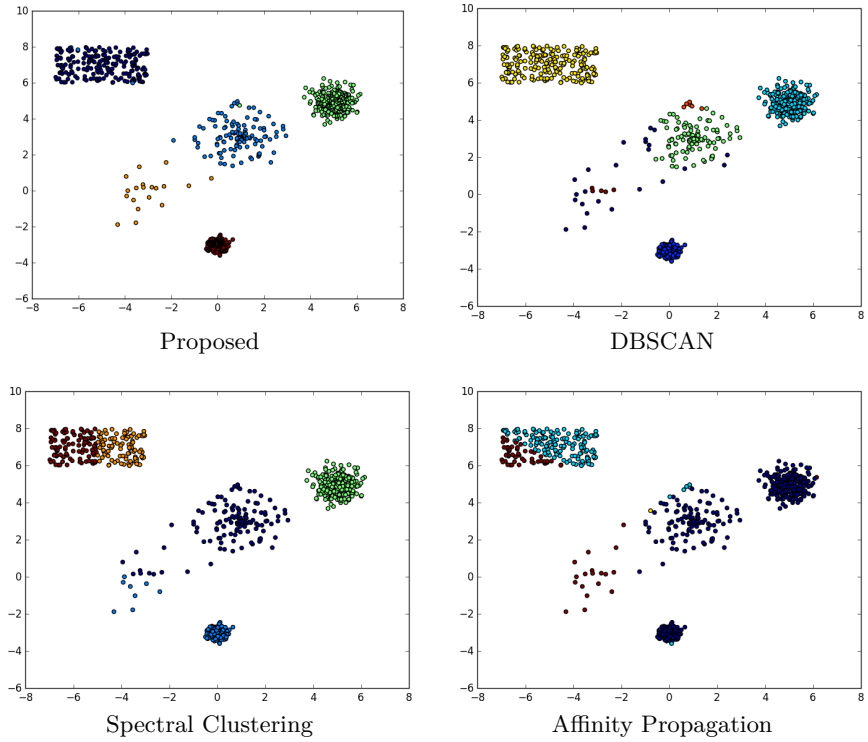


Figure 4.7: Visualization of the clustering obtained for the "ART3" data-set with different algorithms. The proposed approach finds the correct clusters with very few errors. This is not the case of the other algorithms.

Visualizations of the resulting clustering for vectorial data-sets (Figures 4.7 and 4.8) confirm the adequacy of the proposed approach.

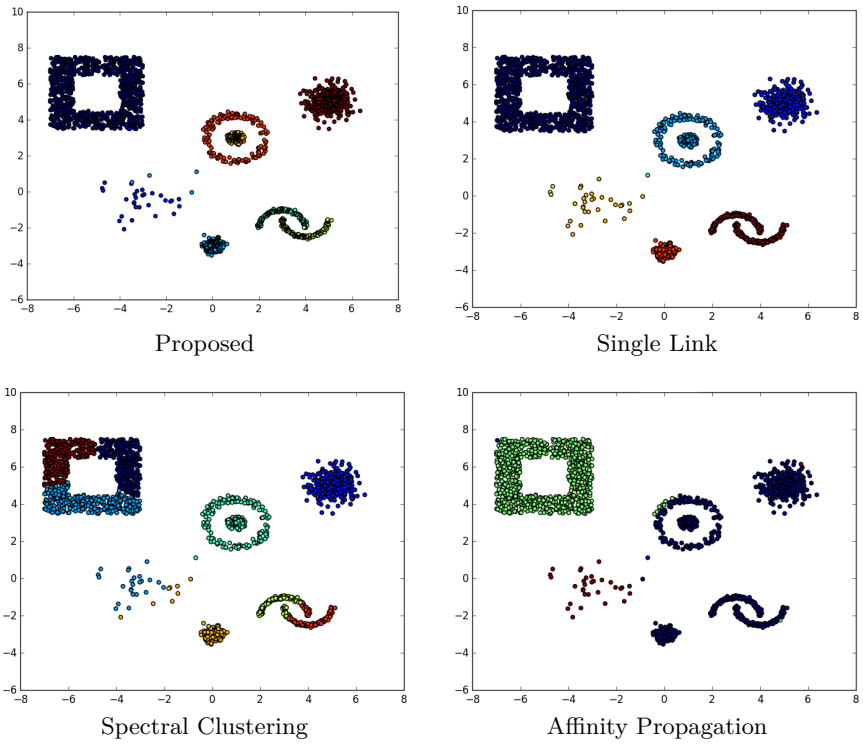


Figure 4.8: Visualization of the clustering obtained for the "ART4" data-set with different algorithms. Only the proposed approach finds the correct clusters with very few errors.

For this data-sets, the clusters' boundaries are defined by distance jumps and are perfectly detected by the proposed algorithm. This is not the case for the other approaches. Indeed, in our experiments, DBSCAN fails to correctly find clusters of different densities and Single Link is unable to separate non-convex clusters. Spectral Clustering and Affinity propagation also have difficulties to detecting clusters with different densities and shapes.

4.3.3 Summary

In this section, we proposed a new approach of clustering adapted to relational data-sets. The first step is to use a reordering technique to transform the relation between data into a one-dimension signal. Then we apply a peaks detection algorithm to this signal in order to find the distance-defined boundaries between clusters. The proposed algorithm performs very well in comparison to the state-of the art algorithms. It has a low complexity, its quality is at least equal to the competitors and there is no parameter value to be defined by the user whereas most other algorithms need to define number of clusters or other parameters values that have to be adapted to each data-set.

4.4 Conclusion

The proposed approaches described in this chapter show interesting results, with the advantage to be able to deal with dynamic data and an automatic detection of the number of clusters. The only user-defined parameter is the time windows. It would be interesting to propose a heuristic method to choose the optimal size of the matrix or the time window, depending on the computational and memory constraints, and an estimation of the rate of changes in the data structure.

The main drawback of the proposed clustering approach is its usual difficulties to detect overlapping clusters. One idea is to focus on being able to detect clusters defined by a variation in densities instead of variations in distances. One idea would be to compute a signal representing the variation of the density in the reordered data and to detect local minimal of density that defines boundaries between clusters. Another approach (potentially complementary) would be to compute prototypes representing the clusters, in a way adapted to relational data with a dynamic structure. This is the topic of the following chapter.

Chapter 5

Prototype-based Clustering for Relational Data

Table of Contents

5.1 Introduction

Prototype-based approaches are very popular in Unsupervised Learning, because of the compactness of the resulting model (the prototypes), the descriptive power of these prototypes and the low computational complexity of the model (each object is only compared to a usually small set of prototypes). This low complexity alone explains the popularity of prototype-based approach in real-life applications.

Usually, the best choice of prototype is the barycenter of the cluster. The prototype is then defined as the object minimizing the sum of square distances with all the objects in the cluster. If the objects are described as numerical vectors in a Euclidean space, the definition of clusters' prototypes is straightforward. In that case, a prototype is a vector defined in the same space, computed as the vectorial mean of the objects belonging to its cluster. In fact, most prototype-based algorithms are only adapted to vectors in a Euclidean space. However, in many cases the objects cannot be easily defined in a Euclidean space without a loss of information and/or a costly preprocessing (e.g. images, networks, sequences, texts). The similarity between such object is usually not a Euclidean distance, and the usual computation of the prototypes is no more valid.

Few works have been done yet in relational prototype-based clustering, but some authors have worked on adapting K-means to relational data [152, 90, 169, 45]. The main issue in relational prototype-based clustering is the definition of the prototypes based solely on the distances between objects. The Partitioning Around Medoids (PAM) algorithm [109] get around the problem by choosing K "prototypes" among the objects of the data-set, allowing a distortion in the clusters representation. In order to propose a better representation of the clusters, in [90] the authors represent the prototypes as a linear combination of the input data. But, as far as processing power and memory usage are concerned, this implementation is very expensive ($\mathcal{O}(N^2)$ for N objects), making it unusable for large sets of data. In [169], the authors proposed to enforce sparser prototyping by considering the prototypes as a linear combination of a subset of objects from the data-set (called support points), succeeding in improving the computational complexity. However, the support points are specific to each cluster and the complete dissimilarity matrix (size $N \times N$) is still needed to be kept in memory.

In this chapter, we will study an alternative method to select the support points to train a relational K-means: The objective is to use one unique set of representatives through the whole learning process, independently from the clusters, in a way that all the prototypes are represented by the same subset of objects, improving the cost of computing (and storing) new support points and more importantly, the cost of computing the new prototypes. We will also introduce the Barycentric Coordinate formalism, in order to unify the representation of the objects and the prototypes and allow a simple incremental learning process for relational

clustering. Finally, we will propose a few theoretical arguments for the choice of the support points.

5.2 Review of Prototype-based Relational Clustering Approaches

We recall below the K-means algorithm and we present the relational K-means algorithms, adapted to relational data. We also highlight the qualities and drawbacks of these algorithms.

5.2.1 K-means

K-means is a prototype-based algorithm [126], which aims to minimize the within-cluster sum of squares. The data in each cluster are represented by a prototype (i.e. a centroid).

K-means works on objects represented by vectors. Let $X = \{x^1, ..., x^N\}$ be the set of vectors representing the N objects, K-means clustering compute a partition C of the N objects into K ($\leq$ N) clusters. The prototypes are computed so as to minimize the sum of square distances (SSD) between each observation and its closest prototype:

$$SSD = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - \mu^k\|^2 \tag{5.1}$$

This minimization problem has been shown to be NP-hard in Euclidian space [9][132]. Its computational complexity for a data-set of N input is [97]: $\mathcal{O}(N^{DK+1} \log N)$. However, local optima can be computed with the Lloyd algorithm [126], which is the standard approach for K-means clustering. This approach has a linear complexity of $\mathcal{O}(NKD)$. Lloyd's algorithm uses an iterative refinement technique: After an initialization of the K prototypes, we assign each object to its closest prototype (*Assignment step*), then we update each prototypes i such that μ^i is the barycenter of the objects in cluster i (*Update step*). We iterate these two steps until the assignments stop changing.

Algorithm 9 K-means

Input: Vectorial Data, K
Output: Prototype's vector μ^k.

Initialize cluster centroids $\mu^1, ..., \mu^k$
while the convergence is not attained **do**
  for each object x^i **do**
    $c^i = \arg\min_j \|x^i - \mu^j\|^2$
  for for each cluster k **do**
    $\mu^k = \frac{1}{|C_k|} \sum_{x \in C_k} x$

The standard formulation of K-means is described in Algorithm 9, where c^i represent the label of the cluster in which the object i is assigned and $|C_k|$ is the number of object in cluster k.

To initialize the cluster centroids (first step), it is possible to choose K objects randomly, and set the cluster centroids to be equal to the vector representation of these objects. However, since K-means algorithm is sensitive to the initial set-up of centroids, in K-means++ [12] the authors propose a variant for choosing the initial values. The idea is to select prototypes iteratively among the data point, each new prototype being the farthest data point from the already chosen prototypes. Let $d(x^i, \mu^j)$ denote the distance from a data point x^i to the prototype μ^j. Then, the initialization process is defined as follow:

1. Choose one prototype μ^1 randomly among the data points.
2. Choose the farthest data point from μ^1 (highest $d(x^i, \mu^1)$) as the second prototype μ^2.
3. Choose the data point with the highest distance from the existing prototypes as the next prototype μ^j
4. Repeat Step 4 until K prototypes have been chosen.

5.2.2 Relational K-means

Relational K-means (sometimes called Naive Relational K-means) cover the case of relational data-sets, where the objects are represented as a distance (or dissimilarity) matrix D. In that case, the prototypes can no longer be defined based on a vectorial representation of the objects, because this representation is unknown. In [169], the authors propose to consider that the objects have an unknown hypothetical representation $X = \{x^1, ..., x^N\}$ in a pseudo-Euclidean space $E*$. A prototype w^k can be associated with a representation μ^k in $E*$ defined as a normalized linear combination of the data hypothetical representations in $E*$: $\mu^k = \sum_{i=1}^{N} \alpha_i^k x^i$ with $\alpha^k \in \mathbb{R}^N$ and $\sum_{i=1}^{N} \alpha_i^k = 1$. It is therefore possible to define the squared distance $d(k, i)$ between each objects o^i and each prototype w^k based only on the coefficients α^k and the dissimilarity matrix D[168]:

$$d(w^k, o^i) = (D\alpha^k)_i - \frac{1}{2}\alpha^{kT} D\alpha^k \tag{5.2}$$

The relational K-means follows the same concept of the classical variant, i.e., alternating assignment and update, while minimizing the sum of square distances (see Algorithm 10). A solution to the minimization problem would be [169]:

$$\alpha^k = \frac{1}{|C_k|}(\delta_{k,1}, ..., \delta_{k,N}) \tag{5.3}$$

where $|C_k|$ is the number of objects in cluster k, and $\delta_{k,1}$ correspond to the Kronecker delta ($\delta_{i,j} = 1$ for $i = j$ and $\delta_{i,j} = 0$ otherwise). Note that the hypothetical representations x and μ of the objects and prototypes are never computed.

Algorithm 10 Relational K-means

Input: D, K
Output: Prototypes' coefficients α^k

```
Initialize the prototypes' coefficients randomly
while the convergence is not attained do
    for each object o^i do
        Assign o^i to the cluster k minimizing d(w^k, o^i)
    for for each cluster k do
        Update α^k using (5.3)
```

The main drawback of the algorithm is its time and memory complexity in $\mathcal{O}(KN^2)$. By considering that in (5.3) each vector α^k is sparse, it is possible to reduce the complexity to $\mathcal{O}(N^2)$ by computing only nonzero terms in (5.2) [169]. However, a complexity in $\mathcal{O}(N^2)$ is far too high to be usable in real applications with large data-sets.

5.2.3 Sparse Relational K-means

In [169], the authors proved that the sparse variation of relational K-means is more efficient than an naive adaptation of K-means for dissimilarity data. For a sparse representation, the prototypes are represented only by a limited number of object, which means that prototypes are a linear combination of a relatively small portion $J_k = (j_{k,1}, ..., j_{k,P})$ of the corresponding cluster, where $o^p_{j_k} \in C_k$ for all p, P being the number of support points chosen.
We are looking for a representation μ^k, the prototype of cluster k, as a normalized linear combination of the support points $o^p_{j_k}$. That can be translated into: For each prototype k, there is $\alpha^k \in \mathbb{R}^N$ such as:

$$\mu^k = \sum_{i=1}^{N} \alpha^k_i o^i \text{ where } \forall i \notin J_k, \alpha^k_i = 0 \text{ and } \sum_{i=1}^{N} \alpha^k_i = 1 \tag{5.4}$$

Considering that the general principle is the same for this sparse variation of relational K-means, the minimization problem for each cluster is still the minimization of Eq. 5.2. The main difference is that now the coefficients α^k have only a small number P of non-null values, as P support points are chosen among the objects for each cluster.

To minimize the sum of square distance in an efficient way, the authors in [169] propose the following approach. By considering $s^k_j = \sum_{i \in C_k} d(o^i, o^j)_{j \in J_k}$, $D_{J_k} = d(o^u, o^v)_{u \in J_k, v \in J_k}$ and $s_{k,J} = (s^k_{j_{k,1}}, ..., s^k_{j_{k,P}})^T$, then applying the method of Lagrange multipliers to find the

local minima, the problem is reduced to the equation:

$$\nabla L_k = s_{k,J} - |C_k| D_{J_k} \beta^k + \lambda \vec{1} = 0 \tag{5.5}$$

where $\vec{1} = (1, ..., 1)^T \in \mathbb{R}^P$. The vector β^k represents the nonzero components of the coefficients α^k. The problem for which we are looking for a local minimum can be represented by the linear system below:

$$\begin{bmatrix} |C_k| D_{J_k} & -\vec{1} \\ \vec{1}^T & 0 \end{bmatrix} \begin{bmatrix} \beta^k \\ \lambda \end{bmatrix} = \begin{bmatrix} s_{k,J} \\ 1 \end{bmatrix} \tag{5.6}$$

Using the sparse representation presented above, we obtain Algorithm 11.

Algorithm 11 Sparse Relational K-mean

Input: K, D
Output: Prototypes' coefficients α^k

Initialize the clusters with random affectations of the objects
while the convergence is not attained **do**
  for each cluster k **do**
    Select P support points among the object in C_k
    Compute D_{J_k} and $s_{k,j}$ from the support points
    Solve equation (5.6) to get β^k
    Compute $-\frac{1}{2} \alpha^{kT} D \ \alpha^k$
    Compute $(D\alpha^k)_i$ for all object o^i
  Assign each object o^i to it's closest prototype using Eq. 5.2

The most expensive operation is the computation of β in $\mathcal{O}(NKP + KP^3)$. Calculation of $(D\beta^k)_i$ is in $\mathcal{O}(NKP)$ and the computation of $-\frac{1}{2} \beta^{kT} D \ \beta^k$ cost $\mathcal{O}(P)$. Finally, the complexity of the assignment step is $\mathcal{O}(NK)$. The global complexity in computation time is linear in function of the number of objects, which is acceptable for the analysis of big data-sets. However, the memory usage is still in $\mathcal{O}(N^2)$, as the whole dissimilarity matrix (size $N \times N$) is needed to compute the distances in Eq. 5.2.

5.2.4 Summary

Existing approaches to enforce sparsity in relational K-means is effective but can be optimized. In particular, the use of a different set of support points for each cluster does not seems ideal as the whole dissimilarity matrix must be kept in memory. In addition, we are interested in incremental (and ultimately dynamic) clustering. The formalism described in this section does not allow a straightforward adaptation to dynamic data-sets.

In the next sections I will present the fixed support points solution chosen to decrease memory usage and processing time. Then I propose a new formalism, based on the Barycentric Coordinate system, to allows an incremental clustering of relational data.

5.3 A New Approach of Sparse Relational K-means

This section presents the new improved sparse relational K-means using a single set of support points for all the clusters.

5.3.1 Proposed Algorithms

The objective in the proposed relational prototype-based clustering is to compute a good representation of the clusters prototypes based on their distances to a unique set of support points instead of a set of support point for each cluster. In a representation space with a dimension d, we theorically only need $d+1$ non-aligned objects to determine the position of any point in the space using only distances. Therefore, a unique set of support points should be enough for a good representation of the prototypes. Of course, choosing $d+1$ support point requires the ability to compute the intrinsic dimension of the data from a similarity matrix (see for example [33]), but it is also possible to choose a fixed number of support points, depending on the available memory and processing time.

The main advantage of a unique set of support point S is that both the computation of the distance and the minimization problem can be simplified. In addition, the full similarity matrix is not required anymore, as we only need the distances between the N objects and the P support points.

Taking into account the reasoning presented in the previous section, the equation (5.7) is a simplified form of the minimization problem.

$$\frac{1}{|C_k|}s_k - D_S\beta^k + \frac{1}{|C_k|}\lambda\vec{1} = 0 \tag{5.7}$$

where $s_k = (s_1^k, ..., s_P^k)^T$ with $s_j^k = \sum_{i \in C_k} d(o^i, o^j)_{j \in S}$ and $D_S = d(o^u, o^v)_{u \in S, v \in S}$. We recall that $d(o^i, o^j)$ is the squared dissimilarity between two objects o^i and o^j.

By keeping in mind that the support points are not cluster dependent in this situation, the problem can be simplified to the linear system below:

$$\begin{bmatrix} D_S & -\vec{1} \\ \vec{1}^T & 0 \end{bmatrix} \begin{bmatrix} \beta^k \\ \frac{1}{|C_k|}\lambda \end{bmatrix} = \begin{bmatrix} \frac{1}{|C_k|}s_k \\ 1 \end{bmatrix} \tag{5.8}$$

The main difference with Eq. 5.6 is that D_S, the dissimilarity matrix between the support point is unique and defined only once at the beginning of the process.

In addition, the distance between prototypes and objects is now simplified into:

$$d(w^k, o^i) = \sum_{p \in S} \beta^k d(o^i, o^p) - \frac{1}{2} \beta^{kT} D_S \beta^k \tag{5.9}$$

where $\beta^k \in \mathbb{R}^P$ is the vector of P coefficients associated to prototype k. The full matrix D is no more needed, as well as the full coefficient vectors α^k.

Based on these optimization, Algorithm 12 now has a memory usage in $\mathcal{O}(NP)$.

Algorithm 12 Sparse Relational K-mean with Fixed Support Points

Input: K, P, the objects O and a dissimilarity function d
Output: Prototypes' coefficients β^k

```
Initialize the clusters with random affectations of the objects
Select P support points among the objects
Compute D_S
while the convergence is not attained do
    for each cluster k do
        Compute s_k
        Solve equation (5.8) to get β^k
        Compute -1/2 β^{kT} D_S β^k
        Compute Σ_{p∈S} β^k d(o^i, o^p) for all object o^i
    Assign each object o^i to its closest prototype using Eq. 5.9
```

Considering that solving the system (5.8) cost $\mathcal{O}(P^3)$ for each cluster k, a variation of the algorithm would consist into separating the system solving step, into inversion of the first term, which can be extracted from the loop considering it is k-independent, and a product of the resulted inverse with the right-side term, which will be kept inside the loop. This modification would replace the k solve operation (costing $\mathcal{O}(KP^3)$), with one inverse and k matrix multiplications (costing $\mathcal{O}(KP^{2.4})$ [51, 180, 119]). The system (5.8) will become:

$$\begin{bmatrix} \beta^k \\ \frac{1}{|C_k|}\lambda \end{bmatrix} = \begin{bmatrix} D_S & -\vec{1} \\ \vec{1}^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{|C_k|} s_k \\ 1 \end{bmatrix} \tag{5.10}$$

and the resulting algorithm would be as described in Algorithm 13.

In addition, the optimal number P of support points is usually lower in our approach (where $P = d + 1$) than in previous algorithms (where $P = K \times d$ [169]). The complexity has therefore been improved, especially when K is high.

Algorithm 13 Optimized Sparse Relational K-mean with Fixed Support Points

Input: K, P, the objects O and a dissimilarity function d
Output: Prototypes' coefficients β^k

Initialize the clusters with random affectations of the objects
Select P support points among the objects
Compute D_S
Compute the inverse used in (5.10)
while the convergence is not attained **do**
    for each cluster k **do**
        Compute s_k
        Compute the product (5.10) to get β^k
        Compute $-\frac{1}{2}\beta^{kT} D_S \ \beta^k$
        Compute $\sum_{p \in S} \beta^k d(o^i, o^p)$ for all object o^i
    Assign each object o^i to its closest prototype using Eq. 5.9

5.3.2 Experimental Validation

5.3.2.1 General Setting

For the experiments, we used some representative vector data-sets, to allow us a comparison with vector K-means. The objects are injected into the relational algorithms after the calculation of corresponding similarity matrices using Euclidean distance. While selecting random support points can give good results in practice, we need to minimize the probability of getting aligned points within the chosen observations. Here, we propose to use the same principle as K-means++ initialization. We select the first support point randomly, the second representative being the furthest one from the first. We continue by pinning down, at each iteration, the closest points from the previous support points, then we select the furthest point selected from this set. Preliminary tests have shown that this initialization increase the quality of the clustering for relational algorithms. We use this initialization for all of the algorithms tested here requiring support points. The coefficients, however, must be initialized randomly.

Considering the conclusion of [169] that sparse algorithms are generally more effective than their naive counterpart, we will analyze the differences between the proposed approaches and the previously proposed sparse implementations, in addition to the regular vectorial K-means. Thus, we compare the overall performance of algorithms 12 and 13 (the sparse relational K-means with fixed support point studied) with the following algorithms:

- Regular K-means for vector data (Algorithm 9)
- Sparse K-means without support points (Algorithm 10)
- Sparse K-means with cluster-specific support points (Algorithm 11)

All the algorithms are implemented using python 2.7.11. Mathematical computations are performed using the python library NumPy 1.10.4, in addition to the module *linalg* from SciPy package for matrix inversion and solving.

We evaluate the clustering results using two external quality indexes, Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI), and the internal Silhouette Coefficient. See Chapter 2 for a description of the quality indexes.

The results reported for processing time and index values are the means of twenty iteration of the algorithms, with a new initialization for each run. It is worth noting that, for each specific iteration, the initialization is common for all the algorithms tested. See Chapter 3 for a description of the experimental data-sets.

5.3.2.2 Data-sets Description and Algorithms Notation

The data-sets used, while not sufficiently massive to simulate real life problems, should provide enough insights in order to examine an evolution for real problems. For the results reported next, we use the abbreviations defined in table 5.1 for each data-set, and the names in table 5.2 for algorithms names. The description of these data-sets is presented in chapter 3.

Table 5.1: Data-sets abbreviations.

Artificial convex data-set	Convex
Iris data-set	Iris
Digits data-set	Digits
Glass data-set	Glass
Wine data-set	Wine

Table 5.2: Algorithms notation.

K-means	K-means
Relational K-means	Rel-KM
Sparse Relational K-means	S-Rel-KM
Sparse Relational K-means with fixed support points	Sparse KM 1
Optimized Sparse Relational K-means with fixed support points	Sparse KM 2

5.3.2.3 Complexity Summary

Table 5.3 shows the complexity of the different algorithms. Relational approaches are usually more complex than vectorial approaches, but among them KM2 is the less complex. In addition, in KM1 and KM2, P is around K time smaller than in the other relational algorithms.

Table 5.3: Computational complexity.

Algorithm	Complexity
K-means	O(NKD)
Rel-KM	O(N^2)
S-Rel-KM	O($NKP + KP^3$)
Sparse KM 1	O($NKP + KP^3$)
Sparse KM 2	O($NKP + KP^{2.4}$)

The fixed cluster representatives studied in this chapter yields promising theoretical optimization. The following part is dedicated to practical experimentation and discussion of the results using a python implementation of the solution.

We computed the processing time, memory consumption and clustering quality for each algorithm and each data-set.

5.3.2.4 Processing Time

Table 5.4 presents time processing for different algorithms and different data-sets. The time usage for Sparse KM 2 (Optimized sparse relational K-means with fixed support points) is the best for convex and Iris data-sets and for the others data-sets is not the best but stays comparative. Each algorithm was implemented in a different function, but the prototypes initialization is common for all the functions and is not included in the duration computed. The time needed for each method/function to complete the learning was calculated using the package *timeit*: we start a timer exactly before running an algorithm script, and we stop it after saving the return results into a variable.
The computations are tested on a Windows 7(x64) machine, with a dual-core CPU clocked at 2.50Ghz (i5-2450M). The program is not multi-threaded.

Table 5.4: Processing time (seconds) for each algorithm and data-set.

Algorithm	Convex	Iris	Digit	Glass	Wine
K-means	475.20	3.98	39.52	4.62	3.78
Rel-KM	119321.45	59.50	3971.75	53.80	38.25
S-Rel-KM	353.93	5.43	392.40	13.26	10.95
Sparse KM 1	409.25	3.80	340.16	9.03	8.98
Sparse KM 2	238.80	3.58	351.54	8.41	8.40

5.3.2.5 Memory Usage

For memory monitoring (Table 5.5), we used python package *memory_profiler*. It provides us with a line-by-line analysis of memory usage for the designated function. Consider-

ing that python memory management is OS dependent, and cannot always be predicted as it does not always release freed memory, we decided to launch each script independently from the other, which mean that unlike the other measures reported, memory usage numbers are based on a different initialization, and are the mean of 5 executions.

Table 5.5: Memory usage results.

Algorithm	Convex	Iris	Digit	Glass	Wine
K-means	0.0080	0.0940	0.0136	0.0088	0.4950
Rel-KM	0.0160	0.1680	0.0238	0.0176	2.1234
S-Rel-KM	0.4220	1.9148	0.5164	0.6898	4.1706
Sparse KM 1	0.3360	0.8386	0.4626	0.4432	2.8736
Sparse KM 2	0.3224	0.9472	0.4642	0.4760	3.1558

5.3.2.6 Quality

ARI [160] is an external measure of the similarity between two data clustering, by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clustering. The adjusted Rand index have a value close to 0.0 for random labeling and exactly 1.0 when the clustering results and the labels are identical. Experimental results are presented in Table 5.6.

Table 5.6: ARI index for each algorithm and each data-set.

Algorithm	Convex	Iris	Digit	Glass	Wine
K-means	0.9991	0.7268	0.5718	0.4650	0.3566
Rel-KM	0.9990	0.7063	0.3986	0.4504	0.3295
S-Rel-KM	0.9994	0.7412	0.5720	0.4526	0.3425
Sparse KM 1	0.9995	0.7391	0.5711	0.4706	0.3473
Sparse KM 2	0.9995	0.7392	0.5692	0.4672	0.3523

NMI [183] is an external clustering quality index, that quantify the mutual information between two clustering, which is a measure of the similarity between two labels of the same data. The results are then normalized to scale the results between 0 (no mutual information) and 1 (perfect correlation). The experimental quality indexes are presented in Table 5.7.

Silhouette [101] is an internal method of interpretation and validation of consistency within clusters of data. Silhouette coefficients are measures of how similar a sample is to its own cluster (cohesion) compared to other clusters (separation). The mean coefficient of all samples is then computed. The silhouette ranges from -1 to 1, where a high value indicates that the observation is well matched to its own cluster and poorly matched to neighboring clusters. Table 5.8 shows the experimental values.

Table 5.7: NMI Index for each algorithm and each data-set.

Algorithm	Convex	Iris	Digit	Glass	Wine
K-means	0.9970	0.7541	0.7059	0.6997	0.4253
Rel-KM	0.9964	0.7660	0.5626	0.6796	0.4046
S-Rel-KM	0.9978	0.7787	0.7075	0.6840	0.4233
Sparse KM 1	0.9983	0.7626	0.7070	0.7039	0.4178
Sparse KM 2	0.9982	0.7677	0.7064	0.6976	0.4223

Table 5.8: Silhouette Index for each algorithm and each data-set.

Algorithm	Convex	Iris	Digit	Glass	Wine
K-means	0.7914	0.5522	0.1768	0.5185	0.5625
Rel-KM	0.7914	0.5431	0.1248	0.5006	0.5646
S-Rel-KM	0.7913	0.5472	0.1727	0.5115	0.5702
Sparse KM 1	0.7912	0.5448	0.1777	0.5154	0.5552
Sparse KM 2	0.7912	0.5464	0.1773	0.5146	0.5589

5.3.2.7 Impact of the Number of Support Points

Our experience with real data-sets show that with random *dimension*+1 representatives initialization, the sparse K-means with shared support points converge in an average of 6,25 iterations.

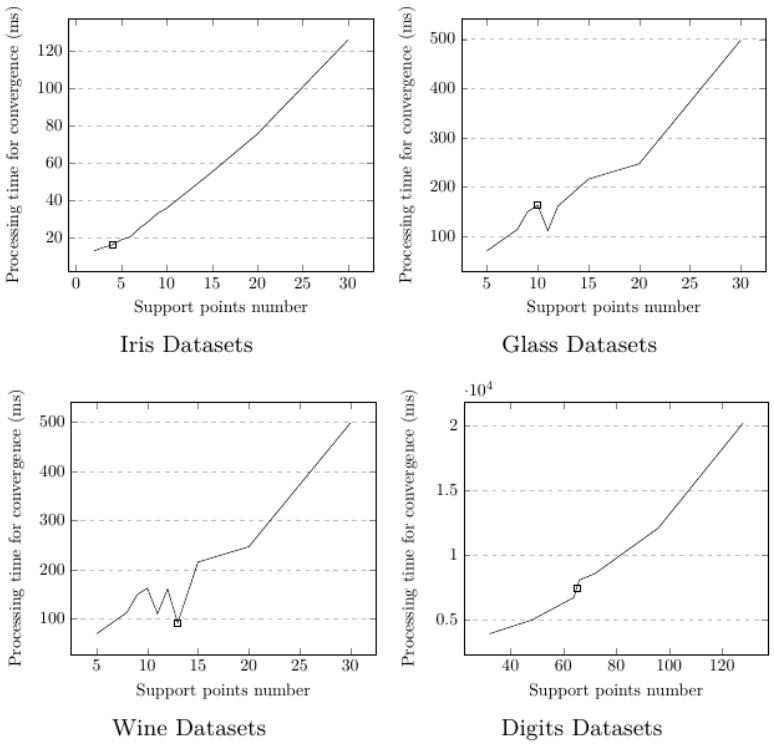


Figure 5.1: Effect of the number of support points on the processing time.

Meanwhile, an initial setting inspired from K-means++ of representatives converge in 5,05 iterations (mean over 20 runs). When changing the number of support points, we notice a small improvement in clustering quality before a decrease when the number is too high (figure 5.2). There is also an increase in the time needed for convergence (figure 5.1). We notice that a good compromise between quality and time complexity is when the number of representatives is equal to $d + 1$ of each data-set (represented by a square in 5.2 and in 5.1).

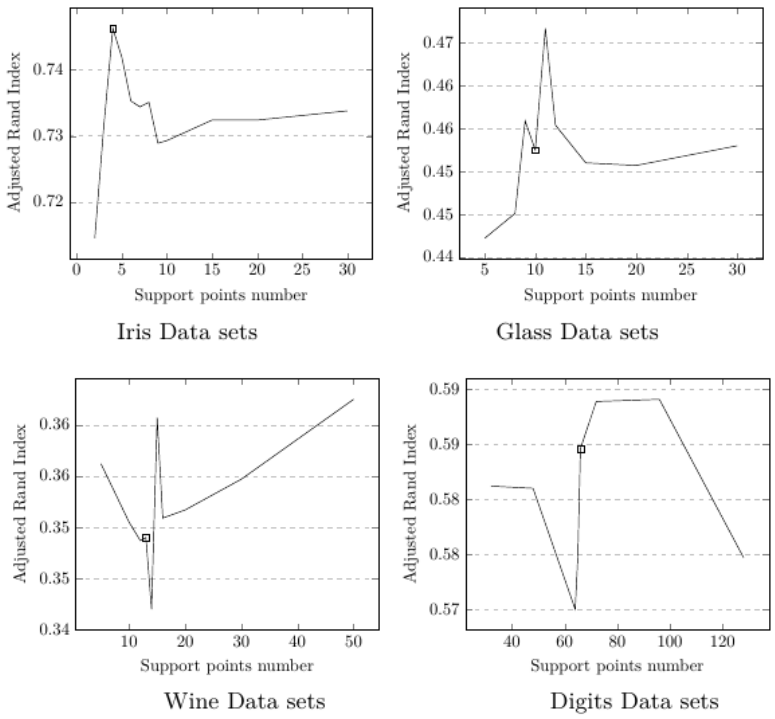


Figure 5.2: Effect of the number of support points on the Adjusted Rand Index.

To summarize, regular sparsity implementation in relational K-means consume a minimal amount of memory, but the processing duration are extremely high, which render this option unusable. The sparse algorithm with distinct support points is time-effective, but the results showed that the computation time can be improved with the usage of fixed support points as we propose here. In addition, the former is more expensive on the memory side by a difference ranging from 4% (Iris data-set), to 33% (Glass data-set), comparing with the latter. The clustering quality is similar for both of them and with the vectorial K-means (see figures 5.3 and 5.4 for an example).

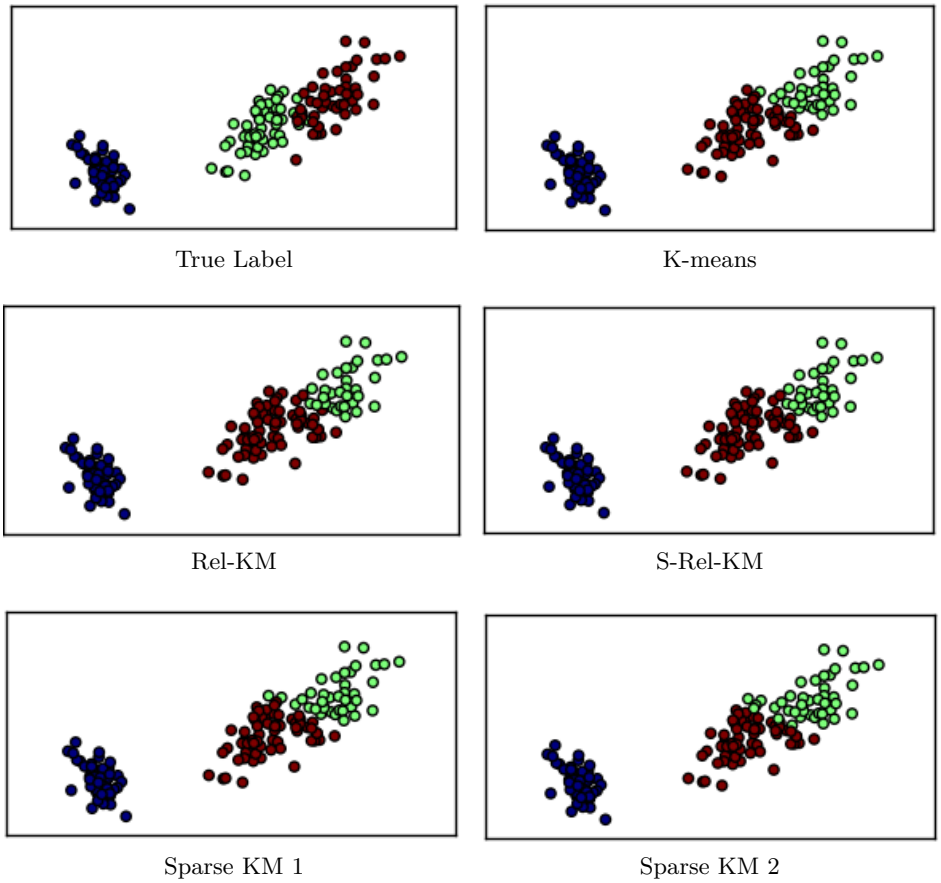


Figure 5.3: Representation of clustering results for Iris data-set (using MDS).

5.3.2.8 Discussion

In our experiments, the external quality indexes ARI (table 5.6) and NMI (table 5.7) are on the same levels for the different versions of K-means tested. More specifically, that means our sparse K-means implementation yields as good results as vectorial K-means and the other relational approaches. In the same manner, the internal quality index Silhouette (table 5.8) shows that, aside from Digit data-set, the clusters created are reasonably structured, and again the results are comparable with K-means and the sparse relational algorithm.

Concerning processing cost, as predicted with the complexity values already mentioned, the relational variations of K-means can be more expensive time-wise than the vectorial version, although it is not always the case. Our algorithms (sparse KM1 and especially sparse KM2) achieve good performances in comparisons to the other relational algorithms. Rel-KM, in particular, can be very slow, especially when the size of the data-set increase. One can note that the cost shows a polynomial increase with the dimension of the data-set.

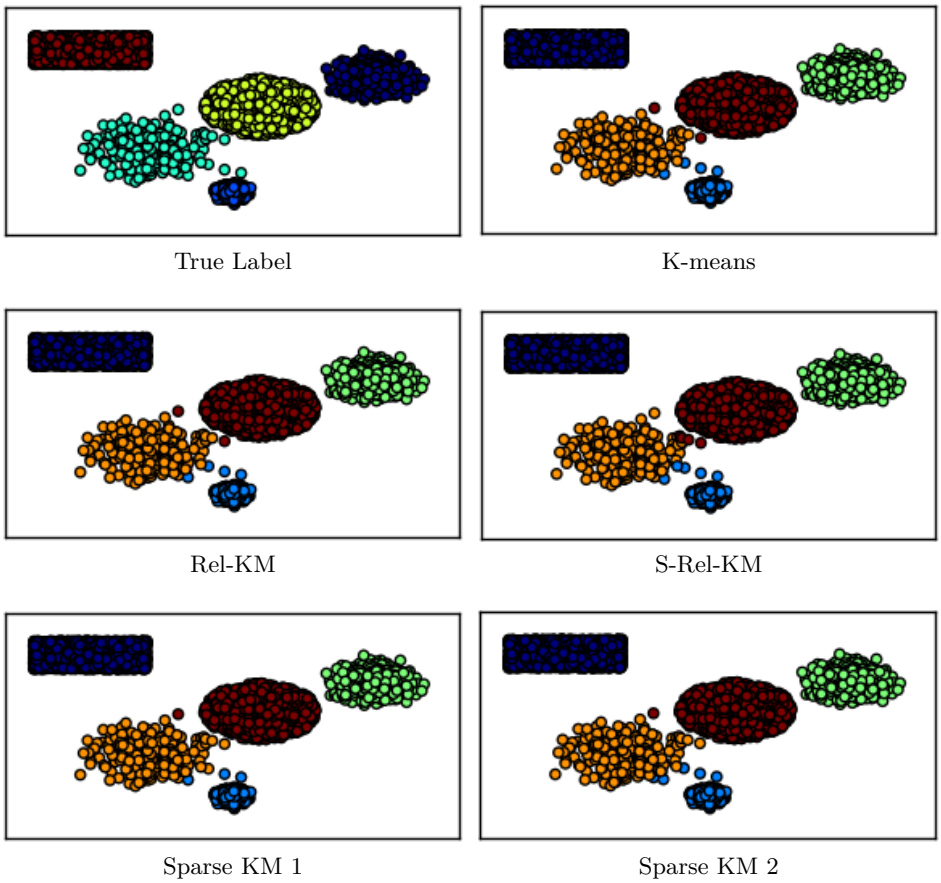


Figure 5.4: Representation of clustering results for the artificial convex data-set.

That can be explained by the decision to select $dimension + 1$ support points to represent the prototypes.

As far as memory usage is concerned, we notice at first sight that the vectorial version is less demanding than the relational ones, which can be explained by the storage of pairwise distances matrix. For the relational approaches, the less memory consuming is the Rel-KM, but at the expense of time processing, as shown earlier. Among the remaining algorithms, Sparse KM1 and even more Sparse KM2 are clearly less memory consuming: we observe a noticeable memory gain when using common support points over cluster-dedicated representatives.

5.3.3 Summary

The sparse relational K-means with fixed representatives is a good candidate to optimize memory usage and processing time for relational data. However, it is not adapted to incremental and dynamic data. We therefore introduce in the next section the Barycentric

Coordinate formalism, in order to unify the representation of the objects and the prototypes and allow a simple incremental learning process for relational clustering.

5.4 Relational Clustering Based on Barycentric Coordinate System

In this section, we present an approach to relational clustering based on the Barycentric Coordinate system [91] to homogenize the representation of both the objects and the prototypes. contrary sparse relational K-means, the relational clustering based on Barycentric Coordinates can deal with scalable data-sets.

5.4.1 Barycentric Coordinate System

In the Barycentric Coordinate system [91], the representation space is defined by a unique set of P support points chosen among the objects O. These support points can be any objects chosen randomly from O and represent a virtual space of dimension $P-1$. In the rare cases when co-linear points are chosen, the actual dimension of the space can be lower. Let $\mathcal{I} \subset \{1,\ldots,N\}$ be an index finite subset, with $P = |\mathcal{I}| \ll N$. We define the set of support points $O_S = o^I, i \in \mathcal{I} \subset O$ associated to an unknown representation in X by $X_S = \{s^i; s^i = x^i, i \in \mathcal{I}\} \subset X$. We aim at representing each cluster k by a prototype w^k associated to a representation μ^k in X. The prototype μ^k of cluster k is defined as a normalized linear combination of X_S (the support points):

$$\mu^k = \sum_{p=1}^{P} \beta_p^k \cdot s^p, \text{ where } \beta^k = (\beta_1^k, \ldots, \beta_p^k)^T \in \mathbb{R}^p, \text{ with} \sum_{p=1}^{P} \beta_p^k = 1. \tag{5.11}$$

This is also the definition of the Barycentric coordinate of an object in the space defined by the support points. In other words, β^k are the Barycentric coordinates of μ^k with respect to the system of support points X_S. Any object o in the database can also be defined using Barycentric coordinates: $o^i = \sum_{p=1}^{P} \beta_p^i s^p$ with the coordinates β^i satisfying $\sum_{p=1}^{P} \beta_p^i = 1$.

In order to evaluate the distance between an object o^i and a prototype w^k, we use the displacement from o^i to w^k and, by using $\sum_{p=1}^{P}(\beta_p^i - \beta_p^k) = 1-1 = 0$, we obtain the following distance:

$$d^2(o^i, w^k) = -\frac{1}{2}(\beta^i - \beta^k)^T \cdot D_S \cdot (\beta^i - \beta^k), \tag{5.12}$$

where $D_S = (d(o^i, o^j))_{i,j\in\mathcal{I}}$ is the dissimilarity matrix between objects corresponding to $\mathcal{I}$, the index set of the support points: it is the dissimilarity matrix between the support points.

We have by hypothesis all the dissimilarities $d(s^i, s^j)$ between each pair of support point. However, in order to compute the distance described in equation (5.12), we need to compute the Barycentric coordinates of the objects in O. In order to obtain the coordinates β^i of an object o^i, with respect to the system of support points O_S, we consider the following $P \times P$

matrix: $A = (A_{i,j})_{1 \leq i,j \leq P}$. A contains the differences between the pairwise dissimilarity of all the support points and the first support point only. We also consider J^i the $P \times 1$ vector of dissimilarity between an object o^i and the support points X_S. More precisely $J^i = (J^i_p)_{1 \leq p \leq P}$ and $J^i_p = d(o,^i s^1) - d(o^i, s^{p+1})$ for $1 \leq i \leq P-1$ and $J^i_P = 1$.

$$A = \begin{pmatrix} d(s^1,s^1)\text{-}d(s^2,s^1) & \dots & d(s^1,s^P)\text{-}d(s^2,s^P) \\ \cdot & \dots & \cdot \\ \cdot & \dots & \cdot \\ \cdot & \dots & \cdot \\ d(s^1,s^1)\text{-}d(s^P,s^1) & \dots & d(s^1,s^P)\text{-}d(s^P,s^P) \\ \\ 1 & \dots & 1 \end{pmatrix}, \quad J^i = \begin{pmatrix} d(o^i,s^1)\text{-}d(o^i,s^2) \\ \cdot \\ \cdot \\ \cdot \\ d(o^i,s^1)\text{-}d(o^i,s^P) \\ 1 \end{pmatrix}. \tag{5.13}$$

By using the symmetry of D_S, we obtain β^i as solution of the following linear system:

$$A \cdot \beta^i = J^i \Rightarrow \beta^i = A^{-1} \cdot J^i. \tag{5.14}$$

Note that the last equation of the system represents the normalized constraint $\sum_{p=1}^{P} \beta^i_p = 1$.

Therefore, we are able to compute the Barycentric coordinates β^i for each data o^i using (5.14). We thus are able to compute the distances between an object and a prototype from equation (5.12). The problem to optimize in order to find the coordinates of each prototype is still a minimization of the sum of square distances. We propose two algorithms to compute the prototypes' coordinates: a batch version, were the data-set is kept in memory during the whole learning process and a stochastic version where objects are presented one by one. The batch version is faster than the stochastic version, at the cost of an increase in memory usage.

5.4.2 Proposed Algorithms

5.4.2.1 Batch Version

The batch version supposes that the whole data-set can be stored in the memory, allowing to compute and store the Barycentric coordinates β^i of all objects. To minimize the sum of square distances between the objects and the prototypes, using equation (5.12), we compute the coordinates of the prototype w^k of the cluster k in the Barycentric coordinates system defined by O_S. Note that the Barycentric coordinates system is associative, allowing the Barycentric coordinates of a set of points to be computed iteratively. Therefore, the Barycentric coordinates of the prototype w^k, which is defined as the barycenter of all points in C_k, is equal to the average of the Barycentric coordinates of all points in C_k. The

Barycentric coordinates of the prototype of C_k are given by:

$$\beta^k = \frac{1}{|C_k|} \sum_{i|o^i \in C_k} \beta^i \tag{5.15}$$

Indeed, recall that β^k is defined as the solution of the minimization of the sum of square distances between objects and prototypes, which can be equivalently written by using (5.12) as follows:

$$\beta^k = \underset{\beta \in \mathbb{R}^P}{\operatorname{argmin}} \left(-\frac{1}{2} \sum_{i|o^i \in C_k} (\beta^i - \beta)^T \cdot D_S \cdot (\beta^i - \beta) \right), \tag{5.16}$$

As the optimization problem (5.16) is the finite sum of $|C_k|$ problem of the same type, we can permute minimization and summation. Therefore (5.16) is equivalent to the following:

$$\beta^k = \underset{\beta \in \mathbb{R}^P}{\operatorname{argmin}} \left(-\frac{1}{2} \left(\frac{1}{|C_k|} \sum_{i|o^i \in C_k} \beta^i - \beta \right)^T \cdot D_S \cdot \left(\frac{1}{|C_k|} \sum_{i|o^i \in C_k} \beta^i - \beta \right) \right), \tag{5.17}$$

which give the solution (5.15). The resulting procedure is given in algorithm 14. Note that, in this algorithm, the inversion of matrix A is only done once (line 3 of the algorithm). In addition, the dissimilarity matrix D (size $N \times N$) is never computed, allowing a low computational complexity and a gain in memory.

The resulting algorithm is the following:

Algorithm 14 Batch Barycentric Relational Clustering

Input: objects O, function d, K, P.
Output: Prototypes' coordinates β^k.

1: Choose randomly P support points from O.
2: Compute β^i using (5.14) for each $o^i \in O$.
3: Choose randomly K coordinates β^i to initialise the β^k.
4: **while** the convergence is not attained **do**
5: Assign each object to its closest prototype using (5.12).
6: Update the μ^k using (5.15).

Note that, in this algorithm, the inversion of matrix A is only done once (line 4 of the algorithm). In addition, the dissimilarity matrix D (size $N \times N$) is never computed, allowing a low computational complexity and a gain in memory.

5.4.2.2 Stochastic Version

In this section, we propose a stochastic version of our algorithm. The idea of stochastic process is to present objects from the data-set one by one randomly. The update of the prototypes is computed incrementally for each object presented. Although stochastic ap-

proaches are usually slightly slower than the batch versions, they have the advantage of allowing a better management of the memory usage as it is not necessary to store the whole data-set in memory for each step of the process. Stochastic processes are also the basis of incremental, on-line and dynamic clustering.

The stochastic algorithm must update incrementally the barycentric coordinates of each prototype μ^k with respect to the support points X_S for each object presented to the system if this object belongs to cluster k. By construction, $\mu^k = \sum_{p=1}^{P} \beta_p^k s^p$ with $\sum_{p=1}^{P} \beta_p^k = 1$, i.e. the prototype are a linear combination of $X_S = \{s^i; s^i = o^i, i \in \mathcal{I}\} \subset X$. From the Barycentric representation theory, it is known that the barycenter of a set of points can be computed iteratively, due to its associativity. Therefore, for a new object o^i belonging to the cluster k, the barycenter of $C_k \cup \{o^i\}$, defined by β^k, is equal to the weighted barycenter of the object o^i and the barycenter of C_k, i.e. the prototype μ^k.

As we can compute the barycentric coordinates of o^i in terms of the support points O_S (see equation (5.14)), the update rule of β^k can be written as: ${\beta^k}_{t+1} = (1-\gamma){\beta^k}_t + \gamma\beta^i$, where γ is the weight (or learning rate) defining the importance of o^i in the new Barycentric coordinates. From the last equation we have:

$${\beta^k}_{t+1} = \beta_t^k - \gamma(\beta^i - \beta_t^k). \tag{5.18}$$

The resulting algorithm is the following:

Algorithm 15 Stochastic Barycentric Relational Clustering

Input: objects O, function d, K, P, γ.
Output: Prototypes' coordinates β^k.

1: Choose randomly P support points from O.
2: Compute matrix A using (5.13).
3: **while** the convergence is not attained **do**
4: Select o^i from O randomly.
5: Compute β^i from (5.14).
6: Assign o^i to its closest prototype μ^{k^*} using (5.12).
7: Update μ^{k^*} by computing β^{k^*} using (5.18).

Note that again the inversion of matrix A is only done once and the computation of the dissimilarity matrix D is not needed. In addition, there is no need to store all the data in memory at all time, as each object is treated separately.

5.4.2.3 Complexity

We compare here the theoretical computational and memory complexity of the proposed approach to other algorithms adapted to relational data. We are only interested here in the complexity relative to the number of objects in the data-set, as the number of prototypes or

support points is usually negligible in comparison. K-Medoids (KMed) [109], Relational K-Means (Rel-KM) [90], HBDSCAN [37] and Affinity propagation (Affinity) [65] have usually a time complexity of $\mathcal{O}(N^2)$, with N the number of objects. Hierarchical Ascendant Clustering (HAC) [133] is known to have a complexity of $\mathcal{O}(N^2 Log(N))$, whereas the spectral clustering approach (Spectral) [174] is in $\mathcal{O}(N^3)$. Some optimizations and approximations have been proposed to reduce the complexity of these algorithms (down to $\mathcal{O}(Nlog(N))$ depending on the data's structure). The time complexity of the Sparse Relational K-Means (S-Rel-KM) is $\mathcal{O}(N)$ after the computation of the dissimilarity matrix (which is $\mathcal{O}(N(N-1)/2)$). The memory complexity of these approaches is usually $\mathcal{O}(N^2)$, as the dissimilarity matrix is needed to be kept in memory, although it can be possible in some conditions to reduce it to $\mathcal{O}(N)$ for HDBSCAN and HAC.

In comparison, the proposed Barycentric Clustering approaches, batch (BC-batch) and stochastic (BC-stoch) versions, have a significantly lower time complexity of $\mathcal{O}(N)$ independently of the data structure, especially as the dissimilarity matrix does not need to be computed. The batch version need to have the objects in memory ($\mathcal{O}(N)$), whereas the memory consumption of the stochastic version is independent of the number of objects, as they can be stored and released one by one during the process.

5.4.3 Experimental Validation

In this section we present the experimental protocol we used to evaluate the proposed algorithm. The algorithm is compared to seven state-of-the-art clustering algorithms adapted to dissimilarity matrices (see sections 2). The approaches have been tested on 13 real and artificial data-sets of various types. The quality of each algorithm has been tested using two quality indexes. We discuss the experimental results obtained by our algorithm in comparison to different approaches in terms of computation time and quality. All algorithms are implemented using the *scikit-learn* library on python 2.7 [191].

5.4.3.1 Database Description

Table 5.9 presents a summary of the experimental data (for more details please see chapter 3). Note that the sizes of the data-sets are kept relatively small because of the time and memory complexity of some of the tested algorithms .

In this work, we used external and internal quality indexes. External indexes provide a measure of similarity between the cluster assignment proposed by the algorithms and the "true" labels, when they are known [59]. Internal indexes are used to measure the goodness of a clustering structure without external information [185]. Here we evaluate the clustering results using an external quality index, the Normalized Mutual Information score (NMI) [182], and the internal Silhouette Coefficient [103].

Table 5.9: Description of the experimental data-sets.

Data-sets	# Objects	Type	# Classes
Art1	10000	Vector (artificial)	4
Art2	1000	Vector (artificial)	4
Art3	800	Vector (artificial)	5
Iris	150	Vector (real)	3
Digits	1797	Vector (real)	10
Wine	178	Vector (real)	3
Prot1	115	Sequence	3
Prot4	129	Sequence	5
Prot5	98	Sequence	3
Hist-mean	1000	Distribution	5
Hist-shape	1000	Distribution	5
Hist-std	1000	Distribution	5
People	300	Concept	3

5.4.3.2 Computational Time

In table 5.10 we compared the computation time of the proposed algorithm (10 support points S) with other algorithms for different data-sets. In addition, to study experimentally the effect of the number of observations on the computational time and the memory usage, we generated Gaussian data with 10 clusters and 10 dimensions. We increased gradually the number of observations to observe the increase of computational time (table 5.11 and figure 5.5). Note that only the batch version of the proposed approach is presented here, as the computational time of the stochastic version is exactly proportional (around 2 times slower in our implementation). We used 10 support points for our approach.

Table 5.10: Computation time (seconds) for each data-set and different algorithms.

	BC-batch (S=10)	HAC	Affinity	HDBSCAN	Spectral	KMed	Rel-KM	S-Rel-KM
ART1	0.10	0.02	0.32	0.93	0.16	0.01	0.06	0.22
ART2	0.12	0.02	0.32	1.06	0.14	0.01	0.06	0.21
ART3	0.18	0.02	0.44	1.02	0.15	0.01	0.16	0.36
Digits	0.40	0.06	1.30	4.99	0.59	0.02	0.47	0.69
Prot1	0.01	0.00	0.05	0.01	0.01	0.00	0.00	0.02
Prot2	0.00	0.00	0.03	0.00	0.01	0.00	0.00	0.01
Prot3	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.01
Prot5	0.02	0.00	0.02	0.01	0.02	0.00	0.01	0.03
Prot6	0.01	0.00	0.01	0.00	0.01	0.00	0.00	0.01
Prot7	0.01	0.00	0.01	0.00	0.01	0.00	0.00	0.02
hist_mean	0.12	0.02	0.32	0.89	0.13	0.01	0.06	0.22
hist_shape	0.14	0.02	0.40	0.89	0.16	0.01	0.11	0.31
hist_std	0.11	0.02	0.46	0.90	0.13	0.01	0.08	0.25
Iris	0.01	0.00	0.01	0.01	0.02	0.00	0.01	0.03
People	0.02	0.00	0.02	0.13	0.03	0.00	0.01	0.06
Wine	0.02	0.00	0.01	0.02	0.02	0.00	0.01	0.04

Table 5.11: Computation time in second for each algorithm with respect to the data-sets size.

Size	500	1,000	2,000	5,000	10,000	20,000	40,000	50,000	100,000	1,000,000	10,000,000
Affinity	0.09	0.34	1.45	9.95	33.75	-	-	-	-	-	-
Spectral	0.05	0.16	0.56	3.91	19.44	-	-	-	-	-	-
HAC	0.00	0.03	0.13	0.80	3.16	13.55	-	-	-	-	-
HDBSCAN	0.08	0.11	0.30	1.52	6.17	29.66	-	-	-	-	-
KMed	0.00	0.03	0.06	0.44	1.97	7.66	35.32	-	-	-	-
Rel-KM	0.06	0.16	0.36	1.33	4.02	14.22	54.01	-	-	-	-
S-Rel-KM	0.14	0.30	0.61	1.70	4.02	11.45	37.10	-	-	-	-
BC-batch	0.20	0.34	0.61	1.45	2.98	5.45	12.49	13.95	27.80	274.86	2,763.08

In table 5.11, "-" means that the algorithm requires more than 16GB of RAM for the process to finish. As expected, the computational time of the proposed algorithm increases much slower than for the other approaches. One can note that the K-medoid algorithm is very fast in comparison to our algorithm when the number of observations is low, but with the increase of the number of observations the computational time increase much faster than in our approach. The proposed algorithm can deal with massive data-sets for a very reasonable time cost and memory consumption; it is not the case for the other algorithms.

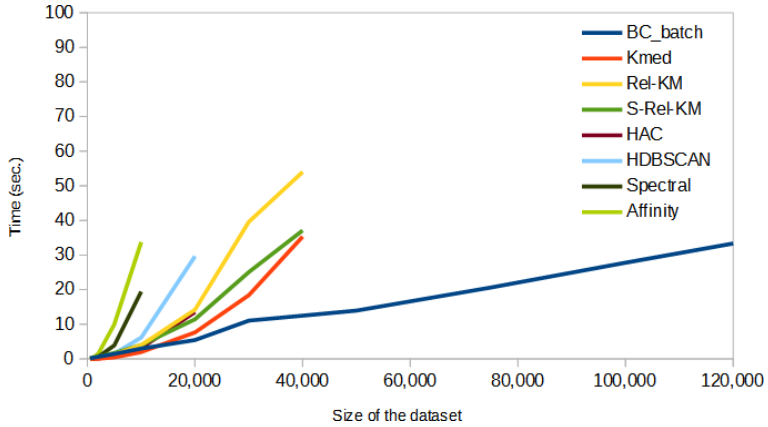


Figure 5.5: Effect of the number of objects on the computing time for different algorithms.

Although the low complexity of our approach is a great advantage for real application, it is important to check if the quality of the resulting clustering is not significantly lower than the quality of the competitors.

5.4.3.3 Quality

In table 5.12 and 5.13 we examined our algorithm in batch and stochastic version with NMI and Silhouette quality indexes (see Chapter 2) and compare it to different algorithms.

We also compared the efficiency of the different algorithms based on a statistical test (Figures 5.6 and 5.7). The goal is to determine if there is enough evidence to reject the

Table 5.12: Values of the NMI score index for each data-set and different algorithms.

	HAC	Affinity	HDBSCAN	Spectral	KMed	Rel-KM	S-Rel-KM	BC-batch	BC-stoch
Art1	0.86	0.56	0.84	0.97	0.97	0.97	0.97	0.97	0.97
Art2	1.00	0.74	1.00	0.89	1.00	1.00	1.00	1.00	1.00
Art3	0.94	0.63	0.98	0.80	0.87	0.83	0.86	0.92	0.92
Iris	0.74	0.47	0.76	0.75	0.79	0.76	0.80	0.80	0.80
Digits	0.04	0.48	0.70	0.69	0.64	0.73	0.70	0.69	0.65
Wine	0.11	0.16	0.09	0.41	0.42	0.39	0.32	0.40	0.41
Prot1	0.96	0.50	0.90	0.96	0.96	0.96	0.96	0.96	0.96
Prot2	0.86	0.77	0.86	0.86	0.86	0.86	0.86	0.81	0.86
Prot3	0.92	0.38	0.78	0.92	0.92	0.92	0.92	0.92	0.92
Hist-mean	1.00	0.60	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Hist-shape	0.03	0.19	0.38	0.73	0.76	0.75	0.70	0.77	0.80
Hist-std	1.00	0.46	1.00	1.00	1.00	1.00	1.00	1.00	1.00
People	0.02	0.00	0.64	0.57	0.79	0.70	0.70	0.91	0.94

null hypothesis, which suggests that all algorithms have the same performance. First, we used the non-parametric Friedman test [66]. This test is based on the global comparison of several algorithms with respect to several data-sets. Then we use the Nemenyi test [148] to compare each pair of algorithms. This test is based on the computation of a statistic on the difference between the average rankings of the algorithms used. The performances of two algorithms compared by this test are significantly different if the difference between their average ranking is greater than or equal to a so-called critical distance (CD). These tests make it possible to have critical diagram representing an ordered ranking of the different algorithms according to their performances. A critical diagram represents a projection of average ranks algorithms on enumerated axis. The algorithms are ordered from left (the best) to right (the worst) and a thick line which connects the classifiers were the average ranks not significantly different (for the level of 5% significance).

Table 5.13: Values of the Silhouette index for each data-set and different algorithms.

	HAC	Affinity	HDBSCAN	Spectral	KMed	Rel-KM	S-Rel-KM	BC-batch	BC-stoch
Art1	0.56	0.43	0.56	0.56	0.56	0.56	0.56	0.56	0.56
Art2	0.79	0.38	0.77	0.69	0.79	0.79	0.79	0.79	0.79
Art3	0.66	0.42	0.59	0.60	0.63	0.64	0.63	0.65	0.63
Iris	0.51	0.55	0.69	0.55	0.55	0.55	0.56	0.56	0.56
Digits	-0.15	0.01	0.05	0.15	0.15	0.19	0.17	0.16	0.17
Wine	0.33	0.51	0.46	0.52	0.52	0.52	0.51	0.52	0.52
Prot1	0.96	-0.05	0.91	0.96	0.96	0.96	0.96	0.96	0.96
Prot2	0.95	0.71	0.58	0.95	0.95	0.95	0.95	0.83	0.95
Prot3	0.99	-0.07	0.80	0.99	0.99	0.99	0.99	0.99	0.99
Hist-mean	0.64	0.08	0.64	0.64	0.64	0.64	0.64	0.64	0.64
Hist-shape	0.15	0.54	0.44	0.23	0.25	0.28	0.27	0.25	0.26
Hist-std	0.65	0.15	0.65	0.65	0.65	0.65	0.65	0.65	0.65
People	0.32	0.00	0.30	0.25	0.38	0.38	0.38	0.27	0.25

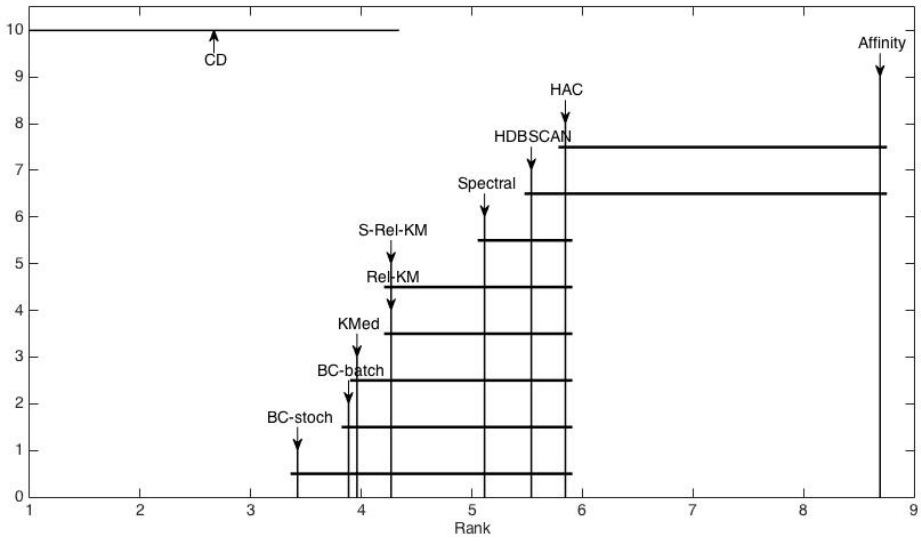


Figure 5.6: Statistical comparison on the NMI scores for the different algorithms.

In this experience, we also used 10 support points. The parameters' values were optimized experimentally for all of the algorithms, but we used a unique set of values for all of the data-sets. The results demonstrate the quality of the proposed approaches in comparison to the state-of-the-art algorithms. The internal and external qualities of our algorithms are, most of the time, at least as good as the competitors on the experimental data-sets. The batch and stochastic versions are very similar in quality.

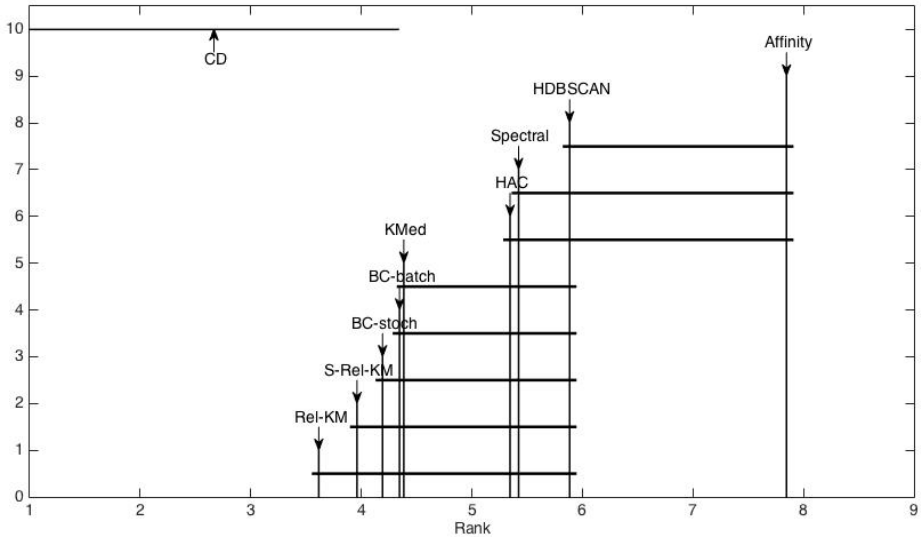


Figure 5.7: Statistical comparison on the Silhouette scores for the different algorithms.

The results of the statistical analysis confirm these conclusions. The quality of the proposed approaches are in general better than the other algorithms for the considered

data-sets, but the differences are not significant. The exception being "affinity propagation", mostly because of its bad performances in some of the sequential data.

The proposed algorithms are therefore as good as other approaches when the size of the data-set is small enough to be deal with in a reasonable time and memory usage by all approaches. However, when the data-sets to analyze start to be too big for traditional approaches, the proposed algorithms seem to be the best candidates for the clustering of such data.

5.4.3.4 Effect of the Number of Support Points

Finally, we studied the effect of the number of support points on the quality of the proposed algorithm. Only results for the batch version are showed here, as the results are similar for both version. As shown in table 5.14 and 5.15, increasing the number of support points usually does not have a major effect on the results' internal and external quality.

Table 5.14: Values of the Normalized Mutual Information index for each data-set with different numbers of support points.

	ART1	**ART2**	**ART3**	**Iris**	**Digits**	**Wine**	**Prot1**	**Prot2**	**Prot3**	**Hist-mean**	**Hist-shape**	**Hist-std**	**People**
S=2	0.80	0.87	0.74	0.87	0.32	0.43	0.96	0.86	0.92	0.96	0.44	0.71	0.62
S=3	0.95	0.89	0.90	0.86	0.43	0.40	0.96	0.86	0.92	1.00	0.60	1.00	0.73
S=5	0.97	1.00	0.92	0.78	0.55	0.40	0.96	0.86	0.92	1.00	0.78	1.00	0.89
S=10	0.97	1.00	0.92	0.80	0.69	0.40	0.96	0.86	0.92	1.00	0.77	1.00	0.91
S=20	0.97	1.00	0.86	0.80	0.69	0.39	0.96	0.86	0.92	1.00	0.79	1.00	0.92
S=100	0.97	1.00	0.86	0.78	0.70	0.40	0.96	0.86	0.92	1.00	0.80	1.00	0.92

Table 5.15: Values of the Silhouette index for each data-set with different numbers of support points.

	ART1	**ART2**	**ART3**	**Iris**	**Digits**	**Wine**	**Prot1**	**Prot2**	**Prot3**	**Hist-mean**	**Hist-shape**	**Hist-std**	**People**
S=2	0.52	0.66	0.44	0.55	0.01	0.52	0.96	0.95	0.99	0.62	0.12	0.3	0.27
S=3	0.55	0.69	0.63	0.55	0.05	0.52	0.96	0.95	0.99	0.64	0.18	0.65	0.33
S=5	0.56	0.79	0.63	0.55	0.11	0.52	0.96	0.95	0.99	0.64	0.24	0.65	0.34
S=10	0.56	0.79	0.65	0.56	0.16	0.52	0.96	0.83	0.99	0.64	0.25	0.65	0.27
S=20	0.56	0.79	0.62	0.56	0.18	0.52	0.96	0.95	0.99	0.64	0.25	0.65	0.25
S=100	0.56	0.79	0.63	0.55	0.18	0.52	0.96	0.95	0.99	0.64	0.26	0.65	0.34

Indeed, with only one support point the quality is not better than random clustering but, starting from a small number of support points, increasing this number does not increase significantly the quality of the result. It means that our algorithm does not need a high number of support points to work; this reduce furthermore the complexity of the approach. In addition, the optimal number of support point can be much lower than the actual dimension of the data-set. "People", for example, has a representation with a thousand of dimensions (size of the bag of words) but 5 to 10 support points seem to be enough to achieve an optimal quality of clustering.

5.4.4 Summary

In this section, we proposed a new prototype-based clustering algorithm adapted to relational data-sets. The new approach is based on the Barycentric Coordinate formalism and presents a linear time and memory complexity without a significant loss in quality in comparison to the state-of-the-art algorithms.

We proposed a batch and a stochastic version of our approach. Although the stochastic version is slower than the batch, it has the advantage of allowing a better management of the memory usage, as it is not necessary to store the whole data-set in memory for each step of the process. Stochastic processes are also the basis of incremental, on-line and dynamic clustering.

5.5 What is the Optimal Number of Support Points and How to Choose Them?

In the approaches proposed in this chapter, it is necessary to first choose a number P of support points to represent the data space and then to select P objects in the data-set to be the support points. The choice of P and the selection of the support points is a challenge, as the quality of the results as well as the speed of the computation may be affected by these choices.

5.5.1 Selection of the Support Points

The role of the support points is to define a compact representation space based on the dissimilarities between said support points and the objects in the data-set. In a pseudo-Euclidean space of dimension d, each object o^i is perfectly defined by its distance to $d+1$ non-colinear objects in this space (the support points). If at least one object o^p is colinear to the others, it means that the distances between o^i and o^p can be expressed as a combination of the distances between o^i and the other support points. Therefore, o^p is not informative in this system and the actual dimension of the represented space is $d-1$.

In this chapter, to avoid co-linearity when choosing the support points, we proposed to use a strategy based on the centroids initialization in K-means++ [12]. The idea was to choose each support point as far as possible to the others. This should effectively minimize the risk of co-linearity, but we have no guaranties. In addition, when the number of support points is much smaller than the number of objects, the risk of co-linearity should be small. This is not really satisfying, but hopefully there is a mathematical way of being sure that the chosen support points are not co-linear: the Cayley-Menger determinants [141].

The Cayley-Menger determinants is defined as follow. Let Ds be a matrix of squared dissimilarities (in our case it is the matrix of pairwise squared dissimilarities between the P supports points).

The Cayley-Menger matrix CM_{D_S} is expressed as:

$$CM_{D_S} = \begin{bmatrix} D_S & \vec{1} \\ \vec{1}^T & 0 \end{bmatrix} \tag{5.19}$$

where $\vec{1}$ is a vector of 1 of length P.

The determinant of this matrix is null if at least one support point can be expressed as a combination of the others. A strategy would then be to add each candidate object to the set of support points one at a time and check each time if the resulting Cayley-Menger determinant is not null. If it becomes null, the candidate is rejected. This operation is repeated until we obtain the chosen number of support points. Remains the question of the choice of the candidate support points.

The choice of the support points can be seen as a dimensionality reduction problem. Indeed, the unknown representation space of the objects will be projected into the space defined by the supports points. If the number of support points is high enough, this projection will not necessarily decrease the intrinsic dimension of the space, but in general we choose a low number of support points for speed and memory efficiency. In that case, the choice of the support points is potentially very important to keep a good representation.

Dimensionality reduction techniques often use transformations to determine the intrinsic dimensionality of the data as well as extracting a new representation minimizing the distortion in the data representation. For this purpose, there are various related techniques, including linear and non-linear methods: in particular Principal Component Analysis (PCA) [27], Multi-Dimensional Scaling (MDS) [31], Random projections [26] or Auto-encoder networks [23]. Each have its advantage and disadvantage.

Principal Components Analysis (PCA)[27] is a linear technique and it performs dimensionality reduction by embedding the data into a linear subspace of lower dimensionality. PCA is stable, there are no additional parameters, and it is guaranteed always to converge to the same optima. However, the PCA approach is only valid for vectorial data-sets and is not adapted to the other type of data-sets. The Multi-Dimensional Scaling (MDS) [31] algorithm, on the other hand, aims to place each object in N in a lower dimensional space such that the between-object distances are preserved as well as possible. This approach works from a dissimilarity matrix and is adapted to relational data. However, its process is very complex ($\mathcal{O}(N^2)$) in time and memory and is not applicable when the number of objects increases. Autoencoders [23] are artificial neural networks used for unsupervised learning of efficient data representation. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction. The disadvantage is that it is difficult to optimize the weights in non-linear autoencoders that have multiple hidden layers. This algorithm is not fast and data-set should be big enough. In addition, each type of data should require a specific autoencoder structure to be efficient.

The most interesting approach in our case is the random projection approach [26]. Random projection is a technique in statistic and mathematics domains to reduce the dimensionality of a set of points which lie in Euclidean space. Random projection is a simple and computationally efficient way to reduce the dimensionality of data by trading a controlled amount of error for faster processing times and smaller model sizes. The reduction approximately preserves the pairwise distances between any two objects of the data-set. There are many advantages for random projection compared to others approaches. First, the random projection has a low complexity $\mathcal{O}(N)$. In addition, if the data-set is very large, it is not necessary to hold it in memory for a random projection, while for PCA or MDS it is compulsory. In addition, it has been shown that the quality of the representation is very close to the more complex approaches [26].

Random projections transform a matrix of vectorial data in order to obtain a new set of coordinates corresponding to randomly chosen axis in the representation space. Interestingly, this process is very similar to a random choice of non-colinear support points in our case. Indeed, choosing P support points is equivalent to define a set of $P-1$ axis, one of the support point acting as the origin of these axis. This strategy is therefore probably optimal in order to keep at the same time a low complexity and a good representation of the data. The main question is then to choose an acceptable number of support points according to the trade-off speed/quality.

5.5.2 Choice of the Number of Support Points

The choice of the number of support point is usually a tricky question. When the number of dimensions d of the initial representation space is known, $d+1$ is an obvious choice, but suffers from two limitations. First, d is often not known in relational data analysis. Secondly, the dimension of the initial representation space may be too big (it can potentially be infinite) for a reasonable computation time and memory load. We need a way of controlling the distortions in pairwise dissimilarities in function of the number of support points, in order to find a good trade-off between speed, memory and accuracy. The Johnson-Lindenstrauss Lemma is perfect for this task.

In [106], the authors presented a lemma, the Johnson-Lindenstrauss Lemma, bounding the distortion error of a projection of data from a big to a lower dimensionality space. [54] gave an elementary proof of the Johnson-Lindenstrauss Lemma. The lemma was vastly used in compressed sensing, manifold learning, dimensionality reduction, and graph embedding. In particular, essentially all the dimension reduction approaches via random projection rely on the Johnson-Lindenstrauss Lemma to justify the quality of the approach.

The goal of Johnson-Lindenstrauss Lemma is to demonstrate, for a representation X in a d-dimension space, that we can find a representation Q in a k-dimension space and a mapping f from X to Q such that the pairwise distances of X are preserved (approximately) under the mapping.

Definition 1 Given a set of N points in a d-dimension space and a projection f onto a random k-dimension linear subspace, a squared distance $\|i-j\|_2$ between two objects i and j is ϵ**-preserved** if, for any $0 \leq \epsilon \leq 1$:

$$(1-\epsilon)\|i-j\|_2 \leq \|f(i)-f(j)\|_2 \leq (1+\epsilon)\|i-j\|_2 \tag{5.20}$$

Theorem 1(Johnson-Lindenstrauss Lemma) Let X be the representation of N objects in a d-dimension space and let $\epsilon > 0$ be a parameter. Let Q be the projection of X onto a random $k-dimension$ linear subspace. If

$$k \geq \frac{4}{\mathbf{1/2\epsilon^2 - 1/3\epsilon^3}}.log(N) \tag{5.21}$$

then all pairwise distances in X are ϵ**-preserved** in Q with probability at least $\frac{1}{2}$.

Therefore, considering that the use of P support points randomly chosen is similar to a random projection in a $P-1$-dimensional space, Eq. 5.21 allows us to either choose a maximal error ϵ to obtain a number of support points $P = k+1$, or to choose a number of support points adapted to the available resources in memory and computation and check the error bound ϵ.

One can note that the choice of P with this approach is independent from the initial dimensionality d (often unknown) and only depend on the size of the data-set. It is also worth noting that the bound proposed by the lemma is very conservative. The actual experiments presented in this manuscript give very good results from much lower number of support point P than the value obtained using the lemma. Still, the Johnson-Lindenstrauss Lemma is a very useful tool to control the trade-off speed and memory vs. quality in our prototype-based relational approaches.

5.5.3 Summary

In this thesis, we deal with massive data-sets and data streams and we need fast and memory efficient algorithms. Choosing the right number of support points is a very important step. How to select these support points and how to choose their number are challenging questions. We showed in this section that a random selection of non-colinear support points is a good trade-off between efficiency and speed. In addition, the Johnson-Lindenstrauss Lemma assures that the pairwise distances are preserved within a controllable error. This error is independent from the initial representation space dimensionality and only depends on the number of support points. It is therefore possible to define an optimal number of support points according to a bounded error rate.

5.6 Conclusion

In this chapter we presented two new algorithms. First, we proposed an approach for sparse relational K-means using fixed support points. This algorithm optimizes the memory usage and computational time. However, this approach is not adapted to dynamic data-sets. To deal with this problem we then proposed an incremental approach of clustering for relational data-set. This approach uses Barycentric Coordinates formalism to define the prototypes.

The choice of the support points is a vital part of relational prototype-based algorithm. Different strategies and some theoretical arguments were discussed, leading to a random selection of non-colinear support points able to deal with large data-sets. We propose the use of the Johnson-Lindenstrauss Lemma to determine the number of support points. This approach has the advantage to be independent to the dimentionality of the initial representation space, which is usually unknown.

The algorithms have a linear time and memory complexity and are adapted to big data-sets. In addition, overlapping clusters is not problem in prototype-based approaches. However, in comparison to the algorithms proposed in Chapter 4, the number of cluster is a now parameter to choose, despite being rarely known in real cases. Finally, despite being incremental, it is not yet adapted to data streams.

We present in the next chapter an algorithm which is adapted to data stream clustering and does not need a fixed number of clusters: the number of cluster is adapted over time based on the dynamics of the data structure.

Chapter 6

Application to the Analysis of Internet User's Behavior

Table of Contents

6.1 Introduction

In this chapter, we present an application of the Barycentric Coordinate approaches proposed in Chapter 5, based the need of the Mindlytix company. Our practical motivation is to perform real-time profiling of connected users. Profiling tasks aim at recognizing the "mindset" of users through their navigation on various websites or their interaction with digital "touch points" (varying ways that a brand interacts and displays information to prospective customers and current customers). It intervenes in the international market for "programmatic advertising" tasks, by assigning a profile to users connecting to a site that can offer advertising, so that the displayed advertising corresponds best to the needs of the users. These profiles are calculated from a very large database of internet browsing which lists URL sequences or touch points visited by a large number of people. Each URL of a "touch point" is characterized by contextual and semantic information. It is necessary to have an "actionable" representation of these opportunities in order to be able to select them according to different criteria (linguistic, conceptual, proximity, etc.). Given the nature of the data in the form of a continuous and very voluminous stream, we wish to be able to implement an automatic and adaptive analysis of the frequent changes and the fluctuating behaviors of the connected users using an adapted representation of the data structure. The clustering algorithms must be adapted to these very large, complex and dynamic databases, in order to detect, on the one hand, informative "concepts" describing the URL and touch points visited in function of the information associated with them, and on the other hand to categorize the URL and touch points according to their similarity.

We propose here an extension of the stochastic algorithm proposed in Chapter 5. This extension is adapted to data stream analysis, allowing a dynamic creation and suppression of prototypes to follow the dynamic of the data structure. This approach is applied to the data provided by Mindlytix, in order to analyze and follow the dynamic of areas of interest over time in user's web navigation.

The second section of this chapter is another application on the company data-sets. This time, the idea is to describe the behavior of individual user and detect changes over time. Two descriptions of the behavior of each user are used: the actual location of the user during his/her online navigation is recorded using geolocation, and the log of his/her navigation is recorded. A signal of distribution of interest or location is constructed and this signal is analyzed with a modified version of the signal-based algorithm presented in Chapter 4.

6.2 Users' Interest

6.2.1 Data-sets

In this section, we present an adaptation of the Barycentric Coordinates approach to deal with complex data streams. The proposed algorithm is applied to analyze the dynamic

of web navigation behavior, as recorded during two weeks in August 2017 among French Internet users. We computed a dynamic clustering of the visited web pages according to two measures of similarity. Different labeling methods were used to characterize the clusters. In addition, the evolution of the stream structure has been recorded and analyzed in order to highlight trends and variation in the users' behaviors over time.

The data have been collected by the company "Mindlytix" from the 22^{th} of August to the 5^{th} of September, 2017. Around 6 million observations have been recorded during this period. Each observation includes a time-stamp in seconds and a URL. For example: $\{(0, google.com), (2, yahoo.com/economy), (18, currencyconverter.com)\}$. In this example, a connection to "google.com" is recorded at the start of the monitoring, then a connection to "yahoo.com/economy" two second later and finally to currencyconverter.com 16 seconds later.

The company also provided a tool to compute two measures of similarity to compare URLs, based on confidential data. The first measure uses semantic information associated to each URL. This measure also allows to compare URLs with single or group of words. The other measure is based on contextual information (URLs often visited during a short period by the same users are similar).

Our objective was to produce a dynamic clustering of this data, in order to monitor the general users' behavior and interest and follow their evolution over time. Such results are very interesting for online marketing companies which constantly need to adapt their advertising strategy to user's "mindset".

6.2.2 Algorithm

The optimization sought concern both the speed and efficiency of the algorithms and the real cost reductions. In practical industrial life, large volumes of data must be processed in a very short time on reduced material resources. Moreover, the databases to be processed are dynamic, and the proposed approach must be able to be updated quickly to detect the emergence of new concepts or the emergence of new user profiles. Finally, the data to be processed are complex because they are characterized by semantic, incomplete and noisy information. These data are very hollow, which requires an adapted approach for the creation of concepts, classes, topological space of representation with metrics not necessarily Euclidean. The solution therefore requires a very compact representation space for all the information.

Algorithm 16 is the pseudo code of the approach we propose for stream data-sets. In this approach, after projection of data in Barycentric space (we do this for each data arriving), for each object o^i, if the distance between o^i and the closest prototype w^* is bigger than a maximum radius we have chosen as a parameter, we create a new prototype by this new object, otherwise, the new object should be assigned to the nearest prototype. At the end of this step we update the prototype by equation 5.18 to reduce the distance between the

prototype and the object in the barycentric coordinates representation. In the next step, for each prototype, if the prototype's age is bigger than the parameter $MaxAge$, we remove the prototype. If it is not the case, we update the prototype using equation 5.18.

Algorithm 16 Barycentric Relational Clustering for Data Streams

Input: Data objects and their timestamps (o^i, t^i), a similarity function d.
Output: Clusters.

Parameters: MaxRadius, MaxAge, MaxSupportPoint, γ.
Initialization: We initialize the first prototype w^0 by first new arrived data with age=0.
for each new data o^i **do**
    if Current number of support point<MaxSupportPoint **then**
        Add object o^i in support points list P.
    Project data (o^i, t^i) in the Barycentric Space using eq. 5.14.
    Compute the distance between object and prototype with eq. 5.12.
    Compute the Matrix J using 5.13.
    Find the nearest prototype w^*.
if Distance(o_i,w^*)>Max Radius **then**
    Create a new prototype w^k with $\beta^k = \beta^i$.
    Set the age_k of w^k to 0.
    w^k is now the closest prototype w^* of o^i.
else
    Update the coefficient β^* of w^* using eq. 5.18.
    Set the age_* of w^* to 0.
for each prototype $w^p \neq w^*$ **do**
    if $age_p > MaxAge$ **then**
        Remove the prototype
    else
        $age_p = age_p + (t^i - t^{i-1})$

6.2.3 Results

In this section, we present and discuss the experimental results obtained on the real data-set using the two similarity measures presented in previous section.

6.2.3.1 Analysis using the "semantic" similarity

Based on the "semantic" similarity, we obtained 712 clusters from the 6 million URLs representing a monitoring period of 2 weeks. Based on the domain knowledge and the desired output, we chose a time scale $MaxAge = 604800$ seconds, a $MaxRadius = 3$ (which is the maximal distance between "similar" URLs with this measure according to our expertise), 100 support points and a value of $\gamma = 0.2$.

In table 6.1, we represented 5 of the 771 clusters obtained from the "semantic" similarity, with a few examples of URLs in each cluster. The URLs grouped in the same clusters

clearly belong to the same area of interest. Cluster #1 and #4 contain URLs sharing the same hostnames (i.e. "lacentrale.fr" for cluster #1) and is therefore quite straightforward. However, clusters #2, #3 and #5 contain URLs with different hostnames yet belonging to the same area of interest. Cluster #2 regroups URLs on the theme of cooking. Cluster #3 is clearly about hair dressing and finally cluster #5 is about dictionary. This is just a small set of clusters chosen randomly, but it shows that our algorithm is capable to regroup URLs with different hostname if they share a common theme.

In this experience, to characterize our clusters, we used a list of *Wikipedia* pages URLs in order to compute the 5 nearest Wikipedia pages to the prototype of each cluster, based on the "semantic" metrics. The 5 labels are then chosen as the title of the Wikipedia page. In table 6.1, the labels associated to each prototype (cluster) are presented in the third column. The fourth column represent a general (annotated by experts) concept for each cluster. For example, cluster #2 is labeled as "Recipe", "Seasoning", "Condiment", so the concept associated for this cluster could be "Cooking".

Table 6.1: Clusters examples based on the semantic similarity, labellization is obtained from Wikipedia pages.

Cluster number	**Associated URLs for each cluster**	**Label**	**Concept**
1	lacentrale.fr/occasion-voiture-marque-mini.html lacentrale.fr/occasion-voiture-modele-porsche-911.html lacentrale.fr/occasion-voiture-marque-dacia.html lacentrale.fr/occasion-moto-marque-triumph-9.html lacentrale.fr/occasion-voiture-marque-bmw-4.html lacentrale.fr/occasion-moto-marque-bmw-21.html	Sedan Second hand Cabriolet	Second-hand Car
2	cuisinevg.fr/endives recettes.de/pomme-de-terre auvertaveclili.fr/soupe-crue-aux-carottes-noix-de-cajou grands-meres.net/crepes-au-saumon larecette.net/pommes-de-terre-tornade-parmesan recettes.de/saute-de-veau-aux-carottes	Recipe Seasoning Condiment	Cooking
3	cheveux-naturels.fr/meches-bresiliennes.php beautiful-boucles.com/coiffure-produits beautiful-boucles.com/gel-pour-definir-ses-boucles perruquescheveuxnaturels.net beautiful-boucles.com	Hair Hair dressing Hair style	Hair dressing
4	lacoccinelle.net/aerosmith-i-don-t-want-to-miss-a-thing.html lacoccinelle.net/lukas-graham-mama-said.html lacoccinelle.net/1070837.html lacoccinelle.net/rag-n-bone-man-guilty.html lacoccinelle.net/annie-lennox-don-t-let-it-bring-you-down.html lacoccinelle.net/riles-i-do-it.html lacoccinelle.net/hayley-kiyoko-gravel-to-tempo.html lacoccinelle.net/ed-sheeran-bibia-be-ye-ye.html	Song Lyrics Popular Song	Song lyrics
5	dico-definitions.com/dictionnaire-mots.php dicocitations.lemonde.fr/Fsynonymes.php fr.wiktionary.org chileconcarlota.en-escale.com/dictionnaire-franco-chilien_pl212.html definition-dictionnaire.com synonymes.com	Conjugation Dictionary Definition	Dictionary

Table 6.2 gives additional example of clusters based on the "semantic" similarity. This time, the labels were computed from on a list of words and expressions projected in the barycentric coordinate system, based on the same similarity measure used to compute the distance between URLs. Again, in these examples, we have URLs with same hostnames and different hostnames in the clusters. Each cluster is represented by the 5 nearest words to its prototype (third column). From these words, a clear concept emerges easily.

Table 6.2: Clusters examples based on the semantic similarity, labellization is obtained from a list of words and expressions.

Cluster Number	Associated URLs for each cluster	Label	Concept
1	afrologize.blogspot.fr/2013/07/comment-prendre-soin-des-tresses-avec.html extensionstopchrono.over-blog.com/article-extensions-a-clip-de-cheveux-naturels-lisses-meches-53213780.html madmoizelle.com/youtubeuses-beaute-afro-francophones-436919 curlidole.fr/4-idees-coiffures-a-realiser-sur-les-cheveux-des-enfants-aux-cheveux-crepus-frises-et-boucles aufeminin.com/idees-maquiller.html,madmoizelle.com/maquiller-yeux-marrons-779447 coiffure-simple.com/2016/11/50-magnifiques-couleurs-cheveux-tendance-2017 beautiful-boucles.com/comment-embarquer-ses-cosmetiques-en-voyageavion-conseils-coiffure-produits	pretty hair for curly hair hairdressing tutorial hairstyle for hair hair hairstyles	Hairdressing
2	meteocity.com/contact meteocity.com/france/demain chasseurdesauvagine.forumactif.fr/Meteo-h4.htm,meteoamikuze.com/Previsions.html meteo123.org meteoblue.com/fr/meteo/prevision/semaine/clermontferrand_france_3024635 meteoblue.com/fr/meteo/prevision/semaine/hendaye_france_3013534 meteoblue.com/fr/meteo/prevision/semaine/pau_france_2988358\	weather forecast snow weather webcams weather weather forecast meteoblue weather	Weather
3	candy-crush-soda-saga.fr.softonic.com/ android/telecharger logo-quiz.fr.uptodown.com/android/telecharger wordalot.fr.uptodown.com/android/telecharger castle-clash.fr.uptodown.com/android/telecharger candy-crush.fr.uptodown.com/android/telecharger gta-iv-san-andreas.fr.uptodown.com/windows/telecharger	download free Android for android download apk download android free download	Android download
4	ilikeyou.com/like angesdemons.fr/home skuat.com/anaislareine eskimi.com/members/ mygreenlovers.com/mail.php lovebook-rencontre.com angesdemons.fr/visit	teen dating site free for singles meet teens free chat Chat for	Dating
5	lacoccinelle.net/1214369-lana-del-rey-love.html lacoccinelle.net/242810-eminem-lose-yourself.html lacoccinelle.net/1124567-ariana-grande-be-alright.html lacoccinelle.net/1180421-shawn-mendes-bad-reputation.html lacoccinelle.net/1123802-ariana-grande-bad-decisions.html lacoccinelle.net/1233199-selena-gomez-bad-liar.html lacoccinelle.net/1216858-ed-sheeran-supermarket-flowers.html	what love is Lyrics lyrics and translation i will love lyrics song translation	Song lyrics
6	halawiyat-okla-maghribiya.ovh/halwa5.php recettes.de/jus-de-fruits halawiyat-okla-maghribiya.ovh/halwa2.php recettechoumicha.canalblog.com emilie25besancon.canalblog.com jardiner-malin.fr/recettes mesrecettesfaciles.fr	recipes turkey recipes recipes of veal cooking recipes	Recipes

6.2.3.2 Analysis using the "contextual" similarity

Table 6.3 represent the results of clustering obtained with the "contextual" similarity measure. In this experience, to automatically label the clusters, we used a list of domains (e.g. "google.com", facebook.fr", etc...) and computed the "contextual" similarity measure between domains and the URLs in our data. We computed the similarity between prototypes and all domain in order to select the four closest domains for each cluster. The concept associated to each cluster is presented in column 3. For example, cluster number #3 is associated to the domains "opodo.fr","govoyage.fr","expedia.fr" and "flights-results.liligo.fr"

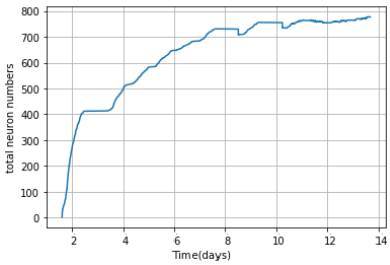
which are French websites proposing to sell flight tickets. The concept associated for this cluster could be "Flight travel".

Table 6.3: Clusters examples based on the contextual similarity, labellization is obtained from a list of domains.

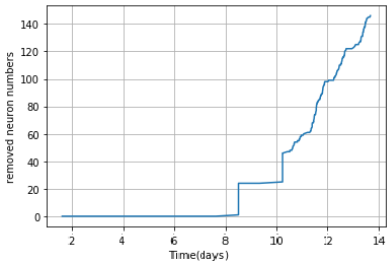
Cluster number	Associated URLs for each cluster	Concept
1	skyrock.net skyrock.com music.skyrock.com skyrock.mobi	Skyrock Music
2	opodo.fr expedia.fr govoyages.com flights-results.liligo.fr	Flight travel
3	larousse.fr linguee.de reverso.net la-conjugaison.nouvelobs.com	Dictionary
4	lepoint.fr europe1.fr bfmtv.com lexpress.fr	News
5	footmercato.net foot01.com m.lequipe.fr maxifoot.fr	Football

To monitor the evolution of the data structure, it is possible to record and store the variation of the number of clusters over time, as illustrated in figure 6.1 a, b and c. Figure 6.1.a shows the total number of clusters created during the two weeks period. In this figure, we can observe that dynamic follow three main phases. At the beginning the number of clusters sharply increase as new observations are presented to the model. Then, after 2-3 days of monitoring, the increase in the number of cluster slows down, as more and more observations are already well represented by the existing clusters. Finally, the number of clusters remain stable after the first a week of monitoring, as the creation of new clusters reach as similar rate as the deletion of irrelevant clusters. The total number of deleted prototypes over time is given in Figure 6.1.b. The figure shows that during the first week there is no deletion (in accordance to our choice of parameters' values). In the second week, however it rises sharply, as more and more cluster created at the beginning of the monitoring are no more representative of the current users' interest. Last diagram of Figure 6.1 presents the prototypes added over the period. At first, this number increase significantly as there

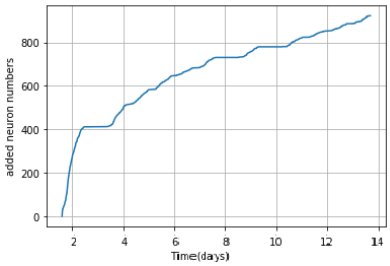
is not enough prototype to represent the diversity of the recorded URLs. After a few days, the growth starts to decrease before reaching a relative stability. This kind of analysis can help to detect global changes of users' interest, or in the contrary stable periods (see for example the plateau at days 2-3 in Figure 6.1.a).



a. Number of prototypes over time



b. Number of prototypes removed

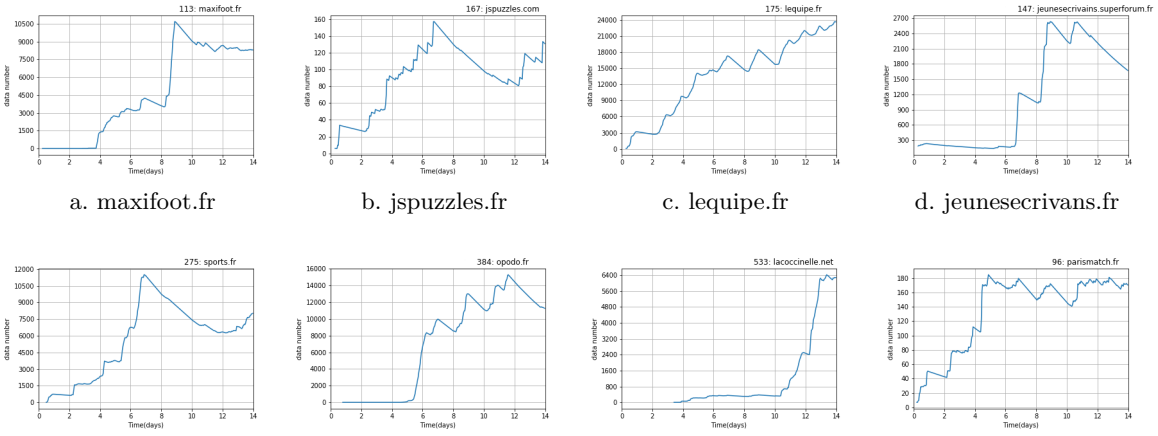


c. Number of prototypes added

Figure 6.1: Evolution of the number of prototypes over time.

One of the important part of data stream clustering is to detect variation in clusters over time. Figures 6.2 are examples of results obtained with the proposed algorithm to illustrate the changes for each cluster. In particular, each figure demonstrates the number of absorbed observations per cluster in a time period. The horizontal axis represents time and the vertical axis represents the number of users visiting URLs belonging to the cluster during the two weeks of user's navigation. Our monitoring starts at 22/08/2017, 2:00:08 a.m., for a period corresponding to the last two weeks of August. We used the same labeling method as in 6.3. Figures6.2 illustrate some examples of clusters' dynamic. For example, for

prototype number 113 (Figure 6.2.a), which represents a cluster of football (soccer) events, we can see that the number of users increase sharply at the end of August (31th August). This effect can be explained by the qualification matches in the European Cup, especially the matches of the French team during this period. Interestingly, the dynamic of cluster 275 (Figure 6.2.a), which regroups URLs associated to general information about sport, show a different pattern. In this cluster we observe a clear peak of interest during the weekend (days 7 and 14). The same pattern is observed in Figure 6.2.b in a cluster associated to puzzle games. We detect here area of interest that seems to be manifested in a cyclic pattern each week.



a. maxifoot.fr b. jspuzzles.fr c. lequipe.fr d. jeunesecrivans.fr

e. sports.fr f. opodo.fr g. lacoccinelle.net h. parismatch.fr

Figure 6.2: Time (days) in function of visiting users in each cluster. The closest domain name is given for each cluster.

Another example of punctual interest can be seen in Figure 6.2.d, representing the evolution of the cluster associated to "writers". Starting from the 6th and 7th day, the user's interest increases sharply. This timing of interest is related to the beginning of the scholar year at the beginning of September, where student start to prepare their future scholar program. Similarly, Figure 6.2.f (prototype number 384) represent a cluster associated to airline companies. We can observe that during the first weekend the number of user grow dramatically. This could be interpreted as a peak of last minute plane ticket buy, because of the end of the summer holiday in France.

In many applications, change detection is very important because we can predict the behavior of clients and propose them the product corresponding to their needs. In online marketing, the advertising must follow the users' interest as accurately as possible. With these examples we can see the dynamic of various area of interest over time. This can help us to understand and predict subjects with increasing popularity or in the contrary area showing a loss of interest. Indeed, clusters with a large number of observations at some time

denotes popular interest which should be taken into account. In addition, regularities in the clusters popularity allows to detect cyclic interest and to predict such variations. Typical cycles are found each day on different hours, or each week on different days: some topics are more popular during the evening or during the weekend.

6.2.4 Summary

In this section, we proposed an extension of this approach able to deal with relational data stream. This algorithm was applied to a real stream of user's web-pages navigation, in order to analyze the structure and dynamics of user's area of interest over time. We tested different measures of similarity between URLs and methods of automatic labeling to characterize the clusters. The results are convincing and encouraging, the clusters are homogeneous with clear associated topics. The dynamics of user's interest can be recorded and visualized for each cluster. Remarkable patterns can be associated to precise events or usual timing and cycles in user's interest.

6.3 Change Detection in Individual Users' Behavior

Tracking the behavior of users is a useful tool in marketing domain. There are different techniques to detect the changes in users' lifestyle or their behaviors. One of the solution is to detect the change in geographic location of users. People who decide to move (changing their home) are very interesting targets to advertising companies to detect and send them appropriate advertisements. Another solution is to follow the interests of a user. By tracking the users, the advertising companies have more chance to sell their offers. The main idea of the proposed approach presented is this section is to represent the behavior of a user by the distribution of the frequented places, according to the postal code of the place of connection, or by the distribution of the categories of the visited URLs, the categories being defined by the clusters URLs calculated previously. To examine the approach, I first tested the algorithm on a simulated data-set and then applied the approach on real data-sets provided by the Mindlytix company.

6.3.1 Data-sets

The proposed approach to detect changes in the users' individual behavior has been tested on an artificial data-set and applied on two real data-sets.

To generate the artificial data-set we considered three categories of behavior: The user's behavior changes over time into a totally new behavior, the user's behavior change over time into a partially different behavior, and the user do not change its behavior. We generated 10000 signals for each of these categories of users. To construct a signal, we first generated two sets of 1 to 5 random labels each, representing the possible behaviors before and after

the change. Only one set is created to simulate the absence of change and to simulate partial change we forced the two set to share 1 or 2 labels. We simulated a period of two months. A hundred random time-stamps were generated over this period. Each time-stamp were associated to a label from the first or the second set, depending on a randomly chosen date of change.

To follow the real changes in individual interest based on the data provided by Mindlytix, we used a data-set of the navigation log of 142794 users giving for each user a list of time-stamps associated to the URL visited at this time, over a period of 30 days. Based on the result of the URLs clustering presented in the previous section (using the contextual similarity), each URL were substituted by a cluster label. This step allows a user navigating between different URLs from the same topics to be considered having stable interest. The time windows for this data-set is fixed to 5 days, meaning that the distribution of URLs' labels visited during a 5 days period defines a user behavior.

Finally, to follow the change in users' physical location habits, Mindlytix provided a data-set of geolocations (postal codes) associated to time-stamps over a period of 74 days for 598 users. The objective for these data is to be able to detect when a user relocates to a different location or spend some time outside its usual area. Here, we chose a size of 10 days for the time windows to avoid detecting very short trips and unusual displacements.

In order to detect changes in the users' behavior, we applied a jump detection algorithm similar to the signal-based clustering algorithm presented in Chapter 4. This algorithm detects unusual "jumps" in a signal characterizing behavioral variations. To construct such signal, were a change in behavior is characterized by a jump, we defined the distribution of labels or postal codes in the first time windows as the reference behavior. Then, the window is shifted one day at a time, in order to produce a series of distribution. For example, if in a time window a user have been detected in France 7 times in Strasbourg (Postal code 67000) and 3 times in Nancy (postal code 54000), the distribution for this user and this time window will be $\{67000 : 70\%, 54000 : 30\%\}$. The signal is created from the dissimilarities between the distributions in the sliding time window and the distribution of reference. The signal thus obtained represents the evolution of the differences with respect to the reference window and makes it possible to detect significant changes in distributions: a move or a change of interest.

The similarity between two probability distributions (reference window and shifted windows) is computed by a metric called Jensen-Shannon divergence [136, 52]. It is based on the Kullback-Leibler divergence, with some notable (and useful) differences, including that it is symmetric and it is always a finite value. The Jensen-Shannon divergence (JS) is a symmetrized and smoothed version of the Kullback-Leibler divergence $D(P \parallel Q)$ between two discrete distributions. It is defined by

$$JS(P \parallel Q) = \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M)$$

Where $M = \frac{1}{2}(P+Q)$. For discrete probability distributions P and Q, the Kullback-Leibler divergence from P to Q is defined [130] to be

$$D_{KL}(P \parallel Q) = -\sum_i P(i) \log \frac{Q(i)}{P(i)} = \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$

Note that any zero probabilities in P or Q are ignored in the computation, meaning that two totally different distributions will have a JS value of 1.

6.3.2 Algorithm

Algorithm 17 describe the multi-scale jump detection approach, which is a slightly modified version of the signal processing clustering algorithm described in Chapter 4. The idea is similar: an iterative smoothing process eliminates random fluctuations in the signal, then unusually high variations are detected.

The signals are piece-wise continuous functions having discontinuities at some locations x_i, i.e.,

$$v(x_i^+) \neq v(x_i^-). \tag{6.1}$$

For that type of functions, there exist many approaches to locate the singularities. In multi-scale coefficients-based approach, a strategy to detect the singularities at level j is based on a criterion that uses the first or the second order differences of v^j. In these approaches, the jump singularities detection is carried out at each level independently. however, we propose a strategy to locate the jump singularities at a given level j by taking into account the detection at other levels. We detect intervals $I_{j,k}$ potentially containing a jump singularity as those satisfying the following inequalities:

$$\begin{aligned} &|\Delta v_k^j| + |\Delta v_{k+1}^j| > |\Delta v_{k+r}^j| + |\Delta v_{k+r+1}^j|,\ r = 1, \cdots, (2N+1)p_j \\ &|\Delta v_{k-1}^j| + |\Delta v_k^j| > |\Delta v_{k-1-r}^j| + |\Delta v_{k-r}^j|,\ r = 1, \cdots, (2N+1)p_j, \end{aligned} \tag{6.2}$$

for some integer $p_j \geq 1$, dependent on j and where v^j is obtained by successive projections of v^J. In that context, the singularities at level j must be separated by more than $(2N+1)p_j$ intervals. With this approach, a singularity can be detected inside two neighboring intervals. Indeed, assume that $I_{j,k}$ and $I_{j,k-1}$ are such that 6.2 is satisfied, then the singularity could be detected inside $I_{j,k}$ and inside $I_{j,k-1}$. To avoid this situation, we consider that $I_{j,k}$ (resp. $I_{j,k-1}$) contains a singularity if it satisfies 6.2 and if:

$$|\Delta v_k^j| + |\Delta v_{k+1}^j| > (\text{ resp. } <)|\Delta v_{k-1}^j| + |\Delta v_k^j|. \tag{6.3}$$

We then compute the number N_j of singularities at level j, and we define $j_{\max}$ as the largest level j such that $N_{j-1} = N_j$. We also define the level $j_{\min}$ as the smallest j such that $N_j = N_{j_{\max}}$.

We finally consider the definition of *admissible singularities*: A singularity detected in $I_{j,k}$, i.e. satisfying 6.2 and 6.3, for $j_{\min} < j < J$ is called admissible if there exists a singularity inside $I_{j+1,2k}$ or $I_{j+1,2k+1}$ and if there exists a singularity in $I_{j-1,\lfloor \frac{k}{2} \rfloor}$. This definition implies that admissible singularities make up chains when j varies.

Then, jump possibly containing intervals are defined by considering the intervals associated with an admissible singularity for levels $j_{\min} + 1$ to $J - 1$ and those associated to a singularity connected to an admissible singularity for levels $j_{\min}$ and J.

Algorithm 17 Changes Detection in Behavior Signal

Input: Signal vector v of length N.
Output: List of detected changes.

Initialize $e = [log2(N)]$
Initialize global list of jumps $L_g = \emptyset$
while e >4 **do**
    Downsampling
    for $i \leftarrow 1, length(v)$ **do**
        $v(i) = \frac{v(2i-1)+v(2i)}{2}$
    Initialize local list of jumps $L_e = \emptyset$
    Compute cost function dv based on first order of finite differences:
    for $i \leftarrow 1, length(v)$ **do**
        $dv(i) = |v(i-1) - v(i)| + |v(i) - v(i+1)|$
    Compute local maxima of cost function dv:
    for $i \leftarrow 1, length(v)$ **do**
        if $dv(i) > \max(dv(i-2), dv(i-1), dv(i+1), dv(i+2))$ **then**
            $L_e \leftarrow L_e + \{i\}$
    $e \leftarrow e - 1$
Define L_g as the intersection of all L_e $L_g = \cap_e^{\log_2(N)} L_e$

6.3.3 Results

The proposed approach has been tested on the artificial data-set for validation, then applied on the real data-sets to analyze the changes in users' behaviors.

6.3.3.1 Artificial Data-sets

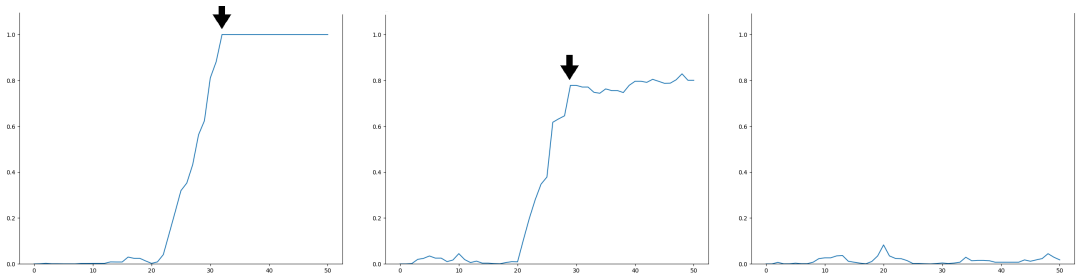
The artificial data have been generated following three cases: full change in behavior, partial change and no change.

Figure 6.3.a is an example of simulated signal for a user who expressed a full change of behavior. The horizontal axis is the time-stamp (days) and the vertical axis is the JS

dissimilarity for the reference window. As you see, the JS increases from the 22th to the 29th day. Then the signal keeps a value of 1 from the 30th day onward, as there is no intersection between the reference distribution and the distributions from the 30th day.

In the case of a partial change, the user express new behaviors in addition to some of its previous. For example, a user who relocate into a new house but keep the same work in its previous location. Figure 6.3.b shows such case. This time, the signal never reaches 1 as there are still some similarities before and after the changes. The change is nonetheless correctly detected by the algorithm.

The last case is when there is no detection of change for a user. In this case there is no significant different between the reference window and the shifted windows and the signal stays steady over the time. The example showed in Figure 6.3.c demonstrates that the similarity computed by the Jensen-Shannon divergence is low. The signal created is stable all over the time with no notable change. This case can describe users who keep a regular activity without any notable variations.



a. Full change b. Partial change c. No change

Figure 6.3: Example of simulated signals of user's behaviors. The arrows indicate the detected changes.

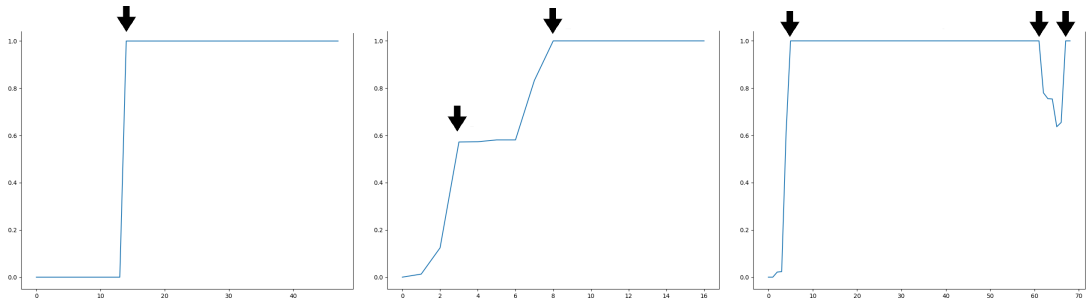
To validate the quality of the detected changes, we computed the mean of the absolute differences between the detected and the predicted date of change. Over the 30000 signals, we observed a mean error of $\pm 1.67\,\%$, which is an acceptable value considering the size of the time windows. We also computed the percentage of undetected change and the percentage of wrongly detected change. The proposed algorithm never detected a change for the 10000 signals without simulated change, and only $0.74\,\%$ of the simulated changes were not detected. Overall the quality of the proposed approach is very satisfying and it should be able to deal with real data in more complex applications.

6.3.3.2 Geolocation

In this section we will describe the results obtained on the geolocation data-sets provided by Mindlytix. We analyzed the change behavior for 598 users during 74 days. In the signal

creation step we used a window with a size of 10 days. We observe some variety in the signals, but there is still some characteristic patterns.

Figure 6.4 illustrates some examples of signals characteristic of a clear relocation. Figure 6.4.a is a good example of simple change in the user's location. It is very similar to the signals we generated for the algorithm validation. In figure 6.4.b the jensen-shannon dissimilarity increases sharply for two days, stays stable for three days, then again rises suddenly. Two changes are detected, the first being a partial change. This kind of signal can be interpreted as a move in two steps, with a period where the user spend time in both locations before moving definitively. The third figure of 6.4 is another example for relocation of a user. However, this time we observe a small period where the user spend some time in its previous location.



a. b. c.

Figure 6.4: Example of obtained signals during a user's relocation. The arrows indicate the detected changes.

Users from another category do not move at all, neither to relocate nor to go to trip or vacations during the recorded period. The dissimilarity between the reference window and the shifted windows stays low all the time. Figure 6.5 is a typical example for this type of user.

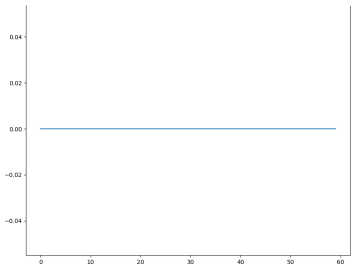
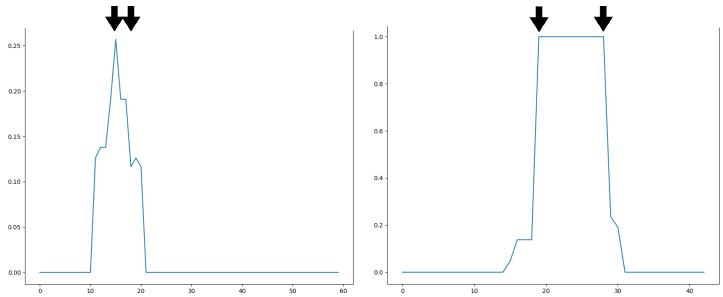


Figure 6.5: Illustration of a signal of a user who does not change its location during the recorded period.

Another interesting case is when the user leaves for a vacation or work for some time, before returning to the place he/she used to live. Figure 6.6 shows two examples for this case.

As you see in Figure 6.6.a, around the 10th day the user starts to move. The dissimilarity between the reference window and the shifted window rises sharply until the 15th day. Then, this dissimilarity decreases rapidly to reach the same distribution as the reference window. It means that this user spent 10 days (the size of the time windows) in another place before coming back. Another example is presented in Figure 6.6.a, which shows a clear example of a user leaving for a 3-week travel and return to his/her initial place. In both examples, the two changes are correctly detected.

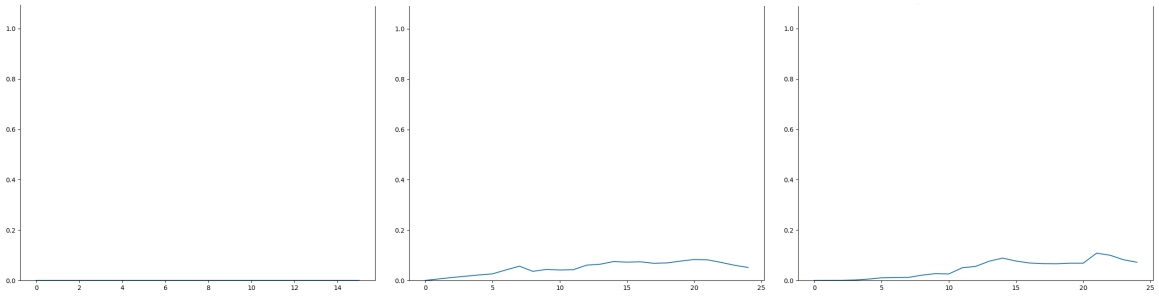


a. b.

Figure 6.6: Example of obtained signals for temporary displacements. The arrows indicate the detected changes.

6.3.3.3 Individual interest

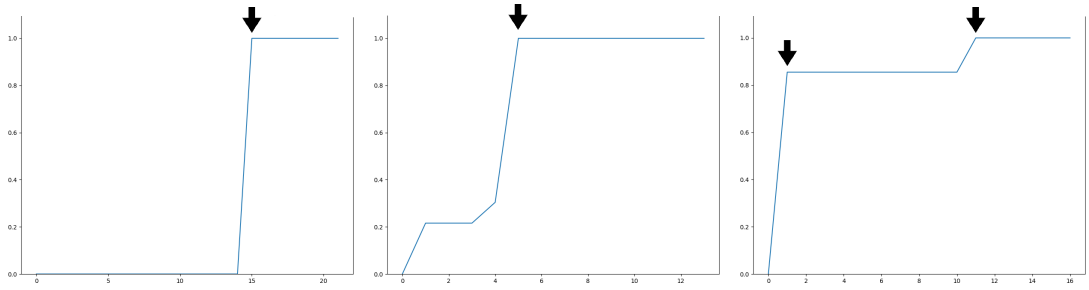
To follow the change of users' interest we used users' navigation log information. We have the URLs visited during 30 days for 142794 users. Each URL have been associated to a cluster in the previous section, and the user's navigation can be expressed into a distribution of visited clusters varying over time.



a. User without change b. User with minor variations c. User with minor variations

Figure 6.7: Examples of stable interest in users' navigation.

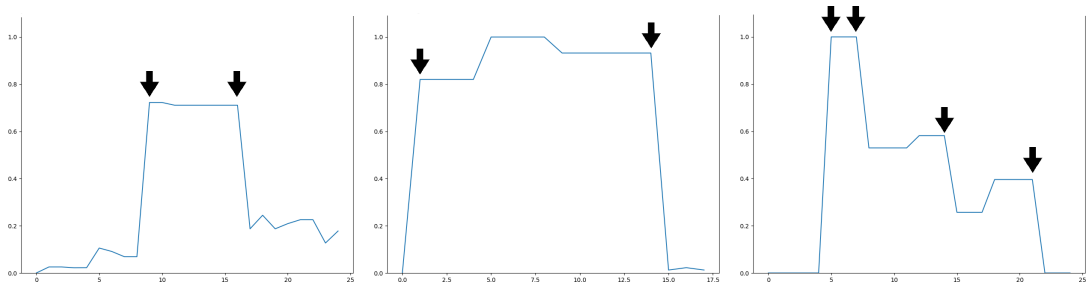
Figure 6.7 illustrate the behavior of users who do not change their interest during one month. As you see, in all three figures the signal is either stable or with only minor variations (undetected by the algorithm).



a. b. c.

Figure 6.8: Examples of changing interest in users' navigation.

Figure 6.8 is an example of results for the detection of change in individual interest, where the users change their interest over time. As can be seen, in Figures 6.8.a the signal of this user remains stable for 14 days, then starts to rise sharply as the user start to navigate in other categories of URLs. In figures 6.8.b and 6.8.c, the change is more gradual before reaching a state of interest fully different from the window of reference. These three figures are typical examples of the different pattern of change in a user's interest.



a. b. c.

Figure 6.9: Examples of temporary change in users' navigation

A third category of observed behavior is a group of users who change their interest for a limited period and then return to their initial interest. Figures 6.9 illustrate this type of users. As you see, these signals go up and stay stable over a period of time and then go down. It means that the dissimilarity between the reference window and the shifted windows increase for a period of time, but at the end of the recorded period the distribution of visited categories of URL returns to a distribution similar to the distribution of reference. Figure 6.9.c shows a particular example of temporary change, were the user return to its initial interests in several steps.

In addition, we computed a clustering of users based on their interest over a chosen time window. For each user, we computed the distribution of visited category of URLs during one month. Then, we applied the Barycentric Coordinates Algorithm 16 on these distribution, using the Jenson-Shannon metric as a distance measure. The Algorithm was parameterized to never forget a cluster and to produce clusters with a maximum radius of 0.2 JS, meaning that the users in a cluster have a at least a similarity of 80% with the prototype and 60% with each other for the most distant members of the cluster. We obtained 661 clusters of users. Each cluster represent a pattern of interest shared among a group of users, so we computed the global distribution of visited categories of URLs over all the members of the cluster. Some examples are shown in Table 6.4. In this table we only show the probabilities $> 10\%$ in the distributions. As one can see, we found some big clusters of typical interest, such as "sport", "news" or "cooking", a well as some interesting combination of interest, such as "women beauty" + "gardening" or "second-hand car" + "real estate".

Table 6.4: Example of clusters of users' interest over a window of one month.

Cluster number	Cluster size	Distribution of interests	
		Label of URLs category	Probability of visit
0	7438	News	97.2 %
5	10649	Webmail	67.2 %
		Search engine	28.0 %
14	1762	Handiwork	86.0 %
24	2396	Cooking	90.2 %
28	1099	Women beauty	18.7 %
		Gardening	10.5 %
55	3327	Sport	58.1 %
267	1381	Second-hand car	26.3 %
		Real estate	11.8 %

6.3.4 Summary

The detection of change in behavior of users is a very interesting information which can help marketing companies to send and sell the right product to users based on their needs. In this section we used the users' geographic location to detect changes in their geographic habits, and the users' navigation logs to detect variation in their interests. We first created a signal for each user based on distributions representing its behavior. Similarity between distributions is computed by the Jenson-Shannon metric. Then, by using a jump detection algorithm, we detected the date where there was a change in the user behavior. We detected different scenarios: during the analyzed period, some users kept the same behavior, some had a clear change in their behaviors (for example they move to another place) and some showed a change in their behavior which lasted only a short period of time (for example a short trip or a temporary change of interest in the visited URLs). The tests performed on the simulated signals showed that the algorithm is efficient for this task: only 0.74% of the changes are not correctly detected, no false positives have been produced and the average error for the prediction of the change dates is less than 2%.

6.4 Conclusion

In this chapter, we proposed an extension of the prototype-based algorithm (introduced in Chapter 5) to be able to deal with relational data stream. This algorithm was applied to a real stream of user's web-pages navigation, in order to analyze the structure and dynamics of user's area of interest over time. The clusters are homogeneous with clear associated topics and the dynamics of user's interest can be recorded and visualized for each cluster.

In addition, we analyzed the change in individual behavior of users based on their navigation and geolocation data. We first created, for each user, a signal of the evolution in the distribution of online user's interest and another signal based on the distribution of physical locations recorded during their navigation. Then, by using a jump detection algorithm similar to the one proposed in Chapter 4, the changes in interest or locations were detected automatically.

The detection of behavior of users is a very interesting and useful subject in marketing. The applications presented here show the adequacy of the approaches proposed in this thesis, but are yet only a few of the possibilities make possible by fast and dynamic relational clustering algorithms.

Chapter 7

General Conclusion and Perspectives

Table of Contents

7.1 Summary and Discussion

Unsupervised learning is an important and complex task. The process of extracting knowledge from data, by itself, is a challenging job. Besides, with the increasing volume of the data-sets, the need for fast algorithms able to deal with this kind of data is more and more important. Additional difficulties arise when the data-set change over time. Indeed, because of the limitation of available memory, it is usually not possible to stock the whole data-set. Furthermore, the structure of data change over the time and the algorithm should be capable to detect these changes. In this thesis, we were interested to relational data. Most clustering algorithms are designed for a specific type of data, such as images, networks or documents. The main objective of this work is to propose algorithms capable to deal with any type of data. For this reason, we focused on relational data, which are described by their similarity or their dissimilarity.

In Chapter 4, we introduced an incremental matrix reordering approach for relational data. In terms of quality and memory cost, we observed a significant improvement compared to the state of the art approaches. This algorithm is capable to represent any change in the data structure. Then, we proposed a new incremental clustering approach based on matrix reordering (therefore adapted to relational data). The specificity of this algorithm is that, by using a signal-based jump detection function, the number of clusters is detected automatically and is no more a parameter to choose. The experimental comparisons show the quality of the proposed approach on a set of real and artificial data-sets.

In Chapter 5, we focused on prototype-based clustering approaches for relational data, based on the Barycentric Coordinates formalism. With this approach, we do not need to stock in memory the whole matrix of similarity (size N^2). The idea is, instead of computing the dissimilarity of all pair of data, to use a subset of representative data, called support points. After testing the approach on real and artificial data-sets, results show a significant improvement in terms of complexity and memory usage. The algorithms are capable to deal with massive data-sets.

In Chapter 6, we proposed an adaptation of the Barycentric Coordinates clustering algorithms able to deal with data stream and we applied the approach on the data-sets provided by the company Mindlytix. The aim was to discover and follow over time clusters of URLs based on user's navigation information. The obtained results and visualizations were validated by the experts and provided new exploitable information to improve the company's efficiency.

In summary, we proposed algorithms adapted to relational data for both static and dynamic application. The reordering approaches provide useful visualization and are parameter free but can be expensive in terms of memory cost and have difficulties to detect properly overlapping clusters, which are usually not correctly represented by the reordering process. The prototype-based approaches are adapted to big volume of data and to data

stream analysis but relies on user-defined parameters (in particular the number of clusters). In order to produce approaches able at the same time to detect the number of clusters and the presence of overlapping clusters, a first idea would be the use of density estimate, in which a high density represent a cluster center, whereas low densities are characteristic of clusters' borders. In the next section we introduce a first step in this direction, coupling prototypes and density to find the borders between clusters.

7.2 Perspective: Coupling Prototypes with Density?

In this section, we investigate the idea of performing a two-level clustering based on a local density estimation for each prototype, in order to detect automatically the number of clusters without losing the advantages of the prototype-based models.

The objective is to develop an approach of clustering based on the Growing Neural Gas algorithm (GNG) [137, 67] to train prototypes capable of following the dynamic of the data. The idea is to learn dynamically an estimation of the local density of objects for each prototype, then to use this density to detect clusters boundaries. The number of clusters is detected automatically and clusters of any type of structure can be detected.

This approach is only valid for vector data for now, but the conversion to a Barycentric approach should not be difficult. We will discuss the weakness and advantages of the new algorithm and we propose some ideas to improve our work.

7.2.1 Growing Neural Gas Algorithm

Growing Neural Gas (GNG) is an algorithm proposed by [137], which is able to compute dynamically a set of prototypes to represent the data in a condensed form.

GNG is able to adapt the number of prototypes (nodes) to the representation need and compute a neighborhood network between prototypes by linking nodes representing similar data. In addition, each connected component of this graph can be seen as a representation of a cluster. Its greatest weakness, however, is its inability to adapt to rapidly changing distributions.

The algorithm GNG + Utility [67] on the other hand, is designed to handle this type of data by removing periodically the least useful nodes and adding relevant new nodes when needed. Basically, it removes nodes that contribute little to reducing the error and inserts new nodes in the graph when they contribute significantly to the error reduction. Algorithm 18 describes the GNG+U approach.

The difference between the GNG and GNG+U algorithm is not large but the impact on performance is significant [67].

Algorithm 18 $GNG + U$

Input: X, k, e_μ, e_n, age_{max}, α, β and λ.

Output: A network of connected neurons representing the data structure.

Initialization: create two nodes positioned randomly with an *error* value and a utility value U set to 0, connected with an edge with $age = 0$.

for each data point x **do**

Locate the two nodes s and t the closest to x.

Update the *error* and U values of s:

$$error_s \leftarrow error_s + \|\mu_s - x\|^2$$

$$U_s \leftarrow U_s + error_t - error_s$$

Update prototype w_s and its topological neighbors (i.e. all nodes connected to s):

$$\mu_s \leftarrow \mu_s + e_\mu(x - \mu_s)$$

$$\mu_n \leftarrow \mu_n + e_n(x - \mu_n), \forall n \in Neighbour(s)$$

with $e_\mu, e_n \in [0, 1]$.

Increment by 1 the *age* of all edges between s and its topological neighbors.

if s and t are connected by an edge **then**

set the *age* of this edge to 0.

else

create an edge between them and set its *age* to 0.

if there are edges with *age* over the age_{max} threshold **then**

Remove them these edges

if there are nodes without edges **then**

Remove these nodes

Remove node i with the smallest value U_i if: $\frac{error_j}{U_i} > k$, where j is the node with the biggest error, and k is a constant parameter.

if the current iteration is a multiple of a parameter λ **then**

Insert a new node r as follows:

1. Find the node u with the highest *error*.
2. Among the neighbors of u, find the node v with the highest *error*.
3. Insert the new node r between u and v as follows: $\mu_r \leftarrow \frac{\mu_u + \mu_v}{2}$.
4. Create edges between u and r and v and r, and remove the edge between u and v.
5. Reduce errors in variables u and v and define the error of r: $error_u \leftarrow \alpha \times error_u$, $error_v \leftarrow \alpha \times error_v$ and $error_r \leftarrow error_u$
6. Initialize the utility value of the new node r: $U_r \leftarrow \frac{U_u + U_v}{2}$

Reduce the *errors* and U of all nodes by a factor β: $error_k \leftarrow error_k - \beta \times error_k$ and $U_k \leftarrow U_k - \beta \times U_k$

if the stop criterion is not met **then**

repeat from 2

7.2.2 Two-Level GNG Based on Density Estimate

We propose a new algorithm that simultaneously learns the prototypes of GNG and their segmentation using information on the data densities. The principle is inspired from DS2L-SOM [36], a clustering algorithm adapted to static data-sets. The new algorithm estimates the local density of data for each node of the GNG, in order to detect the density fluctuations that characterize the boundaries between data groups.

Algorithm 19 Density-based Clustering

Input: $P = C_{i=1..L}$ and $D_{j=1..M}$.
Output: A segmentation of the data (clusters).

1: **for** each component $C_k \in P$ **do**
2: Determine the set $M(Ck)$ of local maximal density:

$$M(C_k) = \{i \in C_k | D_i \geqslant D_j, \forall j \text{ neighbour of } i\}$$

3: Compute the threshold matrix:

$$S = [S(i,j)]_{i,j=1...|M(C_k)|} \text{ with } S(i,j) = \left(\frac{1}{D_i} + \frac{1}{D_j}\right)^{-1}$$

4: **for** each node $i \in C_k$ **do**
5: Label i with an element $label(i)$ of $M(C_k)$, according to an ascending gradient of density along topological connections.
6: **for** each pair of neighbor nodes (i,j) **do**
7: **if** $label(i) \neq label(j)$ and $D_i > S(i,j)$ and $D_j > S(i,j)$ **then**
8: Merge the two groups (the small variation of density is considered as noise).
9: **Return the segmentation** (i.e. the clusters).

The first part of the new algorithm is based on the same assumptions as GNG + U with the addition of:

- For every node n, we include a local variable, the density D_n of this node.
- In GNG + U, Step 4, after the local error and utility of node s have been updated we now add a rule to update the density D_j for each node j.

$$D_j(t) = D_j(t-1) + e^{\frac{\|x-\mu_j\|^2}{2\sigma^2}})$$

  with σ a width parameter.

- In Step 17, when a node is created, initialize the density of the new node r as follow:

$$D_r \leftarrow \frac{D_u + D_v}{2}$$

- In Step 18, the density for all nodes is updated in the same manner as *error* and U, with the same decay constant.

$$D_k \leftarrow D_k - \beta \times D_k$$

The second part of the approach is the off-line computation of the segmentation (i.e. the clusters), using the density information to detect the boundaries between clusters (Algorithm 19). The idea is to detect low density zones within the L connected components C of the GNG+U, in order to characterize the subgroups defined by density. We use, for each pair of adjacent subgroups, a "density-dependent" index is computed to determine whether a low-density area is a reliable indicator of the data structure, or whether it should be regarded as a random fluctuation in density.

7.2.3 Experimental Validation

We compared the proposed algorithm (named "Den") with other two-level approaches in which traditional clustering algorithms [5] are applied on the nodes of GNG+U: K-means ("KM"), Hierarchical Ascendant Clustering with Ward ("Ward") or Average Link ("Avg") distances, affinity propagation ("Aff"), the density-based DBSCAN algorithm ("DBScan") and GNG-U alone ("GNG") with cluster defined by connected component of the graph. The parameters used for these algorithms are $e_w = 0.2$, $e_n = 0.006$, $age_{max} = 50$, $\lambda = 100$, $\alpha = 0.5$ and $k = 30$, $\beta = 0.01$.

7.2.3.1 Description of the experimental data-sets

The approaches were compared on six artificial data-sets, the description of data-sets is presented in table 7.1. Three data-sets ("static_...") are not associated to a changing distribution over time, whereas the others ("Dyn_...") have a time-stamp associated to each object and a dynamic structure over time (for more details please see Chapter 3).

Table 7.1: Description of the experimental data-sets.

Data-sets	# Objects	Type	# Classes
Static_Noconv_2	10000	Vector (artificial)	7
Static_Gauss_2	10000	Vector (artificial)	6
Static_Gauss_10	10000	Vector (artificial)	9
Dyn_Noconv_2	10000	Vector (artificial)	3
Dyn_Gauss_2	10000	Vector (artificial)	4
Dyn_Gauss_10	10000	Vector (artificial)	9

7.2.3.2 Results

Table 7.2 presents the value of the Adjusted Rand Index (ARI) [160] for the 7 different algorithms for each data-set in each of the 10 periods noted T1 to T10 (a time period is represented by 1000 data points). Results based on the Normalized Mutual Information and the Jaccard index are highly similar and are not presented here. We used the true number of clusters for the three algorithms that need this number as a parameter or to extract the clusters from the dendrogram ("KM", "Ward" and "Avg"). The others parameters were chosen to give the best results. Fig. 7.1 represents the mean quality of the algorithms over the 6 data-sets, for each time period.

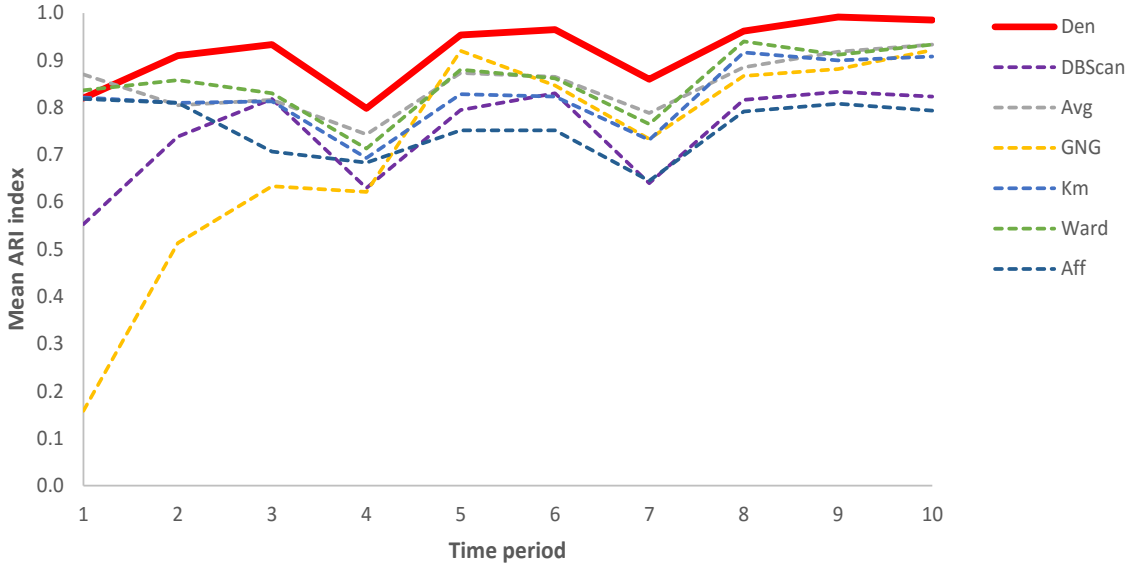


Figure 7.1: Mean value of the Adjusted Rand Index over the 6 data-sets, for each time period and each algorithm.

The results show that our algorithm provides a generally better solution than the other algorithms. For the three algorithms "Km", "Ward" and "Avg", satisfactory quality is observed in general, especially for static Gaussian data-sets, but the number of cluster must be known a priori, which is rarely the case in reality. Amongst algorithms that don't need this value, our algorithm is the only one that perform well for all the data-set, as shown in Figure 7.2. On the contrary, we can observe that the algorithm "GNG" cannot separate the clusters in contact because of the connections between prototypes; "DBSCAN" struggles with clusters of different density; as for "Aff" algorithm, it does not work well on non-convex data-sets.

Table 7.2: Adjusted Rand Index for each data-set, each algorithm and each period.

data-set	Algorithm	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Mean
	Den	0.62	0.66	0.72	0.73	0.98	1.00	0.95	0.79	1.00	0.98	**0.84**
	DBScan	0.17	0.23	0.95	1.00	1.00	0.98	0.98	1.00	0.98	1.00	**0.83**
	Avg	0.65	0.51	0.53	0.67	0.65	0.62	0.66	0.53	0.54	0.66	**0.60**
Static_Noconv_2	**GNG**	0.00	0.08	0.51	0.76	1.00	1.00	0.98	0.79	1.00	1.00	**0.71**
	Km	0.62	0.49	0.46	0.48	0.50	0.50	0.48	0.55	0.45	0.52	**0.50**
	Ward	0.68	0.66	0.55	0.65	0.66	0.63	0.66	0.66	0.50	0.65	**0.63**
	Aff	0.80	0.48	0.47	0.48	0.49	0.46	0.54	0.53	0.48	0.50	**0.52**
	Den	0.81	0.89	0.94	0.94	0.90	0.97	0.93	0.98	0.95	0.93	**0.92**
	DBScan	0.67	0.68	0.69	0.31	0.68	0.65	0.68	0.64	0.66	0.64	**0.63**
	Avg	0.93	0.95	0.96	0.95	0.92	0.95	0.95	0.98	0.97	0.94	**0.95**
Static_Gauss_2	**GNG**	0.00	0.00	0.00	0.28	0.68	0.26	0.60	0.59	0.29	0.64	**0.33**
	Km	0.89	0.92	0.94	0.95	0.88	0.85	0.94	0.95	0.95	0.93	**0.92**
	Ward	0.93	0.95	0.96	0.95	0.92	0.95	0.95	0.98	0.97	0.95	**0.95**
	Aff	0.85	0.92	0.87	0.88	0.88	0.86	0.94	0.88	0.93	0.85	**0.89**
	Den	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
	DBScan	0.02	1.00	0.76	0.49	0.25	0.53	0.36	0.44	0.36	0.30	**0.45**
	Avg	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
Static_Gauss_10	**GNG**	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.89	**0.89**
	Km	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
	Ward	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
	Aff	0.79	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**0.98**
	Den	0.58	1.00	1.00	0.92	1.00	1.00	1.00	1.00	1.00	1.00	**0.95**
	DBScan	0.83	0.61	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**0.94**
	Avg	0.76	0.63	0.62	0.77	1.00	1.00	1.00	1.00	1.00	1.00	**0.88**
Dyn_Nocov_2	**GNG**	0.58	1.00	1.00	0.92	1.00	1.00	1.00	1.00	1.00	1.00	**0.95**
	Km	0.66	0.72	0.70	0.70	1.00	1.00	1.00	1.00	1.00	1.00	**0.88**
	Ward	0.61	0.78	0.68	0.66	1.00	1.00	1.00	1.00	1.00	1.00	**0.87**
	Aff	0.71	0.73	0.29	0.80	0.66	0.69	0.46	0.58	0.68	0.65	**0.63**
	Den	0.91	0.91	0.94	0.69	0.84	0.82	0.96	1.00	1.00	1.00	**0.91**
	DBScan	0.91	0.91	0.51	0.69	0.84	0.82	0.82	0.82	1.00	1.00	**0.83**
	Avg	0.88	0.74	0.79	0.46	0.67	0.62	0.81	0.80	1.00	1.00	**0.78**
Dyn_Gauss_2	**GNG**	0.00	0.00	0.29	0.69	0.84	0.82	0.82	0.82	1.00	1.00	**0.63**
	Km	0.76	0.73	0.78	0.42	0.59	0.59	0.59	1.00	1.00	1.00	**0.75**
	Ward	0.80	0.76	0.79	0.41	0.70	0.59	0.60	1.00	1.00	1.00	**0.77**
	Aff	0.91	0.73	0.61	0.42	0.48	0.50	0.60	0.76	0.76	0.76	**0.65**
	Den	1.00	1.00	1.00	0.51	1.00	1.00	0.32	1.00	1.00	1.00	**0.88**
	DBScan	0.72	1.00	1.00	0.29	1.00	1.00	0.00	1.00	1.00	1.00	**0.80**
	Avg	1.00	1.00	1.00	0.61	1.00	1.00	0.31	1.00	1.00	1.00	**0.89**
Dyn_Gauss_10	**GNG**	0.37	1.00	1.00	0.08	1.00	1.00	0.00	1.00	1.00	1.00	**0.74**
	Km	1.00	1.00	1.00	0.61	1.00	1.00	0.38	1.00	1.00	1.00	**0.90**
	Ward	1.00	1.00	1.00	0.61	1.00	1.00	0.38	1.00	1.00	1.00	**0.90**
	Aff	0.85	1.00	1.00	0.52	1.00	1.00	0.33	1.00	1.00	1.00	**0.87**

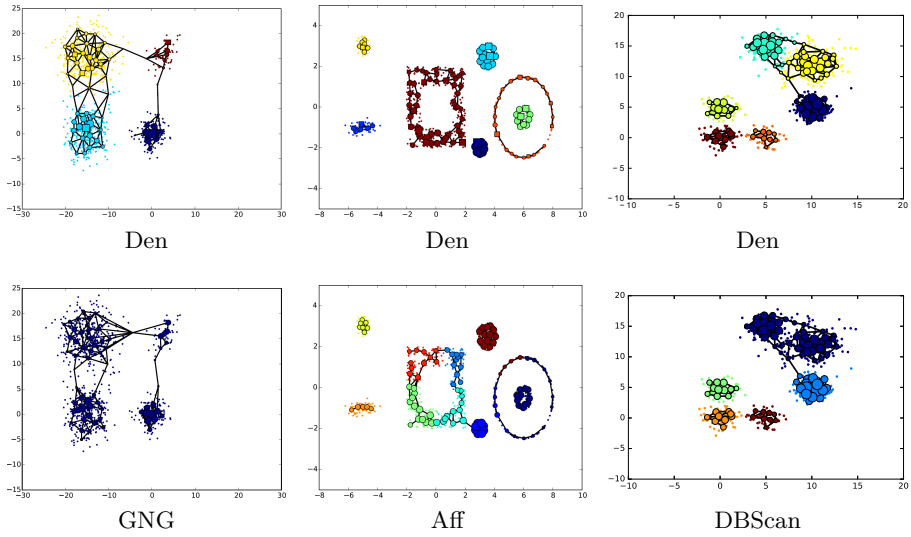


Figure 7.2: Example of clustering results according to the algorithms used. Each color represents a cluster. The graph of GNG+U is shown with node size proportional to the estimated density used in the proposed algorithm.

7.2.4 Summary

We proposed in this section a new two-level algorithm based on GNG-U, able to represent the data structure in real-time and automatically discover the number of clusters at each moment. We showed that our algorithm works well on a set of static and dynamic data-sets and produces better quality results than classical algorithms applied on the prototypes of a GNG-U. In addition to convert and test the approach in the Barycentric formalism, we now need to study the effect of the different parameters to analyze the importance of each of them and find the most effective combinations depending on the data size and the speed of its evolution. It is also necessary to compare our algorithm on real data of larger sizes in order to validate the approach for real applications.

7.3 Future Steps

In the last section of this manuscript, we wish to propose new paths of research to pursue the work undertaken during this Ph.D.

7.3.1 Application to the Detection and Monitoring of User's "Mindset"

We aim to use the real data-sets from the company Mindlytix to improve the visualization and change detection of user's "mindset", i.e. the current "state of mind" of a user, based on its recent navigation sequence and other information collected by the company. Indeed, Mindlytix is interested to detect the changes in users' mental state, such as interests, sentiments or emotions, in order to propose advertisement adapted to their mental state

(mindset profiling job). By profiling the users, we can find people who have the same profile and, by analyzing their mental state, we can precisely understand their needs. Scoring the mental state of users can give us information on about what they feel at the moment they are monitored and predict their next move. Moreover, it can help the companies to send the right and exact advertisement.

7.3.2 Signal-based Clustering Adapted to Overlapping Clusters

In section 4 we presented a clustering approach based on Reordering dissimilarity matrix. This approach works well in case we don't have overlapped data. So, to address this weakness, we thought to learn a set of prototypes using a barycentric version of the 1-D Self-Organizing Map (SOM) and to compute the local density of each node of the SOM. The ordering of the prototypes will be learned automatically thanks to the SOM properties. The obtained signal of density can then be treated so as to detect low density area defining clusters boundaries. It will also be possible to obtain a local reordering of the data for each prototype, to obtain a good visualization of the dissimilarity matrix.

7.3.3 Dynamic Neural Networks using Barycentric Coordinates

In previous section we reviewed GNG-U algorithm which is a powerful algorithm for dynamic data. In addition, in Chapter 5 we introduced an approach based on Barycentric Coordinates to update the prototypes and find the clusters in static data-sets. Since Growing Neural Gas algorithm can deal with dynamic data and can add or remove prototypes based on their utility, I would like to convert this algorithm in order to detect the clusters in dynamic relational data using Barycentric Coordinates to update the structure of the network automatically. In addition, this adaptation can be applied to other Unsupervised Neural Networks such as the Self-Organizing Map [112] and its dynamic variants, allowing powerful visualization of the data structure.

7.3.4 Similarity Learning with Deep Neural Network.

Since we are able to deal with massive relational data with aid of Barycentric Coordinates based on a similarity measure, the next step would be to find this similarity measure automatically. One idea is to train a deep learning algorithm to be capable to produce a general similarity measure for data, defined as the quantity of common information shared by the objects. For example, auto-encoder [23] algorithm can be able to get two objects in input, compress the information and then decompress to try to recover the information about the two object. Comparing the quality of the output can give us an idea how much information these two data share. If the error is low it means that the system successfully compressed the combined information, which should only be possible if the two objects share enough information (i.e. are similar). Other deep learning approaches, such as GAN [73], could also be investigated.

Bibliography

[1] Charu C. Aggarwal. A Framework for Diagnosing Changes in Evolving Data Streams. In *Special Interest Group on Management of Data Conference*, pages 575–586, 2003.

[2] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 81–92. VLDB Endowment, 2003.

[3] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for projected clustering of high dimensional data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 852–863. VLDB Endowment, 2004.

[4] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for on-demand classification of evolving data streams. *IEEE Trans. on Knowl. and Data Eng.*, 18(5):577–589, May 2006.

[5] Charu C. Aggarwal and Chandan K. Reddy, editors. *Data Clustering: Algorithms and Applications*. CRC Press, 2014.

[6] Charu C. Aggarwal and Philip Yu. A Survey of Synopsis Construction Methods in Data Streams. In C. Aggarwal, editor, *Data Streams: Models and Algorithms*, pages 169–207. Springer, 2007.

[7] Charu C. Aggarwal and ChengXiang Zhai. *Mining text data*. Springer Science & Business Media, 2012.

[8] Damminda Alahakoon, Saman Halgamuge, and Bala Srinivasan. Dynamic self-organizing maps with controlled growth for knowledge discovery. *Trans. Neur. Netw.*, 11(3):601–614, May 2000.

[9] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.

[10] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12(4):461–486, 2009.

[11] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, pages 49–60, New York, NY, USA, 1999. ACM.

[12] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[13] Kenneth D. Bailey. *Typologies and taxonomies: an introduction to classification techniques*. Number 102 in Quantitative applications in the social sciences. Sage Publications, 1994.

[14] Timothy L. Bailey and Charles Elkan. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine learning*, 21(1):51–80, 1995.

[15] Antonio Balzanella, Yan Lechevallier, and Rosanna Verde. A new approach for clustering multiple streams of data. In S Ingrassia and R Rocci, editors, *Classification and Data Analysis*, pages 417–420, 2009.

[16] Ziv Bar-Joseph, David K Gifford, and Tommi S Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17(suppl 1):S22–S29, 2001.

[17] Stephen T Barnard, Alex Pothen, and Horst D Simon. A Spectral Algorithm for Envelope Reduction of Sparse Matrices. In *Proceedings of the 1993 ACM/IEEE Conference on Supercomputing*, Supercomputing '93, pages 493–502, New York, NY, USA, 1993. ACM.

[18] Alberto Barrón-Cedeño, Paolo Rosso, Eneko Agirre, and Gorka Labaka. Plagiarism detection across distant language pairs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 37–45, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[19] Derek Beaton and Iren Valora. RADDACL: A Recursive Algorithm for Clustering and Density Discovery on Non-linearly Separable Data. *Prodeeding of International Joint Conference on Neural Networks*, pages 1423–1428, aug 2007.

[20] Michael Behrisch, Benjamin Bach, Nathalie Henry Riche, Tobias Schreck, and Jean-Daniel Fekete. Matrix Reordering Methods for Table and Network Visualization. *Computer Graphics Forum*, 2016.

[21] Shai Ben-David and Margareta Ackerman. Measures of Clustering Quality: A Working Set of Axioms for Clustering. In D Koller, D Schuurmans, Y Bengio, and L Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 121–128. Curran Associates, Inc., 2009.

[22] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[23] Yoshua Bengio. *Learning Deep Architectures for AI*. Now Foundations and Trends, 2009.

[24] James C. Bezdek and Richard J. Hathaway. VAT: a tool for visual assessment of (cluster) tendency. In *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, volume 3, pages 2225–2230, 2002.

[25] Albert Bifet. Adaptive stream mining: Pattern learning and mining from evolving data streams. In *Proceedings of the 2010 Conference on Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*, pages 1–212, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[26] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: Applications to image and text data. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 245–250, New York, NY, USA, 2001. ACM.

[27] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[28] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. GTM: the generative topographic mapping. *Neural Computation*, 10:215–234, 1998.

[29] Christian Bohm, Karin Kailing, Hans-Peter Kriegel, and Peer Kroger. Density connected clustering with local subspace preferences. In *Proceedings of the Fourth IEEE International Conference on Data Mining*, ICDM '04, pages 27–34, Washington, DC, USA, 2004. IEEE Computer Society.

[30] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *CoRR*, abs/1607.0, 2016.

[31] Ingwer. Borg and Patrick.J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[32] Andreas Buja, Deborah F Swayne, Michael L Littman, Nathaniel Dean, Heike Hofmann, and Lisha Chen. Data visualization with multidimensional scaling. *Journal of Computational and Graphical Statistics*, 2008.

[33] Cristian Bustos, Gonzalo Navarro, Nora Reyes, and Rodrigo Paredes. An Empirical Evaluation of Intrinsic Dimension Estimators. In Giuseppe Amato, Richard Connor, Fabrizio Falchi, and Claudio Gennaro, editors, *Similarity Search and Applications: 8th International Conference, SISAP 2015, Glasgow, UK, October 12-14, 2015, Proceedings*, pages 125–137. Springer International Publishing, Cham, 2015.

[34] Guénaël Cabanes and Younès Bennani. Change detection in data streams through unsupervised learning. In *The 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, Australia, June 10-15, 2012*, pages 1–6, 2012.

[35] Guénaël Cabanes, Younès Bennani, Renaud Destenay, and André Hardy. A new topological clustering algorithm for interval data. *Pattern Recognition*, 46(11):3030–3039, 2013.

[36] Guénaël Cabanes, Younès Bennani, and Dominique Fresneau. Enriched topological learning for cluster detection and visualization. *Neural Networks*, 32:186 – 195, 2012. Selected Papers from IJCNN 2011.

[37] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. *Density-Based Clustering Based on Hierarchical Density Estimates*, pages 160–172. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[38] Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(1):5, 2015.

[39] Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. *Density-Based Clustering over an Evolving Data Stream with Noise*, pages 328–339. Society for industrial and applied mathematics, 2006.

[40] Gilles Caraux and Sylvie Pinloche. PermutMatrix: a graphical environment to arrange gene expression profiles in optimal linear order. *Bioinformatics*, 21(7):1280–1281, 2005.

[41] Brad Chapman and Jeffrey Chang. Biopython: Python Tools for Computational Biology. *SIGBIO Newsl.*, 20(2):15–19, 2000.

[42] Peter C. Cheeseman, Matthew Self, James Kelly, Will Taylor, Don Freeman, and John C. Stutz. Bayesian classification. In *AAAI*, volume 88, pages 607–611, 1988.

[43] Chun-houh Chen, Wolfgang Hrdle, Antony Unwin, Chun-houh Chen, Wolfgang Hrdle, and Antony Unwin. *Handbook of Data Visualization (Springer Handbooks of Computational Statistics)*. Springer-Verlag TELOS, Santa Clara, CA, USA, 1 edition, 2008.

[44] Yixin Chen and Li Tu. Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 133–142, New York, NY, USA, 2007. ACM.

[45] Safouane Cherki, Parisa Rastin, Guenael Cabanes, and Basarab Matei. Improved sparse prototyping for relational K-means. In *IEEE Symposium Series on Computational Intelligence (SSCI'16), Athens, Greece*, pages 1–8, 2016.

[46] Tommy W. S. Chow and Sitao Wu. An online cellular probabilistic self-organizing map for static and dynamic data sets. *IEEE Trans. on Circuits and Systems*, 51-I(4):732–747, 2004.

[47] Walczyk Chris. Top 100 Greatest Actors of All Time (The Ultimate List). 2012.

[48] Guillaume Cleuziou, Lionel Martin, Viviane Clavier, and Christel Vrain. DDOC: overlapping clustering of words for document classification. In *SPIRE*, pages 127–128, 2004.

[49] William Jay Conover. *Practical nonparametric statistics.* Wiley, New York, 1981.

[50] The UniProt Consortium. UniProt: a hub for protein information. *Nucleic Acids Research*, 43(D1):D204–D212, 2015.

[51] Don Coppersmith and Shmuel Winograd. Computational algebraic complexity editorial Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

[52] Ido Dagan, Lillian Lee, and Fernando Pereira. Similarity-based methods for word sense disambiguation. In *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics*, EACL '97, pages 56–63, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics.

[53] Xuan Hong Dang, Vincent Lee, Wee Keong Ng, Arridhana Ciptadi, and Kok Leong Ong. An em-based algorithm for clustering data streams in sliding windows. In *Proceedings of the 14th International Conference on Database Systems for Advanced Applications*, DASFAA '09, pages 230–235, Berlin, Heidelberg, 2009. Springer-Verlag.

[54] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of the johnson-lindenstrauss lemma. Technical report, 1999.

[55] Nedim Dedić and Clare Stanier. *Towards Differentiating Business Intelligence, Big Data, Data Analytics and Knowledge Discovery*, pages 114–122. Springer International Publishing, Cham, 2017.

[56] Li Deng and Xiao Li. Machine learning paradigms for speech recognition: An overview. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):1060–1089, May 2013.

[57] Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, Jan 2001.

[58] Chris Ding and Xiaofeng He. Linearized Cluster Assignment via Spectral Ordering. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, page 30, New York, NY, USA, 2004. ACM.

[59] Sandrine Dudoit and Jane Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology*, 3(7):research0036.1, 2002.

[60] Tyrone E. Duncan. On the calculation of mutual information. *SIAM Journal on Applied Mathematics*, 19(1):215–220, 1970.

[61] Denise Earle and Catherine B Hurley. Advances in Dendrogram Seriation for Application to Visualization. *Journal of Computational and Graphical Statistics*, 24(1):1–25, 2015.

[62] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[63] Manfred M Fischer. Expert systems and artificial neural networks for spatial analysis and modelling: Essential components for knowledge-based geographical information systems. *Machine learning*, 1992.

[64] Edward B Fowlkes and Colin L Mallows. A Method for Comparing Two Hierarchical Clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.

[65] Brendan J Frey and Delbert Dueck. Clustering by Passing Messages Between Data Points. *Science*, 315:972–976, 2007.

[66] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[67] Bernd Fritzke. A self-organizing network that can follow non-stationary distributions. In Springer, editor, *Proc. of the International Conference on Artificial Neural Networks '97*, pages 613–618, 1997.

[68] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1606–1611, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[69] Joao Gama. *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 1st edition, 2010.

[70] Johannes Gehrke, Flip Korn, and Divesh Srivastava. On Computing Correlated Aggregates Over Continual Data Streams. In *Special Interest Group on Management of Data Conference*, pages 13–24, 2001.

[71] Izrail Moiseevich Gel'fand and Akiva Moiseevich Yaglom. *Calculation of the amount of information about a random function contained in another such function*. American Mathematical Society Providence, 1959.

[72] Chris Giannella, Jiawei Han, Jian Pei, Xifeng Yan, and Philip S. Yu. Mining frequent patterns in data streams at multiple time granularities, 2002.

[73] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[74] Allan D. Gordon. *Classification*. Chapman and Hall//CRC, Boca Raton, Florida, second edi edition, 1999.

[75] Ardeshir A. Goshtasby. *Similarity and Dissimilarity Measures*, pages 7–66. Springer London, London, 2012.

[76] Gunnar Gruvaeus and Howard Wainer. Two additions to hierarchical cluster analysis. *British Journal of Mathematical and Statistical Psychology*, 25(2):200–206, 1972.

[77] Sudipto Guha and Boulos Harb. Approximation Algorithms for Wavelet Transform Coding of Data Streams. *IEEE Transactions on Information Theory*, 54(2):811–830, 2008.

[78] Sudipto Guha, Chulyun Kim, and Kyuseok Shim. Xwave: Optimal and approximate extended wavelets. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 288–299. VLDB Endowment, 2004.

[79] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE Trans. on Knowl. and Data Eng.*, 15(3):515–528, March 2003.

[80] Sudipto. Guha, Nina. Mishra, Rajeev. Motwani, and Liadan. O'Callaghan. Clustering data streams. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, FOCS '00, pages 359–, Washington, DC, USA, 2000. IEEE Computer Society.

[81] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. *Special Interest Group on Management of Data Record*, 27(2):73–84, 1998.

[82] Michael Hahsler, Christian Buchta, and Kurt Hornik. Infrastructure for seriation, 2016.

[83] Michael Hahsler, Kurt Hornik, and Christian Buchta. Getting things in order: an introduction to the R package seriation. *Journal of Statistical Software*, 25(3):1–34, 2008.

[84] Barbara Hammer, Alexander Hasenfuss, Fabrice Rossi, and Marc Strickert. Topographic Processing of Relational Data. In *Proceedings of 6th International Workshop on Self-Organizing Maps*, Bielefeld (Germany), 2007.

[85] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[86] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[87] Amiram Harten. ENO Schemes with Subcell Resolution. *Journal of Computational Physics*, 83:148–184, 1989.

[88] Erez Hartuv and Ron Shamir. A Clustering Algorithm based on Graph Connectivity. *Information Processing Letters*, 76:175–181, 1999.

[89] Sattar Hashemi, Ying Yang, Zahra Mirzamomen, and Mohammadreza Kangavari. Adapted one-vs-all decision trees for data stream classification, 2009.

[90] Richard J. Hathaway, John W. Davenport, and James C. Bezdek. Relational duals of the c-means clustering algorithms. *Pattern Recognition*, 22(2):205–212, 1989.

[91] Einar Hille. *Analytic Function Theory*. Number vol.˜2 in AMS Chelsea Publishing Series. AMS Chelsea Publishing, 2005.

[92] Alexander Hinneburg and Hans-Henning Gabriel. Denclue 2.0: Fast clustering based on kernel density estimation. In Michael R. Berthold, John Shawe-Taylor, and Nada Lavrač, editors, *Advances in Intelligent Data Analysis VII*, pages 70–80, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[93] Alexander Hinneburg and Daniel A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, KDD'98, pages 58–65. AAAI Press, 1998.

[94] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[95] Lawrence Hubert, Phipps Arabie, and Jacqueline Meulman. *Combinatorial Data Analysis*. Society for Industrial and Applied Mathematics, 2001.

[96] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 97–106, New York, NY, USA, 2001. ACM.

[97] Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted voronoi diagrams and randomization to variance-based k-clustering (Extended Abstract). In *IN SCG94: Proceedings of the Tenth Annual Symposium on Computational Geometry*, pages 332–339. ACM Press, 1994.

[98] Antonio Irpino and Rosanna Verde. A new Wasserstein based distance for the hierarchical clustering of histogram symbolic data. In Batanjeli, Bock, Ferligoj, and Ziberna, editors, *Data Science and Classification*, pages 185–192. Springer, Berlin, 2006.

[99] Antonio Irpino, Rosanna Verde, and Francisco de A. T. de Carvalho. Fuzzy clustering of distributional data with automatic weighting of variable components. *Inf. Sci.*, 406:248–268, 2017.

[100] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[101] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[102] Anil K. Jain and Patrick J. Flynn. Image segmentation using clustering. In *Advances in Image Understanding: A Festschrift for Azriel Rosenfeld*, 65–83. IEEE Press, 1996.

[103] Anil K. Jain and Martin H. C. Law. Data Clustering: A User's Dilemma. In Sankar K Pal, Sanghamitra Bandyopadhyay, and Sambhunath Biswas, editors, *Pattern Recognition and Machine Intelligence*, volume 3776 of *Lecture Notes in Computer Science*, pages 1–10. Springer Berlin Heidelberg, 2005.

[104] Anil K. Jain, Narasimha M. Murty, and Patrick J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[105] Matthew A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[106] William B. Johnson, Joram Lindenstrauss, and Gideon Schechtman. Extensions of lipschitz maps into banach spaces. *Israel Journal of Mathematics*, 54(2):129–138, Jun 1986.

[107] George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, August 1999.

[108] Randa Kassab and Frédéric Alexandre. Incremental data-driven learning of a novelty detection model for one-class classification with application to high-dimensional noisy data. *Machine Learning*, 2007.

[109] Leonard Kaufman and Peter Rousseeuw. *Clustering by means of medoids*. North-Holland, 1987.

[110] Mark G. Kelly, David J. Hand, and Niall M. Adams. The impact of changing populations on classifier performance. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 367–371, New York, NY, USA, 1999. ACM.

[111] Jon M. Kleinberg. An Impossibility Theorem for Clustering. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 463–470. MIT Press, 2003.

[112] Teuvo Kohonen. The Self-Organizing Map. *Proceedings of the IEEE*, pages 1468–1480, 1990.

[113] Jeremy Z. Kolter and Marcus A. Maloof. Using additive expert ensembles to cope with concept drift. In *ICML*, pages 449–456, 2005.

[114] Philipp Kranen, Stephan Günnemann, Sergej Fries, and Thomas Seidl. Mc-tree: Improving bayesian anytime classification. In *Proceedings of the 22Nd International Conference on Scientific and Statistical Database Management*, SSDBM'10, pages 252–269, Berlin, Heidelberg, 2010. Springer-Verlag.

[115] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowl. Discov. Data*, 3(1):1:1–1:58, March 2009.

[116] Ruschendorf L. Wasserstein metric. In Hazewinkel M, editor, *Encyclopedia of mathematics*. Springer, Berlin, 2001.

[117] Thomas K. Landauer and Susan T. Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240, 1997.

[118] Mark Last. Online classification of nonstationary data streams. *Intell. Data Anal.*, 6(2):129–147, April 2002.

[119] François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 296–303, New York, NY, USA, 2014. ACM.

[120] Hyoungjoo Lee and Sungzoon Cho. SOM-Based Novelty Detection Using Novel Data. In Marcus Gallagher, James M Hogan, and Frédéric Maire, editors, *IDEAL*, volume 3578 of *Lecture Notes in Computer Science*, pages 359–366. Springer, 2005.

[121] Daniel Leite, Pyramo Costa, and Fernando Gomide. Evolving granular neural networks from fuzzy data streams. *Neural Netw.*, 38:1–16, February 2013.

[122] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966.

[123] Warren T. Liao. Clustering of time series data—a survey. *Pattern Recognition*, 38(11):1857 – 1874, 2005.

[124] Moshe Lichman. UCI Machine Learning Repository, 2013.

[125] Innar Liiv. Seriation and matrix reordering methods: An historical overview. *Statistical Analysis and Data Mining*, 3(2):70–91, 2010.

[126] Stuart Lloyd. Least Squares Quantization in PCM. *IEEE Trans. Inf. Theor.*, 28(2):129–137, 2006.

[127] Sebastian Lühr and Mihai Lazarescu. Incremental clustering of dynamic data streams using connectivity based representative points. *Data Knowl. Eng.*, 68(1):1–27, January 2009.

[128] Kevin Lund and Curt Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208, Jun 1996.

[129] Kevin Lund, Curt Burgess, and Ruth Ann Atchley. Semantic and associative priming in high-dimensional semantic space. In *Proceedings of the 17th Annual Conference of the Cognitive Science Society*, pages 660–665, 1995.

[130] David J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002.

[131] Frederik Maes, Dirk Vandermeulen, and Paul Suetens. Medical image registration using mutual information. *Proceedings of the IEEE*, 91(10):1699–1722, 2003.

[132] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The Planar k-Means Problem is NP-Hard. In Sandip Das and Ryuhei Uehara, editors, *WALCOM: Algorithms and Computation: Third International Workshop, WALCOM 2009, Kolkata, India, February 18-20, 2009. Proceedings*, pages 274–285. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[133] Oded Maimon and Lior Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[134] Colin L. Mallows. A note on asymptotic joint normality. *Annals of Mathematics Statistics*, 43(2):508–515, 1972.

[135] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *Very Large Data Base*, pages 346–357, 2002.

[136] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.

[137] Thomas M. Martinetz and Klaus J. Schulten. A "neural-gas" network learns topologies. In Teuvo Kohonen, K Mäkisara, O Simula, and J Kangas, editors, *Artificial Neural Networks*, pages 397–402. Elsevier Science Publishers, Amsterdam, 1991.

[138] Basarab Matei and Sylvain Meignen. Nonlinear Cell-Average Multiscale Signal Representations: Application to Signal Denoising. *Signal Processing*, 92:2738–2746, 2012.

[139] David W Matula. Graph Theoretic Techniques for Cluster Analysis Algorithms. In Van J. Ryzin, editor, *Classification and Clustering*, pages 95–129. Academic Press, New York, 1977.

[140] Irina Matveeva. *Generalized Latent Semantic Analysis for Document Representation*. PhD thesis, Chicago, IL, USA, 2008. AAI3300440.

[141] Karl. Menger. New foundation of euclidean geometry. *American Journal of Mathematics*, 53(4):721– 745, 1931.

[142] Rada Mihalcea, Courtney Corley, and Carlo Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 775–780. AAAI Press, 2006.

[143] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[144] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244, 1990.

[145] David W Mount. Sequence and genome analysis. *Bioinformatics: Cold Spring Harbour Laboratory Press: Cold Spring Harbour*, 2, 2004.

[146] J. Ian Munro and Mike S. Paterson. Selection and sorting with limited storage. In *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, pages 253–258, Oct 1978.

[147] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, March 2001.

[148] Peter Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963.

[149] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On Spectral Clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, 2001.

[150] Liadan O'Callaghan, Nina Mishra, Adam Meyerson, Sudipto Guha, and Rajeev Motwani. Streaming-data algorithms for high-quality clustering. In *Proceedings of IEEE International Conference on Data Engineering*, 2002.

[151] Alexander Ocsa, Carlos Bedregal, and Ernesto Cuadros-Vargas. DB-GNG: A constructive Self-Organizing Map based on density. *Prodeeding of International Joint Conference on Neural Networks*, pages 1506–1511, aug 2007.

[152] Carlos Ordonez and Edward Omiecinski. Efficient Disk-Based K-Means Clustering for Relational Databases. *IEEE Trans. on Knowl. and Data Eng.*, 16(8):909–921, 2004.

[153] Sheetal Reddy Pamudurthy, S Chandrakala, and C Chandra Sakhar. Local Density Estimation based Clustering. *Prodeeding of International Joint Conference on Neural Networks*, pages 1338–1343, aug 2007.

[154] Nam Hun Park and Won Suk Lee. Statistical grid-based clustering over data streams. *SIGMOD Rec.*, 33(1):32–37, March 2004.

[155] Nam Hun Park and Won Suk Lee. Cell trees: An adaptive synopsis structure for clustering multi-dimensional on-line data streams. *Data and Knowledge Engineering*, 63(2):528 – 549, 2007.

[156] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[157] Frits H. Post, Gregory M. Nielson, and Georges-Pierre Bonneau. *Data visualization : the state of the art*. Boston : Kluwer, 2003.

[158] Piyush Rai, Hal Daumé, and Suresh Venkatasubramanian. Streamed learning: One-pass svms. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1211–1216, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[159] Sankar Rajagopal. Customer data clustering using data mining technique. *CoRR*, abs/1112.2663, 2011.

[160] William M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[161] Parisa Rastin and Basarab Matei. Incremental Matrix Reordering for Similarity-Based Dynamic Data sets. In *Proceeding of the International Joint Conference on Neural Informatin Processing (ICONIP)*, ICONIP'17, 2017.

[162] Parisa Rastin and Basarab Matei. Prototype-based Clustering for Relational Data using Barycentric Coordinates. In *Proceeding of the International Joint Conference on Neural Networks (IJCNN)*, IJCNN'18, 2018.

[163] Parisa Rastin, Basarab Matei, Guénaël Cabanes, and Ibtissame Ibtissame. Signal-Based Autonomous Clustering for Relational Data. In *Proceeding of the International Joint Conference on Neural Networks (IJCNN)*, IJCNN'17, 2017.

[164] Parisa Rastin, Tong Zhang, and Guénaël Cabanes. *A New Clustering Algorithm for Dynamic Data*, pages 175–182. Springer International Publishing, Cham, 2016.

[165] William S Robinson. *A Method for Chronologically Ordering Archaeological Deposits*. American Antiquity, 1951.

[166] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[167] Daniel J. Rosenkrantz, Richard E. Stearns, and Philip M. Lewis II. An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing*, 6(3):563–581, 1977.

[168] Fabrice Rossi. How Many Dissimilarity/Kernel Self Organizing Map Variants Do We Need? In Thomas Villmann, Frank-Michael Schleif, Marika Kaden, and Mandy Lange, editors, *Advances in Self-Organizing Maps and Learning Vector Quantization (Proceedings of the 10th International Workshop on Self Organizing Maps, WSOM 2014)*, volume 295 of *Advances in Intelligent Systems and Computing*, pages 3–23, Mittweida (Germany), 2014. Springer International Publishing.

[169] Fabrice Rossi, Alexander Hasenfuss, and Barbara Hammer. Accelerating Relational Clustering Algorithms With Sparse Prototype Representation. In *Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 07)*, Bielefeld (Germany), 2007.

[170] Jörg Sander. *Generalized density based clustering for spatial data mining.* Herbert Utz Verlag, 1999.

[171] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, Jun 1998.

[172] Thomas Seidl, Ira Assent, Philipp Kranen, Ralph Krieger, and Jennifer Herrmann. Indexing density models for incremental learning and anytime classification on data streams. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, pages 311–322, New York, NY, USA, 2009. ACM.

[173] Claude E. Shannon and Warren Weaver. *The mathematical theory of communication.* 1949.

[174] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[175] Jonathan A. Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, André C. P. L. F. de Carvalho, and João Gama. Data Stream Clustering: A Survey. *ACM Comput. Surv.*, 46(1):13–31, 2013.

[176] Amit Singhal. Modern Information Retrieval: A Brief Overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[177] Toby Smith and Damminda Alahakoon. *Growing Self-Organizing Map for Online Continuous Clustering*, pages 49–83. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[178] Peter H. A. Sneath and Robert R. Sokal. *Numerical Taxonomy: The Principles and Practice of Numerical Classification.* W.H. Freeman and Company, San Francisco, USA, 1973.

[179] Michael Steinbach, George Karypis, and Vipin Kumar. A Comparison of Document Clustering Techniques. In Marko Grobelnik, Dunja Mladenic, and Natasa Milic-Frayling, editors, *KDD-2000 Workshop on Text Mining, August 20*, pages 109–111, Boston, MA, 2000.

[180] Andrew James Stothers. On the complexity of matrix multiplication, 2010.

[181] Nick W. Street and YongSeog Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 377–382, New York, NY, USA, 2001. ACM.

[182] Alexander Strehl and Joydeep Ghosh. Cluster ensembles: a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, 3:583–617, 2003.

[183] Colin Studholme, Derek L. G. Hill, and David J. Hawkes. An overlap invariant entropy measure of 3D medical image alignment. *Pattern Recognition (PR)*, 32(1):71–86, 1999.

[184] Dimitris K. Tasoulis, Gordon Ross, and Niall M. Adams. Visualising the cluster structure of data streams. In *Proceedings of the 7th International Conference on Intelligent Data Analysis*, IDA'07, pages 81–92, Berlin, Heidelberg, 2007. Springer-Verlag.

[185] Anbupalam Thalamuthu, Indranil Mukhopadhyay, Xiaojing Zheng, and George C. Tseng. Evaluation and comparison of gene clustering methods in microarray analysis. *Bioinformatics*, 22(19):2405, 2006.

[186] Sergios Theodoridis and Konstantinos Koutroumbas. Pattern recognition and neural networks. In *Machine Learning and Its Applications*, pages 169–195. Springer, 2001.

[187] Alexander P. Topchy, Anil K. Jain, and William F. Punch. Clustering Ensembles: Models of Consensus and Weak Partitions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(12):1866–1881, 2005.

[188] Alexey Tsymbal. The problem of concept drift: Definitions and related work. Technical report, 2004.

[189] Komkrit Udommanetanakit, Thanawin Rakthanmanon, and Kitsana Waiyamai. E-stream: Evolution-based technique for stream clustering. In *Proceedings of the 3rd International Conference on Advanced Data Mining and Applications*, ADMA '07, pages 605–615, Berlin, Heidelberg, 2007. Springer-Verlag.

[190] Alfred Ultsch. Clustering with SOM: U*C. In *Proceedings of the Workshop on Self-Organizing Maps*, pages 75–82, 2005.

[191] Guido van Rossum. Extending and embedding the Python interpreter. Report CS-R9527, 1995.

[192] Rosanna Verde, Antonio Balzanella, and Antonio Irpino. Order statistics for histogram data and a box plot visualization tool. In *Symbolic Data Analysis and Visualization: Special Issue in honor of Monique Noirhomme-Fraiture*, pages 29–48, 2015.

[193] Rosanna Verde, Antonio Irpino, and Antonio Balzanella. Dimension reduction techniques for distributional symbolic data. *IEEE Trans. Cybernetics*, 46(2):344–355, 2016.

[194] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[195] Li Wan, Wee Keong Ng, Xuan Hong Dang, Philip S. Yu, and Kuan Zhang. Density-based clustering of data streams at multiple resolutions. *ACM Trans. Knowl. Discov. Data*, 3(3):14:1–14:28, July 2009.

[196] Wei Wang, Jiong Yang, and Richard R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 186–195, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[197] Joe H. Ward. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

[198] Gerhard Widmer and Miroslav Kubat. Learning in the Presence of Concept Drift and Hidden Contexts. *Machine Learning*, 23(1):69–101, 1996.

[199] Wikipedia contributors. Wikipedia, the free encyclopedia, 2017.

[200] Zhenyu Wu and Richard M. Leahy. An approximation method of evaluating the joint likelihood for first-order GMRFs. *IEEE Transactions on Image Processing*, 2(4):520–523, 1993.

[201] Xiaowei Xu, Martin Ester, Hans-Peter Kriegel, and Jörg Sander. A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases. In *ICDE '98: Proceedings of the Fourteenth International Conference on Data Engineering*, pages 324–331, Washington, DC, USA, 1998. IEEE Computer Society.

[202] Shi-Hong Yue, Ping Li, Ji-Dong Guo, and Shui-Geng Zhou. Using Greedy algorithm: DBSCAN revisited II. *Journal of Zhejiang University SCIENCE*, 5(11):1405–1412, 2004.

[203] Shi-Hong Yue, Ping Li, Ji-Dong Guo, and Shui-Geng Zhou. A statistical information-based clustering approach in distance space. *Journal of Zhejiang University SCIENCE*, 6(1):71–78, 2005.

[204] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96, pages 103–114, New York, NY, USA, 1996. ACM.

[205] Aoying Zhou, Feng Cao, Weining Qian, and Cheqing Jin. Tracking clusters in evolving data streams over sliding windows. *Knowl. Inf. Syst.*, 15(2):181–214, May 2008.