UNIVERSITE CATHOLIQUE DE LOUVAIN

LOUVAIN SCHOOL OF MANAGEMENT

TITRE : Clustering of social networks, a managerial tool : comparison of methods

Promoteur : Marco Saerens

Mémoire-recherche présenté par Joëlle Van Damme
en vue de l'obtention du titre de
Master en ingénieur de gestion

ANNEE ACADEMIQUE 2014-2015

I would like to thank my thesis' director Marco Saerens for his help, expertise and assistance during the last two years.

I am also thankful to Betrand Lebichot, teaching assistant, for his availability and deep knowledge of Matlab, both proven determinant for the most technical parts of this thesis.

Finally, I would like to thank my friends whose continuous support kept me on track (and on time!) during the last months.

Contents

1 Introduction

With companies facing a pressure for more efficiency at lower costs everyday [28], we saw the need for accuracy and speed in data analysis grow exponentially over the last years [16]. Therefore, firms require strong analytical tools to gain a competitive advantage in mastering analysis techniques.

Clustering finds its place among those techniques as it allows companies to group customers (or business units, or assets, or...) according to different characteristics, and to customize one's approach of an issue. It is therefore of vital importance to incorporate clustering techniques in the business analytics' world. It is namely used to study social networks and graphs (this is the case with this thesis), and allows to detect dense communities in those graphs and networks. It is particularly used in the telecoms industry, where dense communities allow to identify similar consumers' behaviors, including the probability to join the competition.

Clustering relates to grouping or segmenting a collection of objects into subsets or "clusters", with the characteristics that objects within each cluster are more closely related to one another than objects assigned to different clusters [17]. A clustering technique will therefore provide a partition of the set of objects (the graph, the network) into distinct classes, each containing objects relatively similar to one another, while objects belonging to different classes are less similar.

Different clustering techniques exist to perform that kind of partition.

- The first one I will study is the distance-based k-means, where the distance between the nodes have been computed. It basically corresponds to a k-means-like two-steps iterative algorithm based on a distance (or dissimilarity) matrix, instead of features. The goal will be to partition the nodes by minimizing the total within cluster sum of distances [13].

- The second one is the kernel k-means. The principle remains similar to the distance-based k-means, with the difference that we will use a kernel matrix instead of a distance matrix. This kernel will be computed first via a simple transformation known as the "kernel trick", then using a more complicated kernel: the sigmoid

commute-time kernel. The intuition residing behind the use of kernels and a detailed explanation about the differences between kernel matrices will be developed in section 5.2.1.

- The third algorithm I will study is known as the "Louvain Method", and was developed by Vincent Blondel (quoted in [13]). Here, the criterion to optimize is the modularity (a measure of the structure of graphs), and the logic of the algorithm in slightly different, as we will start with a partition where each node is a cluster in itself and group them if there is an improvement in modularity.

- Finally, the fourth method that I will use is the possibilistic fuzzy c-means algorithm, and was implemented by myself. The intuition behind this method is to attach the memberships of the nodes to an idea of probability, while abandoning the traditional constraint of the c-means algorithm.

This thesis is structured in three parts.

- The first part ("Theoretical Background") will detail the theoretical foundations of machine learning, similarity measures and clustering in particular. I will present the most popular clustering techniques currently used (distance-based k-means, kernel k-means and Louvain method), before displaying my own implementation of a new clustering algorithm: the possibilistic fuzzy c-means algorithm.

- The second part ("Experiments") will compare those different algorithms according to three performance measures: the NMI, the Adjusted Rand Index and the classification rate, and try to identify if one method gives significantly better results than another. For that purpose, I will launch each algorithm 50 times while preserving the average NMI, ARI and classification rate over those 50 runs, those means serving as basis for my comparison. Once this question answered, I will try to improve the results obtained with the first research question by using one of the most popular tools in machine learning: the kernel matrices.

My research questions are therefore firstly, "does the possibilistic fuzzy c-means (PFCM) algorithm give better clustering results than the distance-based k-means, the kernel k-means, the kernel fuzzy c-means and the Louvain method?" and secondly, "does using a Gaussian kernel-based possibilistic c-means yield better results than the "regular"possibilistic c-means?". We will therefore have a first part designed to support the hypotheses, and a second part to test them.

- The third part, "Further work and conclusions", will present some prospective developments and variations that could be worth investigating in the future in order to improve the performance of the clustering techniques.

Finally, I will cover in this thesis some popular applications of clustering in the business world, and explain how mastering those techniques can improve a firm's position.

I immediately must warn the prospective reader that there are a lot of concepts and tools to support my work, some of which that might be difficult to understand for a newcomer in the field of data mining. I therefore tried to explain my ideas in the simplest way possible, and will offer the reader a lot of references to assist him/her in understanding the theoretical foundations of the thesis.

In terms of contribution, I wanted this thesis to be as comprehensive as possible, to make a real step forward in the field of clustering. In particular, I have conceived and implemented in Matlab a new clustering algorithm (the possibilistic fuzzy c-means algorithm) based on an extensive literature review. I also implemented an adapted version of the kernel k-means to study the effect of the fuzziness of the membership (the kernel fuzzy c-means algorithm), developed the code for calculating the commute-time distance, the corrected commute-time distance and the corrected sigmoid commute-time kernel. I validated all my results using not one but three performance measures. I tried to improve those results by varying the different parameters on a wide range, then by trying out different distance measures and kernel matrices to make sure I was obtaining the best possible results. For each method, I launched the algorithm at least 50 times to alleviate the possible effects of the initialization procedure. Finally, I studied my results with a

global perspective, using Friedman's test to determine whether the differences in results were statistically significant on a broader scale. Each step of this thesis was thus criticized along the way and adapted to ensure its robustness.

There is a lot of work and internal validation behind this thesis, to make sure the progress made in the clustering techniques are technically and theoretically sound and reliable.

I hope this thesis will provide the reader with a lot of insights into the world of clustering and trigger an interest for the use of clustering techniques in the business environment.

Part I. Theoretical Background

2 Machine learning

"Machine learning" , also called "data mining", is the field of the multivariate statistics applied to computer science and engineering. Indeed, multivariate statistics are a very powerful tool, used in many domains (biostatistics, econometrics,...), and its application in the computer science domain (data mining) will be at the core of this thesis.
Machine learning can be described as a set of methods that automatically detect patterns in data and are able to use the uncovered patterns to predict future data or to assist in the decision making processes. A good model is defined as one that can accurately predict an outcome [37].
We separate machine learning into two types: the predictive/supervised learning, and the descriptive/unsupervised learning approach.

The goal of the predictive approach is to derive a mapping from an input x to an output y , given the set of pairs $D = (x_i, y_i), i = 1, \ldots, N$. D is called the training set and N is the number of examples. Each input x_i is a vector of numbers representing characteristics (for instance the height and weight of a person). These are called features, or attributes. In general, x_i could be a complex object, such as an image, a sentence, an email, a time serie and so on. Similarly, the form of the output can in principle be anything. However, most methods assume that y_i is either a categorical or nominal variable from some finite set, $y_i \in 1, \ldots, C$ or that y_i is a real-valued scalar. When y_i is categorical, the problem is known as classification. When y_i is real-valued, the problem is known as regression [37].

The second type of machine learning is the descriptive (unsupervised) learning approach. Here, we are only given inputs and the goal is to detect patterns in the data. This is obviously a much less well-defined problem since we are not told what structure to look for. Therefore, instead of looking for known patterns, we will formalize our task as one of density estimation. That is, we will try to build models of the form $p(x_i \mid \theta)$ [37].

Clustering is an example of unsupervised clustering, and relates to grouping or segmenting a collection of objects into subsets or "clusters", with the characteristics that objects within each cluster are more closely related to one another than objects assigned to

different clusters [17].
With that in mind, we can now detail how such techniques can be valuable in a business context.

3 Clustering and management

Clustering becomes more and more present in the business world, as it allows a more precise lecture and analysis of the business environment. Hence, the graduate business engineer mastering those skills will prove himself/herself valuable to an enterprise as he/she will be able to use those tools in various domains of the company. As for the prospective manager, he/she will be able to recognize the potential of clustering methods in every business section, and therefore position its department in the front line of the data-driven business management methods.
Finally, clustering techniques appear to offer a competitive edge in various sectors of the business life. We will cover some of them below.

3.1 Clustering and marketing

Clustering finds some of its most popular applications in marketing. The main one [64] remains the market segmentation, leading to the targeting of customers, retaining of those, customization of the offer, improvement of the customer-relationship and satisfaction studies [33].
These will allow a company to maximize the return on marketing actions by designing a customized offer that targets a specific group (cluster) of clients sharing similar characteristics or consumption behaviors. They will also identify clients who are the most susceptible to join the competitor and give room for the creation of action plans specifically designed to retain those fragile customers. All those actions will improve the customer-relationship.

Market segmentation techniques aim at identifying groups of customers sharing similar requirements and taste. The ultimate refinement of this approach is the formation of "clusters of one", each cluster comprising a sole customer. Current evolutions in e-commerce suggest that this may one day be possible. Such clustering can be performed

on a consumption-basket base, but also on geographical, time-of-purchase, behavioral bases, and so on [55].

Clustering techniques also appear in the analysis of data extracted from market researches.Once a survey is completed, if several questions are found to measure the same aspect, those factors should be combined before any further analysis. After the data reduction, the cluster analysis can be performed so we can decide how many clusters are appropriate for the purpose of the research. Companies should therefore look for differences in the means of factors and name the clusters based on these differences [47]. These differences can be later used to design marketing strategies to ensure that the right customers are approached with the right product in the right way.

Clustering can also be used in more narrowly defined sectors, namely thanks to the growth of social networks, where people share video's, images and interests. We saw the development of image clustering for marketing purposes [1].The idea that uploaded images in social networks reflect consumerâĂŹs activities, interests and opinions finds its roots in visual sociology, where usually four different social reasons for sharing pictures have been identified (namely by van House [62]): images are uploaded to remember and to construct narratives of one's live and a sense of identity, to maintain relationships with others, to ensure that others see one as one wishes to be seen and for self-expression's purposes. Therefore, instead of asking consumers whether they like to go to skiing or sun-bathing, one could ask them to upload a picture of their favorite holiday destination. The same reasoning applies for every possible investigation field, as the images are assumed to contain valuable visual information. The problem of clustering such uploaded images is often to figure out how to organize the search results in order to facilitate the user's image retrieval.

The automatic clustering of images according to similar low-level features (e.g. based on extracted color distributions, textures, or forms) seems to be promising in specific domains of applications . However, most statistical software packages (especially SAS or SPSS) for image feature extraction and clustering are not available [1]. Even in more flexible systems like MATLAB or R, image clustering is only possible with additional packages. This field is thus still to be developed.

3.2 Clustering and sales

Sales is another big consumer of clustering techniques. One of the most widely recognized techinques is the clustering-based forecasting.
Traditionally, the most common forecasting techniques are statistical methods, such as multiple regression models and time series models. These models were proven very effective for data with simple trend and seasonality tendency. However, in the real world, the relationship between the factors or the past time series data (independent variables) and the sales (dependent variable) are nonlinear and quite chaotic [5]. Indeed, few items in a class can lead to large errors in its seasonality estimates. Therefore, correctly identifying classes will work as a buffer mechanism against this phenomenon, and will allow to have a better prediction of the demand per item, and of its price [41].

Clustering can also be used to determine the sales force efficiency. This idea is driven by the hypothesis that a good identification of customerâĂŹs segments will allow the sales teams to adapt their promotion tools and messages to the cluster they are targeting.
Agency relationship will also be better assessed via clustering. Indeed, the principal will be able to divide its agencies according to their performance level, target-groups and goals, and design the eventual corrective measures needed.

3.3 Clustering and finance

Finance departments use clustering techniques as well. Banks, insurance funds and other financial services companies are able to define clusters of clients bearing the same potential risk and therefore design customized credit conditions. They can define different groups of disasters based on their characteristics and design a reimbursement plan for each type of sinister. Finally, they will be able to identify suspicious comportment and group customers who might be at risk of fraud [4 and 33] .

Clustering analysis can be used to study the dependencies in the financial time series. In most cases, returns of financial time series show evidence against the multivariate normal distribution (Embrechts, Frey and McNeil (2005), [10]). Professionals therefore need new tools to measure the degree of dependance, for instance to determine if share closing prices

that move together correspond to companies belonging to the same economic sector. In such situations, clustering methods can be applied to a dissimilarity matrix corresponding to the degree of dependence between the companies. The correlations between stocks are recorded to quantify the degree of dissimilarity and derive the dissimilarity matrix. After having determined clusters of sectors based on the share prices, investors and financial directors can identify sector-related trends and diversify their portfolio if necessary [38]. This theory could be taken further away, and finance departments can even use clusters to describe market cycles in terms of clusters, and to improve their portfolio diversification if clusters are proven to perform better than industrial sectors. Indeed, studies showed that there was a strong (although not complete) similarity between clusters and industrial sectors [39], but that intra-cluster patterns are more robust over time than intrasectors and global patterns. Clusters show high persistence, while its persistence is strongly linked to the industrial sector it over-expresses. Hence, a cluster-based portfolio diversification would lead to a lower volatility than a sector-based one.
Furthermore, it would allow to identify which industrial groups have the greatest influence on the portfolio [66], and in the future, identify within a sector, which company is most likely to be a good indicator of future industry-wide movement.

3.4 Clustering and logistics

Logistics and supply chain functions tend to rely on clustering techniques.
In some cases, customer segmentation is important for the development of advanced logistical distribution strategies in response to the growing complexity in business logistical markets. Cluster-based routes mapping should be performed prior to vehicle dispatching and routing in the process distribution. The resulting supply chain will be more responsive and consumer-oriented (improved level of service and time-to-delivery), leading also to cost reduction (linked to the full and better exploitation of the routes) [18].

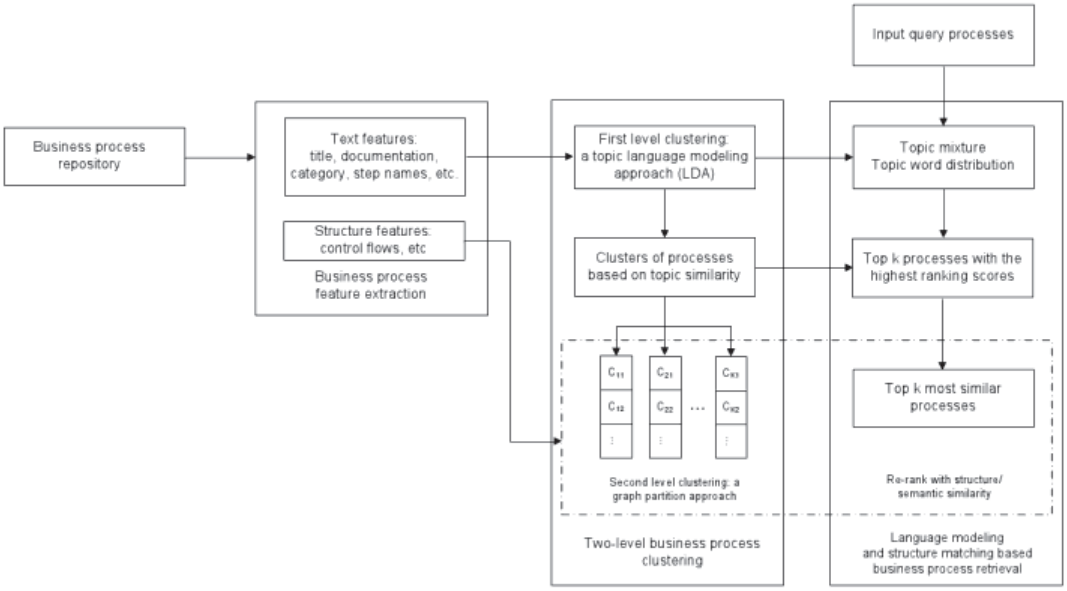
These clustering procedures for routing also showed a great interest in the last stage of the supply chain, the so called "last mile", and therefore turned out to be a great advantage in disaster situations. Balcik and al. [2] define the last mile distribution problem as "the last stage of humanitarian operations that involves delivery from local distribution

centers (tertiary hub) or from central warehouses (secondary hub) to a population in need (beneficiaries)". We can extend this definition to call it "the last mile delivery and pickup problem", where the last mile delivery regards materials transported from warehouses to affected locations and the last mile pickup is the evacuation of injured people. Using clustering (based on the demand for emergency supplies and medical aid), we can find the optimal allocation of warehouses and hospitals. Once the aggregate solution is obtained, the detailed routing problem within each cluster's sub-network is solved by letting relevant parts of the top level solution become problem parameters [42].
That idea, proven efficient in emergency situations, can of course be transposed in other cases to gain strategic advantage in this last mile.

3.5 Clustering and business processes

In a broader view, clustering is used to support business processes and business development.
Business processes are considered one of the core business assets of enterprises, leading sometimes to a significant competitive advantage. They are more and more standardized and automated in process-aware information systems. Since those systems are continuously accumulating business processes, systematic methods of analyzing and improving the processes become a necessity [22]. Large companies tend to have hundreds of business processes. Therefore, highlighting similarities and dissimilarities among them can be useful, namely for process management and maintenance. Commonly occurring activities will be managed more efficiently, similar processes will be reused for other projects implementations, and the commonalities identified will give room for improvement, including opportunities for standardization and consolidation. This might become really handy, for instance in the case of mergers and acquisitions. We can therefore cluster the business processes given their topic, structure and semantic similarities, and most importantly, their structural component [49].
The business processes retrieval procedure is structured as detailed in *figure 1*.

Figure 1: Business Process Retrieval procedure using clustering techniques



source: S. Rinderle-Ma, F. Toumani, and K. Wolf [49], 2011.

Other clustering techniques supporting business processes management rely on the transformation of business process models into vector models based on their structures such as activities and transitions, and the vectors are compared by Cosine similarity measure. Finally, they are clustered by the agglomerative hierarchical clustering algorithm. The results of the clustering experiments can be utilized to re-engineer the process models or support new process design by extracting their common patterns [22].

SME's can take a great advantage from clustering. Through clustering, enterprises can address problems related to their size, production process, marketing, procurement, market information and risks associated with demand fluctuations. They can also improve their competitive position. Once united in a cluster, they may take advantage of external players: presence of suppliers of raw materials, components, machinery and parts, presence of workers with sector-specific skills and so on. Furthermore, with clustering of enterprises, it becomes easier for government, universities, and other development supporting agencies to provide services. Indeed, the services and facilities would be very costly for the providers if given to individual enterprises in dispersed locations (Tambunan 2000

[58]; Humphrey and Schmitz 1995 [20]). Clustering allows to joint actions and increase the scope of projects. In effect, individual enterprises in a cluster can gain collective efficiency. Close proximity facilitates the establishment by enterprises in the locality of industrial links without substantial transaction costs or difficulties.

However, these economic advantages can only be achieved if the cluster has well-developed internal and external networks. Internal networks can be defined as business cooperation or links among enterprises within the cluster. External networks are business and other forms of relation between enterprises inside the cluster and actors outside the cluster such as suppliers of inputs, providers of business services, and so on. There is of course a risk of failure associated with the use of clusters as a catalyst for SME's development. In most failure cases, one or more critical success factors for successful cluster development were either not existing or not addressed correctly. Prerequisite for successful cluster development is the cluster's potential to access growing markets. However, because of the high level of centralization and standardization of the instruments in the policy making, the cluster's existing and potential market linkage has often been neglected in project design [57].

4 Clustering

4.1 Clustering principles

Cluster analysis has a variety of goals. All relate to grouping or segmenting a collection of objects into subsets or "clusters", with the characteristics that objects within each cluster are more closely related to one another than objects assigned to different clusters [17].
A clustering technique will therefore provide a partition of the set of objects into distinct classes. This requires the existence of three different tools: a measure of (dis)similarity between the objects, a criterion measuring the quality of a partition and an optimization technique for computing high-quality partition. The similarity measure could, for instance, be the similarity provided by a kernel (see section 4.2.1) on a graph, while the criterion could be the total within-cluster inertia (a measure of how internally coherent clusters are) induced by the kernel on a graph in the embedding space, as in the case of simple k-means clustering [13].
There are two kinds of inputs we might use when performing a clustering. In similarity-based clustering, the input is an $N \times N$ dissimilarity matrix or distance matrix D. In feature-based clustering, the input is an $N \times D$ feature matrix matrix X. Similarity-based clustering has the advantage that it allows for easy inclusion of domain-specific similarity or kernel functions. Feature-based clustering has the advantage that it is applicable to "raw", potentially noisy data [37].

More formally, this means that, given the set V of node vectors, the hard partition k-clustering of V is a partition of V into k sets (clusters) $C_1, ..., C_k$ so that:

- $C_i \neq \oslash$, $i = 1, ..., k$
- $\cup_{i=1}^{k} C_i = V$
- $C_i \bigcap C_j = \oslash$, $i = j$, $i, j = 1, ..., k$

where each node belongs to a single cluster. An alternative definition allows nodes to belong to all clusters k with a certain degree of membership, called $u_{ij} \in [0, 1]$. Introduced by Zadeh [24] , this is called fuzzy clustering and satisfies the following properties:

$\sum_{i=1}^{K} u_{ij} = 1 \; \forall j$ and $0 < \sum_{j=1}^{N} u_{ij} < N, \; \forall i$ where u_{ij} is the membership coefficient of the jth object in the ith cluster [8] .

According to Fouss, Saerens and Shimbo [13], there exist several different types of clustering algorithms, the most commonly used being :

- Top-down, divisive techniques (also referred as splitting methods). These procedures start from an initial situation where all the nodes of the graph are contained in only one cluster and then split the cluster into pieces by minimizing a criterion.
- Optimization techniques maximizing a criterion assessing the quality of the partition. A popular algorithm pertaining to this class is the kernel k-means, optimizing the within-cluster inertia.
- Bottom-up agglomerative techniques that start from a partition where each node is a cluster by itself. A greedy algorithm is then used to merge the similar (= highly inter-connected) nodes/clusters.

Let us be a bit more precise in those definitions.
When clustering data into groups, the issue of the existence of multiple subgroups might occur. Let K represent the number of clusters. Our first objective is to estimate the distribution over the number of clusters, $p(K \mid D)$. This will tell us if there are any subpopulations within the data.We can approximate the distribution $p(K \mid D)$ by its mode, K^* =argmax$_k p(K \mid D)$. The second goal is to estimate which cluster each point belongs to. If $z_i \in 1, \ldots, K$ represent the cluster to which data point i is assigned, we can infer which cluster each data point belongs to by computing $z_i^* = \text{argmax}_k p((z_i = k \mid x_i), D)$ [37].

4.2 Measuring (dis)similarity and proximity

A dissimilarity matrix D is a $N \times N$ matrix where N is the number of objects and each element $d_{ii'}$ records the proximity between the ith and $i'th$ objects. This matrix is then provided as input to the clustering algorithm. We thus have that $d_{i,i} = 0$ and $d_{i,j} \geq 0$ is a measure of distance between objects i and j. Using a similarity matrix S, we can

convert it to a dissimilarity matrix by applying any monotonically decreasing function,e.g. $D = max(S) - S$ [37].

Sometimes the data is represented directly in terms of proximity (alikeness or affinity) between pairs of objects. These can be either similarities or dissimilarities. Dissimilarities can be computed by evaluating how much objects differ from one another. Most often we have measurements x_{ij} for $i = 1, 2, \ldots, N$, on variables $j = 1, 2, \ldots, p$ (also called "attributes"). Since most of the popular clustering algorithms take a dissimilarity matrix as their input, we must first construct pairwise dissimilarities between the observations. We usually define a dissimilarity $d_j(x_{ij}, x_{i'j})$ between values of the jth attribute, and then define $D(x_i, x_{i'}) = \sqrt{\sum_{j=1}^{p} d_j^2(x_{ij}, x_{i'j})}$ as the dissimilarity between objects I and I'. The most common choice is the squared Euclidean distance: $d_j^2(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2$ [17].

4.3 Performance measures

Different performance measures can be used to determine the quality of a partition.
For the purpose of this thesis, we used three performance measures: the NMI (Normalized Mutual Information) score, the Adjusted Rand Index and the classification rate .

- The NMI algorithm compares the clusters found by our algorithms with the real clusters, and computes to which level they share the same information [71]. This is computed for all 50 iterations, and the mean of those scores is returned for each algorithm. The performance of each algorithm for every dataset is then evaluated by comparing the means of the NMI.

- The Adjusted Rand Index is a second performance measure. Consider a set of n objects $S = \{O_1, O_2, ..., O_n\}$ and suppose that $U = \{u_1, u_2, ..., u_R\}$ and $V = \{v_1, v_2, ..., v_C\}$ represent two different partitions of the objects in S such that $\cup_{i=1}^{R} u_i = S = \cup_{j=1}^{C} v_j$ and $u_i \cap u_{i'} = \phi = v_j \cap v_{j'}$ for $1 \leq i \neq i' \leq R$ and $1 \leq j \neq j' \leq C$. Given two partitions, U and V, with R and C subsets, respectively, a contingency table can be formed to indicate the group overlaps between $U$and V. In such table, a generic entry t_{rc}, represents the number of objects that were classified in the rth subset of partition R and in the cth subset of partition C. From the total number of

possible combinations of pairs $\binom{n}{2}$ from a given set we can represent the results in four different types of pairs [53]:

a - objects in a pair placed in the same group in U and in the same group in V ;

b - objects in a pair placed in the same group in U and in different groups in V ;

c - objects in a pair placed in the same group in V and in different groups in U and;

d - objects in a pair placed in different groups in U and in different groups in V .

The Rand Index can then be easily computed by:

$$RI = \frac{a+d}{a+b+c+d} \qquad \textbf{eq. 4.1}$$

There are some known problems with RI, such as the fact that the expected value of the RI of two random partitions does not take a constant value (say zero) or that the Rand statistic approaches its upper limit of unity as the number of clusters increases. With the intention to overcame these limitations researchers have created several different measures, among those the Adjusted Rand Index. The ARI can be computed by:

$$ARI = \frac{\binom{n}{2}(a+d) - [(a+b)(a+c)+(c+d)(b+d)]}{\binom{n}{2}^2 - [(a+b)(a+c)+(c+d)(b+d)]} \qquad \textbf{eq. 4.2}$$

with expected value zero and maximum value 1 [53].

- The classification rate simply returns the percentage of correct classification. However, this may lead to some erroneous conclusions. Consider the case of a two-class problem with one class having 90% of the cases. If all the outputs of the classification algorithm are from the majority class, we will get a CR value of 90% that can be misleading specially if one intends to detect and classify the minority class. Therefore one should remember that special care must be taken when using CR in problems with low representative classes [53].

5 Algorithms

Let us now describe in detail the different algorithms we are going to use for our clustering experiments. The different algorithms we will study are the distance-based k-means, the kernel k-means, the Louvain method and the possibilistic fuzzy c-means.The following section is based on the book by Fouss, Saerens and Shimbo [13].

5.1 Distance-based k-means

5.1.1 Description

According to Hastie and al. [17], the k-means algorithm is one of the most popular iterative descent clustering methods. It is intended for situations in which all variables are of the quantitative type, and the squared Euclidean distance is chosen as the dissimilarity measure.

It is supposed to be a simple and straightforward method for clustering data [54].

A distance-based k-means basically corresponds to a k-means-like two-steps iterative algorithm based on a distance (or dissimilarity) matrix, instead of features (it is assumed that a symmetric distance matrix, D (also denoted $\triangle$), containing the distances between every pair of nodes, has been pre-computed). The goal will be to partition the nodes by minimizing the total within cluster sum of distances. For this algorithm, the number of clusters m has to be provided by the user. Each cluster ($k = 1, \ldots, m$) is characterized by a prototype (chosen among the nodes belonging to the cluster). This prototype is the "most typical" element of this cluster, and will be chosen as the most central node of the cluster. The variable containing the index of the node chosen as the prototype of class k will be denoted as q_k. The vector of prototype labels is then $q = [q_1, q_2, .., q_m]^T$. We define the within-cluster sum-of-distances of class k as $J_k = \sum_{j \in C_k} \Delta_{q_k j}$ which is the sum of the distances between the nodes of cluster C_k to the prototype of the cluster. It quantifies the compactness of the cluster (it equals 0 when all elements of the cluster are superposed). The criterion should therefore be as low as possible [13].

The total within-cluster sum-of-distances is the sum of the within-cluster sum-of-distances:

$$J = \sum_{k=1}^{m} J_k = \sum_{k=1}^{m} \sum_{J \in C_k} \triangle_{q_k j} \qquad \textbf{eq. 5.1}$$

Let us introduce the membership values u_{ik} , which are equal to 1 if node i is assigned to cluster C_k and to 0 otherwise. The memberships are gathered in a $n \times m$ matrix U. Using the membership matrix, J becomes:

$$J = \sum_{j=1}^{n} \sum_{k=1}^{m} u_{jk} \triangle_{q_k j} \qquad \textbf{eq. 5.2}$$

and our objective is to minimize this criterion with respect to the prototypes and the membership values. We proceed in two steps.

Re-computation of the prototype step. First, we minimize the total sum-of-distances criterion with respect to the prototypes q_k while maintaining the memberships u_{ik} constant. We observe that J is minimal when each q_k minimizes $\sum_{j \in C_k} \triangle_{q_k j}$. Therefore, we must choose q_k such that $q_k = \text{argmin}_{i \in C_k} \{\triangle_{ij}\}$ which aims to choose, as prototype, the node that is most central to the cluster [13].

Re-allocation step. The second step minimizes J with respect to the binary membership values u_{ik} while keeping the prototypes constant. We saw that $\sum_{k=1}^{m} u_{jk} \triangle_{q_k j}$ should be minimal for each J. Since $\sum_{k=1}^{m} u_{jk} = 1$, $\forall j$, and the u_{jk} are binary, this is achieved by setting $u_{jk} = 1$ for the smallest $\Delta_{q_{kj}}$, and $u_{jk} = 0$ for the larger distances. In other words, node j should be assigned to the cluster corresponding to the lowest $\Delta_{q_{kj}}$ [13].
Thus, the optimal cluster allocation k^* of node j is:

$$k_j^* = \text{argmin}_{k \in \{1,\ldots,m\}} \{\triangle_{q_k j}\} \qquad \textbf{eq. 5.3}$$

This choice decreases the criterion J. By denoting the cluster label of node j as $l(j)$, we have to set $l(j) = k_j^*$.
Therefore, iterating these two steps decreases the criterion J at each iteration t by a non-negative amount $\delta J(t) = J(t-1) - J(t) \geq 0$. Since $J > 0$ and cannot become negative, $\sum_{t=0}^{\infty} \delta J(t) < J(0)$ so that $\delta J(t) \to 0$ and the objective function $J(t)$ converges to a local minimum [13].
The algorithm will be further detailed in the following section (see *figure 2*).

5.1.2 Algorithm

The distance-based k-means performs its clustering task given a certain input, before delivering the desired output: the clustering solution.

Input: The $n \times n$ symmetric distance matrix Δ containing distances between nodes and the number of clusters m.

Output: The final $n \times m$ cluster membership matrix U : $u_{ik} = 1$ if node i belongs to cluster k, 0 otherwise.

Figure 2: distance-based k-means algorithm

Initialization: randomly sample m prototypes $[q_1, q_2, \ldots, q_m]^T$ among the nodes. Here, the variable q_k contains the index of the node chosen as kth prototype.

$U \leftarrow Zeros(n, m)$

Repeat

For $i = 1$ **to** n **do**

$k_j^* \leftarrow \text{argmin}_{k \in \{1,\ldots,m\}} \{\triangle_{qkj}\}$

$l(j) \leftarrow k_j^*$ (l is the vector of cluster labels)

End for

For $k = 1$ **to** m **do**

$C_k \leftarrow \{$ Nodes indexes $i : l(i) = k \}$

$q_k \leftarrow \text{argmin}_{i \in C_k} \left\{ \sum_{j \in C_k} \triangle_{ij} \right\}$

End for

$J \leftarrow \sum_{k=1}^{m} \sum_{j \in C_k} \triangle_{qkj}$

until convergence of the objective function

For $i = 1$ **to** n **do**

$u_{il(i)} \leftarrow 1$

End for

Return U

source: adapted from Fouss, Saerens and Shimbo [13], 2014.

5.2 Kernel k-means

5.2.1 The kernels

Some problems cannot be easily be represented by a fixed-size feature vector. One approach to solve this problem is to assume that we have some way of measuring the similarity between objects that does not require preprocessing them into feature vector format. Let $K(x, x') \geq 0$ be a measure of similarity between objects $x, x' \in X$ (X being some abstract space). A commonly used similarity measure is the inner product: $k(x, y) = x^T y$, where x and y are feature vectors in the same embedding space. The function $k(x, y)$ is called a kernel function. It is thus a real-valued function of two arguments, $K(x, x') \in R$ for $x, x' \in X$. Typically, this function is assumed to be symmetric ($K(x, x') = K(x', x)$) and non-negative ($K(x, x') \geq 0$), so it can be interpreted as a measure of similarity [37]. Kernels benefit from some important properties. Firstly, they allow to compute the inner products in the feature space. That is, they project the data into a higher dimensional space where the clusters are more likely to be linearly separable. Another important property is that distances can be extracted from kernels.The kernel is therefore a valuable tool, able to capture a meaningful similarity measure while requiring very few computations in regards to explicit evaluation of the entirely mapped input space into feature space [8]. Given n objects $x_1, x_2, ..., x_n \in \Omega$, we can compute the kernel functions of the $n \times n$ pairs of objects i , j. We obtain a symmetric (n, n) kernel matrix $[K]_{ij} = k_{ij} = k(x_i, x_j)$. As it is positive semi-definite, the following properties are ensured [8]:

- $z^T K z \geq 0$ for all $z \in R_n$;
- all the eigenvalues are non-negative ;
- K is a inner product matrix: $[K]_{ij} = k_{ij} = x_i^T x_j$;
- Kcan be written as a diagonal matrix: $K = U \Lambda U^T$ where U is an orthogonal matrix, and $\Lambda \geq 0$.

In this paper, we will be interested in the kernels on graphs.
Assuming the graph G, we represent each node i by a vector in the r dimensions Euclidean space x_i . K is a semi-definite positive matrix of size $n \times n$ and rank r, with orthonor-

mal eigenvectors u_i and eigenvalues λ_i. From the fundamental spectral-decomposition theorem, we have $K = U\Lambda U^T$, with [8]:

- U, an orthonormal matrix of size $n \times r$, with as columns the normalized eigenvectors u_i of K,
- Λ, the diagonal matrix whose diagonal elements are the eigenvalues λ_i of K sorted by decreasing value. If X is a $n \times r$ matrix whose columns' elements are the product of the normalized eigenvectors of K and the square root corresponding eigenvalue is $\sqrt{\lambda_i}u_i$, then $X = U\Lambda^{\frac{1}{2}}$, and we obtain $K = U\Lambda U^T = (U\Lambda^{\frac{1}{2}})(U\Lambda^{\frac{1}{2}})^T = XX^T$. If x_i is a vector node corresponding to the columns of X^T or the rows of X, then the element i ,j of K can be written as $[K]_{ij} = k_{ij} = x_i^T x_j$ [8].

5.2.1.1 Deriving a kernel from a distance and vice-versa

Let's now see how to derive a kernel from a distance matrix. Before we proceed, we first need to have a look at the reversed equation and derive the Euclidean distances from the kernel. The Euclidean distance between two nodes i and j is defined as [8]:

$$\triangle_{ij}^2 = [\triangle]_{ij} = ||x_i - x_j||^2 = x_i^T x_i + x_j^T x_j - 2x_i^T x_j \qquad \textbf{eq. 5.4}$$

We have seen earlier that a kernel matrix can be expressed as the inner product: $k_{ij} = [K]_{ij} = x_i^T x_j$. Therefore, the distances can be expressed as a inner product function:

$$\triangle_{ij}^2 = k_{ii} + k_{jj} - 2k_{ij} \qquad \textbf{eq. 5.5}$$

We can rewrite this expression in a matrix form using the column vector $diag(K)$ containing the k_{ii}: (the elements on the diagonal of matrix K). The distance matrix D (or $\triangle$) is then:

$$\triangle^{(2)} = diag(K) * e^T + e * diag(K)^T - 2K \qquad \textbf{eq. 5.6}$$

where e is a column vector of ones and $\left[\triangle^{(2)}\right]_{ij} = \triangle_{ij}^2$. The reciprocal relation allows to derive the kernel from the distance matrix. However, the inner product depends on the origin of the system. This problem can easily be fixed by setting the origin of the coordinate system at the centroids of the cloud of point. Indeed, in that case, the sum of the inner

products is zero. Thus, if $\sum_i x_i = 0$ then $\sum_i k_{ij} = \sum_i x_i^T x_j = \sum_i x_j^T x_i = x_j^T(\sum_i x_i) = 0$ [8].

In matrix form, we want to center K by setting the sum of each of its lines and columns to zero: $Ke = 0$ and $e^T K = 0$. Since we derive K from the distance matrix, we start by centering the distance matrix D by multiplying it from the left and right by the matrix $H = (I - \frac{ee^T}{n})$ with $e^T e = n$. We thus have the centered HDH matrix, with the sum of its rows and column equal to zero. The kernel matrix associated to the distances is finally found [13] :

$$K = -\tfrac{1}{2} H \Delta^{(2)} H \qquad \textbf{eq. 5.7}$$

5.2.1.2 Sigmoid commute-time kernel

The sigmoid commute-time is a kernel-based similarity measure. It is is constructed using the sigmoid transformation of the commute-time kernel. It can easily be computed with the pseudo-inverse L^+ of the Laplacian matrix [13] L.

We know that the Laplacian matrix of the graph is $L = D - A$, where $D = Diag(Ae)$. The elements a_{ij} of the adjacency matrix A of the graph are defined as $a_{ij} = w_{ij}$ if node i is connected to node j and 0 otherwise. L is a symmetric and semi-definite matrix with rank $n - 1$. The well-known commute-time distance can be considered as the number of steps taken in a random walk between nodes i and j and back [8]:

$$n(i,j) = v_g (e_i - e_j)^T L^+ (e_j - e_i) \qquad \textbf{eq. 5.8}$$

where v_g is the volume of the graph ($v_g = \sum_{i,j=1}^{n} aij$) and e_i are the basis vectors standing for each node i of the graph in the Euclidean space R^n. L^+ is the inner products matrix of the node vectors in the Euclidean space where these nodes are separated by the commute-time distance. Therefore, it can be rewritten as the kernel [13]:

$$K_{CT} = L^+ \qquad \textbf{eq. 5.9}$$

The elements of the sigmoid commute time kernel are then obtained by the sigmoid transformation on K_{CT}:

$$k_{ij}^{\sigma CT} = \frac{1}{1 + exp(-\alpha l_{ij}^+ / \sigma)} \qquad \textbf{eq. 5.10}$$

where σ is the standard deviation of L^+ and α is a parameter provided by the user [13]. However, attention must be paid to the fact that the sigmoid transformation of a kernel matrix it is not necessarily positive semi-definite. Thus, strictly speaking, it is not a valid kernel matrix. We could therefore argue that using such similarity matrix in kernel k-means algorithms is technically not well-founded. To overcome this limitation, Yen and al. [71] provide arguments in favor of the use of the sigmoid CT : firstly, and most importantly, even if the similarity matrix is not positive semi-definite, the convergence of the kernel k means algorithms to a minimum of the criterion has been proved (see section 5.5.2). The kernel k-means therefore minimizes this criterion, even if it becomes negative. Secondly, systematic comparisons show that the sigmoid commute-time kernel performs much better than the commute-time kernel, because the range of values of the commute-time kernel has quite a large spread. This large spread of values causes the presence of a number of outliers and perturbs the k-means which is known to be quite sensitive to outliers. Thus, taking a sigmoid transformation (which is reduces the spread) appears to be quite beneficial.

5.2.1.3 Free energy distance kernel

The free energy distance is defined as [14]:

$$\triangle_{ij}^{\phi} = \begin{cases} \frac{\phi(i,j)+\phi(j,i)}{2} & \text{if } i \neq j, \ 0 \text{ if } i = j \end{cases} \qquad \textbf{eq. 5.11}$$

where the quantity $\phi(i,j)$ is equal to: $\phi(i,j) = -\frac{1}{\theta} log z_{ij}^h = -\frac{1}{\theta} log(\frac{z_{ij}}{z_{jj}})$. It can be derived from the probability distribution of a random walk between two nodes that minimizes the Helmholtz free energy [44] of a thermodynamical system. Indeed, if we define the temperature with $T = 1/\theta$ and state transition probabilities P_{ij}, the minimization is written as follows [8]:

$$P(\varrho) = \text{argmin}_{\tilde{P}_{ij}(\varrho)} \sum_{\varrho \in P_{ij}} P_{ij}(\varrho)\tilde{c}(\varrho) + \frac{1}{\theta} \sum_{\varrho \in P_{ij}} P_{ij}(\varrho) log(P_{ij}(\varrho)/P_{ij}^{REF}(\varrho)) \qquad \textbf{eq. 5.12}$$

$$\text{s.t.} \sum_{\varrho \in P_{ij}} P_{ij}(\varrho) = 1$$

The free energy distance is then given thanks to the the minimum free energy :

$$\triangle_{sij}^{FE} = \frac{(\phi(P_{ij}^{FE}) + \phi(P_{ij}^{FE}))}{2} \qquad \textbf{eq. 5.13}$$

where the quantity $\phi(P_{ij}^{FE})$ corresponds to $\varphi(i,j) = -\frac{1}{\theta} log z_{ij}^{h}$.
The free energy distance appears to be an adequate distance measure with interesting properties. Indeed, it can be proven that it respects the triangular inequality (proof in [14]) and enjoys the following properties [8]:

- It is a geodetic distance: $\triangle_{ik}^{\varphi} = \triangle_{ij}^{\varphi} + \triangle_{jk}^{\varphi}$ if and only if every path from i to k passes through j [6].
- In an undirected graph G, the distance $\triangle_{ij}^{\varphi}$ converges to the commute-cost distance when θ becomes small, $\theta \to 0^{+}$.
- In an undirected graph G, the distance$\triangle_{ij}^{\varphi}$ converges to the shortest path distance when θ becomes large, $\theta \to \infty$.

5.2.2 Description

The kernel k-means algorithm assumes that a kernel matrix K containing similarities between the nodes of the graph G has been pre-computed.
We must insist that this section is quite technical and relies on some elaborated mathematical principles. It might be difficult for the reader to understand it immediately. We therefore suggest to complete the reading of this thesis with a few resources (namely [13] and [71]) to help the reader understand this chapter more easily. Furthermore, as the principle of the algorithm remains similar to the distance-based k-means (see section 5.1), in case of difficulties in understanding the theoretical foundations, the reader should not hesitate to skip the following lines to go immediately to the algorithm (section 5.2.3), more intuitive.

When dealing with kernel clustering, three different spaces are defined by Fouss and al. [13] :

- The input space is the initial space in which the data are defined. It is irrelevant in our case since we are directly working on a graph structure.

- The embedding space, corresponding to the image of the input space through the mapping. Here, the embedding space corresponds to the space preserving the inner products between the node vectors of the graph, as defined by the kernel matrix. Thus, each node of the graph corresponds to a vector in this embedding space. The dimensionality of this space matches the rank of the kernel matrix.

- The sample space, corresponding to the Euclidean space having, as dimensionality, the number of data samples. In the present case, this is the number of nodes of the graph.

The kernel k-means relies on the four following steps, described by Fouss, Saerens and Shimbo [13]:

1. The computation of a meaningful kernel on the graph.

2. The definition of a criterion, typically the sum of within-cluster inertia, depending on (i) prototype vectors in the embedding space (denoted as g_k) and on (ii) membership values indicating the membership of each node to the cluster (denoted as u_{ik}).

3. The expression the prototype vectors in the embedding space in terms of the prototype vectors in the sample space.

4. The optimization of the criterion with respect to the prototype vectors, h_k , as well as the membership values, u_{ik} . To this end, the standard two-step algorithm used in k-means clustering and variants is applied (see section 4.1.1).

Imagine that we are looking for a partition in m clusters in total (number fixed a priori by the user). The goal is to design an iterative algorithm aiming to minimize our objective function-which, in the case of standard k-means, is the total within-cluster inertia:

$$J(g_1, \ldots, g_m) = \sum_{k=1}^{m} \sum_{i \in C_k} \|x_i - g_k\|^2 \qquad \textbf{eq. 5.14}$$

where the first sum is taken over the m clusters while the second sum is taken on the nodes i belonging to cluster k, $i \in C_k$. x_i is the node vector corresponding to node i and g_k is a prototype vector of cluster k in the embedding space, while $\|x_i - g_k\|$ is the Euclidean distance between the node vector and the cluster prototype it belongs to. The criterion's value (the sum over all classes of the within-cluster inertia) should be as low as possible. The prototype vector g_k of one cluster is defined as a representative of this

cluster k . We denote by X the $n \times p$ (p is the number of features in the embedding space) data matrix containing the transposed node vectors as rows; that is, $X = [x_1, x_2, \ldots, x_n]^T$ [13].

Let us operate the following change of parameter: $g_k \rightarrow X^T h_k$, corresponding to the kernel trick. Let us now recompute the within-class inertia in terms of the h_k and the inner products [13]:

$$\begin{aligned} J(h_1, ..., h_m) &= \sum_{k=1}^{m} \sum_{i \in C_k} (x_i - g_k)^T (x_i - g_k) \\ &= \sum_{k=1}^{m} \sum_{i \in C_k} (x_i^T x_i - 2 x_i^T g_k + g_k^T g_k) \\ &= \sum_{k=1}^{m} \sum_{i \in C_k} (k_{ii} - 2 k_i^T h_k + h_k^T K h_k) \\ &= \sum_{k=1}^{m} \sum_{i \in C_k} (e_i - h_k)^T K (e_i - h_k) \qquad \textbf{eq. 5.15} \end{aligned}$$

where $K = XX^T$, $k_{ii} = [K]_{ii} = x_i^T x_i$, $k_i = X x_i = col_i(K) = K e_i$.

By introducing the membership matrix u_{ik}, J can be rewritten as

$$J(h_1, \ldots, h_m) = \sum_{i=1}^{n} \sum_{k=1}^{m} u_{ik} (h_k - e_i)^T K (h_k - e_i) \qquad \textbf{eq. 5.16}$$

where u_{ik} is equal to 1 if node i is currently assigned to cluster C_k and to 0 otherwise. The k-means iteratively minimizes J, first by re-allocating the node vectors (determination of the u_{ik}) while keeping the prototype vectors fixed, and secondly by re-computing the prototype vectors, h_k , while maintaining the cluster labels fixed [13].

Re-allocation step. Let us first consider that the prototype vectors are fixed. The u_{ik} for each node i can be optimized independently in the sum of equation

$$J(h_1, ..., h_m) = \sum_{i=1}^{n} \sum_{k=1}^{m} u_{ik} (h_k - e_i)^T K (h_k - e_i) \qquad \textbf{eq. 5.17}$$

so that the best cluster allocation for a node i is to assign the node to the cluster for which $(h_k - e_i)^T K (h_k - e_i)$ is a minimum. Therefore, the re-allocation step minimizing J is

$$k_i^* = \text{argmin}_{k\in\{1,..,m\}} \left\{(h_k - e_i)^T K (h_k - e_i)\right\} \qquad \textbf{eq. 5.18}$$

where the variable k_i^* contains the optimal label of node i . Therefore, the elements of the i th row of the membership function become $u_{i,k_i^*} = 1$, and $u_{ik} = 0$ for $k \neq k_i^*$ [13].

Re-computation of the prototypes. For the computation of the prototype vector, taking the gradient of J with respect to h_k and setting the result equal to 0 yields

$$Kh_k = \frac{1}{n_k} \sum_{i\in C_k} Ke_i = K\frac{1}{n_k} \sum_{i\in C_k} e_i \qquad \textbf{eq. 5.19}$$

where n_k is the number of nodes belonging to cluster . We immediately observe from the left-hand side that Kh_k is a linear combination of the k_i , while the right-hand side is also an unknown linear combination of the k_i. Therefore, one particular solution to this system of linear equation is $h_k = \frac{1}{n_k} \sum_{i\in C_k} e_i$. In other words, h_k contains $\frac{1}{n_k}$ if $i \in C_k$ and 0 otherwise; it corresponds to a prototype vector defined for each cluster in the sample space [13].

Therefore, the prototype re-computation step is:

$$h_{ki} = [h_k]_i = \begin{cases} \frac{1}{n_k} & \text{if node } i \in C_k \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{eq. 5.20}$$

This is a natural result since $X^T h_k$ corresponds exactly to the center of gravity of the cluster k in the embedding space [13].

5.2.3 Algorithm

The kernel k-means algorithm is expressed as follows (and represented in *figure 3*).

Input:

-A weighted directed graph G containing n nodes.

-K: a $n \times n$ symmetric similarity matrix associated to G

- The desired number of clusters, m.

Output:

- The $n \times m$ membership matrix U containing the membership of each node i to cluster k, u_{ik}

Figure 3: The kernel K-means algorithm

Initialization: choose m prototype nodes with indices $q_1, q_2, \ldots, q_m$ at random and set $h_k = e_{qk}, k = 1 \ldots m$, where e_i is a $n \times 1$ basis column vector (contains 0's everywhere except at position i where it contains a 1)

Repeat

$U \leftarrow Zeros(n, m)$

For $i = 1$ **to** n **do**

$k^* \leftarrow \text{argmin}_{k \in \{1,\ldots,m\}} \left\{ (h_k - e_i)^T K (h_k - e_i) \right\}$

$u_{ik^*} \leftarrow 1$

End for

For $k = 1$ **to** m **do**

$n_k \leftarrow \sum_{i=1}^{n} u_{ik}$

$h_k \leftarrow \frac{col_k(U)}{n_k}$

End for

until the allocation of the nodes does not change any more

Return U

source: adapted from Fouss, Saerens and Shimbo [13], 2014.

This procedure is iterated until convergence of the criterion.

5.3 The Louvain method

The Louvain method is a clustering technique developed by Vincent Blondel (quoted in [13]), described as follows.

5.3.1 Modularity

Modularity is a measure of the structure of graphs. It was designed to measure the strength of division of a network into modules (or clusters). Networks with high modularity have dense connections between the nodes within modules but sparse connections

between nodes in different modules. Modularity is often used for detecting community structure in networks. However, it has been shown that modularity suffers a resolution limit and, therefore, it is unable to detect small communities [13].
The modularity can be computed by:

$$Q(u_1, u_2, ..., u_m) = \frac{1}{vol(G)} \sum_{k=1}^{m} u_k^T Q u_k \qquad \textbf{eq. 5.21}$$

where Q is the modularity matrix, $Q = (A - \frac{d_o d_i^T}{vol(G)})$, and $u_k = Ue_k$ is a binary membership vector containing a 1 in position i if node i belongs to cluster k and 0 otherwise.

5.3.2 Description

The Louvain method is a greedy bottom-up multi-level algorithm for maximizing the modularity of a weighted undirected graph. The main idea is to perform an iterative local optimization for seeking a local maximum of the modularity. In a second step, a new graph is built whose nodes are the clusters found in the first step. The iterations are repeated on the new graph until no further improvement can be obtained. This results in a clustering of the graph without having to specify a priori the number of clusters, and in a hierarchy of clustering solutions at different levels of aggregation [13].
The Louvain method consists in a succession of steps. Initially, each node is a cluster in its own. Hence, there are as many clusters as nodes,i.e., n clusters in total. After the initialization, the first phase (called the local optimization step) is conducted until a local optimum of the modularity is reached. Then, the second phase, called the coarsening step (whose purpose is to build a new agglomerated graph) is performed. However, the second step will not be detailed in this paper, as it is used for hierarchical clustering [13].
This short introduction should be sufficient for the reader to understand the basic logic of the algorithm. If confronted with difficulties in understanding the following paragraphs, the reader should not hesitate to immediately jump to the simplified presentation of the algorithm (section 5.3.3). More info on the Louvain method can be found in [13].

Local optimization step: during this first phase, each node of G is considered in random order, and tentatively moved to another (neighboring) cluster, while the difference

in modularity is computed. Thus, for each node i , we first identify all the neighboring clusters $N_c(i)$, that is, clusters containing at least one neighbor j of node i ($j \in N(i)$), and $l(i) \neq l(j)$ (the cluster label of i, $l(i)$ is different from the cluster label of j, $l(i)$) [13]. The set of all neighboring clusters of node i is denoted by

$$N_c(i) = C_k \mid \exists j : (j \in N(i)) \wedge (l(j) = k) \wedge (l(j) \neq l(i)) \qquad \textbf{eq. 5.22}$$

(the subscript C means that it returns a set of clusters, not nodes).
The move leading to the largest increase in modularity is confirmed, but only if the difference in modularity is strictly positive. Moreover, if the moved node was alone in its cluster and it is moved, the original cluster disappears. Iterations are performed until no move improves the modularity any more during the whole iteration. The result is a partition of G into clusters (some of them might contain only one single node) [13].

5.3.3 Algorithm

The different steps of the algorithms are detailed as follows and represented in *figure 4*.

Input:

A weighted undirected graph G containing n nodes.

Q: the $n \times n$ modularity matrix associated to G.

Output:

U: the $n \times n$ matrix containing elements $u_{ik} = 1$ if node i belongs to cluster k and 0 otherwise. The kth column of U is u_k

Figure 4: A local optimization procedure for clustering the nodes of a graph G based on modularity (the Louvain method).

Initialization: $m \leftarrow n$ {the initial number of cluster is equal to n }

Set the binary membership vector $\{u_k = e_k\}_{k=1}^{m}$, where e_k is a$n \times 1$ basis column vector (0's everywhere except at position k where it contains a 1). Moreover, set the cluster label of each node i to $\{l(i) = i\}_{i=1}^{n}$ (each node is a cluster).

Repeat

$numberMoves \leftarrow 0$

For $i = 1$ **to** n **do**

Compute the set of all neighboring clusters of node i: $N_C(i) \leftarrow \{C_l \mid \exists j : (j{\in}N(i)){\wedge}(l(j) = l){\wedge}(l(j){\neq}l(i))\}$

For all $C_l \in N_C(i)$ **do**

$\triangle Q(i, C_l) \leftarrow \frac{2}{vol(G)}(e_i + u_l - u_{l(i)})^T Q e_i$

end for

$C_{l*} = \text{argmax}_{C_l \in N_C(i)} \triangle Q(i, C_l)$

if $\triangle Q(i, C_{l*}) > 0$ **then**

$u_{il(i)} \leftarrow 0$

$l(i) \leftarrow l*$; $u_{il*} \leftarrow 1$

$numberMoves \leftarrow numberMoves + 1$

end if

End for

Delete the zero columns of matrix U

Recompute accordingly the labels, $\{l(i) \leftarrow \text{argmax}_l(u_{il})\}_{i=1}^{n}$

until $numberMoves = 0$

Return U

source: adapted from Fouss, Saerens and Shimbo [13],2014.

5.4 Possibilistic fuzzy c-means clustering

This section is not necessarily straightforward for a beginner in statistics, it relies on some important mathematical principles. It might be difficult for the reader to under-

stand it immediately. We therefore suggest to complete the reading of this thesis with a few resources (namely [25] and [43]) to help the reader understand this chapter more easily. Furthermore, as the principle of the algorithm remains similar to the distance-based k-means (see section 5.1), in case of difficulties in understanding the theoretical foundations, the reader should not hesitate to skip the following lines to go immediately to the description of the algorithm (section 5.4.2), more intuitive.

Fuzzy clustering has been shown to be advantageous over crisp clustering in that total commitment of a vector to a given class is not required in each iteration.

However, the fuzzy c-means (FCM) clustering is sensitive to noise and outliers because of the probabilistic constraint. Therefore, Krishnapuram and Keller [25] proposed the possibilistic c-means algorithm (PCM) by abandoning the probabilistic constraint to solve the noise sensitivity.

Their model can be further extended to the possibilistic fuzzy c-means clustering (PFCM, see Pal and al. in [43]). Playing with the different parameters allows us to favor one model (FCM) over the other (PCM) and to keep the opportunity to move from between models, and keep their specificities .

5.4.1 Description

The PCM algorithm is derived from the FCM algorithm by relaxing the constraint on its objective function. Indeed, the FCM objective function is given by [25]:

$$J(L,U) = \sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^m d_{ij}^2 \qquad \textbf{eq. 5.23}$$

subject to $\sum_{i=1}^{C} u_{ij} = 1$ for all j. In that function, $L = (\beta_1, ..., \beta_C)$ is a C-tuple of prototypes, d_{ij}^2 is the distance of feature point x_j to prototype β_i, N is the total number of feature vectors, C is the number of classes, and $U = [u_{ij}]$ is a $C \times N$ matrix, called the fuzzy C-partition matrix. u_{ij} is the grade of membership of the feature point x_j in cluster β_i, and $m \in [1, \infty]$ is a weighting exponent called the fuzzifier. Our objective is to have the memberships for representative feature points as high as possible, while unrepresentative points should have low membership in all clusters. The new PCM objective function satisfying those requirements becomes [25]:

$$J_m(L,U) = \sum_{i=1}^{C}\sum_{j=1}^{N}(u_{ij})^m d_{ij}^2 + \sum_{i=1}^{C}\eta_i \sum_{j=1}^{N}(1-u_{ij})^m \qquad \textbf{eq. 5.24}$$

where η_i are suitable positive numbers. The first term of the equation demands that the distances from the features vectors to the prototypes be the lowest, whereas the second term forces the u_{ij} to be as large as possible, avoiding the trivial solution.Thus, in each iteration, the updated value of u_{ij} depends only on the distance of x_j from β_i [25].
The intuition behind the term $\sum_{j=1}^{N}(1-u_{ij})^m$ is to release the traditional constraint on the membership ($\sum_i u_{ij} = 1$) to give some flexibility to the model, while avoiding he trivial solution (the null solution).
The value of η_i determines the distance at which the membership value of a point in a cluster becomes 0.5. It needs to be chosen depending on the desired "bandwidth" of the possibility (membership) distribution for each cluster. Therefore, this value is the same for all clusters if they are expected to be similar. It also determines the degree to which the second term in the objective function is favored compared with the first. In practice, the following definition works well [25]:

$$\eta_i = K\frac{\sum_{j=1}^{N} u_{ij}^m d_{ij}^2}{\sum_{j=1}^{N} u_{ij}^m} \qquad \textbf{eq. 5.25}$$

We observe that this makes η_i proportional to the average fuzzy intra-cluster distance of the cluster β_i. K is typically chosen to be 1 [25] .

This model can be extended to keep track of its "fuzzy" origin. Indeed, the FCM has problems dealing with noise and outliers, the PCM has the problem of coincident clusters and the FPCM (Fuzzy Possibilistic C-means) has difficulties dealing with big datasets because the typicality values are in that case very small. The apparent problem of FPCM is that it imposes a constraint on the typicality values (the sum of the typicalities over all data points to a particular cluster is 1). We can relax the constraint on the typicality values but maintain the constraint on the memberships. We obtain a new algorithm: the PFCM (Possibilistic Fuzzy C-means). It is a hybridization of PCM and FCM that avoids the recurring problems of PCM, FCM and FPCM [43].

Hence, the modified objective is to minimize the following equation:

$$J_{m,\eta}(U,T,V,X) = \sum_{k=1}^{n}\sum_{i=1}^{c}(au_{ik}^{m} + bt_{ik}^{\eta}) \times \|x_k - v_i\|_A^2 + \sum_{i=1}^{c}\gamma_i \sum_{k=1}^{n}(1-t_{ik})^{\eta} \qquad \textbf{eq. 5.26}$$

subject to the constraints $\sum_{i=1}^{c} u_{ik} = 1 \forall k$, and $0 \leq u_{ik}, t_{ik} \leq 1$. Here $a > 0$, $b > 0$, $m > 1$, $\eta > 1$ and $J_{m,\eta}$ is the objective function. U is the partition matrix, T is the typicality matrix, V is a vector of cluster centers, X is the set of all data points, x represents a single data point, n is the number of data points and c is the number of cluster centers. $\|x_k - v_i\|_A$ (also represented as D_{ikA}) represents any norm used to calculate the distance between i th cluster center and k th data set (here the Euclidean distance). We note that $\|x\|_A = \sqrt{x^t A x}$ is any inner product norm [43].

The constants a and b define the relative importance of fuzzy membership and typicality values in the objective function. If we increase the importance (weight) of the membership, it necessarily forces us to reduce the importance of typicality by the same amount. We can also discuss the importance the parameters a and b. By constraining $a + b = 1$, we lose modeling flexibility. If $b = 0$, and $\gamma_i = 0$ for all i, the objective function is reduced to a FCM optimization problem. Following the usual conditions placed on c-means optimization problems, we get the first-order conditions for extrema of $J_{m,n}$ [26]:

$$u_{ik} = \left(\sum_{j=1}^{c}\left(\frac{D_{ikA}}{D_{jkA}}\right)^{\frac{2}{m-1}}\right)^{-1} \text{and } 1 \leq i \leq c, 1 \leq k \leq n \qquad \textbf{eq. 5.27}$$

$$t_{ik} = \frac{1}{1 + \left(\frac{b}{\gamma_i} D_{ikA}^2\right)^{\frac{1}{(\eta-1)}}} \text{ and } 1 \leq i \leq c, 1 \leq k \leq n \qquad \textbf{eq. 5.28}$$

$$v_i = \frac{\sum_{k=1}^{n}(au_{ik}^{m} + bt_{ik}^{\eta})x_k}{\sum_{k=1}^{n}(au_{ik}^{m} + bt_{ik}^{\eta})} \text{ and } 1 \leq i \leq c \qquad \textbf{eq. 4.29}$$

PFCM behaves like FCM as the exponents grow. That is, irrespective of the values of the constants a and b, the c centroids approach the overall mean as $m \to \infty$ and $\eta \to \infty$.

To reduce the effect of outliers, we should use a higher value for b than a. Similar effects can also be obtained by controlling the choice of η [26].

5.4.2 Algorithm

Piling on the previous rules and constraints, we can establish a family of possibilistic clustering algorithms whose general form is presented in *figure 5*.

Input:

-A weighted directed graph G containing n nodes.

- The desired number of clusters, m.

Output:

- The $n \times m$ membership matrix U containing the membership of each node i to cluster k, u_{ik}

Figure 5 : The possibilistic clustering algorithm

Fix the number of clusters C; fix m, 1<m<∞ ;

Set iteration counter $l = 1$;

Initialize the possibilistic C-partition $U^{(0)}$;

Estimate η_i ;

Repeat

Update the prototypes using $U^{(l)}$, as indicated below;

Compute $U^{(l+1)}$;

Increment l ;

Until ($[\![U^{(l-1)} - U^{(l)}]\!] < \varepsilon$);

{The remaining part of the algorithm is optional and is to be used only when the actual shape of the generated possibility is important}

Re-estimate η_i using $\eta_i = \frac{\sum_{x_j \in (\Pi_i)_\alpha} d_{ij}^2}{\|(\Pi_i)_\alpha\|}$;

Repeat

Update the prototypes using $U^{(l)}$;

Compute $U^{(l+1)}$;

Increment l ;

Until $\left([\![U^{(l-1)} - U^{(l)}]\!] < \varepsilon\right)$;

source: adapted from Krishnapuram and Keller [25], 1993.

The updating of the prototypes depends on the distance measure chosen [25]. If the distance is an inner product induced norm metric as in the case of the FCM algorithm, i.e. if

$$d_{ij}^2 = (x_j - c_i)^T A_i (x_j - c_i) \qquad \textbf{eq. 5.30}$$

where c_i is the center of cluster β_i and A_i is the matrix associated to cluster i , updating the prototype is achieved by using **eq. 5.29** [25].

Part II. Experiments

Now that the theoretical framework is set, we can design the experiments to answer our first research question: **does the possibilistic fuzzy c-means (PFCM) algorithm give better clustering results than the distance-based k-means, the kernel k-means, the kernel fuzzy c-means and the Louvain method**? We will confront the performance of the PFCM against that of the distance-based k-means, kernel k-means (using both a kernel generated by the "kernel trick" on a free energy distance matrix and a sigmoid commute-time kernel) and Louvain method. We will start by tuning our parameters (as the distance measures and kernels presented above rely on different parameters), then run each algorithm on the different datasets, and compare their solution in the light of three performance measures: the NMI, the Adjusted Rand Index and the classification rate. Let us first describe the different datasets.

6 The datasets

6.1 Newsgroup datasets

The newsgroup dataset is formed by the collection of about 20,000 unstructured documents, taken from 20 discussion groups (the newsgroups) of the Usernet diffusion list. For our experiments, nine subsets related to different topics are extracted from the original database, as listed in *figure 6* [71].

Figure 6: Newsgroups datasets description

Topic	Size	Topic	Size	Topic	Size
G-2cl-A		*G-2cl-B*		*G-2cl-C*	
Politics/general	200	Computer/graphics	200	Space/general	200
Sport/baseball	200	Motor/motorcycles	200	Politics/mideast	200
G-3cl-A		*G-3cl-B*		*G-3cl-C*	
Sport/baseball	200	Computer/windows	200	Sport/hockey	200
Space/general	200	Motor/autos	200	Religion/atheism	200
Politics/mideast	200	Religion/general	200	Medicine/general	200
G-5cl-A		*G-5cl-B*		*G-5cl-C*	
Computer/windowsx	200	Computer/graphics	200	Computer/machardware	200
Cryptography/general	200	Computer/pchardware	200	Sport/hockey	200
Politics/mideast	200	Motor/autos	200	Medicine/general	200
Politics/guns	200	Religion/atheism	200	Religion/general	200
Religion/christian	200	Politics/mideast	200	Forsale/general	200

source: Yen and al [71],2009.

For each subset, 200 documents are sampled from different newsgroups. Thus, the first three subsets (G-2cl-A, G-2cl-B, G-2cl-C) contain (2 classes*200 documents =) 400 documents sampled from two newsgroups topics. The next three subsets (G-3cl-A, G-3cl-B, G-3cl-C) contain (3*200 =) 600 documents sampled from three topics and the last three subsets (G-5cl-A, G- 5cl-B, G-5cl-C) contain (5*200 =)1000 documents sampled from five topics. Finally, the adjacency matrix comprising the links between documents is given by the sum of all document-term-document paths connecting all pairs of documents through the terms they have in common. Our clustering experiments will aim at retrieving the correct partition of nodes for each newsgroup (that is, classifying the documents into the right topics).

6.2 Zachary dataset

This network represents the social connections between the 34 members of an American university karate club. The club itself is divided into two different entities and our algorithms are set to identify which club members belong to the first group, and which belong to the second one.

6.3 Political books dataset

This unweighted networks is a network of books about US politics published around the time of the 2004 presidential election. The nodes are linked when the books are frequently being bought by the same readers on the Amazon online bookstore. The 105 books are classified into 3 groups according to whether their political orientation is conservative, liberal, or neutral [40]. The different clustering algorithms should be able to capture (from the graph structure) the true class for each book.

6.4 Football dataset

This dataset displays 115 observations, representing Division I-A US college football games. Edges correspond to games played by the teams against each other during the regular season of fall 2000. We will here distinguish 12 teams (hence, 12 clusters to be found).

6.5 LFR datasets

The LFR dataset contains two graphs computed according to "Directed, weighted and overlapping benchmark graphs for community detection algorithms", written by Andrea Lancichinetti and Santo Fortunato. This program produces binary networks with overlapping nodes.
LFR1 is composed of 600 nodes and 6142 edges, divided in 3 classes.
LFR3 is composed of 600 nodes and 5233 edges, divided in 6 classes.

6.6 School dataset

The school dataset comprises information about a school where students and teachers divided in different classes.
The school_11_classes dataset comprises 10 classes (two classes per education year, from the 1st year to the 5th) and 1 group of teacher.
The school_6_classes is the same dataset where we have grouped the classes according by year (for instance, the classes 4A and 4B are now united into one classe-4- representing a level of education), leading to 5 classes of students and 1 class of teachers.

7 First research question

7.1 First experimental procedure

We will now design the first experimental procedure to answer the first research question.

7.1.1 Tuning of the parameters

The preliminary step to the experiments is to optimize the parameters for each algorithm. We isolated one of the ten datasets from the newsgroup datasets (a newsgroup with 3 classes) and used it for the sole purpose of the optimization of the parameters (for all measures and algorithms tested). We will here assume that the parameters selected on this subset will remain optimized for all other datasets. Hence, the optimization computations are to be done only once.
For each algorithm and for each parameter value, we have computed the average NMI score on 50 runs of the algorithm and kept the best one as the optimal parameter.

- When working with the distance-based k-means, we use the free energy distance (that depends on the parameter value θ) as distance measure. We therefore computed the average NMI score on 50 runs for θ values ranging from 10^{-6} to 10^{3} and kept the best one for the rest of our experiments.

- For the kernel k-means using the kernel trick, we have to look at the association between the θ value of the distance measure (again, the free energy distance), and the exponent of the kernel trick. We therefore compared the average NMI scores for all the possible combinations of θ (ranging from 10^{-6} to 10^{3}) and *exponent* (also ranging from 10^{-6} to 10^{3}). Indeed, as stated previously, the k-means algorithm starts by selecting some initial nodes as prototype vectors. The initial vector nodes are thus randomly selected. Since the final results depend on the initial vectors, a bad pick on the initial vectors may result in a bad cluster performance. Hence, the procedure was repeated 50 times to ensure better chances of finding the best result and alleviating the randomization impact on the clustering results evaluation.

- The sigmoid commute-time (SCT) kernel relies on an α parameter. We therefore

performed 50 runs of the SCT Kernel K-means clustering algorithm while varying the α parameter between10^{-6} to 10^{3}.

- Finally, the possibilistic fuzzy c-means clustering depends on the parameter ϕ, called "fuzzifier", reflecting the degree of fuzziness of the clusters. Again, we varied its value along the $[10^{-6} - 10^{3}]$ scale and kept the best one (highest average NMI score on 50 runs) of them.

The chosen parameter values kept for the experiments are given in *table 1*:

Table 1: Parameters value chosen-first tuning procedure, first research question

Algorithm	Parameter
Distance-based K-means	$\theta = 3$
Kernel trick/Kernel K-means	$\theta = 0.1$ and $exponent = 2$
Kernel Fuzzy C-means	$\phi = 6$
Sigmoid commute-time/Kernel K-means	$\alpha = 7$
Possibilistic Fuzzy C-means	$\phi = 2$

7.1.2 Clustering task

We used the selected parameters for a clustering task on each of the newsgroups datasets as well as on the Zachary dataset, the football dataset, two LFR datasets, the political books datasets and the school dataset. The procedure followed is fairly similar to the tuning phase. We first performed 50 clustering experiments, returning for each dataset and each algorithm the mean NMI score.We then repeated this step, this time computing the mean Adjusted Rand Index and the mean classification rate.

7.1.3 Performance evaluation

As previously mentioned in section 4.3, we will assess the quality of the clustering task using three performance measures: the NMI (Normalized Mutual Information) score, the Adjusted Rand Index and the classification rate. Below is a short reminder of those measures.

- The NMI algorithm compares the clusters found by our algorithms with the real clusters, and computes to which level they share the same information [71].

- The Adjusted Rand Index is a second performance measure created to eliminate the problems related to the Rand Index, such as the fact that the expected value of the RI of two random partitions does not take a constant value (say zero) or that the Rand statistic approaches its upper limit of unity as the number of clusters increases.
- The classification rate simply returns the percentage of correct classification.

7.2 Results analysis

Now that we have tuned our parameters and defined our performance measures, we can look at the results of our experiments.
Hereunder are the the average NMI score, Adjusted Rand Index and Classification Rate for each dataset and for the different methods, followed by a brief discussion of those results.

7.2.1 Results

To facilitate the reader's interpretation of the results, *table 2* offers a quick reminder of the different clustering techniques.

Table 2 : Reminder of the different clustering techniques

Method	Description
DB K-means	A k-means algorithm based on a free energy distance matrix
K_{trick}	A k-means based on a kernel obtained using the kernel trick on a free energy distance matrix
K_{CT}^{S}	A k-means algorithm based on a sigmoid CT kernel matrix
KFCM	A k-means based on a kernel trick matrix with fuzzy memberships
Louvain Method	A greedy algorithm that progresses by studying the variation in modularity
PFCM	A possibilistic algorithm with fuzzy memberships

To present the results, for each dataset, we have tabulated the average performance measure for our different methods; the Distance-Based K-means ("DB K-means"), Kernel-Trick K-means (K_{trick}), the Sigmoid Commute-time Kernel (K_{CT}^{S}), the Kernel Fuzzy C-

means ("KFCM") the Louvain Method and the Possibilistic Fuzzy C-means ("PFCM"').

- **Newsgroup datasets** *(table 3)*

Table 3: Clustering results for the Newsgroup Dataset

First tuning procedure, first research question

Newsgroup-2cl-A	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.3159	**0.8035**	0.7919	0.4324	0.5246	0.6799
ARI	0.3685	**0.8884**	0.8167	0.4569	0.4749	0.7351
Classification Rate	0.7800	**0.9706**	0.9620	0.8183	0.6675	0.9248

Newsgroup-2cl-B	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.3243	**0.5909**	0.5902	0.4053	0.3681	0.5587
ARI	0.3618	**0.6987**	0.6538	0.4559	0.2960	0.6641
Classification Rate	0.7908	**0.9185**	0.9060	0.8287	0.5214	0.8946

Newsgroup-2cl-C	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.5561	**0.8074**	0.7380	0.4341	0.5607	0.6991
ARI	0.6855	**0.8825**	0.8109	0.4997	0.4808	0.7772
Classification Rate	0.8915	**0.9695**	0.9579	0.8202	0.6636	0.9407

Newsgroup-3cl-A	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.4219	**0.7527**	0.6880	0.4586	0.6700	0.5350
ARI	0.4464	**0.7838**	0.7164	0.3536	0.6974	0.5196
Classification Rate	0.7280	**0.9098**	0.8458	0.6374	0.7829	0.7407

Newsgroup-3cl-B	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.2099	**0.7745**	0.7135	0.4590	0.6157	0.5429
ARI	0.2487	**0.8320**	0.7432	0.3990	0.6176	0.4868
Classification Rate	0.5909	**0.9158**	0.8868	0.6899	0.7511	0.7054

Newsgroup-3cl-C	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.2829	**0.7517**	0.6892	0.4116	0.6204	0.5068
ARI	0.3170	**0.8064**	0.7432	0.4121	0.6518	0.4659
Classification Rate	0.6399	**0.9154**	0.8455	0.6772	0.7891	0.7121

Newsgroup-5cl-A	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.2959	0.6512	0.5795	0.3684	**0.6618**	0.3203
ARI	0.2753	**0.6797**	0.4875	0.2870	0.6467	0.2589
Classification Rate	0.5359	**0.7941**	0.7086	0.5338	0.7121	0.4963

Newsgroup-5cl-B	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.2376	**0.6240**	0.5487	0.3752	0.6072	0.3863
ARI	0.2097	**0.5767**	0.4452	0.2745	0.5469	0.2701
Classification Rate	0.4764	**0.7635**	0.6560	0.4690	0.7010	0.4105

Newsgroup-5cl-C	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.1558	**0.5771**	0.5417	0.4017	0.5276	0.4379
ARI	0.1276	**0.5479**	0.4932	0.3324	0.5194	0.3505
Classification Rate	0.4146	**0.7359**	0.6903	0.5439	0.7140	0.5265

As we can see from the tabs above, for each performance measure and almost every newsgroup subset, the Kernel trick/Kernel K-means method gives the best results. We can also see that the performance levels reached are higher with smaller datasets and a smaller number of classes.

- **Zachary dataset** *(table 4)*

Table 4: Clustering results for the Zachary Dataset

First tuning procedure, first research question

Zachary	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.8557	0.5567	0.9809	0.3493	0.7368	**0.9935**
ARI	0.8267	0.6206	0.9799	0.4310	0.7273	**0.9801**
Classification Rate	0.8800	0.8112	0.9824	0.7324	0.8529	**0.9994**

On this smaller dataset, with only two classes, the possibilistic fuzzy c-means clustering algorithm gives the best results according to our different performance measures.

- **Political Books dataset** *(table 5)*

Table 5: Clustering results for the Political Books Dataset

First tuning procedure, first research question

PolBooks	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.4918	0.5732	0.5697	0.4172	0.5130	**0.6069**
ARI	0.4869	0.6429	**0.6719**	0.4722	0.6168	0.6671
Classification Rate	0.7644	0.8055	0.8387	0.7160	0.8048	**0.8480**

On the political books dataset, the possibilistic clustering algorithm gives again the best results according to our performance measures (except for the ARI, for which the sigmoid commute-time kernel gives slightly better results). We also observe that the K^S_{CT} and PFCM yield highly similar results.

- **Football dataset** *(table 6)*

Table 6: Clustering results for the Football Dataset

First tuning procedure, first research question

Football	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.7219	0.8508	**0.8590**	0.7467	0.6306	0.6014
ARI	0.5577	0.7223	**0.7893**	0.5713	0.3426	0.2892
Classification Rate	0.6565	0.3167	**0.8153**	0.6630	0.4593	0.4292

Here, we have a dataset with a large number of classes (12, for only 115 observa-

tions), and the sigmoid commute-time kernel gives us the best results.

- **LFR dataset** *(table 7)*

Table 7: Clustering results for the LFR Dataset

First tuning procedure, first research question

LFR1	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.1393	0.5286	**0.9423**	0.1644	0.9165	0.1087
ARI	0.1523	0.5174	0.9270	0.1562	**0.9432**	0.0687
Classification Rate	0.5182	0.6598	0.9586	0.4906	**0.9823**	0.4325

LFR3	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.4901	0.8569	**0.9390**	0.5621	0.7706	0.4824
ARI	0.4827	0.7385	**0.8864**	0.4448	0.796	0.3901
Classification Rate	0.6386	0.8299	**0.8899**	0.6241	0.8586	0.5903

On those datasets, we clearly observe that the sigmoid commute-time kernel and the Louvain method perform better than the other clustering algorithms.

- **School dataset** *(table 8)*

Table 8: Clustering results for the School Dataset

First tuning procedure, first research question

School 11-cl	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.2307	0.2922	**0.3059**	0.2036	0.2097	0.1577
ARI	0.0801	0.1185	**0.1273**	0.0676	0.0944	0.0597
Classification Rate	0.2883	0.3368	**0.3423**	0.2521	0.2619	0.1775

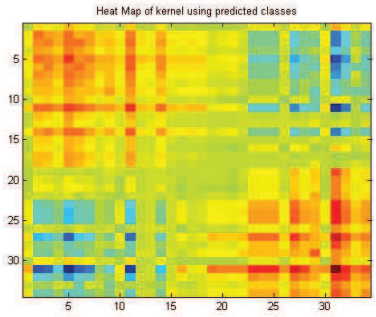
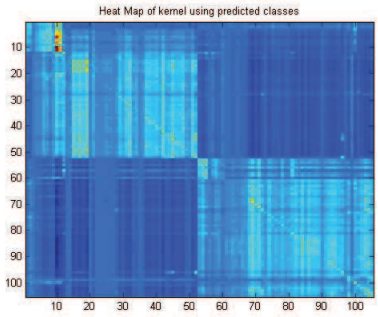
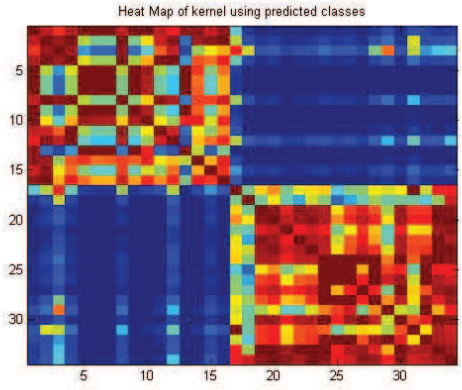
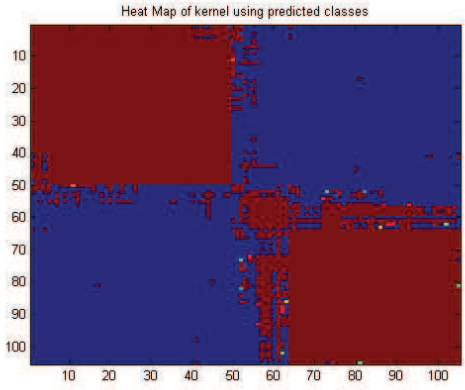
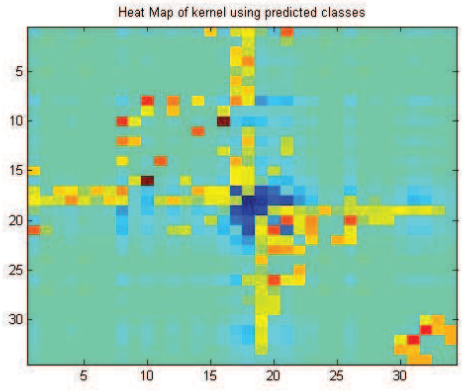
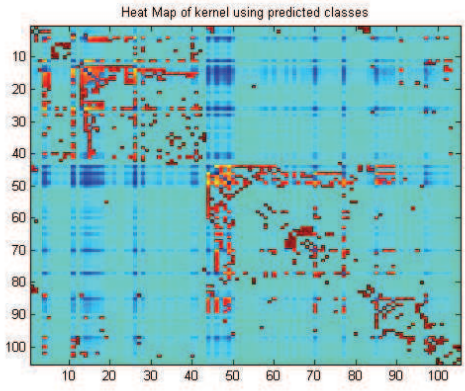
School 6-cl	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.0713	0.1115	**0.1196**	0.0619	0.0986	0.0247
ARI	0.0221	0.0546	**0.0572**	0.0176	0.0506	0.0041
Classification Rate	0.291	0.3124	0.3114	0.2759	**0.3225**	0.2393

On those datasets, we clearly observe that the sigmoid commute-time kernel method performs better than the other clustering algorithms, even though we obtain overall very poor results (below 40%).

7.2.2 Discussion

The discussion of those results can be intuitively lead by a quick look at the clustering results (first in the case of a homogenous successful clustering result among all techniques-as for instance, in the case of the Zachary dataset- , then dissimilar results among the techniques-for instance the political books dataset) as in *figure 7*. We have plot here the different kernel matrices used for the clustering (kernel matrix for the kernel trick, kernel matrix for the sigmoid commute-time and modularity for Louvain method), and the clusters retrieved by the algorithm can be observed as they are aggregated into "blocks" along the diagonal.

Figure 7: comparison of the clustering results via heatmaps on the Zachary and PolBooks datasets

Zachary	**PolBooks**
	
Kernel (Kernel Trick) matrix	Kernel (Kernel Trick) matrix
	
Kernel (Sigmoid commute-time) matrix	Kernel (Sigmoid commute-time) matrix
	
Modularity matrix	Modularity matrix

We distinctly observe that on the left-hand side (Zachary dataset), we have obtained 2 well-separated clusters (two visible squares on the diagonal), especially in the case of the sigmoid commute-time kernel. This reflects the good clustering results obtained during

our experiments. On the right-hand side (Political books dataset), however, we see that the different data points are not as well separated into the three classes, even though we can still distinguish 3 classes on the diagonal. We clearly see a lot of differences between the measures, and a more obvious separation of the points for the sigmoid commute-time kernel compared to the other measures.

We can discuss those results by going back to our research question ("Does the PFCM give better clustering results than the distance-based k-means,the kernel k-means and the Louvain method?"). Our experiments clearly indicate that is **not consistently the case**. The PFCM gave us outstanding results on the small datasets, but fails to compare with the kernel-based methods when the size of the datasets or the number of classes increases. However, me must also notice that, compared to the only other distance-based algorithm ("DB K-means"), it offers superior results.

That leads us to the intuition that modifying the PFCM in order to obtain a kernel based form of possibilistic algorithm might result in an algorithm that would outperform the algorithms presented above. This will be discussed in section 8 ("second research question").

7.2.3 Deepening the experiments

We noticed strong differences between the different techniques, namely with poorer results for the distance-based k-means. We will see if using another distance measure (the commute-time distance and corrected commute-time distance) could improve those results. We will also go back on the sigmoid commute-time kernel to test if using a variant of the commute-time distance could improve its performance.

7.2.3.1 Back to the distance-based k-means

Given the really weak results obtained with the distance-based k-means, we will see if its performance can be improved by using the commute-time distance, and, in a second step, by using the corrected commute-time distance. We will apply the exact same procedure as before, using three performance measures and 50 runs.

7.2.3.1.1 Commute-time distance and corrected commute-time distance

The average commute time is a distance measure between nodes defined as (for any nodes i, j) [13]:

$$\begin{cases} n(i,j) \geq 0 & \\ n(i,j) = 0 & \text{if and only } i = j \\ n(i,j) = n(j,i) & \\ n(i,j) \leq n(i,k) + n(k,j) & \text{for all } k \end{cases}$$

Therefore, the matrix containing the average commute-time distances is computed by:

$$\triangle_{CT} = vol(G)(diag(L^+) * e^T + e * (diag(L^+))^T - 2L^+ \qquad \textbf{eq. 7.1}$$

with $vol(G)$ being the volume of the graph.

This distance matrix can be further developed to obtain the corrected commute time distance [13]:

$$\triangle_{CCT} =$$
$$\triangle_{CT} - vol(G)[D^{-1}ee^T + ee^T D^{-1} + diag(D^{-1}AD^{-1}) * e^T + e * diag(D^{-1}AD^{-1})^T - 2D^{-1}AD^{-1}]$$
$$\textbf{eq. 7.2}$$

7.2.3.1.2 Results

The comparison of the results of the clustering based on the free energy distance, the commute-time distance (CT) and the corrected commute-time (Corrected CT) distance are displayed in *table* 9.

Table 9: Clustering results for the Free Energy, CT and CCT distances

First research question

Dataset	Performance Eval.	FreeEnergy	CT	Corrected CT
Zachary	NMI	**0.8557**	0.3337	0.0200
	ARI	**0.8267**	0.3440	0.0200
	Classification Rate	**0.8800**	0.7112	0.5576
Football	NMI	0.7219	**0.7645**	0.0894
	ARI	0.5577	**0.5864**	0.0190
	Classification Rate	0.6565	**0.6826**	0.1228
LFR1	NMI	**0.1393**	0.0250	0.0000
	ARI	**0.1523**	0.0061	0.0000
	Classification Rate	**0.5182**	0.3383	0.3333
LFR3	NMI	**0.4901**	0.0464	0.0000
	ARI	**0.4827**	0.0027	0.0000
	Classification Rate	**0.6386**	0.2775	0.2745
Newsgroup-2cl-A	NMI	**0.3159**	0.0158	0.0000
	ARI	**0.3685**	0.0000	0.0000
	Classification Rate	**0.7800**	0.5025	0.5000
Newsgroup-2cl-B	NMI	**0.3243**	0.0156	0.0000
	ARI	**0.3618**	$4.302*e^{-6}$	0.0000
	Classification Rate	**0.7908**	0.5027	0.5025
Newsgroup-2cl-C	NMI	**0.5561**	0.0570	0.0000
	ARI	**0.6855**	$6.9796*e^{-4}$	0.0000
	Classification Rate	**0.8915**	0.5029	0.5013
Newsgroup-3cl-A	NMI	**0.4219**	0.0224	$3.1516*e^{-4}$
	ARI	**0.4464**	$2.1394*e^{-6}$	0.0000
	Classification Rate	**0.7280**	0.3362	0.3333
Newsgroup-3cl-B	NMI	**0.2099**	0.0228	0.0000
	ARI	**0.2487**	$7.8592*e^{-6}$	0.0000
	Classification Rate	**0.5909**	0.3371	0.3344
Newsgroup-3cl-C	NMI	**0.2829**	0.0224	0.0000
	ARI	**0.3170**	$-5.0798*e^{-6}$	0.0000

	Classification Rate	**0.6399**	0.3371	0.3361
Newsgroup-5cl-A	NMI	**0.2959**	0.0287	0.0000
	ARI	**0.2753**	-3.5459*e^{-7}	0.0000
	Classification Rate	**0.5359**	0.2032	0.2004
Newsgroup-5cl-B	NMI	**0.2376**	0.0286	0.0000
	ARI	**0.2097**	1.2500*e^{-2}	0.0000
	Classification Rate	**0.4764**	0.2030	0.2002
Newsgroup-5cl-C	NMI	**0.1558**	0.0296	0.0000
	ARI	**0.1276**	1.5100*e^{-1}	0.0000
	Classification Rate	**0.4146**	0.2036	0.2006
Political Books	NMI	0.4918	**0.5839**	0.0000
	ARI	0.4869	**0.6132**	0.0000
	Classification Rate	0.7644	**0.8034**	0.4667
School-11-classes	NMI	**0.2307**	0.1300	0.0000
	ARI	**0.0801**	0.0026	0.0000
	Classification Rate	**0.2883**	0.1378	0.1059
School-6-classes	NMI	0.0713	**0.0735**	0.0000
	ARI	**0.0221**	4.5763*e^{-4}	0.0000
	Classification Rate	**0.291**	0.2169	0.2076

We distinctly observe that the free energy distance gives us the overall best results. We will therefore continue to use is as the reference measure for the rest of our experiments. Furthermore, the results obtained with the commute-time distance and corrected commute-time are very surprising and highly irregular, with very low scores for some datasets and quite high for others. We will in this case have to conclude that this might result from a bug in Matlab. While this might require further investigation, as this experiment was not one of the core questions of this thesis, and given the time and space constraints, we will not push this question further.

7.2.3.2 Back to the Sigmoid commute time kernel

In order to reach the best possible results, we will also test variations on the sigmoid

commute-time kernel. As the kernel matrix computed for our experiments might not meet the conditions of non-negativity, we will first observe the results by multiplying it by its transposed (to force the condition of non-negativity), and secondly we will use the corrected sigmoid commute-time kernel matrix. We will compare those results to the ones obtained in our experiments.

7.2.3.2.1 Transposed sigmoid commute-time kernel and corrected sigmoid commute-time kernel

Our first improved kernel is just the multiplication of the commute-time kernel by its transposed, to force the non-negativity condition. We obtain the following Kernel:

$$K_{TSCT} = K^S_{CT} * (K^S_{CT})^T \qquad \textbf{eq. 7.3}$$

Our second attempt to improve our kernel relies on the corrected commute-time. For large graphs, the commute-time distance tends to depend only on the degrees of the starting and ending nodes. In order to alleviate this drawback, von Luxburg and al. [32] proposed to introduce a correction term, leading to the corrected commute-time distance CCT (see **eq. 7.2**). Consequently, we can use the following equation:

$$K_{CCT} = L^+ + HD^{-1}AD^{-1}H \qquad \textbf{eq. 7.4}$$

where H is the centering matrix, and we will apply afterward the sigmoid transformation to obtain the corrected sigmoid commute-time kernel [13].

7.2.3.2.2 Results

The results of our experiments using the sigmoid commute time kernel, the sigmoid commute-time kernel multiplied by its transposed and the corrected kernel sigmoid commute-time are given in *table 10*:

Table 10: Clustering results for the K^s_{CT}, K^s_{CT}*****$K^s_{CT}{}^T$and Corrected K^s_{CT}

First research question

Dataset	Performance Eval.	K^s_{CT}	$K^s_{CT} * {K^s_{CT}}^T$	Corrected K^s_{CT}
Zachary	NMI	0.9809	**0.9916**	0.9820
	ARI	0.9799	**0.9914**	0.9754
	Classification Rate	0.9824	**1.0000**	0.9894
Football	NMI	**0.8590**	0.8005	0.8562
	ARI	**0.7893**	0.6642	0.7812
	Classification Rate	0.8153	0.7176	**0.8214**
LFR1	NMI	**0.9423**	0.3172	0.9356
	ARI	**0.9270**	0.2227	0.9265
	Classification Rate	**0.9586**	0.5578	0.9138
LFR3	NMI	**0.9390**	0.9117	0.9363
	ARI	**0.8864**	0.7840	0.8796
	Classification Rate	**0.8899**	0.8472	0.8752
Newsgroup-2cl-A	NMI	0.7919	**0.8067**	0.7655
	ARI	0.8167	0.7724	**0.8390**
	Classification Rate	**0.9620**	0.9215	0.9351
Newsgroup-2cl-B	NMI	0.5902	0.5145	**0.5926**
	ARI	0.6538	0.6237	**0.6747**
	Classification Rate	**0.9060**	0.8802	0.8996
Newsgroup-2cl-C	NMI	0.7380	**0.7733**	0.7113
	ARI	0.8109	**0.8572**	0.8134
	Classification Rate	0.9579	**0.9635**	0.9581
Newsgroup-3cl-A	NMI	0.6880	**0.7518**	0.7032
	ARI	0.7164	**0.8204**	0.7224
	Classification Rate	0.8458	**0.8910**	0.8685
Newsgroup-3cl-B	NMI	**0.7135**	0.6932	0.7041
	ARI	**0.7432**	0.7283	0.7123
	Classification Rate	0.8868	**0.8908**	0.8458
Newsgroup-3cl-C	NMI	0.6892	0.5937	**0.7203**
	ARI	**0.7432**	0.6283	0.7371

	Classification Rate	0.8455	0.7898	**0.8711**
Newsgroup-5cl-A	NMI	0.5795	0.5764	**0.5902**
	ARI	0.4875	0.4546	**0.4924**
	Classification Rate	0.7086	0.6962	**0.7202**
Newsgroup-5cl-B	NMI	**0.5487**	0.5336	0.5262
	ARI	**0.4452**	0.4397	0.4344
	Classification Rate	**0.6560**	0.6198	0.6201
Newsgroup-5cl-C	NMI	**0.5417**	0.5262	0.5120
	ARI	**0.4932**	0.4636	0.4521
	Classification Rate	**0.6903**	0.6560	0.6747
Political Books	NMI	0.5697	**0.6116**	0.5717
	ARI	0.6719	**0.7200**	0.6666
	Classification Rate	0.8387	**0.8699**	0.8410
School-11-classes	NMI	0.3059	0.2800	**0.3077**
	ARI	0.1273	0.1128	**0.1301**
	Classification Rate	**0.3423**	0.3175	0.3403
School-6-classes	NMI	0.1196	**0.1570**	0.1154
	ARI	0.0572	**0.0637**	0.0555
	Classification Rate	0.3114	0.2582	**0.3177**

We can observe here that the results are rather close between our different kernels.We will continue to use the "regular" sigmoid commute-time kernel for the rest of our experiments, as this method gives us the best results, as that is the criteria we have decided to pursue, keeping in mind that both the corrected sigmoid commute-time kernel and the combined $K^s_{CT} * {K^s_{CT}}^T$ give us more valid results (in the sense that the kernels will be positive semi-definite). However, as mentioned in section 5.2.1.2, we can neglect that effect.

7.3 Second experimental procedure

We noticed through all those experiments that the results strongly differed from one dataset to the other. This might indicate the presence of a bias in our tuning procedure.

Therefore, we also will come back to our tuning procedure, and compare once again the algorithms after tuning each parameter on each dataset. This time, we will vary the value of each parameter for each dataset and algorithm, and keep as reference value the parameter giving the highest NMI score for a given dataset and a given clustering algorithm. This will allow us to compare our different algorithms based on their best result obtained after an optimal and adapted tuning.

7.3.1 New parameter tuning

We will, for each dataset, run 50 times each algorithm while varying the parameter value, as we did in the first experiment. We will keep as optimal parameter the one leading to the highest value of NMI score. The results of this "new tuning" is given in *table 11*:

Table 11: New parameter values

Second tuning procedure, first research question

	DB Kmeans	**Kernel Kmeans**	**SCT Kmeans**	**KFCM**	**PFCM**
Zachary	$\theta = 9$	$\theta = 9$, expo $= 5$	$\alpha = 8$	$\phi = 7$	$\phi = 9$
football	$\theta = 10^{-1}$	$\theta = 10^{-1}$, expo $= 6$	$\alpha = 3$	$\phi = 10$	$\phi = 10^{-}1$
LFR1	$\theta = 7$	$\theta = 7$, expo $= 10$	$\alpha = 7$	$\phi = 10$	$\phi = 7$
LFR3	$\theta = 2$	$\theta = 2$, expo $= 10$	$\alpha = 7$	$\phi = 9$	$\phi = 9$
news2cl1	$\theta = 2$	$\theta = 2$, expo $= 4$	$\alpha = 5$	$\phi = 4$	$\phi = 9$
news2cl2	$\theta = 5$	$\theta = 5$, expo $= 2$	$\alpha = 4$	$\phi = 8$	$\phi = 5$
news2cl3	$\theta = 10^{-1}$	$\theta = 10^{-1}$, expo $= 1$	$\alpha = 10$	$\phi = 4$	$\phi = 9$
news3cl1	$\theta = 6$	$\theta = 6$, expo $= 9$	$\alpha = 7$	$\phi = 2$	$\phi = 10$
news3cl2	$\theta = 9$	$\theta = 9$, expo $= 1$	$\alpha = 8$	$\phi = 10^{-1}$	$\phi = 9$
news3cl3	$\theta = 7$	$\theta = 7$, expo $= 6$	$\alpha = 10$	$\phi = 5$	$\phi = 9$
news5cl1	$\theta = 10$	$\theta = 10$, expo $= 7$	$\alpha = 10$	$\phi = 9$	$\phi = 3$
news5cl2	$\theta = 3$	$\theta = 3$, expo $= 4$	$\alpha = 3$	$\phi = 3$	$\phi = 2$
newscl53	$\theta = 10^{-2}$	$\theta = 10^{-2}$, expo $= 7$	$\alpha = 5$	$\phi = 3$	$\phi = 3$
polbooks	$\theta = 10^{-2}$	$\theta = 10^{-2}$, expo $= 3$	$\alpha = 1000$	$\phi = 10^{-6}$	$\phi = 10$
School11cl	$\theta = 10$	$\theta = 10$, expo $= 3$	$\alpha = 9$	$\phi = 8$	$\phi = 10^{-}2$
School6cl	$\theta = 10$	$\theta = 10$, expo $= 3$	$\alpha = 10$	$\phi = 10$	$\phi = 10^{-}2$

7.3.2 Experiments with the new parameters values

We are now able to re-launch our algorithms, using the same procedure as before but with the new parameter value. Again, we computed for each dataset the average NMI score,

Adjusted Rand Index and Classification rate for the 50 trial runs. Here are the results for each dataset.

- **Newsgroup datasets** *(table 12)*

Table 12: Clustering results for the Newsgroup Dataset

Second tuning procedure, first research question

Newsgroup-2cl-A	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.3447	**0.932**	0.7878	0.5061	0.5232	0.9080
ARI	0.3265	0.7469	**0.7809**	0.5089	0.4696	0.5727
Classification Rate	0.7698	**0.9321**	0.9443	0.8515	0.6648	0.8662

Newsgroup-2cl-B	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.3452	**0.8933**	0.5883	0.4232	0.3762	0.8904
ARI	0.3691	0.6175	**0.6201**	0.3388	0.2835	0.5996
Classification Rate	0.8337	**0.8931**	0.8713	0.8328	0.5199	0.8865

Newsgroup-2cl-C	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.6449	**0.9707**	0.8139	0.5127	0.5554	0.9209
ARI	0.6708	**0.8784**	0.8465	0.5264	0.4789	0.7205
Classification Rate	0.9088	**0.9683**	0.9585	0.8393	0.6636	0.9244

Newsgroup-3cl-A	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.4178	**0.9213**	0.7005	0.4692	0.6719	0.7156
ARI	0.4719	**0.7595**	0.7106	0.3310	0.6950	0.4571
Classification Rate	0.7299	**0.8906**	0.8689	0.6399	0.7836	0.7051

Newsgroup-3cl-B	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.2472	**0.8799**	0.7326	0.4783	0.6252	0.7402
ARI	0.2313	0.6411	**0.7182**	0.4075	0.6153	0.4755
Classification Rate	0.6114	0.8416	**0.9049**	0.7047	0.7404	0.7292

Newsgroup-3cl-C	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.3232	**0.9000**	0.7397	0.4273	0.6293	0.7326
ARI	0.3148	0.6978	**0.7306**	0.4005	0.6422	0.4424
Classification Rate	0.6736	**0.9000**	0.8793	0.6815	0.7802	0.7089

Newsgroup-5cl-A	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.3048	**0.8530**	0.6061	0.3913	0.6644	0.5276
ARI	0.2956	**0.6663**	0.5238	0.3046	0.6485	0.2711
Classification Rate	0.5441	**0.8579**	0.7149	0.5737	0.7422	0.5085

Newsgroup-5cl-B	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.2578	**0.7269**	0.5644	0.3850	0.6079	0.4374
ARI	0.2022	**0.5585**	0.4656	0.2528	0.5443	0.2527
Classification Rate	0.4980	**0.7363**	0.6244	0.4667	0.7014	0.4280

Newsgroup-5cl-C	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.1966	**0.7275**	0.5451	0.4055	0.5286	0.5498
ARI	0.1293	**0.5466**	0.4824	0.3139	0.5253	0.3362
Classification Rate	0.4064	**0.7173**	0.6970	0.5283	0.6938	0.5325

As we can see from the tabs above, for each performance measure and almost every newsgroup subset, the kernel-based methods give the best results.

- **Zachary dataset** *(table 13)*

Table 13: Clustering results for the Zachary Dataset

Second tuning procedure, first research question

Zachary	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.9260	**0.9982**	0.9770	0.5984	0.7992	0.9429
ARI	0.8999	0.9804	**0.9914**	0.2628	0.7260	0.9429
Classification Rate	0.8894	0.9759	**0.9794**	0.7741	0.8141	0.9424

On this smaller dataset, the sigmoid commute-time kernel algorithm gives the best results.

- **Political Books dataset** *(table 14)*

Table 14: Clustering results for the Political Books Dataset

Second tuning procedure, first research question

PolBooks	DB K-means	Kernel trick	K_{CT}^{S}	KFCM	Louvain	PFCM
NMI	0.6531	0.8131	0.5889	0.6095	0.5317	**0.8470**
ARI	**0.6613**	0.6425	0.6841	0.0000	0.5990	0.6604
Classification Rate	0.8894	0.9759	**0.9794**	0.4667	0.8141	0.9424

On the political books dataset, we observe very different results depending on the performance measure chosen. We can also observe that the PFCM offers interesting results.

- **Football dataset** *(table 15)*

Table 15: Clustering results for the Football Dataset

Second tuning procedure, first research question

Football	DB K-means	Kernel trick	K_{CT}^{S}	KFCM	Louvain	PFCM
NMI	0.7757	0.8397	**0.8847**	0.7288	0.6316	0.5200
ARI	0.5846	**0.7992**	0.7689	0.5705	0.3585	0.3987
Classification Rate	0.6883	**0.8417**	0.8323	0.6830	0.4593	0.5217

On this dataset with 12 classes, the kernel-based methods give us the best results.

- **LFR dataset** *(table 16)*

Table 16: Clustering results for the LFR Dataset

Second tuning procedure, first research question

LFR1	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.1744	**0.9794**	0.9428	0.5540	0.9187	0.7339
ARI	0.1847	0.9320	0.8622	0.1307	**0.9497**	0.3198
Classification Rate	0.5786	0.9744	0.9020	0.5123	**0.9798**	0.6979

LFR3	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.5310	**0.9954**	0.9401	0.6452	0.7711	0.6984
ARI	0.4885	**0.9748**	0.8447	0.4603	0.7762	0.5671
Classification Rate	0.6504	**0.9900**	0.8843	0.6026	0.8603	0.7007

On those datasets, we clearly observe that the kernel-trick/k-means algorithm and the Louvain method perform better than the other clustering algorithms.

- **School dataset** *(table 17)*

Table 17: Clustering results for the School Dataset

Second tuning procedure, first research question

School 11-cl	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.2928	**0.3417**	0.3087	0.1867	0.2104	0.2202
ARI	0.0847	0.1279	**0.1315**	0.0733	0.0911	0.0638
Classification Rate	0.2793	0.3396	**0.3473**	0.2525	0.2619	0.2189

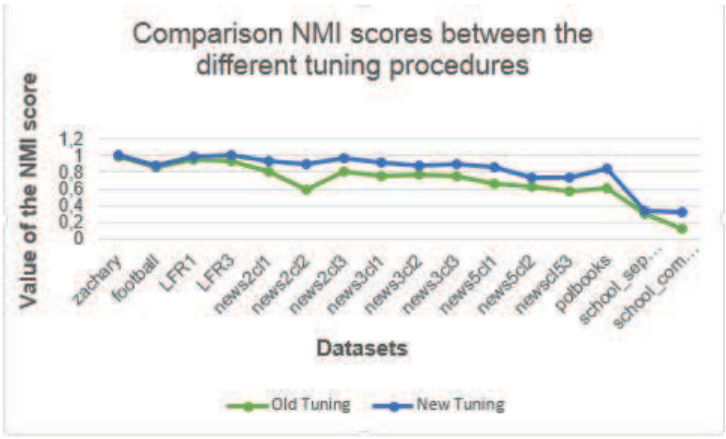
School 6-cl	DB K-means	Kernel trick	K^S_{CT}	KFCM	Louvain	PFCM
NMI	0.2940	**0.3223**	0.1198	0.0518	0.0981	0.2599
ARI	0.0270	**0.0596**	0.0589	0.0216	0.0529	0.0089
Classification Rate	0.2853	**0.3217**	0.3195	0.2779	0.3197	0.2609

On those datasets, we clearly observe that the kernel-based methods perform better than the other clustering algorithms.

7.4 Comparison of the two procedures

We can now compare the results obtained with the different tuning procedures.
First, let us observe the highest level of performance for each dataset using the NMI as performance measure (the results for the ARI and the classification rate are in the appendix, respectively *I.III.I* and *I.III.II*), as given in *figure 8*.

Figure 8 :Comparison of the NMI scores for both tuning procedures



We can easily observe that our new tuning method gives overall higher results. We will therefore continue to use the parameters values yielding the highest possible result for the NMI score.

We also observe that the SCT K-means gives us the best results, just before the kernel k-means. This seems to indicate a superiority of the kernel-based methods, leading to our second research question: does using a Gaussian kernel-based possibilistic c-means yield better results than the "regular" possibilistic c-means?

8 Second research question

The first research question convinced us of the interest residing with the Kernel-based methods. Therefore, we should now try to establish whether turning the possibilistic c-means into a kernel possibilistic c-means significantly improves its performance. This idea leads us to our second research question: does using a Gaussian kernel-based possibilistic c-means yield better results than the "regular"possibilistic c-means?

8.1 Kernel Possibilistic C-means

Let us derive an algorithm from the kernel fuzzy clustering algorithm and develop a kernel possibilistic c-means algorithm. As in the kernel fuzzy c-means clustering (KFCM), we adopt a Gaussian kernel function.
We obtain a new objective function [76]:

$$J_m(U,V) = \sum_{i=1}^{c}\sum_{k=1}^{n} u_{ik}^m \left[\!\left[\phi(x_k) - \phi(v_i)\right]\!\right]^2 + \sum_{i=1}^{c}\eta_i \sum_{k=1}^{n}(1-u_{ik})^m \qquad \textbf{eq. 8.1}$$

where c is the number of clusters, n the number of data points, u_{ik} the membership of x_k in class i , m the fuzzifier and V the set of prototypes. Finally, $K(x,y) = \phi(x)^T\phi(y)$ is an inner product kernel function.
Under those circumstances, the updating of membership is achieved using [76]:

$$u_{ik} = \frac{1}{1 + (2(1-K(x_k,v_i))/\eta_i)^{\frac{1}{(m-1)}}} \qquad \textbf{eq. 8.2}$$

The updating of v_i is computed as

$$v_i = \frac{\sum_{k=1}^{n} u_{ik}^m K(x_k,v_i)x_k}{\sum_{k=1}^{n} u_{ik}^m K(x_k,v_i)} \qquad \textbf{eq. 8.3}$$

and the η_i are estimated using

$$\eta_i = K\frac{\sum_{k=1}^{n} u_{ik}^m 2(1-K(x_k,v_i))}{\sum_{k=1}^{n} u_{ik}^m} \qquad \textbf{eq. 8.4}$$

where K is often chosen to be 1 [76].

With that in mind, we can establish (see *figure 9*) a new algorithm for the Kernel Possibilistic Fuzzy C-means (KPFCM):

Figure 9: the kernel possibilistic fuzzy-means algorithm-KPFCM

Fix c, t_{max}, $m > 1$ and $\varepsilon > 0$ for some positive constant;

Initialize u_{ik}^0 using KFCM algorithm;

Estimate η_i ;

For $t = 1, 2, ..., t_{max}$ **do**

Update all prototypes v_i^t ;

Update all memberships u_{ik}^t ;

Compute $E^t = \max_{i,k} \left| u_{ik}^t - u_{ik}^{t-1} \right|$;

$t = t + 1$;

Until $E^t \leq \varepsilon$;

source: adapted from Zhang and Chen [76], 2003.

8.2 Experiments

We can now compare, again with our three performance criteria, the efficacy of the KPFCM versus that of the PFCM (see *table 18*). The experimental procedure remains unchanged (50 runs on each algorithm to find the best parameter values, and an extra 50 runs to find the mean performance measure).

Table 18: Comparison of clustering results - PFCM vs. KPFCM

Second research question

Dataset	Performance Eval.	PFCM	KPFCM
Zachary	NMI	**0.9935**	0.735
	ARI	**0.9801**	0.5397
	Classification Rate	**0.9994**	0.8712

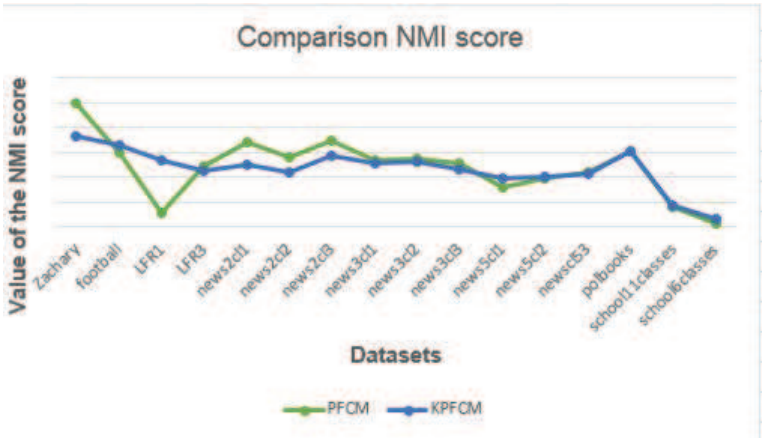
Football	NMI	0.6014	**0.6594**
	ARI	0.2892	**0.4537**
	Classification Rate	0.4292	**0.5663**
LFR1	NMI	0.1087	**0.5392**
	ARI	**0.0687**	0.000
	Classification Rate	**0.4325**	0.3333
LFR3	NMI	**0.4824**	0.457
	ARI	0.3901	**0.6436**
	Classification Rate	0.5903	**0.7392**
Newsgroup-2cl-A	NMI	**0.6799**	0.4962
	ARI	**0.7351**	0.5564
	Classification Rate	**0.9248**	0.8309
Newsgroup-2cl-B	NMI	**0.5587**	0.4409
	ARI	**0.6641**	0.4642
	Classification Rate	**0.8946**	0.7983
Newsgroup-2cl-C	NMI	**0.6991**	0.569
	ARI	**0.7772**	0.5687
	Classification Rate	**0.9407**	0.8871
Newsgroup-3cl-A	NMI	**0.535**	0.5157
	ARI	**0.5196**	0.4076
	Classification Rate	**0.7407**	0.7138
Newsgroup-3cl-B	NMI	**0.5429**	0.5186
	ARI	**0.4868**	0.4653
	Classification Rate	0.7054	**0.7202**
Newsgroup-3cl-C	NMI	**0.5068**	0.467
	ARI	**0.4659**	0.4269
	Classification Rate	0.7121	**0.7291**
Newsgroup-5cl-A	NMI	0.3203	**0.3939**
	ARI	0.2589	**0.2935**
	Classification Rate	0.4963	**0.5574**

Newsgroup-5cl-B	NMI	0.3863	**0.3996**
	ARI	0.2701	**0.2832**
	Classification Rate	0.4105	**0.4890**
Newsgroup-5cl-C	NMI	**0.4379**	0.4265
	ARI	**0.3505**	0.3433
	Classification Rate	0.5265	**0.559**
Political Books	NMI	0.6069	**0.6108**
	ARI	**0.6671**	0.0000
	Classification Rate	**0.8480**	0.4667
School-11-classes	NMI	0.1577	**0.1693**
	ARI	0.0597	**0.0709**
	Classification Rate	0.1775	**0.2594**
School-6-classes	NMI	0.0247	**0.0611**
	ARI	0.0041	**0.0136**
	Classification Rate	0.2393	**0.2755**

8.3 Discussion of the results

We can compare the two methods in a very visual way (see *figure 10*), firstly by looking at their performance level for each dataset .

Figure 10: comparison of the NMI score for PFCM and KPFCM



From the graphic above, we cannot deduce that the KPFCM yields better results than

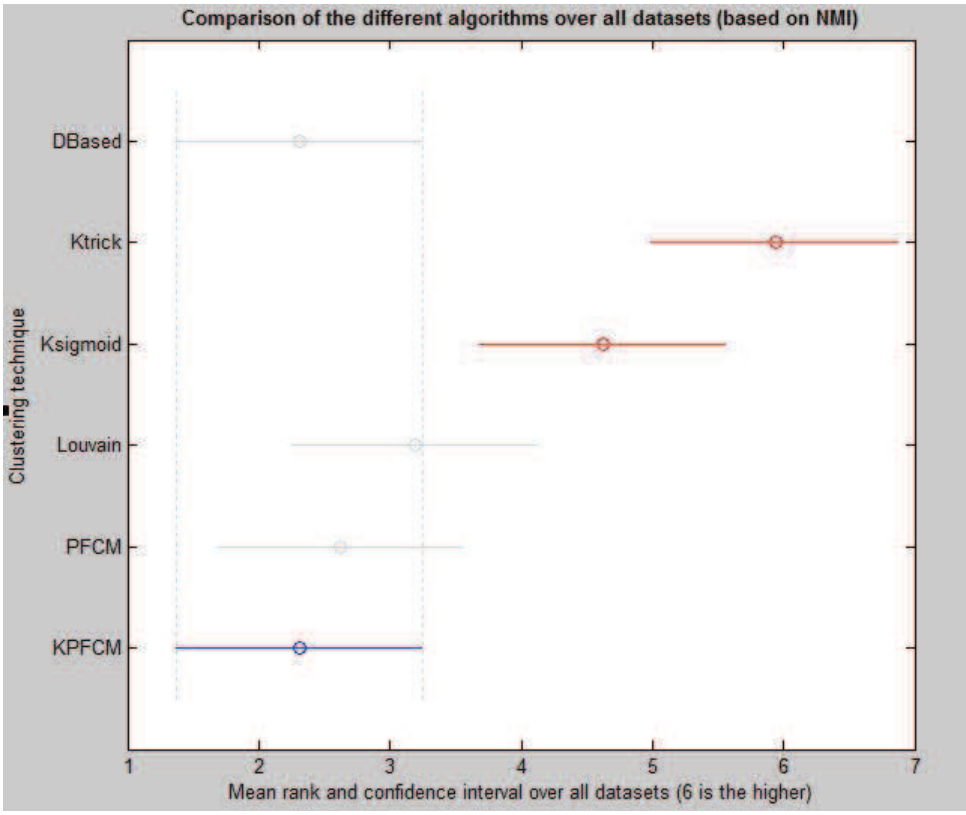
the PFCM, even though they seem to be more consistent. We can thus answer our second research question and state that **using a Gaussian kernel-based possibilistic c-means does not yield significant better results than the "regular"possibilistic c-means**. Of course, using a different kernel method might change that statement, but little theory is yet available on the use of other kernel functions for possibilistic clustering, or on the necessary adaptations of the objective functions and variables.

8.4 Reuniting the two research questions

We can finally summarize our findings to generalize our conclusions at a broader level, that is, compare our techniques over all datasets.

For that purpose, we will start by assigning points to each clustering technique (distance-based k-means, kernel-trick k-means, sigmoid-commute-time kernel k-means, Louvain method, PFCM and KPFCM). We will give 6 points for the technique yielding the highest result, and 1 for the technique leading to the lowest one. We will repeat this step for each dataset. That will give us a mean rank for each clustering algorithm over all datasets.
We will then compare them using Friedman's test. Friedman's test is similar to a classical balanced two-way ANOVA, but it tests only for column (= the techniques) effects after adjusting for possible row (= the datasets) effects. We will then use a multiple comparison test to obtain the critical difference measures (the point at which the differences between our elements become significant).
In other words, we will now compare our different techniques based on their mean rank, and see if some differences are really significant.
More precisely, to go back on our research questions, we will try to determine if the PFCM is significantly different from the other techniques, and if the KPFCM is significantly different from the PFCM on a global scale. The results of those tests are displayed in *figure 11* (for the NMI only, for our other performance measures, see *appendices II.I and II.II*).

Figure 11: Comparison of the clustering techniques (NMI) over all datasets



Clustering techniques are considered significantly different from one another if the mean rank (the "bullet" on the graph) of the first one is located outside the critical difference limit (the "whiskers") of the other.

In our figure, it is therefore pretty clear that the kernel trick and sigmoid CT kernel k-means perform **significantly better** than all the other algorithms (over all datasets), including the PFCM and the KPFCM. We can also see that there is **no significant difference** between the PFCM and the KPFCM.This information thus confirms what our tests datasets per datasets already established, and our research questions are definitely closed.

Some results are particularly interesting. The first one is the dominance of the free energy distance over other distance measures. The second one is linked to the kernel k-trick.

The results of the k-trick kernel k-means are extremely interesting. Indeed, it gives very good results and was obviously improved by the switch towards a new tuning procedure. We can conclude that the parameter θ played an important role (as it is the only change

brought to the tuning procedure), and is probably quite sensitive to the size of the graph. We could exploit this intuition by normalizing and dividing the data by the average of the costs. That way, this size-sensitivity would disappear.

Another noticeable result is that we observe good results from different algorithms in very different datasets (e.g.: the newsgroups datasets are very sparse, while the LFR are "artificial" networks, constructed differently). This brings us closer to the idea that it could be possible to identify which method to use depending on the characteristics of the graph. The existence of a unified "ideal" solution (i.e. a clustering algorithm that would outperform the others on every dataset) being rejected so far by the experiments, we could investigate this "tailored" approach (choosing the method based on the characteristics of the datasets).

Part III. Further work and conclusions

If our experiments did not prove the superiority of the possibilistic fuzzy c-means, there are several tracks that might give us room for improvement.
The first one is to try to derive another kernel-based possibilistic algorithm (for instance, the sigmoid commute-time kernel proven very successful), even though there is little literature on the use of other kernel functions than the Gaussian kernel.
Other leads would directly tackle the theoretical foundation of the PCM as proposed by Krishnapuram and Keller. We will discuss these in the following section.

9 Improving the PCM

The PCM approach proposed by Krishnapuram and Keller was proven effective. However, their method still relies on FCM to calculate the parameter η_i .Yang and Wu [70] proposed to improve that algorithm by computing directly the typicality values and avoid running the FCM beforehand to compute the parameters η_i . They introduced a new Possibilistic Clustering Algorithm : the PCA. After them, Wu and al. [67] realized that the PCA was very sensitive to the initialization procedure and sometimes generated coincident clusters. They therefore proposed an Unsupervised Possibilistic Fuzzy Clustering (UPFC) to solve the problem, combining features from the PCA and the FCM.

9.1 PCA algorithm

9.1.1 Description

The major critique addressed to Krishnapuram and Keller's approach is that it depends heavily on its parameters. Therefore, Yang and Wu [70] suggested another possibilistic clustering approach (PCA) based on the FCM objective function, the partition to noise and outliers. The resulting membership function becomes an exponential function (to gain robustness), or a mountain function (so that the prototypes can be easily handled). The PCA will therefore use an objective function based on the partition coefficient (PC) and partition entropy (PE) validity indexes.

The first validity index associated with FCM is the partition coefficient (PC), defined as

$$PC(c) = \frac{1}{n}\sum_{i=1}^{c}\sum_{j=1}^{n}\mu_{ij}^2 \qquad \textbf{eq. 9.1}$$

where the membership functions μ_i are defined as $\mu_{ij} = \mu_i(x_j)$. We obtain that $\frac{1}{c} \leq PC(c) \leq 1$.

The second validity index, the partition entropy (PE), is defined as

$$PE(c) = -\frac{1}{n}\sum_{i=1}^{c}\sum_{j=1}^{n}\mu_{ij}log_2\mu_{ij} \qquad \textbf{eq. 9.2}$$

where $0 \leq PE(c) \leq log_2\mu_{ij}$.

With that in mind, we can construct a new objective function for the PCA:

$$J_{PCA}(\mu, a) = \sum_{i=1}^{c}\sum_{j=1}^{n}\mu_{ij}^{m}[\![x_j - a_i]\!]^2 + \frac{\beta}{m^2\sqrt{c}}\sum_{i=1}^{c}\sum_{j=1}^{n}(\mu_{ij}^m log\mu_{ij}^m - \mu_{ij}^m) \qquad \textbf{eq. 9.3}$$

where β, m and c are all positive. The first term is equivalent to the FCM objective function and the second term is constructed by an analog of the PE validity index ($\mu_{ij}^m log\mu_{ij}^m$), and an analog of the PC validity index (μ_{ij}^m).
We deduce the following update equations:

$$\mu_{ij} = exp(-\frac{m\sqrt{c}\,[\![x_j - a_i]\!]^2}{\beta}) \qquad \textbf{eq. 9.4}$$

and

$$a_i = \frac{\sum_{j=1}^{n}\mu_{ij}^m x_j}{\sum_{j=1}^{n}\mu_{ij}^m} \qquad \textbf{eq. 9.5}$$

with $i = 1, ..., c$ and $J = 1, ..., c$.

When minimizing the objective function, we obtain that $J'_{PCA}(\mu) = -\frac{\beta}{m^2\sqrt{c}}\sum_{i=1}^{c}\sum_{j=1}^{n}\mu_{ij}^m$. We can therefore see that the objective of J_{PCA} is to find prototypes such that the sum of the membership functions is maximized. Therefore, the cluster centers obtained by PCA

correspond to the peaks of the potential function. Since the potential function can be considered as a density function, the prototypes of PCA are equivalent to the modes of the estimated density [70].

The parameter β is a normalization term measuring the degree of separation of the data set, and we can define it as the sample co-variance. That is $\beta = \frac{\sum_{j=1}^{n} [\![x_j - \bar{x}]\!]^2}{n}$.

The role of the parameter m corresponds to the fuzzifier m in FCM. If m tends to zero, $lim_{m \to 0} \mu_{ij} = 1$ and $lim_{m \to 0} a_i = \bar{x}$ for all i, j . In this case, the sample mean will be the unique optimizer of the function and no clusters will be found (or, alternatively, the c clusters coincide to one cluster). If m tends to infinity, most data points will have very small membership values even if they are very close to one of these c clusters centers [70].

9.1.2 Algorithm

Given the constraints and objectives given above, we can construct a new possibilistic clustering algorithm, as given in *figure 12*.

Figure 12: The possibilistic clustering algorithm-PCA

Initialize $a_i^{(0)}$, $i = 1, ..., c$ and set $\varepsilon > 0$;

Set iteration counter $l = 0$;

Repeat

Step 1. Compute $\mu_{ij}^{(l+1)}$;

Step 2. Compute $a_i^{(l+1)}$;

Increment l;

Until $max_i \left[\!\left[a_i^{(l+1)} - a_i^{(l)} \right]\!\right] < \varepsilon$

source: adapted from Yang and Wu [70], 2006.

9.2 Unsupervised Possibilistic Fuzzy Clustering

9.2.1 Description

UPFC [67] is an extension of PCA, but integrates the benefits of FCM and PCA. Its objective is to minimize the following objective function (for a set of l points in p dimensional

space):

$$J_{UPFC}(u,v) =$$
$$\sum_{i=1}^{c}\sum_{j=1}^{l}(au_{ij,FCM}^{m}+bu_{ij,PCA}^{n})\left[\!\left[x_j-v_i\right]\!\right]^2+\frac{\beta}{n^2\sqrt{c}}\sum_{i=1}^{c}\sum_{j=1}^{l}(u_{ij,PCA}^{n}logu_{ij,PCA}^{n}-u_{ij,PCA}^{n})$$
eq. 9.6

where the $u_{ij,FCM}^{m}$ are the membership values of x_j in class i and the $u_{ij,PCA}^{m}$ are the possibilistic values of x_j in class i.

We obtain the following updating equations by solving the Lagrangian function:

$$u_{ij,FCM}=\left[\sum_{k=1}^{c}(\frac{\left[\!\left[x_j-v_i\right]\!\right]}{\left[\!\left[x_j-v_k\right]\!\right]})^{\frac{2}{m-1}}\right]^{-1},\forall i,j \qquad \textbf{eq. 9.7}$$

$$u_{ij,PCA}=exp(-\frac{bn\sqrt{c}\left[\!\left[x_j-v_i\right]\!\right]^2}{\beta}),\forall i,j \qquad \textbf{eq. 9.8}$$

$$v_i=\frac{\sum_{j=1}^{l}(au_{ij,FCM}^{m}+bu_{ij,PCA}^{n})x_j}{\sum_{j=1}^{l}(au_{ij,FCM}^{m}+bu_{ij,PCA}^{n})},\forall i \qquad \textbf{eq. 9.9}$$

and the parameter β is calculated in the same way as in the PCA approach.

9.2.2 Algorithm

With that in mind, we can establish (see *figure 13*) an algorithm for the UPFC.

Figure 13: Unsupervised Possibilistic Fuzzy Clustering

Step 1 Initialization

1) Fix c, m, n, a and b, $1 < c < l$, $m, n > 1$ and set $\varepsilon > 0$ and iteration counter $r = 1$ and iteration counter r_{max};

2) Initialize $v_i^{(0)}$, $i = 1, ..., c$;

3) Compute parameter β ;

Step 2 Repeat

1)Update $u_{ij,FCM}^{r+1}$ and $u_{ij,PCA}^{r+1}$;

2) Update v_i^{r+1} ;

3) Increase r ;

Step 3 Until $(\llbracket v_i^{r+1} - v_i^r \rrbracket < \varepsilon)$ or $(r > r_{max})$

source: adapted from Wu and al. [67], 2010.

All these theories, yet to be tested, criticized and more detailed in the literature, will probably lead the future in possibilistic clustering techniques.

10 Conclusion

This thesis finds its place right at the intersection between the scientific world and the managerial world. Indeed, I was able to produce a new way of clustering data, test it, improve it and analyze the results from a critical point view, while keeping in mind the business implications it could trigger.
I first introduced the basic notions of machine learning such as clustering, similarities and dissimilarities, kernel matrices and so on. I then explored the current techniques used for clustering, displaying the different algorithms I was going to test: the distance-based k-means, kernel k-means, Louvain method and possibilistic clustering.
I tackled the experimentation part, after tuning our different parameters, trying to be as critical as possible in regard to the evaluation and validation of the results. I therefore did not hesitate to change the tuning procedure, to use different distance measures or different kernel matrices. All those precautions allowed me to draw valid and sound conclusions from those experiments.
I finished by offering some potential improvements that could be brought to the possibilistic clustering algorithm as it is now, and by stating different applications of clustering in the business world.

With this paper, I have shown that the possibilistic fuzzy c-means clustering offers interesting results. Indeed, for some datasets, it gave me the best clustering results (or close to the best). On other datasets, however, it did not outperform the existing algorithms. I was able to highlight the opportunities linked to the use of kernels. Even though my experiments showed in the end that transforming the PFCM into the KPFCM (using a Gaussian kernel transformation) did not make a significant difference, the obvious supremacy of the kernel-based methods strongly encourage to pursue the work in this direction, and to test other kernel forms of the possibilistic clustering algorithm. Another noticeable result was that we observed good results from different algorithms in very different datasets (e.g.: the newsgroups datasets are very sparse, while the LFR are "artificial" networks, thus constructed differently). This brings us closer to the idea that it could be possible to identify which method to use depending on the characteristics of the

graph. The existence of a unified "ideal" solution (i.e. a clustering algorithm that would outperform the others on every dataset) being rejected so far by my experiments, we could investigate this "tailored" approach (choosing the method based on the characteristics of the datasets).
I strongly advise to continue this thesis's work by integrating the newest theories on possibilistic c-means algorithms, exploring different kernel opportunities and pursue this idea of selecting the clustering method based on the characteristics of the graph.

I would also like to come back on the skills I have developed to start, construct and finish this thesis.
The first challenge was to discover the world of clustering, and understand the logic and language of clustering algorithms. Getting familiar with and master previous researcher's codes and algorithm was another complicated step. I had to translate the theories into a logical new algorithm. Then came the fourth and biggest challenge: implementing those algorithms. I had to discover Matlab âĂŹs universe and code algorithms for the first time. After implementing those algorithms, the difficulty was to determine how to estimate their validity. Finally, vulgarizing a highly technical content, and diving into the LATEX environment was also a novelty for me.
All those skills I developed are quite unusual for a business engineer, but I tend to think that they are extremely useful, and valued by companies. Therefore, I would advice any student with an interest in information systems to tackle this kind of project. Being able to deduce a logic and a succession of steps from a theoretical paper, to be flexible regarding the tools to use, and the ability to program are without a doubt great additions to a management background, as it will form a complete multitasking profile.

The need for outstanding clustering techniques and professional data miners will only grow in the future, and spread to all areas of the business world; from marketing to sales, supply chain to finance, to allow smoother mergers and acquisitions procedures, or business process re-engineering. The data mining fever will spread fast through the business world, and clustering will find its place in the core of this movement.

References

[1] D. Baier and I. Daniel, "Image clustering for marketing purposes," in *Challenges at the interface of data analysis, computer science, and optimization*, pp. 487–494, Springer, 2012.

[2] B. Balcik, B. M. Beamon, C. C. Krejci, K. M. Muramatsu, and M. Ramirez, "Coordination in humanitarian relief chains: Practices, challenges and opportunities," *International Journal of Production Economics*, vol. 126, no. 1, pp. 22–34, 2010.

[3] A. Bhat, "Possibility fuzzy c-means clustering for expression invariant face recognition," *International Journal on Cybernetics & Informatics (IJCI)*, vol. 3, April 2014.

[4] F. Cai, N.-A. Le-Khac, and M.-T. Kechadi, "Clustering approaches for financial data analysis: a survey," in *Proceedings of the 8th International Conference on Data Mining,(DM12), Las Vegas, Nevada, USA*, pp. 105–111, 2012.

[5] P.-C. Chang, C.-H. Liu, and Y.-W. Wang, "A hybrid model by clustering and evolving fuzzy rules for sales decision supports in printed circuit board industry," *Decision Support Systems*, vol. 42, no. 3, pp. 1254–1269, 2006.

[6] P. Chebotarev, "A class of graph-geodetic distances generalizing the shortest-path and the resistance distances," *Discrete Applied Mathematics*, vol. 159, no. 5, pp. 295–302, 2011.

[7] A. F. da Costa, B. A. Pimentel, and R. M. de Souza, "Clustering interval data through kernel-induced feature space," *Journal of Intelligent Information Systems*, vol. 40, no. 1, pp. 109–140, 2013.

[8] L. Delhaze, "A comparison of different models and techniques for network data mining," Master's thesis, Louvain School of Management, 2013.

[9] C. Domeniconi, J. Peng, and B. Yan, "Composite kernels for semi-supervised clustering," *Knowledge and Information Systems*, vol. 28, pp. 99–116, 07 2011.

[10] P. Embrechts, R. Frey, and A. McNeil, "Quantitative risk management," *Princeton Series in Finance, Princeton*, vol. 10, 2005.

[11] C. Ferri, J. Hernández-Orallo, and R. Modroiu, "An experimental comparison of performance measures for classification," *Pattern Recognition Letters*, vol. 30, no. 1, pp. 27–38, 2009.

[12] F. Fouss, A. Pirotte, and M. Saerens, "A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation," in *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*, pp. 550–556, IEEE, 2005.

[13] F. Fouss, M. Saerens, and M. Shimbo, *Algorithms and Models for Network Data & Link analysis*. to be published, 2014.

[14] K. Françoisse, I. Kivimäki, A. Mantrach, F. Rossi, and M. Saerens, "A bag-of-paths framework for network data analysis," *arXiv preprint arXiv:1302.6766*, 2013.

[15] M. Girvan and M. E. Newman, "American college football network.," *Proc. Natl. Acad. Sci. USA*, vol. 99, no. 7821, 2002.

[16] J. F. Hair, A. H. Money, P. Samouel, and M. Page, "Research methods for business," *Education+ Training*, vol. 49, no. 4, pp. 336–337, 2007.

[17] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*, vol. 2. Springer, 2009.

[18] T.-L. Hu and J.-B. Sheu, "A fuzzy-based customer classification method for demand-responsive logistical distribution operations," *Fuzzy Sets and Systems*, vol. 139, no. 2, pp. 431–450, 2003.

[19] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[20] J. Humphrey and H. Schmitz, *Principles for promoting clusters & networks of SMEs*, vol. 1. UNIDO Vienna, 1995.

[21] Z.-X. Ji, Q.-S. Sun, and D.-S. Xia, "A modified possibilistic fuzzy c means clustering algorithm for bias field estimation and segmentation of brain mr image," *Computerized Medical Imaging and Graphics*, vol. 35, no. 5, pp. 383–397, 2011.

[22] J.-Y. Jung, J. Bae, and L. Liu, "Hierarchical clustering of business process models," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 12, pp. 1349–4198, 2009.

[23] A. M. Khattak, A. Khan, S. Lee, and Y.-K. Lee, "Analyzing association rule mining and clustering on sales day data with xlminer and weka," *International Journal of Database Theory and Application*, vol. 3, no. 1, pp. 13–22, 2010.

[24] G. J. Klir and T. A. Folger, "Fuzzy sets, uncertainty, and information," *Prentice-Hall*, 1988.

[25] R. Krishnapuram and J. M. Keller, "A possibilistic approach to clustering," *Fuzzy Systems, IEEE Transactions on*, vol. 1, no. 2, pp. 98–110, 1993.

[26] N. Kumari, B. Sharma, and D. Gaur, "Implementation of possibilistic fuzzy c-means clustering algorithm in matlab," *International Journal of Scientific & Engineering Research*, vol. 3, no. 11, pp. 1–9, 2012.

[27] A. Lancichinetti and S. Fortunato, "Community detection algorithms: a comparative analysis," *Physical review E*, vol. 80, no. 5, p. 056117, 2009.

[28] R. E. Litan and A. M. Rivlin, "Projecting the economic impact of the internet," *American Economic Review*, pp. 313–317, 2001.

[29] G. Lombart, "Le data miner, ce professionnel que les e-marchands s'arrachent," September 2012. Online on the e-commerce mag website at http://www.ecommercemag.fr/Thematique/profession-1013/rh-formations-10060/Breves/Le-data-miner-ce-professionnel-que-les-e-marchands-s-arrachent-48179.htm.

[30] D. Lusseau, "Evidence for social role in a dolphin social network," 2004. Online on arxiv at http://arxiv.org/ftp/q-bio/papers/0607/0607048.pdf.

[31] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "an undirected social network of frequent associations between 62 dolphins in a community living off doubtful sound," *Behavioral Ecology and Sociobiology*, no. 54, pp. 396–405, 2003.

[32] U. V. Luxburg, A. Radl, and M. Hein, "Getting lost in space: Large sample analysis of the resistance distance," in *Advances in Neural Information Processing Systems*, pp. 2622–2630, 2010.

[33] D. Marketing, "Définition data mining," January 2015. Online on the Definitions Marketing website at http://www.definitions-marketing.com/Definition-Datamining.

[34] B. Minasny, "Fuzzy kmeans," January 2004. Online on the mathworks website at http://www.mathworks.com/matlabcentral/fileexchange/4353-fuzzy-k-means/content/distmat0.m.

[35] S. Miyamoto, "Algorithms for generating clusters with nonlinear boundaries," *Laxenburg, Austria*.

[36] K. Mizutani and S. Miyamoto, "Possibilistic approach to kernel-based fuzzy c-means clustering with entropy regularization," in *Modeling Decisions for Artificial Intelligence*, pp. 144–155, Springer, 2005.

[37] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[38] A. T. Y. Musetti, "Clustering methods for financial time series," *Swiss Federal Institute of Technology*, 2012.

[39] N. Musmeci, "Dynamical analysis of clustering on financial market data," May 2013. Online document via http://www.lorentzcenter.nl/.

[40] M. Newman, "Network data," April 2013. Online on a personal website at http://www-personal.umich.edu/ mejn/netdata/.

[41] M. K. Nitin Patel and R. Ramakrishnan, "Clustering models to improve forecasts in retail merchandising," 2004. Online on the cytel website.

[42] L. Ozdamar and O. Demir, "A hierarchical clustering and routing procedure for large scale disaster relief logistics planning," *Transportation Research Part E: Logistics and Transportation Review*, vol. 48, no. 3, pp. 591–602, 2012.

[43] N. R. Pal, K. Pal, J. M. Keller, and J. C. Bezdek, "A possibilistic fuzzy c-means clustering algorithm," *Fuzzy Systems, IEEE Transactions on*, vol. 13, no. 4, pp. 517–530, 2005.

[44] L. Peliti, *Statistical mechanics in a nutshell*. Princeton University Press, 2011.

[45] J. Peng and Y. Wei, "Approximating k-means-type clustering via semidefinite programming," *SIAM Journal on Optimization*, vol. 18, no. 1, pp. 186–205, 2007.

[46] G. Punj and D. W. Stewart, "Cluster analysis in marketing research: review and suggestions for application," *Journal of marketing research*, pp. 134–148, 1983.

[47] Qualtrics, "Clustering in market research," April 2009. Online on the qualtrics website at http://www.qualtrics.com/blog/clustering-in-market-research/.

[48] I. C. Research/SAS, "Supply-chain analytics:beyond erp & scm," 2010. Online via SAS website.

[49] S. Rinderle-Ma, F. Toumani, and K. Wolf, *Business Process Management: 9th International Conference, BPM 2011, Clermont-Ferrand, France, August 30-September 2, 2011, Proceedings*, vol. 6896. Springer Science & Business Media, 2011.

[50] M. Saerens, Y. Achbany, F. Fouss, and L. Yen, "Randomized shortest-path problems: Two related models," *Neural Computation*, vol. 21, no. 8, pp. 2363–2404, 2009.

[51] J. Sakuma and S. Kobayashi, "Large-scale k-means clustering with user-centric privacy-preservation," *Knowledge and information systems*, vol. 25, no. 2, pp. 253–279, 2010.

[52] S. Salvador and P. Chan, "Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms," in *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pp. 576–584, IEEE, 2004.

[53] J. M. Santos and M. Embrechts, "On the use of the adjusted rand index as a metric for evaluating supervised classification," in *Artificial Neural Networks–ICANN 2009*, pp. 175–184, Springer, 2009.

[54] A. Schenker, *Graph-theoretic techniques for web content mining*, vol. 62. World Scientific, 2005.

[55] J. Shields, "Getting to know your customers by clustering on product purchase patterns," 2006. Online on the lexjansen website via http://www.lexjansen.com/nesug/nesug06/po/po20.pdf.

[56] V. Shrivastava *et al.*, "A study of various clustering algorithms on retail sales data," *International Journal*, vol. 1, no. 2, 2012.

[57] T. Tambunan, "Promoting small and medium enterprises with a clustering approach: A policy experience from indonesia," *Journal of Small Business Management*, vol. 43, no. 2, pp. 138–154, 2005.

[58] T. Tambunan, *Development of small-scale industries during the new order government in Indonesia*. Ashgate Publishing, 2000.

[59] S. C. Tan and J. P. San Lau, "Time series clustering: A superior alternative for market basket analysis," in *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)*, pp. 241–248, Springer, 2014.

[60] J. Tang, S. Scellato, M. Musolesi, C. Mascolo, and V. Latora, "Small-world behavior in time-varying graphs," *arXiv preprint arXiv:0909.1712*, 2009.

[61] S. Tokushige, H. Yadohisa, and K. Inada, "Crisp and fuzzy k-means clustering algorithms for multivariate functional data," *Computational Statistics*, vol. 22, no. 1, pp. 1–16, 2007.

[62] N. A. Van House, "Flickr and public image-sharing: distant closeness and photo exhibition," in *CHI'07 extended abstracts on Human factors in computing systems*, pp. 2717–2722, ACM, 2007.

[63] C.-d. Wang, J.-h. Lai, and J.-y. Zhu, "Conscience online learning: an efficient approach for robust kernel-based clustering," *Knowledge and Information Systems*, vol. 31, pp. 79–104, 04 2012.

[64] M. Wedel and W. A. Kamakura, "Introduction to the special issue on market segmentation," *Intern. J. of Research in Marketing*, vol. 19, pp. 181–183, 2002.

[65] D. Wegener and S. Rüping, "On integrating data mining into business processes," in *Business Information Systems*, pp. 183–194, Springer, 2010.

[66] T. Wittman, "Time-series clustering and association analysis of financial data," *Elect. Eng. Res. Lab., Univ. Texas, Austin, Tech. Rep. CS*, vol. 8980, 2002.

[67] X. Wu, B. Wu, J. Sun, and H. Fu, "Unsupervised possibilistic fuzzy clustering," *J. Info. and Comp. Sci*, vol. 7, no. 5, pp. 1075–1080, 2010.

[68] M.-S. Yang and C.-F. Su, "On parameter estimation for normal mixtures based on fuzzy clustering algorithms," *Fuzzy Sets and Systems*, vol. 68, no. 1, pp. 13–28, 1994.

[69] M.-S. Yang and H.-S. Tsai, "A gaussian kernel-based fuzzy c-means algorithm with a spatial bias correction," *Pattern recognition letters*, vol. 29, no. 12, pp. 1713–1725, 2008.

[70] M.-S. Yang and K.-L. Wu, "Unsupervised possibilistic clustering," *Pattern Recognition*, vol. 39, no. 1, pp. 5–21, 2006.

[71] L. Yen, F. Fouss, C. Decaestecker, P. Francq, and M. Saerens, "Graph nodes clustering with the sigmoid commute-time kernel: A comparative study," *Data & Knowledge Engineering*, vol. 68, no. 3, pp. 338–361, 2009.

[72] L. Yen, F. Fouss, C. Decaestecker, P. Francq, and M. Saerens, "Graph nodes clustering based on the commute-time kernel," in *Advances in Knowledge Discovery and Data Mining*, pp. 1037–1045, Springer, 2007.

[73] L. Yen, M. Saerens, A. Mantrach, and M. Shimbo, "A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 785–793, ACM, 2008.

[74] E. A. Zanaty, "An adaptive fuzzy c-means algorithm for improving mri segmentation," *Open Journal of Medical Imaging*, vol. 3, p. 125, 2013.

[75] C. Zhang, Y. Zhou, and T. Martin, "A validity index for fuzzy and possibilistic c-means algorithm," in *Proc. IPMU*, pp. 877–882, 2008.

[76] D. Q. Zhang and S. C. Chen, "Kernel-based fuzzy and possibilistic c-means clustering," in *Proceedings of the International Conference Artificial Neural Network*, pp. 122–125, 2003.

Part IV. Appendices

Appendix I : tuning procedures

I.I Results of the first tuning procedure

I.I.I Ktrick

Ktrick	*exponent*													
theta	**10^-6**	**10^-2**	**10^-1**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1000**
10^-6	0.0333	0.0255	0.0388	0.0286	0.029	0.0275	0.0262	0.0265	0.0153	0.0054	0.0014	0.0025	0.0035	0,00000
10^-2	0.1214	0.1138	0.4786	0.5191	0.5425	0.5609	0.5414	0.5547	0.5473	0.5057	0.502	0.4661	0.427	0,00000
10^-1	0.1051	0.1089	0.3322	0.6386	**0.6828**	0.6753	0.6705	0.6817	0.6741	0.6543	0.6516	0.6159	0.6095	0,00000
0.5	0.0955	0.0954	0.3532	0.609	0.6391	0.6303	0.6286	0.5971	0.6109	0.612	0.6174	0.6212	0.5961	0,00000
1	0.0926	0.1092	0.3884	0.5952	0.5755	0.5885	0.5976	0.5914	0.5931	0.6077	0.6137	0.6078	0.5904	0,00000
3	0.0994	0.092	0.3757	0.5489	0.5398	0.579	0.5735	0.5563	0.5852	0.6024	0.5777	0.5884	0.5832	0,00000
10	0.0888	0.0978	0.3252	0.5221	0.5363	0.5618	0.5611	0.5644	0.567	0.5745	0.5789	0.5837	0.5464	0,00000
1000	/	/	/	/	/	/	/	/	/	/	/	/	/	/

I.I.II Distance-based k-means

theta	**10^-6**	**10^-2**	**10^-1**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1000**
DBased	0.0225	0.1255	0.1479	0.1504	0.1437	**0.1694**	0.1468	0.1359	0.1387	0.1544	0.1509	0.1403	0.1394	/

I.I.III Possibilistic clustering

phi	**10^-6**	**10^-2**	**10^-1**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1000**
Possib	0.3593	0.3479	0.3521	0	**0.3532**	0.3496	0.3497	0.3504	0.351	0.3397	0.3401	0.3524	0.3463	0.3458

I.I.IV SCT Kernel

alpha	**10^-6**	**10^-2**	**10^-1**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1000**
SCT	0.0337	0.0264	0.0319	0.3937	0.4246	0.4543	0.4737	0.492	0.5068	**0.5147**	0.5111	0.4897	0.5142	0.4855

I.II Results of the second tuning procedure

I.II.I Ktrick

Ktrick	*exponent*													
dataset	**10^-6**	**10^-2**	**10^-1**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1000**
zach	0.7324	0.7588	0.74	0.9247	0.97	0.9824	0.9871	**0.9982**	0.9559	0.9388	0.9265	0.9029	0.8582	0.5294
foot	0.603	0.5962	0.6172	0.736	0.7842	0.8073	0.8063	0.823	**0.8397**	0.832	0.8162	0.7677	0.7073	0.113
LFR1	0.4869	0.4768	0.6097	0.9742	0.9632	0.9716	0.9783	0.9771	0.9633	0.9641	0.9582	0.9744	**0.9794**	0.3333
LFR3	0.5454	0.5278	0.7111	0.9453	0.9654	0.9749	0.9682	0.9867	0.9759	**0.9941**	0.9747	0.9869	0.9954	0.27
N2A	0.6304	0.649	0.8463	0.923	0.9315	0.9253	**0.932**	0.9302	0.9296	0.9273	0.9229	0.9038	0.8147	0.5
N2B	0.6468	0.6289	0.8152	0.888	**0.8933**	0.8863	0.8874	0.893	0.8888	0.8924	0.8918	0.8906	0.8894	0.5025
N2C	0.7353	0.7141	0.9205	**0.9707**	0.9697	0.9676	0.9684	0.9689	0.9607	0.9699	0.9699	0.9682	0.965	0.5013
N3A	0.5749	0.565	0.8299	0.854	0.8986	0.8967	0.8973	0.8734	0.9128	0.8878	0.9008	**0.9213**	0.8985	0.3333
N3B	0.5073	0.507	0.7334	**0.8799**	0.8459	0.8592	0.8764	0.8676	0.8537	0.8512	0.8417	0.8097	0.6037	0.3344
N3C	0.5496	0.5347	0.7963	**0.8979**	0.8834	0.8917	0.8916	0.8898	0.9	0.8902	0.8483	0.6568	0.6274	0.3361
N5A	0.4459	0.4509	0.7241	0.795	0.8097	0.7997	0.7851	0.8232	0.847	**0.8530**	0.8337	0.7975	0.6275	0.2004
N5B	0.4117	0.4293	0.6273	0.7086	0.7232	0.7012	**0.7269**	0.724	0.7265	0.7137	0.6575	0.5554	0.4876	0.2002
N5C	0.4156	0.4219	0.4693	0.6732	0.6984	0.7188	0.6911	0.707	0.7040	**0.7275**	0.7244	0.7246	0.6985	0.2006
PolB	0.7484	0.7333	0.6958	**0.8181**	0.8063	0.8131	0.8097	0.8017	0.7910	0.8074	0.7853	0.7530	0.7573	0.4667
sc.11	0.2684	0.2654	0.269	0.3358	0.3395	0.3417	0.3390	0.3386	0.3391	0.3379	0.3396	0.3351	**0.3397**	0.1059
sc.6	0.2809	0.2777	0.2847	0.3184	0.3180	**0.3223**	0.3199	0.3157	0.3175	0.3141	0.3170	0.3063	0.3022	0.2076

I.II.II Distance-based k-means

DBKmeans	*theta*													
dataset	**10^-6**	**10^-2**	**10^-1**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1000**
zach	0.1187	0.2185	0.4571	0.5989	0.7272	0.7555	0.9096	**0.9255**	0.7779	0.87	0.9072	0.926	0.7208	/
foot	0.7477	0.7707	**0.7757**	0.7281	0.736	0.7311	0.732	0.7513	0.7291	0.7385	0.7373	0.7264	0.7291	/
LFR1	0.0297	0.022	0.0306	0.1233	0.1444	0.1606	0.1605	0.1641	0.1613	**0.1744**	0.1624	0.1568	0.1608	/
LFR3	0.3913	0.0696	0.2445	0.4802	**0.5310**	0.4866	0.4776	0.4953	0.4960	0.4840	0.4914	0.4768	0.4937	/
N2A	0.0158	0.1676	0.3127	0.3053	0.3447	0.3219	0.3219	**0.3534**	0.3087	0.2939	0.3054	0.276	0.3443	/
N2B	0.0302	0.1152	0.2179	0.2192	0.2834	0.2945	0.3332	**0.3452**	0.3218	0.3316	0.3261	0.2913	0.2951	/
N2C	0.0159	0.3629	**0.6449**	0.6279	0.5503	0.5739	0.5402	0.5406	0.5038	0.4827	0.5616	0.5385	0.4861	/
N3A	0.0223	0.2245	0.3133	0.4053	0.4137	0.4097	0.4300	**0.4380**	0.4178	0.4325	0.4275	0.4097	0.3877	/
N3B	0.0336	0.1639	**0.2682**	0.1881	0.2101	0.2191	0.2156	0.2084	0.2041	0.2171	0.2232	0.2472	0.2197	/
N3C	0.0286	0.2412	0.315	0.2888	0.3002	0.3041	0.3052	0.3077	0.3048	**0.3232**	0.3106	0.3047	0.3012	/
N5A	0.0282	0.2309	0.2469	0.2578	0.2714	0.272	0.2831	0.2981	0.2871	0.3016	0.2967	0.3008	**0.3048**	/
N5B	0.0289	0.1991	0.2030	0.2401	0.2388	**0.2578**	0.2493	0.2528	0.2384	0.2535	0.252	0.2479	0.2468	/
N5C	0.0287	**0.1966**	0.1824	0.1783	0.168	0.1671	0.1615	0.1771	0.17	0.1819	0.1598	0.1567	0.1713	/
PolB	0.259	**0.6531**	0.5678	0.5109	0.5074	0.5023	0.5002	0.4890	0.4900	0.488	0.4814	0.4843	0.4974	/
sc.11	0.2118	0.2381	0.2675	0.2916	0.2905	0.2836	0.2797	0.2871	0.2894	0.2871	**0.2930**	0.2892	0.2928	/
sc.6	0.2297	0.2446	0.2881	0.2948	0.2924	0.2868	**0.2962**	0.2896	0.2898	0.2879	0.2926	0.2829	0.294	/

I.II.III Possibilistic clustering

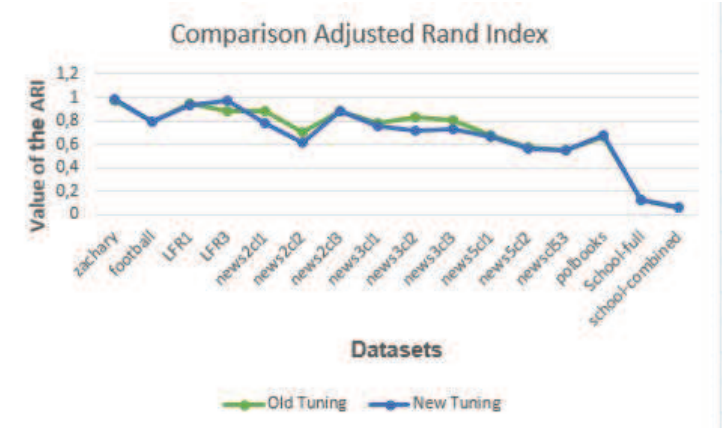
Possibilistic	*phi*													
datasets	**10^-6**	**10^-2**	**10^-1**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1000**
zach	0.5294	0.7576	0.8171	0.5294	0.9412	0.9412	0.9412	0.9418	0.9412	0.9412	0.9418	**0.9429**	0.9394	0.6524
foot	0.113	0.5052	**0.5200**	0.113	0.4402	0.4607	0.4784	0.4809	0.5017	0.5037	0.4892	0.5136	0.4918	0.113
LFR1	0.3502	0.496	0.4405	0.3333	0.6847	0.7038	0.7179	0.7112	0.6971	**0.7339**	0.7196	0.7149	0.7061	0.3889
LFR3	0.27	0.6026	0.6128	0.27	0.6855	0.654	0.638	0.6445	0.6326	0.6649	0.6901	**0.6984**	0.6876	0.301
N2A	0.5	0.8252	0.8701	0.5	0.8299	0.8819	0.876	0.8807	0.9007	0.8948	**0.9041**	0.908	0.9016	0.8357
N2B	0.5025	0.8659	0.8353	0.5025	0.8859	**0.8977**	0.8701	0.8904	0.8688	0.8724	0.8779	0.8843	0.8477	0.7438
N2C	0.5013	0.8722	0.8754	0.5013	0.8598	0.8972	0.9084	0.9032	0.9189	0.9204	0.9204	**0.9209**	0.9162	0.8625
N3A	0.3487	0.6366	0.6072	0.3333	0.5806	0.6280	0.6492	0.6453	0.6813	0.6989	0.7090	0.6928	**0.7156**	0.4439
N3B	0.3469	0.6749	0.6997	0.3344	0.67	0.6987	0.7141	**0.7312**	0.7181	0.7080	0.7141	0.7402	0.7048	0.4464
N3C	0.3506	0.6798	0.6831	0.3361	0.6858	0.6999	0.7078	0.7158	0.7164	0.6978	0.6857	**0.7326**	0.7036	0.439
N5A	0.2137	0.4968	0.4957	0.2004	0.5118	**0.5276**	0.5145	0.5012	0.4863	0.4960	0.4818	0.4913	0.4895	0.2975
N5B	0.2102	0.4214	0.4129	0.2002	0.4374	**0.4378**	0.4272	0.4102	0.3994	0.4038	0.4077	0.4035	0.3987	0.2732
N5C	0.2100	0.5235	0.4972	0.2006	0.5134	**0.5498**	0.5471	0.534	0.5445	0.5223	0.5442	0.5477	0.5397	0.2605
PolB	0.4682	0.8225	0.8398	0.4667	0.8459	0.8448	0.8423	**0.8480**	0.8457	0.845	0.8450	0.8465	0.847	0.5808
sc.11	0.1192	**0.2202**	0.2172	0.1059	0.1875	0.1923	0.1973	0.1965	0.1944	0.1982	0.1973	0.2	0.1986	0.1059
sc.6	0.2076	**0.2599**	0.2592	0.2076	0.241	0.2455	0.2448	0.2496	0.2502	0.2496	0.2467	0.2486	0.2488	0.2076

I.II.IV SCT Kernel

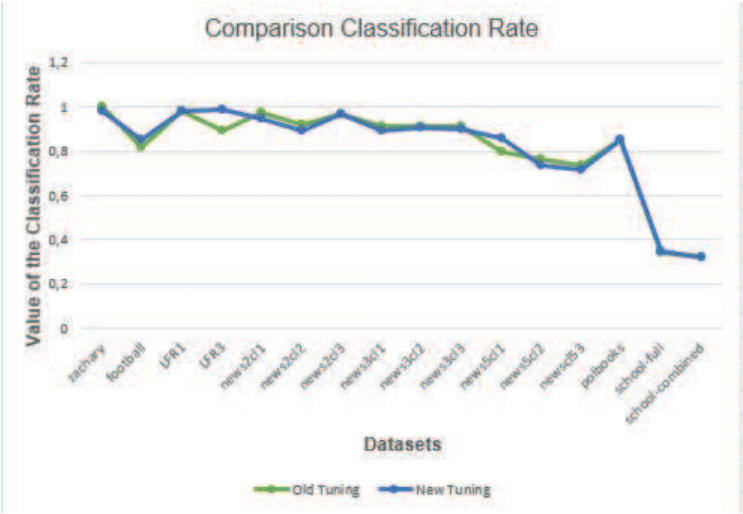
SCTKernel	*alpha*													
datasets	**10^-6**	**10^-2**	**10^-1**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1000**
zach	0.2916	0.2884	0.2505	0.6656	0.9273	0.8727	0.9087	**0.9820**	0.9406	0.964	0.977	0.957	0.8819	0.8458
foot	0.747	0.7351	0.7528	0.8492	0.8722	0.8847	0.8700	**0.8764**	0.8681	0.8759	0.8507	0.8543	0.8636	0.7109
LFR1	0.022	0.0216	0.0271	0.3291	0.8123	0.8872	0.9199	0.9107	0.8854	**0.9428**	0.9234	0.9296	0.9334	0.4416
LFR3	0.0603	0.0556	0.0924	0.8359	0.9211	0.9155	0.9383	0.9295	0.9263	**0.9401**	0.9310	0.9293	0.9173	0.8969
N2A	0.0194	0.0203	0.0314	0.5733	0.676	0.7315	0.7021	**0.7878**	0.7707	0.7580	0.7705	0.7543	0.7697	0.5609
N2B	0.0218	0.0239	0.0225	0.4158	0.4988	0.5511	**0.5883**	0.5680	0.5744	0.5822	0.5638	0.5809	0.5782	0.4565
N2C	0.0178	0.0206	0.0288	0.6266	0.6135	0.682	0.7028	0.7670	0.7865	0.7473	0.7872	0.7812	**0.8139**	0.5604
N3A	0.027	0.0393	0.0336	0.5997	0.6775	0.6363	0.6687	0.6603	0.6755	**0.7005**	0.6829	0.6928	0.6840	0.6746
N3B	0.0345	0.0382	0.0435	0.6061	0.6234	0.7018	0.6706	0.7103	0.7135	0.7082	**0.7326**	0.7098	0.6961	0.5159
N3C	0.0349	0.0273	0.0452	0.5072	0.5789	0.6335	0.6538	0.6774	0.6754	0.7096	0.7024	0.7216	**0.7397**	0.6356
N5A	0.0378	0.0393	0.0444	0.5347	0.5556	0.5691	0.5762	0.5741	0.5846	0.5971	0.5917	0.5992	**0.6061**	0.5898
N5B	0.0381	0.0362	0.0429	0.4918	0.5352	**0.5644**	0.5606	0.5572	0.5553	0.5597	0.5515	0.5554	0.5409	0.4730
N5C	0.0415	0.0455	0.0415	0.4865	0.5291	0.5314	0.5332	**0.5451**	0.5439	0.5412	0.5402	0.544	0.5371	0.4913
PolB	0.4683	0.4502	0.4636	0.5739	0.5763	0.5707	0.5714	0.5678	0.5684	0.5687	0.5713	0.5744	0.5715	**0.5889**
sc.11	0.1367	0.1413	0.1384	0.2829	0.2967	0.3013	0.303	0.3036	0.3073	0.3027	0.3051	**0.3087**	0.3065	0.2921
sc.6	0.0692	0.0702	0.0733	0.1097	0.1152	0.1170	0.1187	0.118	0.1195	0.1197	0.1186	0.1190	**0.1198**	0.1173

I.III Comparison of the two tuning procedures

I.III.I Comparison on the ARI

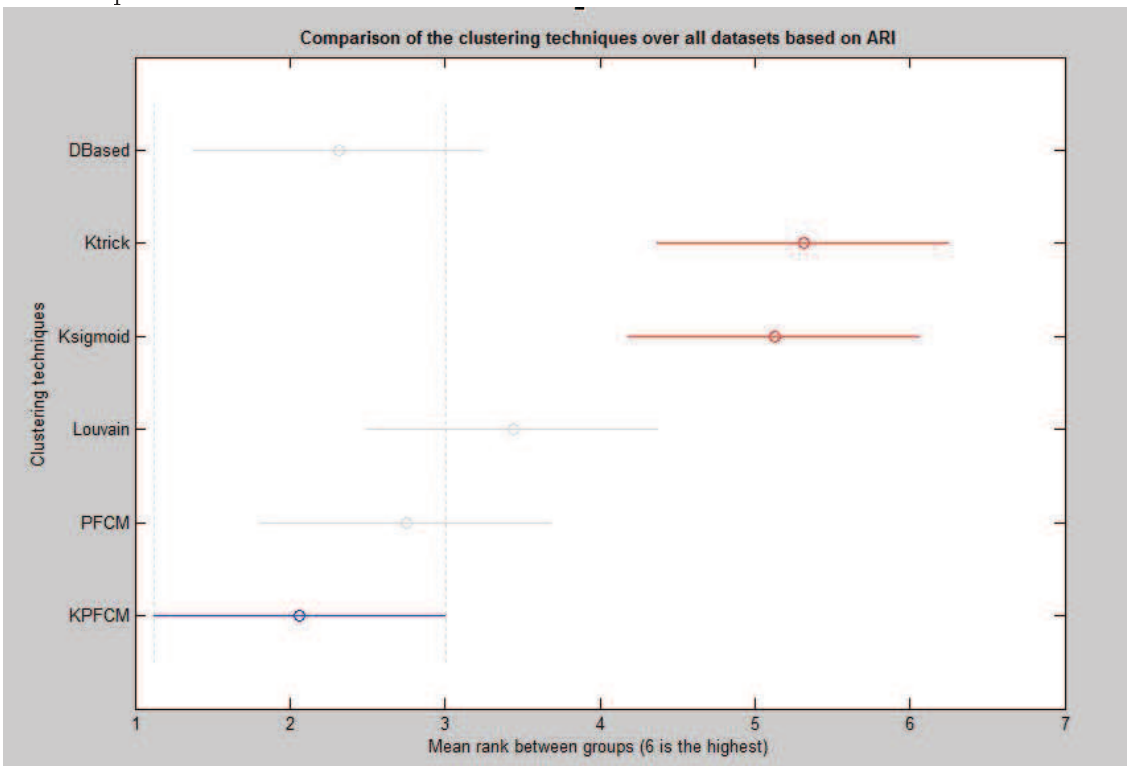


I.III.II Comparison of the Classification rate



Appendix II : comparison of methods

II.I Comparison of the ARI over all datasets



II.II Comparison of the classification rate over all datasets