Μέθοδοι Εξόρυξης Γνώσης από Συλλογές Εγγράφων

Η ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης

του Τμήματος Πληροφορικής Εξεταστική Επιτροπή

από τον

Αργύρη Καλογεράτο

ως μέρος των Υποχρεώσεων για τη λήψη του

ΔΙΔΑΚΤΟΡΙΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

Απρίλιος 2013

Τριμελής Συμβουλευτική Επιτροπή

- Αριστείδης Λύκας, Αναπληρωτής Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων (Επιβλέπων).
- Κωνσταντίνος Μπλέκας, Επίκουρος Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων.
- Ανδρέας-Γεώργιος Σταφυλοπάτης, Καθηγητής της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Ε.Μ.Π.

Επταμελής Εξεταστική Επιτροπή

- Αριστείδης Λύκας, Αναπληρωτής Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων (Επιβλέπων).
- Κωνσταντίνος Μπλέκας, Επίκουρος Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων.
- Ανδρέας-Γεώργιος Σταφυλοπάτης, Καθηγητής της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Ε.Μ.Π.
- Ευαγγελία Πιτουρά, Αναπληρώτρια Καθηγήτρια του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων.
- Παναγιώτης Τσαπάρας, Επίκουρος Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων.
- Μιχαήλ Βαζιργιάννης, Καθηγητής του Τμήματος Πληροφορικής του Οικονομικού Πανεπιστημίου Αθηνών.
- Δημήτριος Γουνόπουλος, Καθηγητής του Τμήματος Πληροφορικής και Τηλεπικοινωνιών του Ε.Κ.Π.Α.

Dedication

I dedicate this work to my family and to Augousta.

I also dedicate it to all people struggling for socially accessible Public Education of high quality.

Αφιερώνω την εργασία αυτή στην οικογένεια μου και στην Αυγούστα.

Επίσης την αφιερώνω στους ανθρώπους που αγωνίζονται για τη Δημόσια Δωρεάν Παιδεία ώστε να είναι κοινωνικά προσβάσιμη και υψηλής ποιότητας.

Acknowledgements

I thank the advisory committee, Assoc. Prof. Aristidis Likas, Assist. Prof. Konstantinos Blekas, and Prof. Andreas-Georgios Stafylopatis, for their cooperation and support. I would also like to thank the rest of the members of the examination comittee, Assoc. Prof. Evaggelia Pitoura, Assist. Prof. Panayiotis Tsaparas, Prof. Michalis Vazirgiannis, and Prof. Dimitrios Gunopulos.

I would like to thank especially my advisor Assoc. Prof. Aristidis Likas for his valuable assistance and contribution to my research during the past years. But most of all, I should thank the person Mr. Aristidis Likas for being all these years consistent to the academic values and practices which created a exceptional environment to develop myself. This was a great lesson and helped me maintaining my determination for the future.

I would also like to thank Dr. Dimitrios Tzikas, Dr. Vasileios Chasanis, and the PhD candidate Grigorios Tzortzis, from the Department of Computer Science of University of Ioannina, for doing their best to create a nice environment to work into.

I also thank my very close friends Andreas Vasilakis, Georgios Margaritis, Nikolaos Papanikos, and Andromachi Hatzieleftheriou, all PhD candidates at the Department of Computer Science of University of Ioannina, for the endless but always interesting talks we had and all the great times we had in Ioannina.

Finally, I would like to thank my parents Odysseas and Svetlana, and my brother Nickolas, for always believing and unconditionally supporting me. I am grateful to my parents for the additional reason that they were the primary 'funding source' that made possible for me to reach this far with my education. I should especially thank Augousta for being supportive and full of understanding to me during my work for this thesis.

Contents

List of Figures

List of Tables

Abstract

Kalogeratos, Argyris, O.

PhD, Computer Science Department, University of Ioannina, Greece. April 2013.

Knowledge Extraction Methods from Document Collections.

Thesis Supervisor: Aristidis Likas.

This thesis studies the problem of document clustering. Given a document collection, at first, preprocessing, and feature extraction take place. As a result, each document is usually represented using a vector space model where the non-negative dimension weights describe the significance of the respective term features. The properties of such a feature space are: i) the high dimensionality that is of the order of thousands of features, and ii) sparsity which reaches 99%. In this dissertation, methods are studied and developed for document representation and knowledge extraction regarding the cluster structure of a dataset.

At first, a vector space model is presented which, without supervision, revisits the traditional assumption about the term independence. A Global Term Context Vector is computed for each term feature of the collection, which embeds the context in which a term appears in the documents (term co-occurrences). Next, a semantic matrix is constructed based on which the document vectors are mapped in a denser feature space of the same dimensionality. The effectiveness of the proposed representation model is experimentally studied in the context of document clustering.

The second contribution of this work is the k-synthetic prototypes clustering method that is based on the spherical k-means. Its novelty lays at the introduction of the synthetic

prototypes as cluster representatives. The proposed incremental approach for the computation of a synthetic prototype uses the K nearest neighbors of the cluster medoid. The interesting property of this approach is that it favors the representation of the documents of the dominant class in the cluster. In this way the clustering algorithm manages to overcome problems caused by bad initializations. In the experimental study, this method is compared to a series of widely-used document clustering techniques.

In the chapter that follows, incremental clustering algorithms are studied, which add the $k+1$ data based on a solution containing k clusters. A general framework for incremental clustering is presented which applies partial update of the solution when introducing the $k+1$ cluster. This framework covers known incremental algorithms, such as bisecting k-means, global k-means, and various extensions of them. Next, global k-synthetic prototypes algorithm is proposed which is experimentally compared to existing incremental approaches achieving better clustering results on document collections.

The next chapter concerns the problem of estimating the number of clusters in a dataset. Dip-dist criterion is proposed which considers each object of a cluster as a '*viewer*' and applies a univariate statistic hypothesis test, the dip-test, to examine unimodality in the distribution of the distances between the viewer and the rest of the objects in cluster. This criterion is incorporated by the incremental dip-means method. The only assumption of this method is the unimodality of all clusters. Important advantages are: i) the unimodality test is applied on univariate distance vectors, and ii) it can be directly applied with kernel-based methods, since only the pairwise distances are involved in the computations. Experimental results on artificial and real datasets indicate the effectiveness of our method and its superiority over analogous approaches.

Περίληψη

Αργυρης Καλογεράτος του Οδυσσέα και της Σβετλάνας.

PhD, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Απρίλιος, 2013.

Μέθοδοι Εξαγωγής Γνώσης από Συλλογές Εγγράφων.

Επιβλέποντας: Αριστείδης Λύκας.

Η παρούσα διατριβή ασχολείται με το πρόβλημα της ομαδοποίησης εγγράφων (document clustering). Δοθείσης μία συλλογής εγγράφων φυσικής γλώσσας (corpus), καταρχήν εφαρμόζεται προεπεξεργασία και εξαγωγή χαρακτηριστικών όρων (terms). Ως αποτέλεσμα, κάθε εγγραφο συνήθως αναπαρίσταται με ένα διανυσματικό μοντέλο (vector space model) όπου το μη αρνητικό βάρος κάθε διάστασης περιγράφει τη σημαντικότητα του αντίστοιχου χαρακτηριστικού όρου. Οι ιδιότητες αυτού του χώρου αναπαράστασης είναι: α) η πολύ υψηλή διάσταση της τάξης των χιλιάδων χαρακτηριστικών, και β) η αραιότητα που αγγίζει το 99% (high dimensionality and sparsity). Στη διατριβή μελετώνται και αναπτύσσονται μέθοδοι αναπαράστασης και εξαγωγής πληροφορίας σχετικά με τη δομή ομάδων στη συλλογή εγγράφων (cluster structure).

Αρχικά προτείνεται ένα μοντέλο διανυσματικής αναπαράστασης εγγράφων, το οποίο, δίχως επίβλεψη, επανεξετάζει την παραδοσιακή υπόθεση ανεξαρτησίας των όρων (term independence). Για κάθε όρο του λεξικού εξάγεται το αντίστοιχο γενικευμένο διάνυσμα συμφραζομένων όρων (global term context vector) το οποίο ενσωματώνει τη συμφραζόμενη πληροφορία γύρω από τις εμφανίσεις του όρου στα έγγραφα (συν-εμφανίσεις όρων). Στη συνέχεια, κατασκευάζεται ένας σημασιολογικός πίνακας βάσει του οποίου προβάλλονται τα διανύσματα δεδομένων σε έναν πυκνότερο χώρο ίδιας διάστασης. Στο στάδιο αυτό μελετή-

θηκε η συμβολή του προτεινόμενου μοντέλου αναπαράστασης στην ομαδοποίηση εγγράφων.

Ύστερα παρουσιάζεται η μέθοδος ομαδοποίησης εγγράφων k-συνθετικών πρωτοτύπων (k-synthetic prototypes). Η μέθοδος βασίζεται στον σφαιρικό k-μέσων (spherical k-means) με την πρωτοτυπία ότι εισάγει τους συνθετικούς αντιπροσώπους για τις ομάδες. Η προτεινόμενη αυξητική προσέγγιση για τον υπολογισμό ενός συνθετικού αντιπροσώπου χρησιμοποιεί τα K αντικείμενα που βρίσκονται εγγύτερα στο ενδιάμεσο αντικείμενο μίας ομάδας (medoid). Η ενδιαφέρουσα ιδιότητα αυτής της προσέγγισης είναι ότι ευνοεί την αναπαράσταση της κυρίαρχης κλάσης δεδομένων σε μία ομάδα επιτρέποντας με αυτό τον τρόπο την αποφυγή λύσεων τοπικών ελαχίστων λόγω κακής αρχικοποίησης. Στην πειραματική μελέτη συγκρίνεται η μέθοδος αυτή με μία σειρά από ευρέως χρησιμοποιούμενους αλγόριθμους ομαδοποίησης.

Στη συνέχεια, μελετώνται αλγόριθμοι αυξητικής ομαδοποίησης (incremental clustering), οι οποίοι εισάγουν την $k+1$ ομάδα δεδομένων βασιζόμενοι στη λύση k ομάδων. Αρχικά παρουσιάζεται ένα γενικό πλαίσιο (clustering framework) που εφαρμόζει τη μερική ενημέρωση της $k+1$ λύσης κατά την εισαγωγή μίας νέας ομάδας (partial update). Το πλαίσιο αυτό καλύπτει γνωστούς αυξητικούς αλγορίθμους, όπως ο διαμεριστικός k-μέσων (bisecting k-means), ο γενικευμένος k-μέσων (global k-means), και διάφορες παραλλαγές τους. Προτείνεται, δε, ο γενικευμένος αλγόριθμος k-συνθετικών πρωτοτύπων (global k-synthetic prototypes) ο οποίος συγκρίνεται πειραματικά με υπάρχουσες αυξητικές προσεγγίσεις επιδεικνύοντας καλύτερα αποτελέσματα ομαδοποίησης σε συλλογές εγγράφων.

Η τελευταία ενότητα της διατριβής αφορά το πρόβλημα εκτίμησης του αριθμού των ομάδων σε ένα σύνολο δεδομένων. Για την προσέγγιση του προβλήματος προτείνεται το κριτήριο dip-dist το οποίο θεωρεί κάθε αντικείμενο της υπό εξέταση ομάδας ως ΄παρατηρητή΄ και εφαρμόζει ένα στατιστικό τεστ μονοτροπικότητας (unimodality dip-test) στην κατανομή των αποστάσεων μεταξύ του παρατηρητή και των υπολοίπων αντικειμένων της ομάδας. Στη συνέχεια, περιγράφεται η αυξητική μέθοδος διπ-μέσων (dip-means) της οποίας η μοναδική υπόθεση είναι η μονοτροπικότητα κάθε ομάδας. Τα πλεονεκτήματα της προτεινόμενης προσέγγισης είναι ότι το στατιστικό τεστ εφαρμόζεται σε 1Δ κατανομές, ενώ θα μπορούσε να χρησιμοποιηθεί και σε μεθόδους που βασίζονται στον πίνακα ομοιότητας (kernel-based methods), όπου δεν απαιτούνται τα πραγματικά διανύσματα δεδομένων.

Chapter 1

Introduction

1.1 Knowledge mining from document collections

During the last years the electronic means of communication have acquired a dominant role in developed societies. The plethora of services provided on the world wide web (WWW), such as electronic social networks, have made it the primary communication and entertainment tool for many people. One of the most important changes happened was that the user is now both *content consumer* and *producer* at the same time.

Nowadays, in the era of cloud computing, the data being produced, stored, and processed electronically, are massive in volume and present an increasing rate of growth. Electronic publishing, digital libraries with text articles, e-books, images and videos, e-mails, broadcast news articles, user blogs, and other conventional websites, are just some of the activities that need to manage large volumes of data. This data management burdens users, that have to spend time to organize or search content, and of course the

computer systems. Either in a local scale or in the large-scale of modern distributed systems, manual data management and processing are of unbearable economic cost and sometimes even impossible to be done in reasonable time. On the other hand, naive automatic methodologies fail to scale to real-life complex problems in terms of accuracy of results or computational cost. It is clear that, despite the improvement of computer systems performance, this computing power itself cannot meet the evolving modern needs. Efficient automatic or semi-automatic methods for *content-based document management tasks*, organization, and information retrieval, are of great significance.

Another great challenge where *machine learning* (ML) and *data mining* (DM) procedures can contribute to human knowledge and science is encapsulated in the quotation "*data can create new data*". Specifically, various scientific problems can be investigated by the means of processing large volumes of recorded information relevant to the problem. In this way, directly or more often after post-processing and external evaluation of the extracted information, new knowledge may be acquired (e.g. classifiers, interesting rules, feature correlations). In the worst scenario, some hints may be obtained to help the setup of focused further research. Examples are the analysis of human genetic material (DNA, RNA) for the identification of suspicious genes for various diseases, or the analysis of hundreds of thousands images from the web to extract visual features for object/scene recognition tasks. Large datasets have being collected the last decades and, obviously, the bottleneck towards taking advantage of such data volumes lays at the side of computer systems and the efficiency of the methods developed by computer scientists.

Text is the basic format in which information is represented, thus, the processing operations should be able to handle properly this type of data (e.g. transmission, archiving, indexing, and searching). Organizing and mining information from text data has become one of the most active scientific fields of ML and DM communities, usually called collectively as *text mining* (TM). With the term '*document*' we refer to the general data instance which may include information of the following types: text, images, videos, or any other composite multimedia content. Composite documents with various such data types is actually the most usual case in web. Computer algorithms cannot use the raw

format of these documents, hence a representation is required in a standard format such as the *vector space model* (VSM) [1, 2] where a vector stores the weight of significance for each feature of a data object. The extracted text features can be the different words, or more complex composite features. The *bag of words* (BOW) is the most traditional text document representation model, where the set of word-terms is called *vocabulary* and forms a *vector space* (VS).

It must be noted that both images and videos (a sequence of image frames) can also be represented in a similar way called *bag of visual words* (BOVW). Here, a *visual vocabulary* is constructed by processing low-level visual features (e.g. color, texture, shape) from the dataset, and the data objects are then mapped in the respective VS [3, 4].

One fundamental difference between text and other multimedia content is that text provides better low level features. Word-terms have specific discrete encoding with written letters and can be directly used to define the feature space of BOW model. Contrary, it is not that trivial to extract good quality low-level features from images and videos. Additionally, those features are usually represented in a continuous vector space (e.g. SIFT features [5]) and a vector quantization is required to transform them into a discrete feature set that could then be used in BOW model. The quantization is usually performed by means of clustering the low-level features to form the visual words. The mapping of features into the visual words can be done using the hard membership to one cluster or, alternatively, it is possible to introduce ambiguity using soft assignment to clusters [6]. Nevertheless, the *semantic-gap* is present, at any case, and multimedia content-based management faces problems similar to those for text documents when extracting the high-level semantics [7]. In fact, *semantic enrichment and smoothing* methods proposed for video representation have clear origin from text mining, such as the *locally weighted visual BOW* [8]. Another important difference is that, in contrast to the 1d term sequence of text, image features have 2d spatial structure that enables the use of spatial feature matching [9]. By projecting this spatial structure onto a proper direction, *visual sentences* can be created [10] and then language modeling is applicable.

Another paradigm of the importance of text processing is that text is the dominant

format of *descriptive metadata*. *Tagging* is a common practice where human assigns a set of textual terms to a data object, termed as *tags* or *text annotations*. Although, in the digital world, tagging was introduced for text mining and the web search, it is currently widely-used practice also for other multimedia content in order to help representation and retrieval. The need to increase the utilization of textual tags, and the need to tackle problems caused by misuse of tags by authors, have led to the development of various fully automated tagging methods, or others for tag recommendation to authors [11, 12].

All the above explain why text mining is one the of the most active ML and DM fields and the reason why numerous state-of-the-art methods that have been developed for text are successfully used in other data processing domains. However, the idea of *automatic organization of texts* comes from the early '60s. Until the late '80s, the most popular approach to partition a set of documents into groups was *knowledge engineering* (KE). A set of rules in *disjunctive normal form* (DNF) was used to determine the category of a document:

if (*DNF rule*) **then** (*category*).

A rule may examine the presence of a term, or the co-occurrence of various terms in the document, and each class is described by a set of such rules. Domain experts and engineers were responsible for the creation of the rules of the *knowledge base*. The main drawback of this approach is known from the field of *expert systems* and is called *knowledge acquisition bottleneck*. More specifically, the expert system cannot create new rules automatically, thus, it does not generalize/adapt to similar problems other than the problem it was trained on. Any change in its parameters (e.g. addition of categories, or new terms) requires the domain experts to re-intervene manually and update the ruleset. Moreover, the knowledge sources can be unreliable since domain experts may provide incomplete or incorrect information. Finally, the knowledge base is hard to build and very expensive to maintain. On the other hand, these systems provide interpretable decisions which is a requirement of high priority in some applications.

1.2 Machine learning for document management

Considerable research activity has been conducted since the early '90s. Back then, the first steps were made for machine learning techniques [13], [14] which present the advantage that the specification of the rules is severely limited (e.g. with classification the experts should classify manually a small set of documents that are examples used for training), or completely eliminated (using an unsupervised clustering approach). In this dissertation we focus on unsupervised document representation and clustering.

A fundamental difference between ML methods is the existence or absence of supervision. *Supervised learning* methods use an auxiliary dataset that contains example objects of the data categories we need to identify in the unknown data. In contrast, *unsupervised learning* methods attempt to discover the underlying group structure of the objects by directly processing the unknown input data. One should realize that unsupervised learning is not just the last option for the case where there is no '*explanatory*' labeled data provided by human:

- supervised learning aims to '*imitate*' the human perception on a problem by discovering the important rules to reproduce the indicated behavior, whereas,
- unsupervised learning aims to capture relations between objects and then to discover the intrinsic structure of data, imposed by those relations.

In other words, even if we have supervised information, we may still choose to apply unsupervised learning in order to extract information we are not presently aware. Furthermore, it is interesting to mention that the two approaches can be complementary. In many applications, unsupervised techniques are used during data preprocessing in order to estimate individual parameters of the problem. This is done independently to the supervised technique that is finally applied. Reversely, it is possible to reinforce the unsupervised learning procedure using a labeled dataset along with the unknown data (partially labeled dataset). This latter hybrid approach is called *semi-supervised learning*. Among the popular supervised learning problems are *classification* and *regression*, while respectively for unsupervised learning, *clustering* and *density estimation*.

A second conceptual difference among ML methods is that they may adopt a *discriminative learning* or, alternatively, a *generative learning* approach to solve a problem. A *generative model* learns the problem and its decision is based on how interpretable is a case under the different possible scenarios it is trained to handle. A *discriminative model* follows a much simpler way: it does not learn the problem itself, but focuses on learning the differences between the possible scenarios in order to discriminate them. This categorization is mostly mentioned in the context of supervised learning, but it applies also in the unsupervised setting. Recently, those two approaches are being used in combination to enhance the learning process [15].

Other characteristics may divide the ML techniques further, but vertically to the aforementioned cases. For example, if they can work with overlapping data categories (e.g. in classification and clustering), handle outliers or noise, handle categorical data, or data representations other than vectors, and others.

Unlike other ML problems where data are represented with a small set of features, even small text datasets carry large vocabularies and certain undesirable effects arise due to the *curse of dimensionality* [14]. The *high dimensional and sparse* (HDS) feature space in combination with linguistic phenomena such as *polysemy*, *homosemy* and *metaphors*, constitute an adverse setting for clustering methods. When a labeled training dataset is provided, several statistical options are available for feature selection [16, 17], even in case of multi-labeled data objects [18]. On the other hand, it is more complicated to select features in an unsupervised setting, which is usually achieved using heuristics [19–22]. Methods such as *latent semantic indexing* (LSI) [23], or *latent Dirichlet allocation* [24] (LDA), may discover the term correlations but they map the data into a feature space of much lower dimensionality. Vector space representations of other multimedia content present similar weaknesses due to the large number of features required to describe the data.

In general, it is a widely-known issue to have a difference between the human understanding about a data object, i.e. its actual perceived meaning and the information of its corresponding representation in a well-defined mathematical space which enables com-

puter processing. This is called as *semantic gap* and its extend depends on the complexity of the original data and the efficiency of the involved representation model. In what concerns documents, the abstract and complex semantics of text and multimedia content is very difficult to be encoded by a formal representation, and this is one of the directions on which there has been a lot of research activity recently. The term semantic gap is also used to express the difficulty to move from a data representation with low-level features to another representation of higher semantic level (e.g. concept-based representation for text or images).

1.2.1 Document classification

In *classification*, or *categorization*, the aim is to identify specific document categories in an unknown input dataset. In the simplest case of one category, a single-class classifier is trained to determine whether an object belongs to that category. The output decision can be *binary*, or *probabilistic*, also called as *hard* and *soft decision*, respectively. It is straightforward for soft decisions to be produced by a *generative classifier* since they are inherent (e.g. naive Bayes). However, specialized techniques may produce such class interpretation weights from a *discriminative classifier*, e.g. with a calculation that uses the distance of an object from a separation hyperplane as in *support vector machines* (SVM) [25].

Multi-class classification is implemented either using a single classifier that can handle more than one categories (e.g. neural networks, decision trees etc), or using multiple two-class classifiers that are employed as components of the classification model, each one responsible for one data category (e.g. one-versus-all SVM). Since there is a rich literature available for two-class classification, the use of multiple such classifiers offers a direct generalization, although their combination usually requires careful setup.

Formally, classification is the process where, provided a dataset D=$\{d_1,...,d_N\}$ consisting of |D|=N objects and a set L=$\{l_1,...,l_M\}$ with |L|=M predefined category labels, a binary value is assigned to each pair (d_i, l_j). The problem can be formulated as the

approximation of an unknown mapping function f: D×L→{$true$,$false$}, which would perfectly assign objects to their true categories. This is approximated by the function implemented by the classifiers, f_c: D×L →{$true$,$false$}, which attempts to minimize the decision disagreement between f_c and f. The learning of the approximation function is conducted using examples provided in a training set Tr={$(d_i,\ K_i)$, i=1,...,$|N_{\text{Tr}}|$}. The requirement for a training set makes classification a supervised ML technique.

Many popular classifiers have been applied to text documents and other multimedia content, such as *support vector machines*, *neural networks*, *naive Bayes*, *linear least square fit*, *k-nearest neighbors*, *inductive logic programming*, *genetic algorithms*, *rule-based systems*, and *statistical analysis*. Extensive comparisons of text classification methods can be found in [26, 27]. As for *regression*, the predicted output is not a discrete label but a set of real numbers. It is less popularly applied in document datasets, however there are such methodologies in literature [28].

Classification finds practical application in a wide variety of domains, such as news filtering, organization and retrieval, opinion mining, e-mail categorization and Spam filtering. All the aforementioned organization and retrieval problems can be defined analogously for other multimedia data types.

1.2.2 Document clustering

Document clustering is an unsupervised learning approach that automatically segregates similar documents of a corpus into the same group, called *cluster*, and dissimilar documents to different clusters. It is employed in both contexts of data analysis [29]:

- in *exploratory data analysis* where the aim is to discover patterns in the input data which then would help to formulate hypotheses about the data properties
- in *confirmatory data analysis* where the target is to validate empirically a given hypothesis by analyzing the input data.

This document management approach has become very popular due to the nature of modern problems. More specifically, when the order of the information volume is hundreds

of thousands of documents (even millions), with dynamic changes in their thematic groups, then supervision is particularly disadvantageous and costly in many respects. Computationally, it is NP-hard to find the optimal grouping of data even for 2-dimensional data [30], or the 2-clustering case [31]. However, there exist efficient algorithms that approximate the solution in $O(N^2)$ time.

Formally, provided a dataset D with N unlabeled documents, a solution C is seeked that partitions the dataset into M clusters of similar objects, where $C=\{c_j: j=1,...,M\}$ and c_j is the set of objects assigned to j-th cluster (definition for *hard clustering*). The number of clusters M is usually predetermined and provided as an input parameter to the clustering algorithm. However it is highly desirable for a method to be able to determine the number of clusters without external information. Although there are plenty of efficient clustering algorithms, finding the number of clusters is still a problem for which there exist no general and effective approach [32]. Clustering utilizes low-level structural information, the relations between objects (*similarities*, or *dissimilarity*), to infer the high-level group structure of data. A function that measures the pairwise object relation is called (*dis*)*similarity function* and plays a crucial role in the performance of the clustering process.

Ideally, each cluster would correspond to one underlying class of objects, whereas it is worth to note that in complex problems there are more than one '*correct*' or '*reasonable*', data partitions. Therefore, the extracted cluster structure is not necessarily expected to coincide with human perception. The positive effect of this issue is that clustering may discover new knowledge about data relationships and structure. On the other hand, the negative effects are mainly three. First, most clustering algorithms conclude a clustering structure irrespectively to whether an actual structure exists in data. Hence, it is advisable, before applying clustering, the data analyst to investigate the '*clusterability*' [33–35] of the dataset in order to decide whether there is interesting cluster structure to be extracted. Of course this is another difficult problem that depends on the characteristics of the clustering algorithm we use. The second difficulty posed by the multiple correct clusterings is how to define a proper *evaluation process* for the quality of clustering solu-

tions without taking into account the context of each clustering method. In other words, how can the analyst select a result among various clusterings produced by conceptually diverse algorithms? Third, it has been shown that it is not easy to define a unified clustering framework for all methods due to its in principle incompatibility [36]. All the above indicate that data clustering is an ill-posed problem without sound general theoretical background. This is a direction on which much research is recently focused.

There is a large number of clustering methodologies with different characteristics [37–40]. In literature, these algorithms are separated into three broad categories:

- *hierarchical clustering*, which produce nested groups following a general-to-specific (or top-down) approach. Similarly, it is possible to adopt a specific-to-general (or bottom-up) approach.
- *partitional clustering* (also non-hierarchical or flat) which iteratively improve the quality of clustering based on some kind of unsupervised clustering evaluation criteria.
- *density-based clustering* which recognize continuous dense areas in data space as clusters. These groups may have arbitrary shape.

One could revise the above categorization and introduce *incremental clustering*. In order to build a solution with M clusters, incremental methods start with one cluster containing all data objects (or a given small number of clusters) and incrementally add more clusters until the desirable number of clusters M is reached. It is essentially different to top-down hierarchical clustering because in incremental clustering no cluster hierarchy is preserved.

It is possible to define other categorizations of clustering methods, for instance, according to whether a method assigns each object in only one or more than one clusters. In this case, we may distinguish *hard* and *soft clustering*, respectively. Also methods may differ on whether they work with all dataset known in advance (*offline clustering*), or they assume that documents arrive sequentially (*online clustering*). *Kernel-based clustering* methods require only the pairwise object similarities (*kernel matrix*), while typical methods require the actual data vectors. Finally, for high dimensional data, there are methods

that discover clusters by determining *subspaces* of the feature space where clusters are more clearly observable (*subspace clustering*) [37, 40].

The most widely-used algorithms for document clustering are those based on popular general clustering methods, such as k-means [41] and hierarchical agglomerative clustering [42]. These methods may appropriately modified to adapt to the special needs of high dimensionality and sparsity of document feature spaces. Finally, subspace clustering methods and generative topic models have recently shown to perform well.

Some modern information management applications of document clustering are:

- grouping of data to assist storage, caching, indexing and retrieval in large-scale systems,
- feature space summarization and codebook generation for representing high dimensional multimedia data,
- automatic creation (or enrichment) of ontologies, knowledge bases, or general taxonomies of information entities,
- word sense disambiguation,
- recommendation systems,
- visualization and browsing document collections,
- automatic summarization of texts, or groups of texts,
- segmentation of data streams in events/stories/topics,
- automatic metadata generation (tagging).

1.3 Thesis contribution

This dissertation deals with the problem of clustering documents. The main difference of this problem compared to general clustering is the nature of the data need to be processed. Document data are high dimensional and sparse (HDS) and put additional difficulties to the already difficult problem of data clustering. We study the document clustering problem in various perspectives:

- the vector representation, where the traditional feature (term) independence in VSM seems to be an over-simplistic assumption,
- the prototype-based cluster representation and the document clustering algorithm,
- finally, we revisit one of the most important, however still open, problems of data clustering: the estimation of the number of clusters.

The organization of the rest of the thesis follows.

In Chapter 2, we provide the important preliminaries and background regarding the preprocessing, representation and clustering of document collections. We also provide a detailed presentation of the related state-of-the-art approaches.

In Chapter 3, we present the *global term context vector model* (GTCVM) for text document representation [43]. It is an extension to VSM approach that maps document vectors onto a new feature space based on *term similarity*, where clustering can achieve better solutions. The method proceeds as follows: i) it captures local contextual information for each term occurrence in the term sequences of documents; ii) the local contexts for the occurrences of a term are combined to define the global context of that term; iii) a proper semantic matrix is constructed using the global context of all terms; iv) this matrix is further used to linearly map traditional VSM (bag of words - BOW) document vectors onto a '*semantically smoothed*' feature space where problems such as text document clustering can be solved more efficiently. We present an experimental study demonstrating the improvement of clustering results when the proposed GTCVM representation is used compared to traditional VSM-based approaches.

In Chapter 4, we investigate the centroid-based cluster representation for HDS data. We propose the idea of *synthetic cluster prototype* that is computed by i) first selecting a subset of cluster objects (cluster members), then ii) computing the representative of these objects and, finally, iii) selecting important features. Further, we introduce the *MedoidKNN* synthetic prototype that favors the representation of the dominant data class in a cluster. These synthetic cluster prototypes are incorporated into the generic spherical k-means procedure leading to a robust clustering method called *k-synthetic*

prototypes (*k-sp*) [44]. Comparative experimental evaluation demonstrates the robustness of the approach especially for small datasets and clusters overlapping in many dimensions and its superior performance against traditional and subspace clustering methods.

In Chapter 5, we present a framework for incremental prototype-based clustering that is based on *partial updates* (PU) on a given solution. A PU is defined by the *activity state* (active or inactive) of clusters, objects, and their prototypes, indicating whether they are kept fixed in a certain k-means iteration. Two widely-known incremental clustering approaches, *global k-means* and *divisive k-means*, are revisited and unified according to this analysis. Focusing on HDS spherical data, we discuss in detail the difficulties encountered when increasing the order of a current k-clustering solution by adding one new component. Then, the use of synthetic cluster prototypes is extended for incremental prototype-based clustering. To this end, we propose the novel *global k-synthetic prototypes* (*gk-sp*) clustering algorithm, which iterates similarly to the global k-means algorithm. The gk-sp method uses the *k-synthetic prototypes* method for fine-tunning the k-solution, and introduces a partial update scheme to setup the initial $k+1$ prototypes for the refining phase. Similarly, the *bisecting k-sp* (*bk-sp*) and *global bisecting k-sp* (*gbk-sp*) are also proposed. Experiments on well-known and artificial datasets illustrate that the proposed gk-sp method outperforms other competitive incremental and flat methods of the k-means family, in terms of clustering error and external clustering evaluation measures.

In Chapter 6, we deal with the problem of estimating the number of clusters in a dataset which is a key problem in data clustering. For this purpose, we present *dip-means*, a novel robust incremental method to learn the number of data clusters [45]. This method can be used as a wrapper around any iterative clustering algorithm of k-means family. In contrast to many popular methods which make assumptions about the underlying cluster distributions, dip-means only assumes a fundamental cluster property: each cluster to admit a unimodal distribution. The proposed algorithm considers each cluster member as an individual '*viewer*' and applies a univariate statistic hypothesis test for unimodality, the *dip-test*, on the distribution of distances between the viewer and the cluster members. Important advantages are: i) the unimodality test is applied on

univariate distance vectors, ii) it can be directly applied with kernel-based methods, since only the pairwise distances are involved in the computations. Experimental results on artificial and real datasets indicate the effectiveness of our method and its superiority over analogous approaches.

Finally, in Chapter 7 we provide an overall review of the results of our research and indicate interesting directions for future work.

Chapter 2

Background and Preliminaries

2.1 Characteristics of natural language document collections

Natural languages are complicated codes capable to encode non-trivial information. Humans use languages to communicate and hereafter, this is the kind of natural language to which we mainly refer. Every such language evolves in time with respect to syntax, vocabulary, and word meanings, to meet the communication needs.

Since we mainly refer to text documents, it should be noted that we use the general terminology of ML along with terminology from text mining. Thus, a *document* is a data object, the *document vocabulary* refers to the set of all the distinct features a document may have, and the *corpus vocabulary* to the features extracted from all the dataset which is document collection (corpus). Next, we describe the basic issues concerning the *natural*

language processing (NLP) required to apply of machine learning methods in the case of documents.

2.1.1 Linguistic phenomena and complex semantics

Three linguistic phenomena can induce severe ambiguity in the automated processing of text. *Polysemy* is the phenomenon where a term has different meaning depending on the context of its appearance in a text. In what concerns data clustering, if there are more than one groups in the dataset that base their formation on such words, then it would be difficult to discriminate those groups. For example, in the field of computer science we may find many words, such as '*fork*', '*pipe*', '*disk*', '*memory*', etc., that have completely different meaning in a context out of that field.

Homonymy is the phenomenon where several terms correspond to an identical concept, generally, or when they appear in a particular context. As an example, the words '*car*', '*auto*', '*vehicle*', '*automobile*', or the words '*street*', '*avenue*', and '*highway*'. It is also common to use abbreviations, or acronyms, instead of the original word or phrase respectively. The generalized problem is that each word corresponds to a concept that is related with other concepts. All concepts can be thought to form a *conceptual hierarchy*, e.g. a '*car*' is primarily a '*machine*', then a '*vehicle*' and then a '*car*'. This means that two terms may have meaning similar to some relative extent, e.g. '*car*', '*motorbike*', '*bus*' are all vehicles and means of transport. Another indicative example is that it is very usual to refer to an object indirectly by mentioning its brand and a system is desirable to be able to realize that; for example, "*riding a Harley Davidson*" and "*riding a BMW*" means more or less the same thing.

Composite terms refer to the cases where more than one words are combined into a term that has a special meaning, e.g. '*Olympic Games*', '*New York*', '*city block*', '*machine learning*', etc. It is a very common phenomenon and can be treated mainly using a list of such terms created manually. Automated techniques also exist; after collecting information about the probability of two consecutive words to form a composite term

(e.g. '*New York*'), then the context of a particular appearance can indicate whether the composite term should be identified.

All these problems are tackled using methods from the fields of natural language processing and computational linguistics. More specifically, *word sense disambiguation* (WSD) uses clustering to group parts of text in which a term appears. Such a grouping may reveal the conceptually different uses of a term [46]. However, the number of clusters is a very critical parameter. Other approaches to tackle the semantic ambiguity of terms are based on the well-known WordNet knowledge base [47] or *Wikipedia* [48]. Related methods can be found in [49–51], respectively. Even the search results obtained from a search engine can be utilised for the purpose of WSD.

2.1.2 Statistics: High dimensionality and sparsity

Text data are naturally represented using a large number of different words. It may include standard words that could be met in a dictionary, idiomatisms, composite terms, etc. A text document can be considered as a sequence of individual word terms structured in chapters, sections, paragraphs and, at the lowest level, in sentences. The order of magnitude of the vocabulary length of a normal sized document collection (e.g. about 10.000 documents of news groups articles) can be of the order of 10 thousands to 100 thousands different words (note: when considering all the different raw terms without preprocessing). This implies that any representation of this information would use many different features, and hence we have a space of high dimensionality. Generally, as dimensionality increases, the space where data are represented increases exponentially and ML requires a number of data objects of the respective order to train with. It is well-known that in high dimensions ML methods encounter certain undesirable effects which arise due to the *curse of dimensionality* [15].

Moreover, each document does not contain all the different vocabulary terms that are present in the overall corpus. It has been observed that a document may have less than 1% of the global corpus vocabulary [52] (non-zero vector dimensions). Furthermore, there

are terms in the corpus vocabulary that do not appear in a given document although they are relevant to its content. This is due to the fact that each document usually is a *semantically narrow instance* of a much more general document class, called *topic*.

2.1.3 Dynamics: Power-laws in natural languages

Text is not a photograph taken from a natural scenery; it is written by an author aiming to help readers understand its meaning. The human writing process induces interesting dynamic characteristics in text and, as a consequence, special statistical properties. It has been empirically observed that, in a large text, the frequency of a term is a *power-law*[1] of the frequency-based rank. This is the *Zipf's power-law* [53] and implies where, if T is the length of the text considered as a term sequence, r is the rank of a term (its position in the ordering), and $n(r)$ the number of term appearances in the sequence, then the frequency of that term is given by:

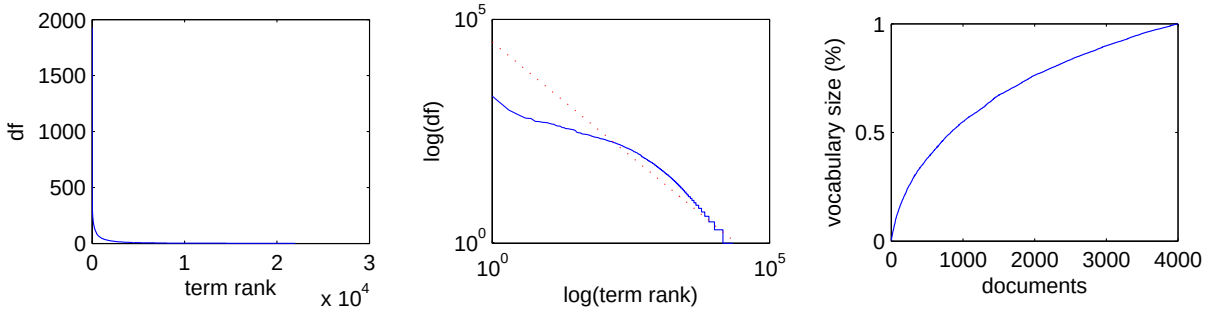
$$f(r) \approx \frac{n(r)}{T}. \tag{2.1}$$

The origin of the power-law is the observation that, in large text, the second most frequent term has about 1/2 frequency of the most frequent term, the third most common has about 1/3 the frequency of the most frequent term, etc. However, this does not perfectly coincide with what is observed in actual data. The small deviations are associated with how rich is the vocabulary of a language, or the writing style of the author. For this reason, Zipf's power-law is formulated in a parametric statistical model:

$$f(r) = \frac{c}{r^{\beta}} \Leftrightarrow \tag{2.2}$$

$$\log f(r) = -\beta \log r + \log c. \tag{2.3}$$

[1]The mathematical relation between two quantities is called *power-law* if the value of one of the quantities is a power function of the other quantity.



(a) (b) (c)

Figure 2.1: Statistics from a text dataset with 4000 documents from 8 classes. (a) the document frequency of terms as a function of their rank (Zipf's powerlaw), (c) the respective log-log scale of (a), and c) the increase of vocabulary as a function of the number of documents considered in a dataset (Heaps' powerlaw).

Interestingly, the logarithmic form shows the linear relationship between the frequency of a term and its rank. The original formulation is obtained if c and β is set to 1.

Another important empirical observation is that the vocabulary growth of a text, namely the number of distinct terms, is linear to the text length. This is stated by the *Heaps' power-law* [54], which is closely connected with Zipf's law and often are met together in many domains [55, 56]. Particularly, both have been confirmed in several Indo-European languages, whereas they do not fit so well in languages like Chinese, Japanese and Korean that have limited dictionary sizes [57].

Let us denote as V_t the vocabulary length when the length of the term sequence of a text is t, and the parameter $a \in [0,1]$ is the probability to introduce a new term (not previously appeared in the text), then the power-law is expressed as follows:

$$|V_t| = \alpha \; t^{\nu}, \;\; \nu \in (0,1). \tag{2.4}$$

Again, the parameters a and ν help the model to fit into slightly deviated cases.

The aforementioned empirical laws refer to a single long text, however, a collection with many documents can be considered as such. This consideration is asymptotically correct since, the topics of the documents can differ and they are not created by *one continuous writing process*. As the dataset size becomes larger, both laws still apply.

Heaps' law also confirms that the negative effects of the curse of dimensionality are more intense in small datasets due to the higher *relative dimensionality* (see Sec. 4.2.1). In addition, if N_t is the number of documents processed by an information management system at a time instance t, then from Eq. 2.4 we may express the order of the respective processed corpus vocabulary [52] and, thus, the respective computational and memory requirements:

$$|V_t| = \alpha \ t^{\nu}, \ \nu \in (0, 1). \tag{2.5}$$

In Fig. 2.1a and Fig. 2.1b the document frequency distribution of terms is plotted in original (long tailed) and log-log scale for a dataset of 4000 documents (stemmed vocabulary, 8 classes), while Fig. 2.1c presents the increase of vocabulary as a function of the number of the processed documents of the dataset.

2.2 Overview of a document clustering system

Each clustering application can be decomposed into five components which depend on each other according to the following order:

1. the *information retrieval procedure* (IR) extracts document features from input raw data which may have arbitrary format (HTML, XML, plain text, etc).
2. the *document representation model* maps in a data space to enable the processing by computer algorithms.
3. the *pairwise similarity/dissimilarity measure* that expresses the degree to which two data objects have some characteristics in common.
4. the *cluster model* is the mathematical representation of documents in a cluster (e.g. a term frequency vector, a probability density function, etc).
5. the *clustering algorithm* that partitions data objects into clusters relying on all the above.

The basic background regarding those components is discussed in the following sections of the chapter.

2.3 Data preparation

2.4 Preprocessing

In order to apply any clustering algorithm, the raw collection of text documents must be first preprocessed and represented in a suitable feature space. This process usually takes place offline, before the application of any learning algorithm. We should remark that the design of the preprocessing step depends on the problem we want to solve and on which semantic level we need to extract information. For example, there are cases where the syntax, or the sentence structure of the text, play important role. In other cases, more coarse concepts are required and hence we discard syntax structure and lexical fine-details. Document clustering that we deal with, usually works at a coarse semantic level that is described by unordered term frequencies (see Sec. 2.5), hence belongs to the second category. Although there are many text preprocessing and preparation software available, we implemented and used our own preprocessor. The presentation that follows refers to English language which is the language of the document collections we used.

A *parser* is responsible to process the raw character stream. In the case where the input is a structured language, such as HTML, or XML in general, then the initial step is to extract the informative text parts along with the metadata by parsing the markup elements. Otherwise, the parser tokenizes the input into individual *word tokens* in lower case. Moreover, a set of preprocessing procedures can be applied; some of them are considered to be traditional, while others are optional and used less often.

Then, *stemming* is applied, which aims to replace each word by its corresponding *morphological stem*. For instance, the words '*player*', '*playing*', '*played*' are all related with the verb '*play*', their stem. Porter's stemming algorithm, that we used, is rule-based and is the most popular approach for this purpose [58]. There are statistical stemmers [59], as well. For a recent comparative study for various stemmers see [60]. Stemming transformation, with the multiple-to-one term mapping, makes each document vocabulary shorter and more compact. In this way, the length of global corpus vocabulary is also

reduced and some terms to become more discriminative.

It is a standard approach to eliminate various trivial terms. It has been observed that the 10 most common words in the English language are about 20-30% of tokens in a text [61]. Words such as '*is*', '*the*', '*to*', '*for*', '*and*', '*of*' are present in almost every sentence of a text and are not characteristic of any topic. A *stopword list* is used to identify and discard terms such as '*the*', '*and*', '*of*', etc. Using the same fixed list to apply information retrieval across many different document collections may not be a good choice. There are stopword lists for various purposes and differ in terms of their length, and the inclusion of words that are non-informative under certain context. There are also approaches for automatic construction of such lists in unlabeled or labeled datasets [62, 63] that can be also used to enrich a standard fixed stopword list. We used a rather general-purpose list that contains about 570 terms.

Another list can be used to recognize various composite multiword terms, in order to treat them as a single feature. Automatic methods are available to achieve this, however we just merged multiword terms that were separated by a character such as '-' (e.g. '*intra-cluster*' or '*state-of-the-art*').

In contrast to the high *document frequency* (denoted as df) of the stopwords, other terms which appear in a small number of documents are also candidates for elimination. This approach is called *document frequency thresholding* (DFT) [64]. All these terms have usually very low *discriminative power*. A cut-off threshold $1 \geq df \geq 5$ is often used in practice. Nevertheless, the removal of these terms does not always increase the efficiency of ML methods, in fact what is usually observed is the opposite. However, the drastic reduction of the vocabulary length compensates for a slight deterioration in accuracy.

2.5 Document Representation

After preprocessing the N documents, the V derived word stems constitute the corpus *term vocabulary*, denoted as $\mathcal{V}=\{\nu_1, ..., \nu_V\}$. Thus, the finite term sequence of T vocabu-

lary terms of a text document is denoted as:

$$d^{seq} = \langle d^{\text{seq}}(1), ..., d^{\text{seq}}(T) \rangle, \text{ with } d^{\text{seq}}(i) \in \mathcal{V}. \tag{2.6}$$

For example, the phrase '*The dog chases a cat and a mouse!*', after stemming and elimination of stopwords, is a sequence d^{seq}=$\langle dog,\ chase,\ cat,\ mous \rangle$.

2.5.1 The Vector Space Model

Despite the fact that it is reasonable to seek for complex representations for text data, such as graphs [65–67], the *vector space model* (VSM) is the most widely-used representation where each document is represented by a vector of weights corresponding to text features. According to the typical VSM approach, the *bag of words* (BOW), a document is represented by a vector $d \in \mathbb{R}^V$, where each word term ν_i of the vocabulary is associated with a single vector dimension.

The feature weights can be binary, or, more often, computed by a frequency-based weighting function. The most popular weighting scheme is the *normalized* $tf{\times}idf$ that introduces the *inverse document frequency* as an external weight to enforce the terms that have discrimination power and appear in a small number of documents [68]. For the ν_i vocabulary term, the term frequency is defined using the indicator function $\mathrm{I}(\cdot)$ as tf=$\sum_{j=1}^{T} \mathrm{I}(d^{\text{seq}}(j) = \nu_i)$, and the idf_i=$\log(N/df_i)$ where N denotes the total number of documents and df_i denotes the term document frequency of term ν_i (see Sec. 2.4). Thus, the normalized $tf{\times}idf$ BOW vector is a mapping of the term sequence d^{seq} defined as follows:

$$\varphi_{\text{BOW}} : d^{\text{seq}} \to d = h \cdot (tf_1\ idf_1, ..., tf_V\ idf_V) \in \mathbb{R}^V, \tag{2.7}$$

$$h_{\text{L}_1} = \left[\sum_{i=1}^{|V|} tf_i \log \frac{N}{df_i} \right]^{-1} \text{and} \tag{2.8}$$

$$h_{\text{L}_2} = \left[\sum_{i=1}^{|V|} tf_i^2 \log^2 \frac{N}{df_i} \right]^{-1/2}, \tag{2.9}$$

where normalization is performed with respect to L_1-norm or L_2-norm using Eq. 2.8 (h=h_{L_1}) or Eq. 2.9 (h=h_{L_2}), respectively.

Vector normalization prevents a bias towards documents with longer term sequences and the two options have different geometrical properties. In the first case, we obtain a probability vector which is a point on the *V-dimensional simplex*. The second maps the data vector on the surface of the positive quadrant of the *V-dimensional hypersphere*. *Non-negative spherical data* is a special case of *directional data* that contain only positive feature weights and vector magnitude is not regarded critical for their analysis. The document collection can then be represented using the N document vectors as rows of the *document-term matrix* D, which is a $N \times V$ matrix whose rows and columns are indexed by the documents and the vocabulary terms, respectively.

The advantages of VSM is that it maps data into a well-defined multi-dimensional feature space and avoids the computationally expensive preprocessing required to build complex representation structures. We should note that the data structures that are used to achieve efficient processing on sparse high dimensional vectors are not simple (e.g. hash tables, or tree structures).

The main criticism against BOW is the assumed *term independence*, which ignores the term correlations in natural languages. However, BOW is only an instance of VSM and there is plenty of research on developing more efficient VSM variations (see Chapter 3). Such a simple but quite efficient method is the *generalized vector space model* (GVSM) [69] which represents a document in the similarity space, i.e. d'=$dD^\top$.

In a VS there are several alternatives to quantify the semantic (dis)similarity between document pairs. The *Minkowski* family of *metric functions* is defined by:

$$dist^{(mink)}(d_i,\, d_j) = \left[\sum_{q=1}^{V} |d_{iq} - d_{jq}|^q\right]^{1/q} \in [0, \infty], \tag{2.10}$$

and the four conditions satisfied by any metric function $dist(\cdot)$ are:

1. $dist(d_i, d_j) \geq 0$ *non-negativity*
2. $dist(d_i, d_j) = 0$ iff $d_i = d_j$ *coincidence*

3. $dist(d_i, d_j) = dist(d_j, d_i)$ *symmetry*

4. $dist(d_i, d_q) \leq dist(d_i, d_j) + dist(d_j, d_q)$ *triangle inequality.*

The obtained L_1-distance function for q=1 is known as *City-block distance* (among other names), while *Euclidean* is the L_2-distance derived for q=2 is one of the standard functions used for document datasets. For multi-dimensional spaces weighted Euclidean versions are applicable that have the form $\sqrt{(d_i - d_j)W(d_i - d_j)^\top}$. W is a $V \times V$ weight matrix which can be diagonal, containing a global weight for each feature, or a full matrix. *Mahalanobis* distance is of the latter case and uses the inverse covariance matrix, $W = \Sigma^{-1}$. It is not necessary for a distance function to satisfy all the above restrictive conditions in order to be applicable in a clustering procedure. Recently, *Bregman divergences* is a family of distance functions that have been considered in a general clustering framework [70]. These functions are not necessarily symmetric nor do they satisfy the triangle inequality property (4), while they have strong connection with various families of exponential distributions. It can be shown that Euclidean, Mahalanobis, and Kullback–Leibler divergence belong to this family of functions.

Among the various alternatives, *Cosine similarity* has shown to be an effective measure [41, 71] for document clustering. It computes the cosine of the angle θ between the two document vectors:

$$sim^{(cos)}(d_i,\, d_j) = \cos\big(\theta(d_i,\, d_j)\big) = \frac{d_i^\top}{\|d_i\|_2} \cdot \frac{d_j}{\|d_j\|_2} \in [0, 1]. \tag{2.11}$$

Unit similarity value implies the two documents are described by identical distributions of term frequencies. In practice, all document vectors are normalized in the preparation step, thus cosine similarity computation reduces to dot-product $d_i^\top d_j$ computation.

This latter measure, especially when applied on non-negative spherical data, determines the same K-nearest neighbors (KNN) ranking for a reference object with the ranking determined by Euclidean distance. This means that it is straightforward to use efficient KNN search methods which are designed based on Euclidean distance measure. Moreover, in contrast to Cosine, Euclidean is a metric where triangular inequality holds, and this

property enables faster KNN search [72, 73].

Other popular choices for text are *Tanimoto*, *extended Jaccard*, *Dice coefficient*, and *Simple matching coefficient*, all set-theoretic functions and widely-used for *binary feature vectors* (BFV) (note: the second and third are quite similar to Cosine since they are based on dot-product computation). The idea is to measure some kind of *information intersection* (overlap) and it can also be extended to arbitrary non-negative weighted vectors. BFV is useful for representing very small segments of text such as search queries, or a set of descriptive tags, where the terms frequencies are not important. Moreover, in very large-scale IR systems, BFV representation along with hashing techniques enable the efficient approximation of pairwise object similarities. Such an approach is applied in [74] for approximate KNN search. Small texts, in general, is a special case of texts where interesting problems arise concerning their representation and similarity calculations [75].

Similarity and dissimilarity are conceptually complementary to each other. A distance measure $dist(\cdot) \in [0,1]$ can be converted to the corresponding similarity measure by $sim(\cdot)=1-dist(\cdot) \in [0,1]$, or using various other simple calculations. When $dist(\cdot)$ is not bounded, e.g. $dist^{(eucl)} \in [0,\infty]$, then a monotonically decreasing function can be used to convert it in to a similarity value in [0,1]. For this purpose, a *kernel function*, such as *Gaussian* or *Laplacian*, could also be employed.

2.6 Clustering

2.6.1 Algorithms

Clustering using k-means family of methods

The k-means procedure is a generic clustering approach that assumes a prototype to represent each cluster and an objective function Φ(C) that evaluates the quality of a partition C, which is defined as the collection of sets c_j containing the objects assigned to the j-th cluster. In order to solve a problem with k clusters, the k prototypes are usually

initialized by randomly selecting k objects as cluster centroids (*Forgy's approach*) and then the algorithm iterates to optimize an objective function (called clustering error):

1. *Reassignment step*: each object is assigned to the cluster whose prototype is nearest to the object.
2. *Prototype batch update step*: given the assignment of objects to clusters, each cluster prototype is updated in a way that optimizes the objective function.

The k-means algorithm minimizes the sum of squared Euclidean distances between the objects of the clusters and the centroid prototypes Eq. 2.12, where the centroids are computed as the arithmetic mean $\mu_j = (1/n_j)\sum_{d_i \in c_j} d_i$ of the n_j objects of that cluster:

$$\Phi_{\text{SSE}}(\text{C}) = \sum_{j=1}^{k} \sum_{d_i \in c_j} \left\| \mu_j - d_i \right\|_2^2. \tag{2.12}$$

It converges to a local minimum of $\Phi_{\text{SSE}}(\text{C})$ and the quality of the solution depends strongly on the initial conditions. Its time complexity is $O(tNV)$, where t is the number of iterations until convergence. The form of cluster prototypes constitutes a choice that also affects the solution quality. *k-medoids* is a robust method that represents a cluster with the *medoid* object defined as the object that has the maximum average similarity to the objects of its cluster:

$$m_j = \arg\max_{d_i \in c_j} \left\{ \frac{1}{n_j} \sum_{d_q \in c_j} sim(d_i, d_q) \right\}. \tag{2.13}$$

In k-medoids, the medoid prototypes are used in Eq. 2.12. Note that in Euclidean space there is the disadvantage in complexity $O(n_j^2)$ required to determine the medoid of a cluster.

Spherical k-means (*spk-means*) is a variant of k-means that utilizes the Cosine similarity for the data vectors normalized with respect to L_2-norm. The maximized objective function is the *clustering Cohesion* (COH). The optimal prototype for a cluster is its *normalized centroid* $u_j = s_j / \left\| s_j \right\|_2$, where $s_j = \sum_{d_i \in c_j} d_i$, and the overall clustering Cohesion

of a partition C is given by:

$$\Phi_{\mathrm{COH}}(\mathrm{C}) = \sum_{j=1}^{k} \sum_{d_i \in c_j} u_j^\top d_i = \sum_{j=1}^{k} u_j^\top s_j = \sum_{j=1}^{k} \frac{s_j^\top s_j}{\|s_j\|_2} = \sum_{j=1}^{k} \|s_j\|_2 \,. \tag{2.14}$$

A lot of research effort has been focused on the careful initialization of this family of algorithms, due to its importance for the final clustering quality [76–80]. Among the typical object-based seeding techniques is the deterministic *Kaufman heuristic* (or *k-farthest heuristic*) [81] that tries to spread the initial centroids away from each other. It selects the most centrally located object as the first centroid and each additional centroid is determined to be the object farthest from the objects-centroids already selected. *k-means++* [78], on the other hand, introduces stochasticity: it starts with the uniform random selection of one object as the first centroid, then each additional centroid is initialized using a weighted probability distribution. Specifically, the probability for a candidate object to be selected as a new centroid is proportional to the squared distance between the object and its nearest centroid previously selected. In [78] it is shown that this initialization guarantees an O(log$k$) approximation to the optimal k-partition. However, all the above initialization methods select objects as seeds and this may not be efficient in the text feature space, since a document usually contains a very small percentage of the vocabulary terms.

Incremental clustering is a strategy that introduces one cluster each time in an already formed solution of lower order. In other words, each clustering k-solution is exploited to initialize the prototypes of the k+1 clustering problem. The advantage is that it provides a way to search for a good initialization and avoid the naive random restarts that is inefficient especially when the number of clusters is large. On the other hand, the computational burden increases and it is important to use efficient techniques to reduce the search space. Popular incremental methods are the *bisecting k-means* [82, 83] and *global k-means* [84]. The former, at each incremental step, selects a cluster according to an inhomogeneity criterion and then uses 2-means to split that cluster in two parts. The latter approach, along with the k already computed prototypes, it considers each of the

N objects of the dataset as the initial prototype of the k+1 cluster. The best clustering produced from these initializations is the resulting k+1-partition. Incremental clustering is further studied in Sec. 7.

Other clustering methods

Spectral clustering [85] is based on spectral analysis of the similarity matrix of the dataset. The basic idea is to project the data in the subspace spanned by the k largest eigenvectors of the Laplacian matrix L, which is computed from the similarity matrix $A^{(N\times N)}$ of pairwise document similarities. The similarity matrix A is computed using the cosine similarity measure. The Laplacian matrix is computed as $L{=}D^{-1/2}AD^{-1/2}$, where D is a diagonal matrix with $D_{ii}{=}\sum_{j=1}^{N}A_{ij}$ the sum of i-th row of similarities. To solve for k clusters, the algorithm proceeds with the construction of a matrix $X^{(N\times k)}{=}\{x_i : i{=}1,\ldots,k\}$ whose columns correspond to the k largest eigenvectors of L. X is then normalized so that each row has unit length in Euclidean space, let $Z^{(N\times k)}$ be the obtained normalized matrix. Finally, the clustering procedure takes place in the embedding space, i.e. the rows of Z are clustered using the standard k-means algorithm, assuming that i-th row of Z represents the i-th document.

Special algorithms have also been developed to deal with HDS feature spaces. The aim is to find clusters in subspaces of data instead of the entire feature space and it is referred to as *subspace clustering* Its key characteristic is the simultaneous determination of the object membership to clusters and the subspace of each cluster. Surveys on subspace clustering in high dimensional spaces can be found in [37, 40]. Further discussion on this category of methods is provided in Sec. 4.2.2.

To provide a more complete report on the state-of-the-art of the clustering literature applied in HDS data spaces, we should mention the effective generative probabilistic topic models, such *probabilistic latent semantic indexing* [86], *latent Dirichlet allocation* [24, 87, 88], *Dirichlet compound multinomial* [89, 90], and *mixture of von Mises-Fisher distributions* [91]. Note that a topic does not actually coincide with a cluster, thus the probabilistic topic-modeling can be viewed as a representation method the output of which

can be partitioned using a clustering algorithm.

Clustering refinement

Clustering refinement is the *post-processing* procedure aiming to improve the clusters produced by a clustering algorithm[2]. This is applicable to the flat, incremental, or hierarchical approaches. The refinement algorithm may be a specialized algorithm that proceeds with small changes in the clusters, such as single object reassignment [92] or swapping the cluster memberships for pairs of objects [93]. It is also a practical choice to refine the produced clusters using a clustering method of different characteristics to the initial one. For instance, it has been proposed to initialize k-means using agglomerative clustering [42], genetic algorithms [94], [95], and simulated annealing [96], among others. An alternative approach is the *hybridized centroid-medoid heuristic* [97] that applies a small number of k-means iterations and tries to replace a centroid with a medoid belonging in a set of candidate medoids precomputed offline.

2.6.2 Performance evaluation

Cluster validation is the procedure that evaluates the quality of the obtained clustering results. One may realize that, since there are various different definitions about which is an interesting cluster structure to search for, the objective evaluation of a solution is not an easy problem [98]. All evaluation measures exhibit some bias towards their underlying assumptions. We should note that the number of clusters is one of the most important factors that affect an evaluation and it is generally difficult to compare clusterings with different number of groups.

The most straightforward evaluation approach is the *external validation*, where supervised information is used to determine whether the result resembles with human perception for the problem. The required information is a labeled dataset that describes an intuitively correct solution to the partitioning problem (the so-called *ground truth*).

[2]Note that in literature the term '*refinement*' is also used to describe the iterative optimization of an objective function

Then, several measures could be utilized to assess the agreement between the labels and the clustering result. Most of them compute the degree to which objects with the same cluster label are grouped together, while objects with different cluster labels are assigned to different groups. There are numerous external evaluation measures. Next we describe some popular ones that have been used in the experiments of the following chapters.

We define the following notation: C the partition of data objects into k clusters (clustering solution) $c_1,\ldots,c_k$, $\mathrm{C}^{(L)}$ the grouping based on ground truth document labels $c_1^{(L)},\ldots,c_k^{(L)}$ (true classes), N the number of documents in a dataset, N_i the size of $c_i^{(L)}$, n_j the size of c_j, and n_{ij} the number of documents belonging to $c_i^{(L)}$ that are clustered in c_j. Let us further denote the probabilities $p(c_j)$=n_j/N, $p(c_i^{(L)})$=$n_i^{(L)}/N$, and $p(c_i^{(L)}, c_j)$=n_{ij}/N. The [0,1]-*normalized mutual information* (NMI) measure, as used in [99], is computed by normalizing the mutual information between C and $\mathrm{C}^{(L)}$ wrt the maximum entropy of clusters $H(C^{(L)})$, or classes $H(C)$:

$$NMI(\mathrm{C}^{(L)}, \mathrm{C}) = \frac{\sum_{\substack{c_i \in \mathrm{C}^{(L)}, \\ c_j \in \mathrm{C}}} p(c_i, c_j^{(L)}) \log_2 \dfrac{p(c_i^{(L)}, c_j)}{p(c_i^{(L)}) p(c_j)}}{\max\left\{H(\mathrm{C}^{(L)}),\ H(\mathrm{C})\right\}} \in [0, 1]. \tag{2.15}$$

When C and $\mathrm{C}^{(L)}$ are independent, the value of NMI equals to zero, while it equals to one when the two partitions contain identical clusters.

The F_1-*measure*, or simply F, is the harmonic mean of the *precision* and *recall* measures of the solution. Let the $precision_{ij}$ and $recall_{ij}$ for each (class i, cluster j) pair, then the respective $F(i,j)$ is given by:

$$f(i,j) = 2\frac{precision_{ij} \cdot recall_{ij}}{precision_{ij} + recall_{ij}}, \tag{2.16}$$

and the final F measure is obtained by the weighted average:

$$F(\mathrm{C}^{(L)}, \mathrm{C}) = \sum_{i=1}^{k} p(c_j) \max\{f(i,j)\}. \tag{2.17}$$

Higher values of F indicate better clustering solutions.

The *Purity* of a cluster can be interpreted as the classification accuracy by assuming that all objects of a cluster are assigned to its dominant class. The clustering Purity is the weighted average of cluster-wise purity:

$$Purity(\mathrm{C}^{(L)}, \mathrm{C}) = \frac{1}{N}\sum_{j=1}^{k} \max_{i=1,\ldots k} \{n_{ij}\} \in [0, 1]. \tag{2.18}$$

In order to compare the ground truth labeling and the grouping produced by clustering, we also utilized the *Variation of Information* (VI) metric [100] and the *Adjusted Rand Index* (ARI) [101]. Better clustering is indicated by lower values of VI and higher for ARI. Note that these measures can be extended to cover the case where the number of data classes is not the same as the number of clusters.

As discussed in Chapter 1, one of the major contributions of clustering in data analysis is the fact that it can discover the cluster structure in data that human might not be able to evaluate by themselves (inability to provide labels). But even if we are aware of what we are looking for, it generally preferable to use unsupervised, called *internal evaluation measures*. These measures compute quantities that involve the relations between data objects themselves. Intuitively, a good clustering solution should present high *separation* and high *compactness*. The first, implies that the clusters should be well-separated in the space, and the second that the objects of each cluster should be close to each other. Thus, given the pairwise (dis)similarities and the discovered cluster structure, we may compute quantities that express these fundamental concepts.

Nevertheless, the clustering methods that are based on the optimization of an objective function provide this value of the objective $\Phi(\mathrm{C})$ that can be used as a clustering quality criterion. The limitation is that, in this way, it is not possible to compare clusterings with different objective functions, and of course in cases of solutions with different number of clusters. However, if all the compared algorithms use the same assumptions under which they seek for a clustering solution, then direct comparison is possible. For example, we can compare clustering results from k-means, bisecting k-means, and global k-means (with

same k value), since all of them use in fact the same objective function.

Chapter 3

Improving Document Clustering Using Global Term Context Vectors

3.1 Introduction

In this chapter, we present the *global term context vector model* (GTCVM) document representation model [43]. It is an entirely corpus-based extension to the traditional vector space model and incorporates *contextual information* for each vocabulary term (feature dimension). First, the *local context* for each term occurrence in the term sequences of

documents is captured and represented in vector space by exploiting the idea of the *locally weighted bag of words* (LoWBOW) [102]. Then all the local contexts of a term are combined to form its *global context vector*. Global context vectors constitute a *semantic matrix* which efficiently maps the traditional VSM document vectors onto a semantically richer feature space of same dimensionality to the original.

As indicated by our experimental study, in the new space, superior clustering solutions are achieved using well-known clustering algorithms such as the spherical k-means [41] or spectral clustering [85].

3.2 Extensions to VSM

In Sec. 2.5.1 we presented the basics regarding the vector space model (VSM) where each document is represented by a vector of weights corresponding to text features [1, 2]. Many variations of VSM have been proposed that differ in what they consider as a feature, or '*term*' [103]. The the most common approach is to consider different words as distinct terms, which is the widely-known the bag of words (BOW) model. This model, despite having a series of advantages, such as generality and simplicity, it cannot model efficiently the rich semantic content of text. An extension is the *bag of phrases model* (BOP) [104] that extracts a set of informative phrases or *word n-grams* (n consecutive words). Especially for noisy document collections, e.g. containing many spelling errors, or collections whose language is not known in advance, it is often better to use VSM to model the distribution of *character n-grams* in documents. In this chapter, we consider word features and we refer to them as *terms*, however, the described procedures can be directly extended to more complex text features.

The disadvantage of considering multiword features, as BOP does, or generally combinations of multiple low-level features, is the fact that as phrases become longer they clearly obtain superior semantic value but, at the same time, they become statistically inferior with respect to single-word representations [105]. A category of methods developed

aiming on tackling this difficulty recognize the frequent *wordsets* (unordered itemsets) in a document collection [106–108], while the method proposed in [109] exploits the frequent word subsequences (ordered) that are stored in a *generalized suffix tree* (GST) for each document.

Modern variations of VSM are used to tackle the difficulties occurring due to high dimensional and sparse (HDS) feature spaces, by projecting the document vectors onto a new feature space called *concept space*. Each *concept* is represented as a *concept vector* of relations between the concept and the vocabulary terms. Generally, this approach of document mapping can be expressed as:

$$\varphi_{\text{VSM}} : d \to d' = Sd \in \mathbb{R}^{V'},\ V' \leq V, \tag{3.1}$$

where the $V' \times V$ matrix S stores the concept vectors as rows. This projection matrix is also known as *semantic matrix*. The Cosine similarity between two normalized document images in the concept space can be computed as a dot-product:

$$sim_{\text{sem}}^{(\cos)}(d'_i, d'_j) = (\overline{Sd_i})^\top(\overline{Sd_j}) = (h_i^S Sd_i)^\top(h_j^S Sd_j) = h_i^S h_j^S (d_i^\top S^\top S d_j), \tag{3.2}$$

where the scalar normalization coefficient for each document is $h_i^S = 1/\|Sd_i\|_2$. The similarity defined in Eq. 3.2 can be interpreted in two ways:

i) as a dot product of the document images $(\overline{Sd_i})^\top(\overline{Sd_j})$ that both belong to the new space $\mathbb{R}^{V'}$ or, alternatively,

ii) as a composite measure that takes into account the pairwise correlations between the original features expressed by the matrix $S^\top S$.

There is a variety of methods proposing alternative ways to define the semantic matrix though many of them are based on the above linear mapping. The widely-used *latent semantic indexing* (LSI) [23] projects the document vectors onto a space spanned by the eigenvectors corresponding to the V' largest eigenvalues of the matrix $D^\top D$, where D is the $N \times V$ document-term matrix. The eigenvectors are extracted by the means of

singular value decomposition (SVD) on matrix $D^\top$ and they capture the latent semantic information of the feature space. In this case, each eigenvector is a different concept vector and V' is a user parameter much smaller than V, while there is also a considerable computational cost to perform the SVD. In *concept indexing* [110], the concept vectors are the centroids of a V'-partition obtained by applying document clustering. In [111], statistical information such as the covariance matrix is combined with traditional mapping approaches into latent space (e.g. LSI, PCA) to compose a hybrid vector mapping.

A computationally simpler alternative that utilizes the document-term matrix D as a semantic matrix is the *generalized vector space model* (GVSM) [69], i.e. $S_{\text{GVSM}}{=}D$ and the image of a document is given by $d'{=}Dd$. By examining the product $Dd \in \mathbb{R}^{N\times 1}$, we can conclude that a GVSM projected document vector d' has lower dimensionality if $N{\leq}V$. Moreover, if both d and D are properly normalized, then image vector d' consists of the N Cosine similarities between the document vector d and the rest of the $N{-}1$ documents in the collection. This observation implies that the GVSM works in the *document similarity space* by considering each document as a different concept. On the other hand, the respective product $S_{\text{GVSM}}^\top S_{\text{GVSM}}{=}D^\top D$ (used in Eq. 3.2) is a $V{\times}V$ *term similarity matrix* whose r-th row has the dot-product similarities between term ν_r and the rest of the $V{-}1$ of vocabulary terms. Note that terms become more similar as their corresponding normalized frequency distributions into the N documents are more alike. Based on the GVSM model, it has been proposed to build *local semantic matrices* for each cluster during document clustering [112].

A rather different approach proposed in [113] for information retrieval is the *context vector model* (CVM) where, instead of a few concise concept vectors, it computes the context in which each of the V vocabulary terms appears in the dataset, called *term context vector* (*tcv*). This model computes a $V{\times}V$ matrix S_{CVM} containing the term context vectors as rows. Each tcv_i vector aims to capture the V pairwise similarities of term ν_i to the rest of the vocabulary terms. Such similarity is computed using a *co-occurrence frequency measure*. Each matrix element $[S_{\text{GVSM}}]_{ij}$ stores the similarity between

terms ν_i and ν_j computed as:

$$[S_{\text{CVM}}]_{ij} = \begin{cases} 1 & , i = j \\ \dfrac{\sum_{r=1}^{N} tf_{ri}\, tf_{rj}}{\sum_{r=1}^{N}(tf_{ri} \sum_{q=1,\ q\neq i}^{V} tf_{rq})} & , i \neq j. \end{cases} \tag{3.3}$$

Note that this measure is not symmetric, generally $[S_{\text{CVM}}]_{ij} \neq [S_{\text{CVM}}]_{ji}$, due to the denominator that normalizes the pairwise similarity to [0, 1] with respect to the '*total amount*' of similarity between term ν_i and the other vocabulary terms. The rows of matrix S_{CVM} can be normalized with respect to the Euclidean norm and each document image is then computed as the centroid of the normalized context vectors of all terms appearing in that document:

$$\varphi_{\text{CVM}} : d \rightarrow d' = \sum_{i=1}^{V} tf_i\, tcv_i, \tag{3.4}$$

where tf_i is the frequency of term ν_i. The motivation for using term context vectors is to capture the semantic content of a document based on the co-occurrence frequency of terms in the same document, averaged over the whole corpus. The CVM representation is less sparse than BOW. Moreover, weights such as *idf* can be incorporated to the transformed document vectors computed using Eq. 3.4. In [113] several more complicated weighting alternatives have been tested in the context of information retrieval that in our text document clustering experiments did not perform better than the standard *idf* weights.

In a higher semantic level than term co-occurrences, additional information for vocabulary terms provided by ontologies has also been exploited to compute the term similarities and to construct a proper semantic matrix. *WordNet* [47] and *Wikipedia* [48] have been used for this purpose in [49, 50] and [51], respectively.

3.3 Discussion on VSM variations

Summarizing the properties of the above mentioned vector-based document representations, in the traditional BOW approach, the dimensions of the term feature space are

considered to be independent to each other. Such an assumption is very simplistic, since there exist semantic relations among terms that are ignored. The VSM-extensions aim to achieve *semantic smoothing*, a process that redistributes the term weights of a vector model, or map data in a new feature space, by taking into account the correlations between terms. For instance, if the term '*child*' appears in a document, then it could be assumed that the term '*kid*' is also related to the specific document, or even terms like '*boy*', '*girl*', '*toy*'. The resulting representation model is also a VSM, but the document vectors become less sparse and the independence of features is mitigated in an indirect way. The smoothing is usually achieved by a linear mapping of data vectors to a new feature space using a semantic matrix S. It is convenient to think that the new document vector $d'=Sd$ contains the dot product similarities between the original BOW vector d and the rows of the semantic matrix S.

A basic difference between the various semantic smoothing methods is related to the dimension of the new feature space which is determined by the number V' of row vectors of matrix S. In case their number is less than the size V of the vocabulary, such vectors are called as *concept vectors* and are usually produced using the LSI method. Each concept vector has a distribution of weights associated to the V original terms that define their contribution of to the corresponding *concept*. Of course the resulting representation of the smoothed vector d' is less interpretable than the original and there is always a problem of determining the proper number of concept vectors.

An alternative approach for semantic smoothing assumes that each row vector of matrix S is associated with one vocabulary term. Unlike a concept vector that describes abstract semantics of higher level, here, the elements of each vector describe the relation of this term to the other terms. Those relations constitute the so called *term context*, thus the respective vector is called *term context vector*. Each element of the mapped vector d' will contain the dot product similarity between document d and the corresponding term context vector, i.e. for each term v_i the element d'_i provides the degree to which the original document d contains the term v_i and its context, instead of just computing its frequency as happens in the BOW representation. Note also that in BOW representation,

a dot product would give zero similarity for two documents that do not have common terms. On the contrary, the dot product between a document vector and a term context vector of a term v_i that does not appear in that document may give a non-zero similarity. This happens if the document contains at least one term v_j with non-zero weight in the context of term v_i. For this reason, the smoothed representation d' is usually less sparse that d and retains their interpretability of dimensions. Moreover, concept-based methods may be applied on the new representations.

The motivation of our work is to establish the importance of *term context vectors* and to define an efficient way to compute them. The CVM method considers that the term context is computed based on term co-occurrence frequency at the document-level. It does not take into account the sequential nature of text and thus ignore the local distance of terms when computing term context. On the other hand, the proposed GTCVM extends the previous approach by considering term context at three levels:

i) it uses the notion of *local term context vector* ($ltcv$) to model the context around the location in the text sequence where a term appears. These vectors are computed using a local smoothing kernel as suggested in the $LoWBOW$ approach [102] which is described in the next section. The kernel *takes into account the distance* in which other terms appear around the sequence location under consideration.

ii) it computes the *document term context vector* ($dtcv$) for each term that summarizes the term context at the document-level.

iii) it computes the final *global term context vector* ($gtcv$) for each term representing the overall term context at corpus-level. The $gtcv$ vectors constitute the rows of the semantic matrix S. Thus the intuition behind GTCVM approach is to capture the local term context from term sequences and then to construct a representation for global term context by averaging $ltcvs$ at the document and corpus-level.

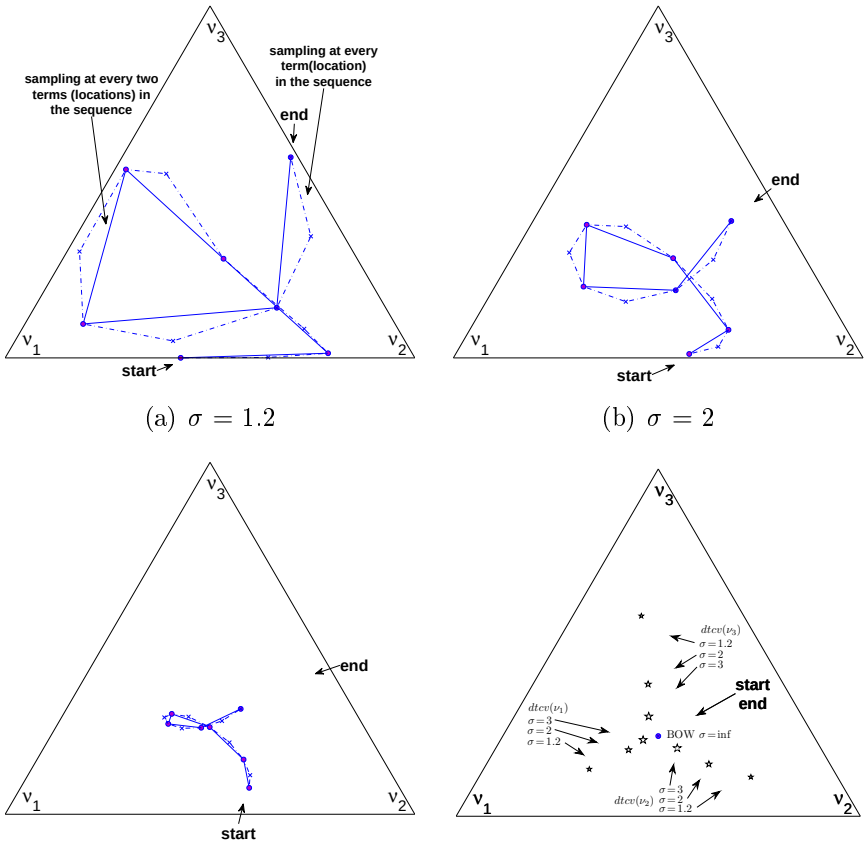
3.4 Utilizing local contextual information

A text document can be considered as a finite term sequence of its T consecutive terms denoted as $d^{\text{seq}}=\langle d^{\text{seq}}(1), ..., d^{\text{seq}}(T)\rangle$ but, except for bag of phrases (BOP), so far in this chapter the previously mentioned VSM-extensions ignore this property. A category of methods have been proposed aiming to capture local information directly from the term sequence of a document. The representation proposed in [114], first considers a segmentation of the sequence that is done by dragging a window of n terms along the sequence and computing the local BOW vectors for each of the overlapping segments. All these local BOW vectors constitute the document representation called *local word bag* (LWB). To compute the similarity between a pair of documents, the authors introduce a variant of the *vg-pyramid matching kernel* [115] that maps the two sets of local BOW vectors to a multi-resolution histogram, and then computes a weighted histogram intersection.

Another approach for text representation presented in [102], is the *locally weighted bag of words* (LoWBOW) that preserves local contextual information of text documents by the effective modeling of the text sequential structure. At first, a number of L equally distant locations are defined in the term sequence. Each sequence location ℓ_i, i=1, ..., L, is then associated with a local histogram which is a point in the multinomial simplex $\mathbb{P}_{V-1}$, where V is the number of vocabulary terms. More specifically, for $(V-1)\geq 0$, the $\mathbb{P}_{V-1}$ space is the $(V-1)$-dimensional subset of $\mathbb{R}^V$ that contains all probability vectors (histograms) over V objects (for a discussion on the multinomial simplex see the Appendix of [102]):

$$\mathbb{P}_{V-1} = \left\{ H \in \mathbb{R}^V : \; H_i \geq 0, \; \forall i = 1, ..., V \text{ and } \sum_{i=1}^{V} H_i = 1 \right\}. \tag{3.5}$$

Contrary to LWB, in LoWBOW the local histogram is computed using a *smoothing kernel* to weight the contribution of terms appearing around the referenced location in the term sequence, and to assign more importance to closely neighboring terms. Denoting as $H_{\delta(d^{\text{seq}}(t))}$ the *trivial term histogram* of V terms whose probability mass is concentrated



(a) $\sigma = 1.2$ (b) $\sigma = 2$

(c) $\sigma = 3$ (d)

Figure 3.1: A toy example where the sequence $\langle \nu_1,\ \nu_2,\ \nu_2,\ \nu_2,\ \nu_1,\ \nu_3,\ \nu_3,\ \nu_1,\ \nu_1,\ \nu_1,\ \nu_2,\ \nu_2,\ \nu_3 \rangle$ is considered that uses three different terms ν_1, ν_2, ν_3 (vocabulary length: V=3). The subfigures present LoWBOW curves in the $(V-1)$-dimensional simplex for increasing values of the parameter σ that induce more smoothing to the curve. Each point of the curve corresponds to a local histogram computed at a sequence location. The more a term affects the local context at a location in the sequence, the more the curve point (the *lowbow* histogram related to that location) moves towards the respective corner of the simplex. For σ=0 local histograms correspond to simplex corners, thus the curve moves from corner to corner of the simplex. Two different sampling rates for LoWBOW representation are illustrated: sampling at every term location in the sequence (dashed line) which is the our strategy to collect contextual information for each term, and sampling every two terms (solid line). d) For σ=∞, the LoWBOW curve reduces to a single point that coincides with the BOW histogram of the sequence. In (d) we present as 'stars' the average *ltcv* histograms for each term (*dtcv* histograms) for the three different values of σ and α=0.6 for all terms. As the value of σ increases, the *dtcv* histograms of all terms become more similar tending to coincide with the BOW representation.

only at the term that occurs at the location t in d^{seq}:

$$\left[H_{\delta(d^{\text{seq}}(t))}\right]_i = \begin{cases} 1, & \nu_i = d^{\text{seq}}(t) \\ 0, & \nu_i \neq d^{\text{seq}}(t) \end{cases}, \ i = 1, ..., V, \tag{3.6}$$

then the locally smoothed histogram at a location ℓ in the d^{seq} term sequence is computed as in [102]:

$$lowbow(d^{\text{seq}}, \ell) = \sum_{t=1}^{T} H_{\delta(d^{\text{seq}}(t))} K_{\ell,\sigma}(t), \tag{3.7}$$

where T is the length of d^{seq}. $K_{\ell,\sigma}(t)$ denotes the weight for location t in sequence given by a discrete Gaussian weighting kernel function of mean value ℓ and standard deviation σ. Specifically, the weighting function is a Gaussian probability density function restricted in $[1, T]$ and renormalized so that $\sum_{t=1}^{T} K_{\ell,\sigma}(t) = 1$. It is easy to verify that the result of the histogram smoothing of Eq. 3.7 is also a histogram.

It must be noted that for σ=0 the *lowbow* histogram (Eq. 3.7) coincides with the trivial histogram $H_{\delta(d^{\text{seq}}(\ell))}$, where all the probability mass is concentrated at the term at location ℓ. As σ grows, part of the probability mass is transfered to the terms occurring near location ℓ. In this way, the *lowbow* histogram at location ℓ is enriched with information about the terms occurring in the neighborhood of ℓ. The smoothing parameter σ adjusts the 'locality' of term semantics that is taken into account by the model. Thus, instead of mining unordered local vectors as in [114], the LoWBOW approach embeds the term sequence of a document in the $\mathbb{P}_{V-1}$ simplex. The sequence of the L locally smoothed histograms (denoted as *lowbow histograms*) form a curve in the (V-1)-dimensional simplex (denoted as *LoWBOW curve*). Fig. 3.1 illustrates the LoWBOW curves generated for a toy example and describes the role of parameter σ. In this figure we aim to illustrate i) the LoWBOW curve representation, i.e. the curve that corresponds to a sequence of histograms (local context vectors), where each local context vector is computed at a specific location of the sequence and corresponds to a point in the (V-1)-dimensional simplex; ii) the impact of the smoothing coefficient σ on the computed local context vectors. This figure illustrates that the increase of smoothing makes the *lowbow* histograms (points of the curve) more similar. This can also be verified by observing that as smoothing increases, the curve becomes more concentrated around a central location of the simplex. For σ=∞ all histograms become similar to the BOW representation and the curve reduces to a single point. On the contrary, for σ=0 the histograms correspond to simplex corners.

A similarity measure between LoWBOW curves has been proposed in [102] that assumes a sequential correspondence between two documents and computes the sum of the similarities between the L pairs of LoWBOW histograms. Obviously, it is expected for this similarity measure to underestimate the thematic similarity between documents that follow different order in the presentation of similar semantic content.

3.5 A semantic matrix based on global term context vectors

In this section we present the global term context vector model (GTCVM) approach for capturing the semantics of the original term feature space of a document collection. The method computes the contextual information of each vocabulary term, that is subsequently utilized in order to create a semantic matrix. In analogy with CVM, our approach reduces data sparsity but not dimensionality. The interpretability of the derived vector dimensions remains as strong as in the BOW model as the value of each dimension of the mapped vector corresponds to one vocabulary term. Methods that reduce data dimensionality could also be applied on the new representations at a subsequent phase. Compared to CVM, GTCVM generalizes the way the term context is computed by taking into account the distance between terms in the term sequence of each document. This is achieved by exploiting the idea of LoWBOW to describe the local contextual information at a certain location in a term sequence. It must be noted that our method borrows from the LoWBOW approach only way the local histogram is computed at each location of the term sequence and does not make use of the LoWBOW curve representation.

More specifically, we define the *local term context vector* ($ltcv$) as a histogram associated with the exact occurrence of term $d^{\text{seq}}(\ell)$ at location ℓ in a sequence d^{seq}. Hence, one $ltcv$ vector is computed at every location in the term sequence, i.e. ℓ=1, ...,T. Note that GTCVM does not preserve any curve representation. This means that we are not interested in the temporal order of the local term context vectors. The $ltcv(d^{\text{seq}}, \ell)$ is a modified $lowbow(d^{\text{seq}}, \ell)$ probability vector that represents contextual information around

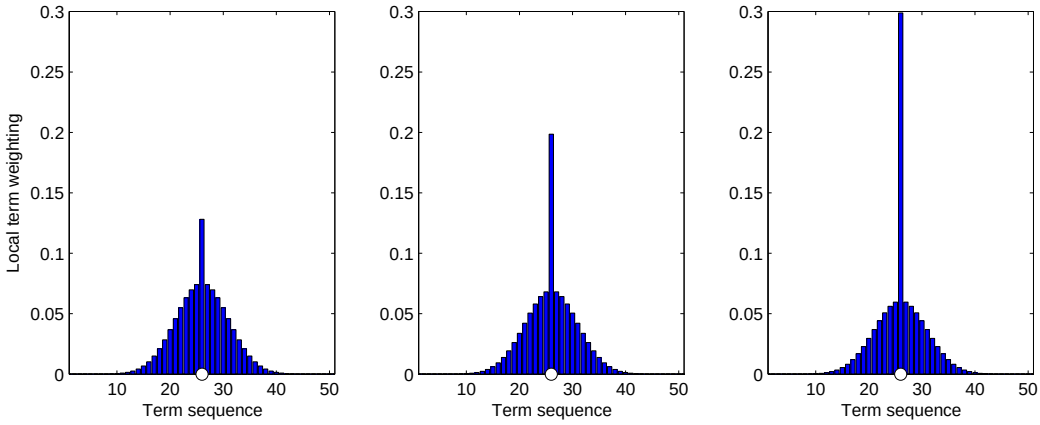


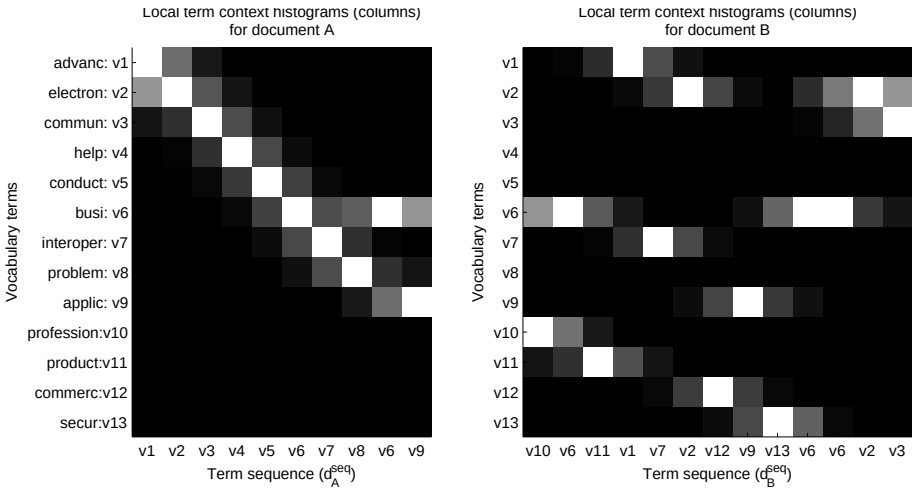
Figure 3.2: Various weight distributions for the neighboring terms around a reference term occurring in the middle of a term sequence of length 50. The distributions are obtained by varying the value of parameter α in Eq. 3.8. This distribution defines the contribution of each term to the context of the specific reference term. The scale value of the local kernel is set to σ=5, while self-weight α is set to 0.05 (left), 0.10 (middle), 0.2 (right).

location ℓ, while adjusting explicitly the *self-weight* $\alpha_{d^{\text{seq}}(\ell)}$ of the reference term appearing at location ℓ:

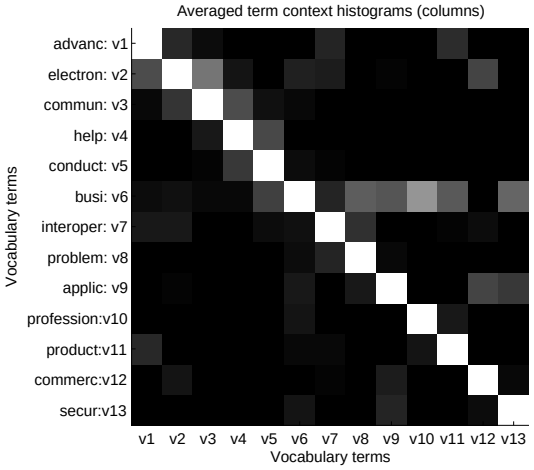
$$[ltcv(d^{\text{seq}},\ell)]_i = \begin{cases} \alpha_{d^{\text{seq}}(\ell)} & ,\nu_i = d^{\text{seq}}(\ell), \\ (1-\alpha_{d^{\text{seq}}(\ell)}) \cdot \frac{idf_i \cdot [lowbow(d^{\text{seq}},\ell)]_i}{\sum_{j=1,j\neq i}^{V} idf_j \cdot [lowbow(d^{\text{seq}},\ell)]_j} & ,\nu_i \neq d^{\text{seq}}(\ell). \end{cases} \tag{3.8}$$

The self-weight ($0 \leq \alpha_{d^{\text{seq}}(\ell)} \leq 1$) adjusts the relative importance between contextual information (computed using the *lowbow* histogram) and the self-representation of each term. Fig. 3.2 illustrates an example of how the value of parameter α affects the local term weighting around a reference term in a sequence. When the parameter σ of the Gaussian smoothing kernel is set to zero, or α=1, the $ltcv(d^{\text{seq}},\ell)$ reduces to a trivial histogram $H_{\delta(d^{(seq)}(\ell))}$ (see Eq. 3.6). The other extreme is the infinite σ value, where for small α values all the *ltcv* computed in a document d become similar to the tf histogram for that document.

The latter observation is the reason for considering an explicit self-weight in Eq. 3.8, because a flat smoothing kernel obtained for large σ value can make a *lowbow* vector to have improperly low self-weight for the reference term. For example, if a term appears



(a)



(b)

Term Sequence A

"The **advance**s in **electron**ic **commun**ications **help** in **conduct**ing **busi**ness without **interoper**ation **problem**s between **busi**ness

Term Sequence B

"Our **professiona**l **buisi**ness **product**s **advance** the **interoper**ability of **electron**ic **commerce** **applic**ations and **securi**ty of **busi**ness-to-**busi**ness **electron**ic **commun**ications."

	Document A			*Document B*		
	BOW	d'	differ.	BOW	d'	differ.
advanc	0.1	0.079	-0.021	0.077	0.077	0.000
electron	0.1	0.085	-0.015	0.154	0.107	-0.047
commun	0.1	0.085	-0.015	0.077	0.076	-0.001
help	0.1	0.087	-0.013		0.026	0.026
conduct	0.1	0.095	-0.005		0.033	0.033
busi	0.2	0.127	-0.073	0.231	0.140	-0.090
interoper	0.1	0.089	-0.011	0.077	0.073	-0.004
problem	0.1	0.098	-0.002		0.049	0.049
applic	0.1	0.087	-0.013	0.077	0.084	0.007
profession		0.049	0.049	0.077	0.094	0.017
product		0.041	0.041	0.077	0.083	0.006
commerc		0.030	0.030	0.077	0.072	-0.005
secur		0.047	0.047	0.077	0.088	0.011

(c)

Figure 3.3: An example of how *ltcv* histograms are used to summarize the overall context in which a term appears in the two term sequences of (c) using Eq. 3.12. a) The term sequences (x-axis) of documents A, B are presented and the corresponding *ltcv* are illustrated as grey-scaled *columns*. Those vectors are computed at every location in the sequence using a Gaussian smoothing kernel with σ=1 and α=0.6 for all terms. Brighter intensity at cell i, j indicates higher contribution of the term ν_i to the local context of the term appearing at location j in the sequence. b) The resulting transposed semantic matrix ($S^\top$), where the gray-scaled columns illustrate the global contextual information for each vocabulary term computed by averaging the respective local context histograms (Eq. 3.11). c) The two initial term sequences (the stem of each non-trivial term is emphasized). Assuming the same *idf* weight for each vocabulary term, the table presents the BOW vector, the transformed vector d' using Eq. 3.12 as well as the effect of semantic smoothing ($diff = BOW - d'$) on document vectors. The redistribution of term weights, that results by the proposed mapping, reveals is done in such a way that low frequency terms are gaining weight against the more frequent ones. Note also that the similarity between the two documents is 0.756 for the BOW model and 0.896 for the GTCVM.

once in a document, then the *lowbow* vector with $\sigma{=}\infty$ at that location would contain very low weight for that term. Generally, the value of α_ν determines how much the context vector of term ν should be dominated by the self-weight of term ν. In our method we set this parameter independently for each individual term as a function of its idf_ν component:

$$\alpha_\nu = \lambda + (1-\lambda)\cdot\left(1 - \frac{idf_\nu}{logN}\right), \quad \lambda \in [0,1], \tag{3.9}$$

where λ is a lower bound for all a_ν, ν=1,...,V (in our experiments we used λ=0.2). The rationale for the above equation is that for terms with high document frequency (i.e. low idf_ν), we assign high α_ν values that suppress the local context in the respective context vectors. In other words, the context is considered more important for terms that occur in fewer documents. In Fig. 3.3a, we present an example illustrating the *ltcv* vectors of two term sequences presented in Fig. 3.3c.

We further define the *document term context vector* (*dtcv*) as a probability vector that summarizes the context of a specific term at the document-level by averaging the *ltcv* histograms corresponding to the occurrences of this term in the document. More specifically, suppose that a term ν appears $no_{i,\nu} > 0$ times in the term sequence d_i^{seq} (i.e. in the i-th document) which is of length T_i. Then the *dtcv* of this term ν for document i is computed as:

$$dtcv(d_i,\, \nu) = \frac{1}{no_{\nu,i}} \sum_{j=1}^{no_{i,\nu}} ltcv(d_i^{\text{seq}},\, \ell_{i,\nu}(j)), \tag{3.10}$$

where $\ell_{i,\nu}(j)$ is an integer value in $[1, ..., T_i]$ denoting the location of the j-th occurrence of ν in d_i^{seq}.

Next, the *global term context vector* (*gtcv*), is defined for a vocabulary term ν so as to represent the overall contextual information for all appearances of ν in the corpus of all N term sequences (documents):

$$gtcv(\nu) = h_{gtcv(\nu)}\left(\sum_{i=1}^{N} tf_{i,\nu}\; dtcv(d_i^{\text{seq}},\, \nu)\right). \tag{3.11}$$

The coefficient $h_{\text{GTCVM}(\nu)}$ normalizes the vector $gtcv(\nu)$ with respect to the Euclidean norm,

and $tf_{i,\nu}$ is the frequency of the term ν in the i-th document. Thus, the $gtcv(\nu)$ of term ν is computed using a weighted average of the document context vectors $dtcv(d_i^{\text{seq}}, \nu)$ obtained for each document i in which term ν appears. Thus, in contrast to LoWBOW curve approach which focuses on the sequence of local histograms that describe the writing structure of a document, our method focuses on the extraction of the global semantic context of a term by averaging the local contextual information at all the corpus locations where this term appears.

Finally, the extracted global contextual information is used to construct the $V \times V$ semantic matrix S_{GTCVM} where each row ν is the $gtcv(\nu)$ vector of the corresponding vocabulary term ν. Fig. 3.1d provides an example of illustrating the $dtcv(d_i^{\text{seq}}, \nu)$ vectors for each document (the points denoted as 'stars'). Fig. 3.3b illustrates the final $gtcv$ vectors obtained by averaging the document-level contexts for each vocabulary term.

To map a document using the proposed GTCVM approach, we compute the vector d' where each element ν is Cosine similarity between the BOW representation d of the document and the global term context vector $gtcv(\nu)$:

$$\varphi_{\text{GTCVM}} : d \rightarrow d' = S_{\text{GTCVM}}\, d,\ d' \in \mathbb{R}^V. \tag{3.12}$$

Note that the transformed document vector d' is V-dimensional that retains the interpretability, since each dimension still corresponds to a unique vocabulary term. Moreover, if σ=0 and α>0, then $S_{gtcv}\, d$=d. Looking at Eq. 3.2, the product $S_{gtcv}^{\top} S_{gtcv}$ essentially computes a Term Similarity Matrix where the similarity between two terms is based on the distribution of term weights in their respective global term context vectors, i.e., on the similarity of their global context histograms. The table of Fig. 3.3c illustrates the effect of redistribution (compared to BOW) of the term weights (semantic smoothing) in the transformed document vectors achieved by the proposed mapping.

The procedure of representing the input documents using GTCVM takes place in the preprocessing phase. Let T_i the length of the i-th document and V_i its vocabulary. Let also V the size of the whole corpus vocabulary. Then the cost to compute one $ltcv$ vector at a

location of the term sequence using Eq. 3.8, and to add its V_i non-zero dimensions to the respective *dtcv*, is O(T_i+V_i). This is done T_i times and the final *dtcv* of each different term of the document is added to the respective the *gtcv* rows. Thus, using proper notation for the average length $\overline{T_i}$ and vocabulary length $\overline{V_i}$ of the documents in a corpus, the cost of constructing the semantic matrix can be expressed as O($N{\cdot}\overline{T_i}{\cdot}(\overline{T_i}{+}2{\cdot}\overline{V_i})$). However, since $\overline{V_i}{\leq}\overline{T_i}{\ll}V$, the overall computational cost of the GTCVM is determined by the O($N{\cdot}V^2$) cost of the matrix multiplication of the mapping of Eq. 3.12.

3.6 Clustering experiments

Our experimental setup was based on five different datasets: D_1-D_4 are subsets of the 20-Newsgroups[1], while D_5 is the mod-apte split [116] version of the Reuters-21578[2] benchmark document collection where the 10 classes with larger number of training examples are kept. The characteristics of these datasets are presented in Tab. 3.1. The preprocessing of datasets included the removal of all tags, headers and metadata from the documents, while applied word stemming and discarded terms appearing in less than five documents. It is worth mentioning how we preprocessed the term sequences of documents. We considered a *dummy term* that replaced in the sequences all the low-frequency terms that were discarded so as to maintain the relative distance between the terms that remained in each sequence. For similar reasons, two dummy terms were considered at the end of every sentence denoted by characters as (e.g. '.', '?', '!'). The dummy term is ignored when constructing the final data vectors.

For each dataset, we have considered several data mappings φ and after each mapping the *spherical k-means* (*spk-means*) [41] and *spectral clustering* (*spectral-c*) [85] algorithms (see Sec. 2.6.1) were applied to cluster the mapped documents vectors into the k predefined number of clusters corresponding to the different topics (classes) in a collection. Spk-means uses the Cosine similarity and maximizes the *Cohesion* of the clusters $C{=}\{c_1,...,c_k\}$

[1] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.tar.gz.
[2] http://www.daviddlewis.com/resources/testcollections/reuters21578/reuters21578.tar.gz

Table 3.1: Characteristics of text document collections. N denotes the number of documents, V is the size of the global vocabulary and $\overline{V_i}$ the average document vocabulary, *Balance* is the ratio of the smallest to the largest class and $\overline{T_i}$ is the average length of the term sequences of documents.

Name	**Topics**	**Classes**	N	*Balance*	V	$\overline{V_i}$	$\overline{T_i}$
D_1	20–NGs: graphics, windows.x, motor, baseball, space, mideast	6	2000	200/400	4343	48.8	110
D_2	20–NGs: atheism, autos, baseball, electronics, med, mac, motor, politics.misc	7	3500	500/500	6442	52.6	108
D_3	20–NGs: atheism, christian, guns, mideast	4	1600	400/400	4080	62	131
D_4	20–NGs: forsale, autos, baseball, motor, hockey	5	1250	250/250	4762	44.1	104
D_5	Reuters–21578: acq, corn, crude, earn, grain, interest, money-fx, ship, trade, wheat	10	9979	237/3964	5613	39.1	76

Table 3.2: NMI values of the clustering solution for VSM (BOW), GVSM, CVM and the proposed GTCVM (for several values of σ) document representations using the spk-means algorithm.

Method	σ	**D_1**			**D_2**			**D_3**			**D_4**			**D_5**		
		avg	**best**	**avg$_{10\%}$**	**avg**	**best**	**avg$_{10\%}$**	**avg**	**best**	**avg$_{10\%}$**	**avg**	**best**	**avg$_{10\%}$**	**avg**	**best**	**avg$_{10\%}$**
BOW	–	.722	.821	.594	.748	.829	.638	.537	.548	.379	.625	.779	.505	.552	.562	.535
GTCVM	1	.749	.854	.601	.767	.845	.638	.544	.564	.372	.667	.793	.515	.570	.578	.561
	2	.756	.871	.631	.765	.852	.657	.563	.574	.396	.670	.832	.539	.572	.580	.561
	5	.773	.881	.687	.777	.864	.662	.577	.602	.400	**.688**	**.851**	.539	.589	**.633**	.578
	10	**.777**	**.886**	.685	**.781**	**.873**	.672	**.590**	**.621**	.424	.684	.849	.540	**.590**	.630	.580
	30	.761	.879	.659	.776	.863	.653	.579	.590	.369	.683	.842	.518	.576	.612	.568
	inf	.760	.862	.631	.772	.862	.639	.574	.586	.366	.681	.840	.521	.576	.610	.566
GVSM	–	.752	.832	.611	.747	.822	.637	.556	.576	.419	.670	.827	.547	.575	.580	.573
CVM	–	.750	.841	.612	.754	.851	.659	.547	.604	.400	.672	.824	.541	.578	.581	.575

Table 3.3: F_1-measure values of the spk-means clustering solution for the different representation methods.

Method	σ	**D_1**			**D_2**			**D_3**			**D_4**			**D_5**		
		avg	**best**	**avg$_{10\%}$**	**avg**	**best**	**avg$_{10\%}$**	**avg**	**best**	**avg$_{10\%}$**	**avg**	**best**	**avg$_{10\%}$**	**avg**	**best**	**avg$_{10\%}$**
BOW	–	.779	.920	.685	.780	.901	.645	.703	.706	.570	.735	.918	.558	.675	.697	.646
GTCVM	1	.806	.940	.688	.790	.921	.650	.709	.713	.576	.755	.920	.561	.691	.695	.677
	2	.814	.946	.688	.792	.924	.674	.721	.728	.580	.764	.938	.598	.698	.714	.672
	5	.828	.953	.722	.817	.929	.665	.736	.737	.597	**.773**	**.948**	.611	**.712**	**.751**	.681
	10	**.832**	**.954**	.733	**.820**	**.936**	.603	**.737**	**.739**	.603	.773	.947	.581	.712	.749	.681
	30	.814	.950	.747	.794	.929	.657	.725	.727	.576	.766	.944	.579	.698	.746	.666
	inf	.813	.942	.689	.792	.926	.651	.722	.728	.576	.765	.944	.581	.698	.744	.666
GVSM	–	.790	.923	.705	.783	.903	.640	.706	.71	.576	.750	.943	.591	.687	.720	.672
CVM	–	.765	.941	.672	.790	.930	.672	.708	.725	.576	.751	.934	.604	.685	.716	.669

(Eq. 2.14). Clustering evaluation was based on the supervised measure *normalized mutual information* (NMI) and the *F_1-measure* (see Sec. 2.6.2 for details).

Tab. 3.2, 3.3, 3.5, and 3.6 present the results from the experiments conducted for each collection. Specifically, we compared the classic BOW representation, the GVSM, the proposed GTCVM method (with λ=0.2 in Eq. 3.9), that represents the documents as

Table 3.4: The p and t values of the statistical significance t-test of the difference in k-means performance using GTCVM (σ=10) and the compared representation methods, with respect to the two evaluation measures. Values of p smaller than the significance level of 0.05 (5%) indicate significant superiority of GTCVM.

GTCVM (σ=10) **vs**	$\mathbf{D_1}$		$\mathbf{D_2}$		$\mathbf{D_3}$		$\mathbf{D_4}$		$\mathbf{D_5}$	
	p-val	**t-val**	**p-val**	**t-val**	**p-val**	**t-val**	**p-val**	**t-val**	**p-val**	**t-val**
BOW_{NMI}	$.011 \cdot 10^{-6}$	5.98	$.075 \cdot 10^{-3}$	4.05	$.025 \cdot 10^{-6}$	5.81	$.080 \cdot 10^{-8}$	6.45	.0000	12.8
$GVSM_{NMI}$	.0008	2.68	$.081 \cdot 10^{-3}$	4.02	$.050 \cdot 10^{-3}$	4.15	.085	1.73	$.056 \cdot 10^{-5}$	5.17
CVM_{NMI}	.0051	2.83	.0010	3.33	$.052 \cdot 10^{-4}$	4.65	.1659	1.39	$.077 \cdot 10^{-3}$	4.04
BOW_{F_1}	$.020 \cdot 10^{-5}$	5.39	$.050 \cdot 10^{-2}$	3.54	$.046 \cdot 10^{-2}$	3.56	.0010	3.32	.0000	12.8
$GVSM_{F_1}$	$.037 \cdot 10^{-3}$	4.22	.0021	3.11	$.067 \cdot 10^{-2}$	3.45	.0329	2.15	.0000	9.06
CVM_{F_1}	$.081 \cdot 10^{-3}$	4.02	$.06 \cdot 10^{-8}$	6.50	.0027	3.04	.0314	2.18	.0000	9.31

Table 3.5: NMI values of the clustering solution for VSM (BOW), GVSM, CVM and the proposed GTCVM (for several values of σ) document representations using the spectral clustering algorithm.

Method	σ	$\mathbf{D_1}$			$\mathbf{D_2}$			$\mathbf{D_3}$			$\mathbf{D_4}$			$\mathbf{D_5}$		
		avg	**best**	$\mathbf{avg_{10\%}}$	**avg**	**best**	$\mathbf{avg_{10\%}}$	**avg**	**best**	$\mathbf{avg_{10\%}}$	**avg**	**best**	$\mathbf{avg_{10\%}}$	**avg**	**best**	$\mathbf{avg_{10\%}}$
BOW	–	.753	.761	.750	.781	.788	.737	.569	.585	.555	.718	.780	.631	.558	.559	.506
GTCVM	1	.770	.774	.769	.790	.795	.750	.614	.626	.600	.735	.779	.642	.560	.561	.516
	2	.781	.785	.760	.790	.794	.757	.625	.632	.601	.752	.789	.649	.562	.564	.523
	5	.794	.804	.790	**.833**	**.853**	.763	.639	.640	.619	**.768**	**.827**	.669	.579	**.600**	.557
	10	**.807**	**.814**	.801	.833	.853	.761	**.645**	**.648**	.620	.758	.819	.661	**.581**	.589	.558
	30	.791	.796	.769	.807	.832	.743	.613	.613	.609	.755	.797	.647	.567	.582	.535
	inf	.774	.782	.767	.794	.794	.722	.619	.619	.610	.749	.793	.637	.560	.568	.530
GVSM	–	.756	.770	.702	.794	.830	.747	.593	.595	.586	.722	.780	.637	.548	.554	.513
CVM	–	.761	.768	.751	.801	.823	.760	.605	.606	.590	.728	.794	.642	.557	.566	.519

Table 3.6: F_1-measure values of the spectral clustering solution for the different representation methods.

Method	σ	$\mathbf{D_1}$			$\mathbf{D_2}$			$\mathbf{D_3}$			$\mathbf{D_4}$			$\mathbf{D_5}$		
		avg	**best**	$\mathbf{avg_{10\%}}$	**avg**	**best**	$\mathbf{avg_{10\%}}$	**avg**	**best**	$\mathbf{avg_{10\%}}$	**avg**	**best**	$\mathbf{avg_{10\%}}$	**avg**	**best**	$\mathbf{avg_{10\%}}$
BOW	–	.801	.811	.780	.819	.822	.767	.710	.723	.701	.808	.911	.697	.666	.669	.654
GTCVM	1	.811	.819	.809	.822	.832	.772	.729	.741	.728	.834	.915	.722	.694	.703	.663
	2	.818	.823	.806	.837	.841	.779	.733	.746	.732	.865	.922	.725	.689	.703	.652
	5	.837	.840	.818	.887	**.927**	.792	.744	.756	.737	**870**	**.930**	.740	**.716**	**.727**	.647
	10	**.840**	**.842**	.826	**.890**	.925	.788	**.754**	**.759**	.742	.865	.929	.736	.710	.725	.654
	30	.823	.826	.809	.856	.886	.769	.726	.735	.725	.864	.925	.705	.704	.701	.642
	inf	.814	.817	.806	.826	.832	.734	.728	.735	.729	.859	.922	.703	.692	.686	.653
GVSM	–	.756	.770	.702	.826	.901	.780	.709	.714	.724	.823	.916	.705	.642	.657	.654
CVM	–	.761	.768	.779	.831	.897	.791	.725	.725	.723	.825	.916	.713	.673	.678	.654

described in Eq. 3.12 and the CVM as proposed in [113], where document vectors are computed based on Eq. 3.4 with *idf* weights. More specifically, for each collection, each representation method was tested for 100 runs of spk-means (Tab. 3.2, 3.3) and spectral-c (Tab. 3.5, 3.6). To provide fair comparative results, for each document collection all methods were initialized using the same random document seeds. The average of all runs (*avg*), the average of the worst 10% of the clustering solutions ($avg_{10\%}$), and the best

Table 3.7: The p and t values of the statistical significance t-test of the difference in spectral clustering performance using GTCVM (σ=10) and the compared representation methods, with respect to the two evaluation measures. Values of p smaller than the significance level of 0.05 (5%) indicate significant superiority of GTCVM.

GTCVM (σ=10) **vs**	$\mathbf{D_1}$		$\mathbf{D_2}$		$\mathbf{D_3}$		$\mathbf{D_4}$		$\mathbf{D_5}$	
	p-val	**t-val**	**p-val**	**t-val**	**p-val**	**t-val**	**p-val**	**t-val**	**p-val**	**t-val**
BOW_{NMI}	.0000	27.3	.0000	13.8	.0000	620.	$.026{\cdot}10^{-4}$	4.85	.0000	8.03
GVSM_{NMI}	.0000	16.7	.0000	7.51	.0000	130.	$.129{\cdot}10^{-5}$	4.99	.0000	12.1
CVM_{NMI}	.0000	19.3	$.150{\cdot}10^{-8}$	6.35	.0000	138.	$.316{\cdot}10^{-3}$	3.67	.0000	8.83
BOW_{F_1}	.0000	24.1	.0000	11.4	.0000	875.	$.123{\cdot}10^{-4}$	4.48	.0000	19.1
GVSM_{F_1}	.0000	15.1	.0000	7.53	.0000	410.	$.113{\cdot}10^{-2}$	3.31	.0000	30.7
CVM_{F_1}	.0000	18.7	.0000	7.11	.0000	268.	$.115{\cdot}10^{-3}$	3.94	.0000	14.1

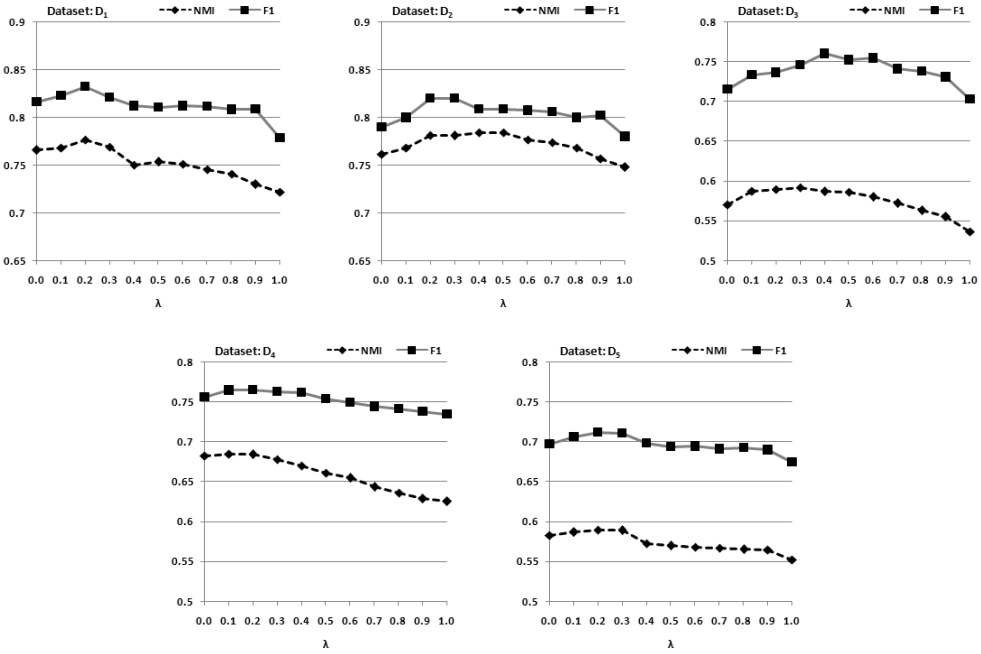


Figure 3.4: The effect of varying the parameter λ on the spk-means clustering performance for each dataset. Eq. 3.9 is used to determine the term self-weight α_ν when computing the *ltcv* histograms.

values are reported for each performance measure. The worst 10% concerns the 10% of the solutions with the lowest Cohesion, while the best clustering solution is that having the maximum Cohesion in the 100 runs (for spectral-c the sum of squared distances is considered for this purpose). Moreover, in Fig. 3.4 we present the average clustering performance of spk-means with respect to the value of λ parameter of Eq. 3.9 where, although not best for all cases, the value 0.2 we used seems to be a reasonable choice for all the datasets we have considered. Note that similar effect was observed for spectral-c method.

In order to illustrate the statistical significance of the obtained results, the well-known *t-test* was applied for each dataset to determine the significance of the performance difference between our methods and the compared methods. We have considered the case where σ=10 for the Gaussian kernel for all datasets. Within a confidence interval of 95% and for the value of degrees of freedom equal to 198 (for two sets of 100 experiments each), the critical value for t is t_c=1.972 (p_c=5% for p value). This means that if the computed $t \geq t_c$, then the null hypothesis is rejected ($p \geq 5\%$, respectively), i.e. our method is superior, otherwise the *null hypothesis* is accepted. As it can be observed from the results of the statistical tests for spk-means presented in Table 3.4, the performance superiority of GTCVM is clearly significant in four out of five datasets with respect to all other methods. For dataset D_4 the tests indicate that GTCVM, although still better than BOW, has less significant difference in performance compared to GVSM and CVM. Table 3.4 provides the respective t-test results for the spectral-c method where, also due to the lower standard deviation of the results using all document representation methods, the GTCVM demonstrates significantly better results than the compared representations.

The experimental results indicate that our method outperforms the traditional BOW approach in all cases, even for small values of smoothing parameter σ (e.g. σ=1 or 2). This substantiates our rationale that the clustering procedure is assisted by the proposed semantic smoothing which takes into account the local contextual information associated with a term occurrence. GTCVM requires moderate values for the parameter σ to achieve better performance. The same is observed for the quality (in terms of NMI or F_1) of the best solution (i.e. the one with maximum Cohesion) found in the 100 runs, where moderate values of σ (i.e. σ=5 or 10) result in better GTCVM performance. Moreover, the clustering results for a wide range of values of the smoothing parameter σ indicate that the method is quite robust to the specification of this parameter. GTCVM behaves similarly to BOW when a low value is set for σ, while when this value becomes very high the discriminative information of the global term context vectors is reduced. This was demonstrated using spk-means and spectral clustering methods. Among them, the latter in all cases except from D_5 presented better average clustering solutions in terms of both

evaluation measures NMI and F_1, while interestingly, spk-means was superior in terms of the best clustering solutions in most cases (with the exception of D_3) despite operating in a feature space of a much larger size.

3.7 Conclusions

We have presented the global term context vector model (GTCVM) document representation, an extension to the vector space model (VSM) that determines a proper feature space to project the typical VSM document vector representations. Our approach is entirely corpus-based and operates in the preprocessing in a sequence of four steps:

i) captures local contextual information associated with each term occurrence in the term sequences of documents,

ii) summarizes the local context vectors of each term into the respective global term context vectors,

iii) constructs the semantic matrix for a problem using the global term context vectors, and finally

iv) projects documents using the semantic matrix.

The proposed approach achieves semantic smoothing by reducing data sparsity, while retaining the original dimensionality. The derived representation maintains the initial interpretability since each dimension is associated with a single vocabulary term.

In the experimental document clustering study, we compared the proposed representation with the typical VSM, the Generalized-VSM and CVM, using Cosine similarity. The statistical analysis of the obtained results indicates that assists well-known clustering algorithms, such as spherical k-means and spectral clustering, to achieve better clustering solutions compared to other representation methods.

Chapter 4

Clustering Using Synthetic Cluster Prototypes

4.1 Introduction

In this chapter we put forth the idea that, although the centroids are the optimal cluster prototypes with respect to certain objective functions (e.g. based on Cosine similarity), their optimality could also become a drawback in *high dimensional and sparse* (HDS) feature spaces and in cases of low data quality (e.g. outliers, noise). Especially, as the number of data objects becomes smaller compared to the complexity of a clustering problem (i.e. number of clusters, dimensionality), the centroids become less appropriate

cluster representatives. Text documents constitute a typical example of data where such an adverse setting is met.

We present the *synthetic prototype*, a novel type of cluster representative that, given the object assignment to clusters, is computed in two steps: i) a *reference prototype* is constructed for the cluster and then ii) *feature selection* is applied on it. We propose the so-called *MedoidKNN* reference prototype which is based on a subset of K objects of a cluster that are close to its medoid. This synthetic prototype favors the representation of the objects of the *dominant class* in a cluster, i.e. the class to which the majority of the cluster objects belong. Finally, we modify the generic sp$k$-means iterative procedure by incorporating synthetic prototypes. This leads to a novel, effective and quite simple clustering method called *k-synthetic prototypes* (*k-sp*) [44].

We conducted an extensive evaluation of the k-sp method examining several options for the synthetic prototypes and comparing it to several traditional clustering methods such as spherical k-means, agglomerative, spectral clustering and two soft subspace clustering methods.

4.2 Background and context

4.2.1 Text representation and representation spaces

The properties of the vector space in which text documents are represented are closely related to the underlying nature of human language. The HDS properties are derived by

i) the very large feature sets that are needed to represent text data, and

ii) the fact that each document is a *semantically narrow instance* of a much more general document class.

For example, two authors may express exactly the same ideas using generally different words or expressions. The text properties have been discussed in detail in Sec. 2.1. This section focuses on the impact of these properties to the pairwise object (dis)similarities,

i.e. the low-level information that any clustering algorithm exploits.

In an HDS space, documents of the same class present average pairwise similarity comparable in magnitude to the similarity between documents from different classes [99, 117]. For instance, let d_x, d_y, and d_z three documents of the same class; it is possible for d_x to share a set of terms with d_y and a different set of terms with d_z whereas, at the same time, d_y and d_z may exhibit no vocabulary intersection. This would be expected to hold mostly for pairs belonging to different classes. In this context, certain qualitative issues arise regarding the direct determination of a large number of nearest neighbors to an object [99, 117]. For example, if an object has non-zero similarity with K objects in the dataset (or a cluster), then the direct determination of its nearest $K'>K$ objects would unavoidably make guesses.

Document clustering differentiates from the high dimensional data clustering problems that seek for a single *global subspace of features* where there are observable clusters; different document clusters are formed in generally different subspaces. Small text datasets should be treated as cases of special interest. According to *Heaps' power-law* [54], the increase of the corpus vocabulary is sublinear to the number of included documents. In order to further analyze this issue, we empirically define the *relative dimensionality* (rd) of the feature space based on the number of features V, the number of data objects N, and the number of clusters k:

$$rd = \log \frac{kV}{N}. \tag{4.1}$$

This quantity may also provide an apriori empirical estimation of the '*difficulty*' of the learning process. Due to the sublinear relation between N and V, rd is expected to be much larger for small datasets than for larger ones. The large vocabulary diversity even between documents of the same class, is an additional justification for the difficulty of clustering small document datasets. Note that, for a fixed k, rd is a monotonically decreasing function as the size of the dataset increases.

4.2.2 Text document subspace clustering

The different topics are usually described by generally different subsets of terms which, in combination with the high sparsity of the feature space, lead to the hypothesis that the underlying cluster structure may be better to be sought in subspaces of the original feature space. The feature selection that is applied in the preprocessing phase actually computes a single *global subspace* where data clustering is performed. A more *fuzzy feature selection* would assign a global weight to each dimension. *Subspace clustering* can be thought as to be an extension to feature selection in the sense that it determines a subspace explicitly for each cluster during clustering.

In brief and according to [40], the main categorization of subspace clustering methods is based on the relation between the axes of the subspaces they seek and the axes of the original feature space. One approach, called *generalized subspace clustering*, is to seek for arbitrarily oriented subspaces. Their major difficulty is to deal with the infinite search space of the candidate subspaces. A second and more widely-used approach is constrained to seek for subspaces with axes parallel to the original. The *projected subspace clustering* lets no intersection between the dimensions that span the different subspaces and hence, 2^d-1 possible subspaces must be examined. The subcategory that lets different axis-parallel subspaces to have dimensions in common is called *soft projected clustering* and usually different feature weights in [0,1] are assigned for each cluster. The latter subcategory can be further split based on the searching approach adopted regarding the feature set a method starts to work with. *Top-down* approaches start with the full set of features and iteratively try to determine narrow subspaces for each cluster. On the other hand, *bottom-up* approaches start from single dimension subspaces and use a strategy similar to mining frequent itemset to increase their dimensionality.

Apparently, there are important methodological differences in the literature of subspace clustering, but a thorough analysis is beyond the scope of this work. In the rest of this section we will discuss the recent research on top-down soft projected subspace clustering methods that develop *feature weighting mechanisms* and incorporate them to

k-means, and have also been tested on the document clustering problem.

An abstract framework is presented in [118] that, using multiple feature vectors to represent each data object, is able to integrate the heterogeneous feature spaces in the k-means algorithm. A *convex-k-means* algorithm is proposed that is based on a convex objective function constructed as a weighted combination of the distortions of each individual feature subspace. The algorithm simultaneously minimizes the average within-cluster dispersion and maximizes the average between-cluster dispersion along all of the feature spaces. A method that received much attention is *clustering on subsets of attributes* (COSA) [119]. It is an iterative algorithm that considers a feature weight vector to each data point, initially containing equal weights for all features. Larger weights are assigned to features that present small dispersion in a neighborhood around the reference object, which means that are more important. The next step is to use these weights to compute some other weights corresponding to each pair of objects that, in turn, update the distances for the computation of the nearest neighbors. The algorithm stops iterating when weight vectors corresponding to objects become stable. COSA outputs a pairwise distance matrix based on a weighted inverse exponential distance and any distance-based clustering method can produce the final clusters. The algorithm requires the user specification of the size of neighborhood to consider, a second parameter that controls the fade of the exponential feature weighting, while the major issue is that all the $N \times V$ parameters should be estimated during the process.

Some other algorithms were then developed that consider one feature weighting vector for each cluster. *Feature weighting k-means* (*fwk-means*) [120] aims to minimize the following objective function:

$$\Phi_{\text{fwkm}}(\text{C}) = \sum_{j=1}^{k} \sum_{d_i \in c_j} \sum_{l=1}^{V} w_{jl}^{h} \left[(\mu_{jl} - d_{il})^2 + \sigma \right], \tag{4.2}$$

subject to

$$\sum_{l=1}^{V} w_{jl} = 1, \quad 0 \leq w_{jl} \leq 1, \ j = 1, ..., k, \tag{4.3}$$

where μ_j is the L_1-normalized centroid of the j-th cluster and $h>1$ a parameter that must be set in advance. The term $w_{jl}^h(\mu_{jl} - d_{il})^2$ computes the distance between the centroid μ_{ji} and a document d_i on the specific l-th feature dimension. Initially, the weights are set to $1/V$ and the k centroids are set in a random fashion. The optimization is then performed by iterating the following steps until convergence:

1. Object assignment to their nearest cluster using the computed centroids and the feature weights.
2. Computation of the cluster centroids using the computed feature weights.
3. Computation of the feature weights for each cluster using the computed cluster centroids.

Given the cluster centroids and the k feature weighting vectors of the previous iteration, the optimal weight of the l-th feature for cluster c_j is computed by:

$$w_{jl} = \left[\sum_{t=1}^{V} \left(\frac{\sum_{d_i \in c_j} w_{jl} \left[(\mu_{jl} - d_{il})^2 + \sigma \right]}{\sum_{d_i \in c_j} w_{jl} \left[(\mu_{jt} - d_{it})^2 + \sigma \right]} \right)^{1/(h-1)} \right]^{-1}, \tag{4.4}$$

where σ is the average dispersion of the vocabulary measured offline in a sample of N_{sample} data objects. fw$k$-means adds this value because a feature weight is not computable if its dispersion in a cluster is zero. If we let mfv_l to be the mean feature value of the l-th feature in the data sample then σ is given by:

$$\sigma = \frac{1}{N_{\text{sample}} V} \sum_{d_i \in c_{\text{sample}}} \sum_{l=1}^{V} (d_{il} - mfv_l)^2. \tag{4.5}$$

Locally adaptive clustering (LAC) algorithm presented in [121] is quite similar to the *Entropy weighting k-means* (*ewk-means*) [122]. Both share some ideas with COSA, whereas the feature weighting vectors are assigned to clusters instead of objects. Moreover, their search strategy is more alike to fw$k$-means. A modified objective function is utilized, which is to add the *weight entropy* $e_j = \sum_{l=1}^{V} w_{jl} \log w_{jl}$ corresponding to each cluster in order penalize the identification of clusters in subspaces spanned by very few

features. The objective function of ew$k$-means is:

$$\Phi_{\text{ewkm}}(\text{C}) = \sum_{j=1}^{k} \left[\sum_{d_i \in c_j} \sum_{l=1}^{V} w_{jl}(\mu_{jl} - d_{il})^2 + \gamma \, e_j \right], \ \gamma \geq 0 \tag{4.6}$$

subject to Eq. 4.3 and the value of γ controls the focus of the objective function on the feature weight entropy. The iterative optimization is identical to that of fw$k$-means and differ only on the weight computation:

$$w_{jl} = \frac{\exp(-disp_{ji}/\gamma)}{\sum_{t=1}^{V} \exp(-disp_{jt}/\gamma)}, \tag{4.7}$$

where

$$disp_{jl} = \sum_{d_i \in c_j} (\mu_{jl} - d_{il})^2. \tag{4.8}$$

COSA and fw$k$-means require the tuning of the value of the parameter controlling the size of the subspaces that are sought (the value of γ in ew$k$-means). LAC introduces an ensemble approach that combines multiple clustering solutions discovered by LAC using different γ values, which produces a superior result than that of the participating solutions. The feature weights of these methods enable the modeling of more complex cluster shapes than the spherical of traditional k-means. However, the parameters that need to be estimated are doubled compared to k-means: $2k \times V$ for the feature weights and the cluster centroids. This parameter increase unavoidably causes a large increase to the number of local minima of the search space. Recently, an adaptive weight-adjusting principle was adopted in [123], which at each step adds a Δw_{jl} to each w_{jl} weight computed based on the extend of contribution of the weight to the clustering quality. Finally, in [124] an algorithm similar to LAC and fw$k$-means is presented, also allowing the incorporation of constraints derived from a labeled data subset.

4.3 The k-synthetic prototypes clustering method

4.3.1 Clustering using centroids and medoids

From an optimization point of view, the normalized centroid is the prototype that maximizes cluster's Cohesion Eq. 2.14. However, this optimality may become a drawback in such a feature space, especially at early clustering iterations where clusters have low homogeneousity due to random initialization. More specifically, there exist two undesirable phenomena concerning the use of centroids. At a data object level, the *self-similarity* phenomenon implies that the similarity of a document with itself becomes the dominant factor for deciding about its nearest cluster [41, 125]. This is explained by observing the similarity between a normalized centroid u_j of the cluster c_j and a member document d:

$$u_j^\top d = \frac{1}{\left\|\sum_{d_i \in c_j} d_i\right\|_2}\left(d^\top d + \sum_{\substack{d_i \in c_j \\ d_i \neq d}} d^\top d_i\right). \tag{4.9}$$

Due to sparsity, the term $d^\top d_i{=}1$ can be large in magnitude compared to the sum of similarities between d and the documents of c_j, or the documents of other clusters. In an extreme case, a document $d \in c_j$ which has non-zero similarity only with documents from clusters other than c_j, may still determine c_j as its nearest cluster, since due to the self-similarity term it may hold that:

$$\frac{d^\top d}{\left\|\sum_{d_i \in c_j} d_i\right\|_2} > \frac{\sum_{d_i \in c_l} d^\top d_i}{\left\|\sum_{d_i \in c_l} d_i\right\|_2} \tag{4.10}$$

Hence, d may remain in an inappropriate cluster. This phenomenon appears more intense in cases where there is a small number of objects per cluster in combination with high sparsity.

The second phenomenon is the *feature over-aggregation* that occurs when computing a centroid for an impure cluster. Supposing that there is a feature subset f_j^+ strongly related to each document class j, and a usually much larger subset f_j^- containing the remaining $V{-}|f_j^+|$ terms, then the learning process aims to find a cluster prototype, i.e. a

weight vector in $\mathbb{R}^V$, being discriminative for that class. This means that for each cluster the clustering algorithm should try to determine the $|f_j^+|$ representative features for its dominant class and to estimate their relative weight distribution in the possible presence of $|f_j^-|$ irrelevant features that should be assigned with very low weights. The effectiveness of such an algorithm may be greatly affected by the level of the relative significance of the features of f_j^+ to that of f_j^- in a cluster at a particular iteration, which can be formally expressed by the following ratio:

$$\delta_j = \frac{\sum_{i \in f_j^+} u_{ji}}{\sum_{i=1}^{V} u_{ji}}. \tag{4.11}$$

Feature over-aggregation appears at the initial iterations where very low δ-ratio values are observed in the clusters of poor quality. This prevents the prototypes from becoming more class discriminative, since the non-informative features also affect the object assignment to clusters and hence the problem is retained.

Both self-similarity and feature over-aggregation constrain the local search flexibility of the k-means procedure and lead to poor solutions strongly dependent on initial conditions, where often documents from two or more classes are assigned to the same cluster.

In what concerns the use of medoid as cluster prototype, it does not present the self-similarity and feature over-aggregation effect. However, as mentioned in Sec. 4.2.1, since each document is a specific semantically narrow instance of the more general topics of its class, it contains a very small fraction of vocabulary terms. Thus it is unlike for a *single document* to be a good cluster representative.

4.3.2 Synthetic cluster prototypes

Traditionally, feature selection (in our case term selection) takes place in the preprocessing phase. However, we adopt a dynamic selection scheme implemented in the form of *synthetic cluster prototypes*, which are computed by first selecting objects and then features from each cluster (Fig. 4.1). As clustering proceeds we exploit the information progressively produced in the formed clusters to retain the important features for each

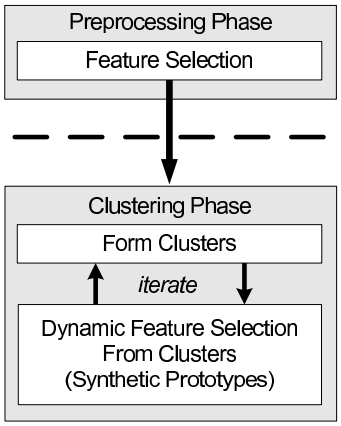


Figure 4.1: The k-sp framework using synthetic prototypes.

cluster. To compute a synthetic prototype we must define:

i) a *reference prototype*, an initial representative of the cluster constructed by a *subset* of its objects, and

ii) *feature selection on prototypes* in order to select features from the reference cluster prototype.

The L_2-normalized cluster representative derived by filtering the features of a reference prototype is a *synthetic prototype*. These prototypes are generic, in the sense that they can be constructed by considering any reference prototype or feature selection scheme. Omitting the feature selection step is also a viable option, thus a reference prototype is also a synthetic prototype. In this case feature selection is achieved implicitly since the reference prototype is computed using a subset of the cluster objects and it may not contain all the vocabulary terms.

The proposed clustering algorithm is called *k-synthetic prototypes* (*k-sp*) and incorporates the synthetic prototypes into the sp$k$-means procedure. Note that sp$k$-means is a *special case* of k-sp where the cluster centroids are used as reference prototypes and no feature selection is applied. By using synthetic prototypes the k-sp procedure aims to discover dynamically a different feature subspace in which each document class can be better separated but, at the same time as we explain, to mitigate the negative effects of the self-similarity and the feature over-aggregation phenomena. The explicit feature selection scheme we have considered is the simple thresholding on the feature weights of

a reference prototype to keep the P most significant features of a cluster (see Sec. 4.3.3). Contrary to the typical preprocessing feature selection techniques, k-sp does not affect the original data objects and hence, does not constrain future iterations with previous cluster representations. In a later phase, one could consider much more detail (i.e. more objects and features) from the clusters to fine-tune the solution.

A straightforward option for reference prototype is the Centroid$^{(r)}$[1] of a cluster. The assumption behind this choice is that many of the representative features for the dominant class in a cluster would have high weights in the respective centroid. Thus, the feature selection on it would keep the highly descriptive features for this class. Obviously, this is not true for a cluster containing documents of more than one class none of which is clearly dominant (Fig. 4.2b).

We propose *MedoidKNN*$^{(r)}$, an approach to construct the reference prototypes by computing the centroid of a *subset* Y of documents assigned to a cluster that are descriptive of its dominant class. The set Y can be formed by selecting the K documents of the cluster being the *nearest neighbors to the medoid* of that cluster, including the medoid itself. As explained in Section 4.2.1, it would not be very efficient to directly determine a large number of nearest neighbors of a medoid using its pairwise similarities, since the medoid document may contain only a part of the features present in the cluster. This issue is further discussed on real world examples in Sec. 4.4.3. Therefore, we propose an incremental procedure to form the set Y that avoids computing a large number of nearest neighbors directly from the medoid object. Let λ be the number of desired steps and β_i, i=1,...,λ a sequence of values such that $0<\beta_i<\beta_{i+1}<...<\beta_\lambda=1$. Starting with the medoid $Y_0=\{m\}$, each next subset Y_i (for $i\geq 1$), is formed by the $\lceil \beta_i K \rceil$ documents nearest to the centroid of subset Y_{i-1}. For a two-step example with β_1=0.2, and β_2=1:

i) first, the medoid of the c_j cluster is determine, then

ii) the $\lceil 0.2K \rceil$ objects in c_j are determined that are nearest to the medoid and compute their centroid rp_1, and

[1] In cases where we need to be more specific we denote explicitly with the superscripts (r) and (s) the reference and the synthetic prototypes, respectively.

iii) the K objects in c_j nearest to rp_1 are located, and rp_2 is computed which is the final MedoidKNN$^{(r)}$. Notice that for K=n_j, the rp coincides with the centroid of cluster c_j, while for K=1 it is the cluster medoid.

Typically, up to three steps (λ=3) are sufficient to determine a proper final set Y_λ.

One could argue that the set Y should contain the nearest documents to the cluster centroid and not to the medoid. As a matter of fact, the medoid is close to centroid in a homogeneous cluster and the nearest objects to medoid may also be the nearest objects to the centroid. However, if there are objects of more than one class in a cluster, the medoid-based construction of Y is more probable to lead to a sharp preference for one of the overlapping classes (see Fig. 4.2). This argument is strengthened by a usually holding property called *intracluster rNN-consistency*: any data object in a cluster and its r nearest objects in the same cluster will belong to the same class with high probability. We should remark that intracluster rNN-consistency is expected to be higher than the rNN-consistency of the whole dataset that can be similarly defined [126].

Another advantage of k-sp method is that by ignoring some documents that are far from the synthetic prototypes, it provides robustness and ensures that possible outlier and noisy objects will not affect any cluster representation (similarly for noisy features). These objects are not discarded from the dataset. Besides, one object may be ignored as a noisy-outlier at an iteration when computing a cluster representative, while it could be later considered as one core object in case it is reassigned to another cluster, or its current cluster changes dramatically, and the object is now located near the new cluster medoid.

The k-sp exhibits some similarity with the soft subspace clustering methods. The object selection of the reference prototype defines implicitly a feature subspace for a cluster while the feature selection on it explicitly prunes this subspace. Instead of using a separate feature weighting mechanism per cluster, which also doubles the parameters need to be estimated, k-sp uses a heuristic way to directly determine better vector representations for the clusters. Using object selection it actually tries to favor the representation of the dominant class in a cluster which implicitly results in subspace cluster representation.

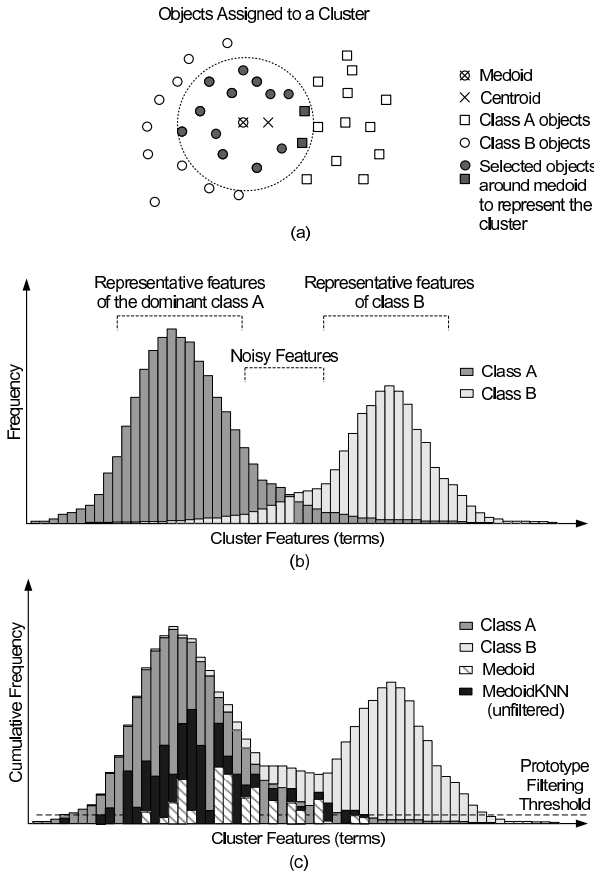


Figure 4.2: A cluster example that combines two data classes. It illustrates the rationale of using objects around the cluster medoid to favor the representation of the dominant class A and to enable the reassignment of the objects of the other class(es) to other clusters. (a) Object-level view of a cluster where the medoid's nearest neighbors belong mostly to the dominant class. (b) Feature-level view of a multidimensional cluster that illustrates the imaginary histogram of the feature frequency for each of the classes. On the horizontal axis, we suppose an ordering where features that exist in both class (probably noisy) lay between the two peaks of representative class features. (c) The histogram of the cumulative feature frequency over both classes. The respective distributions are also presented for the medoid and the MedoidKNN$^{(r)}$ cluster prototypes.

Another worth mentioning difference is that we claim that after having concluded to a set of synthetic representatives defined in certain feature subspaces, then we may take into account the complete feature space to refine the clustering.

Algorithm 1 provides the pseudocode for the k-sp method that incorporates the synthetic prototypes, constructed using Algorithm 2, into the sp$k$-means algorithm. The clustering Cohesion is computed with respect to the synthetic prototypes. It must be noted that k-sp cannot guarantee the monotonicity of convergence. In the case of Centroid$^{(s)}$, we

Algorithm 1 k-Synthetic Prototypes Clustering Method

function $kSP\,(k,\, p_{\text{obj}},\, p_{\text{feat}},\, ref\_flag)$
input: the number of clusters k, two parameters p_{obj}, p_{feat} (see Algorithm 2), a flag ref_flag that enables refinement
output: the k clusters and the set of final prototypes
let: a partition C, the synthetic cluster prototypes S, and the respective clustering cohesion H
$ConstructSP\,(\text{C},\, p_{\text{obj}},\, p_{\text{feat}})$ Algorithm 2 for each cluster of the partition C
$RefineSolution\,(\text{C})$ k-sp using Centroid$^{(s)}$ prototypes (spk-means) initialized by partition C
end let

1: $\{\text{C},\, S\} \leftarrow InitializeClusters\,(\,)$
2: $H \leftarrow Cohesion\,(\text{C},\, S)$
3: **repeat**
4: $\quad\{\text{C}^{(\text{prev})},\, S^{(\text{prev})},\, H^{(\text{prev})}\} \leftarrow \{\text{C},\, S,\, H\}$
5: $\quad\text{C} \leftarrow AssignDocsToClusters\,(\,)$
6: $\quad S \leftarrow ConstructSP\,(\text{C},\, p_{\text{obj}},\, p_{\text{feat}})$
7: $\quad H \leftarrow Cohesion\,(\text{C},\, S)$
8: **until** $C \equiv \text{C}^{(\text{prev})}$ **or** $H \leq H^{(\text{prev})}$
9: **if** $H < H^{(\text{prev})}$ **then**
10: $\quad\{\text{C},\, S,\, H\} \leftarrow \{\text{C}^{(\text{prev})},\, S^{(prev)},\, H^{(\text{prev})}\}$
11: **end if**
12: **if** $ref\_flag == \text{TRUE}$ **then**
13: $\quad\text{C} \leftarrow RefineSolution\,(\text{C})$
14: **end if**
15: **return** $\{\text{C},\, S\}$

compute the cluster centroid as reference prototype that maximizes the cluster Cohesion $\Phi_{\text{COH}}(c_j)$, but this optimality is lost after filtering its features. Similarly, for MedoidKNN$^{(s)}$ prototypes, it is not possible to guarantee that cluster Cohesion will increase at all iterations and it is essential for k-sp to monitor the objective function and to terminate the procedure if a deterioration of the overall Cohesion is observed (the condition $H{<}H^{(\text{prev})}$ in Algorithm 1). In this case, the clusters of the previous iteration are considered as the solution to the problem produced by the main k-sp procedure.

4.3.3 Definition of parameters

The k-sp parameters for computing the MedoidKNN$^{(s)}$ prototype can be defined with respect to the *volume of cluster information*, namely the number of cluster members n_j and the distribution of feature weights aggregated in the reference prototype of a cluster. Two parameters must be specified by the user: $p_{\text{obj}}, p_{\text{feat}} \in [0,\, 1]$. The number of medoid neighbors K_j is computed as:

$$K_j = \lceil p_{\text{obj}}\; n_j \rceil\,. \tag{4.12}$$

Note that different number of neighbors are considered for each cluster c_j, while only documents with non-zero similarity to rp are selected (this is implemented by the function $NNDocs\,(c,\, rp,\, r)$ in Algorithm 2). In what concerns the feature selection, an option is to find the $P_j{=}\lceil p_{\text{feat}} V_j^{(r)} \rceil$ terms of highest frequency in the reference prototype of rp_j that would cost $O(V_j^{(r)})$. Our implementation uses a more efficient approach which is to select the highest weighted features (including the *idf* component) that contain a fraction p_{feat} of the total feature weight sum $\sum_{i=1}^{V} rp_{ji}$ (*total information*) of the reference prototype vector rp_j. Let $y(i)$, $i{=}1,...,P_j$, a function that indexes the selected features which represent the specified p_{feat} information fraction, then P_j is described by:

$$P_j \leq V_j^{(r)} : \frac{\sum_{i=1}^{P_j} rp_{jy(i)}}{\sum_{i=1}^{V_j^{(r)}} rp_{ji}} \simeq p_{\text{feat}}. \tag{4.13}$$

The more uniform the weight distribution of rp_j, the more features are selected to represent the c_j cluster. Typically, the cost of this operation is $O(V_j^{(r)} \log(V_j^{(r)}))$, due to the need of weight ordering. However, this can be reduced to $O(V_j^{(r)} + z \log z)$ by splitting the range of feature weight values of a cluster into several intervals (bins), where only a small number of features z contained in one bin may be needed to be ordered and then to select the most informative subset out of them.

4.3.4 Refining the solution of k-synthetic prototypes

The robustness of the proposed k-sp method is the result of its ability to overcome adverse situations in initial clustering iterations and hence to avoid poor locally optimal solutions. After the termination of the *basic* procedure of k-sp method, the result may be further *refined* by considering the centroids of the obtained clusters as the initial prototypes for a final run of k-sp that now coincides with the regular sp$k$-means (this option is enabled by the flag ref_flag in Algorithm 1). This refinement strategy i) aims to improve the result of k-sp method by using more detailed information for homogeneous clusters already produced by the basic k-sp phase, ii) assists in reducing the sensitivity of the k-sp to parameter definition K and P (see Sec. 4.4), and iii) constitutes a straightforward

Algorithm 2 MedoidKNN Synthetic Prototype Construction

function *ConstructSP* (c, p_{obj}, p_{feat}, λ, β)
input: a cluster c, a threshold $p_{\text{obj}} \in [0, 1]$ that determines the number of documents used for reference prototype construction, $p_{\text{feat}} \in [0, 1]$ for feature selection on it, the number of steps λ, and a vector β of length λ that control the incremental construction (see Sec. 4.3.2)
output: the synthetic prototype MedoidKNN$^{(s)}$ for cluster c
let: n_c the number of documents in cluster c, and m_c its medoid object
$NNDocs\,(c, rp, r)$ determines the rNNs to rp vector in cluster c with non-zero similarity
$Centroid\,(\mathrm{Y}_c)$ computes the centroid of a set Y_c
$FSonRP\,(rp, p_{\text{feat}})$ applies feature selection on the reference prototype rp based on the parameter p_{feat} and normalizes the final prototype to unit length (L_2-norm)
end let

$\mathrm{Y}_c \leftarrow \{m_c\}$
$rp \leftarrow m_c$
$K_c \leftarrow \lceil p_{\text{obj}}\, n_c \rceil$
if $K_c > 1$ **then**
  do for i=1,...,λ
    $\mathrm{Y}_c \leftarrow NNDocs\,(c, rp, \lceil \beta_i K_c \rceil)$
    $rp \leftarrow Centroid\,(\mathrm{Y}_c)$
  end for
end if
$sp \leftarrow FSonRP\,(rp, p_{\text{feat}})$
return $\{sp\}$

approach to choose the best clustering solution among those obtained for different k-sp parameter settings by *comparing the values of the objective function after the refinement step*. This procedure is described in the next section.

The experimentally observed improvement achieved by refinement supports our basic assumption that centroids do not provide sufficient flexibility when clusters are not homogeneous and object reassignments should be encouraged. To tackle this problem one could try to improve the initialization of an iterative method with specialized object-based seeding techniques, or using the clusters produced by a clustering method of different characteristics as the initial partition. Interestingly, the k-sp method is *self-refined* by simply using different values for method parameters, since sp$k$-means is a special k-sp case. The clustering improvement achieved by k-sp refinement phase also confirms that self-similarity and feature over-aggregation play a crucial negative role mostly due the clusters' impurity at the initial iterations of the search procedure. The clusters obtained by the basic phase of k-sp need only a few refining reassignments, thus the self-similarity phenomenon is not a very important issue. Moreover, each respective cluster centroid would have a high δ-ratio (Eq. 4.11) that enables the fine-tuning of its V feature weights

which would lead to an improvement in its class-discrimination.

4.3.5 Selecting the k-sp parameters

An additional advantage of the refinement phase of k-sp, which uses the centroids as cluster prototypes, is that it enables the direct comparison of the results obtained using different values for k-sp parameters. The latter is a very important aspect of k-sp, since it allows the selection of the best setting for parameters p_{obj} and p_{feat}. More specifically, the user could specify two sets of candidate parameter values, the set $\text{S}_{p_{\text{obj}}}$ for p_{obj} and the set $\text{S}_{p_{\text{feat}}}$ for p_{feat}. Then, using the same random initial conditions, k-sp runs several times for each combination of the two parameter values and by monitoring the average value of the *refined objective function* (Eq. 2.14), we can determine which parameter values provide the best average performance. The procedure can be summarized by the following steps:

1. The sets of values $\text{S}_{p_{\text{obj}}}$ and $\text{S}_{p_{\text{feat}}}$ are specified by the user.
2. Run k-sp with refinement (Algorithm 1) several times for each combination of parameter values $p_{\text{obj}} \in \text{S}_{p_{\text{obj}}}$ and $p_{\text{feat}} \in \text{S}_{p_{\text{feat}}}$.
3. Compare the average value of the refined objective function of each set to determine the best k-sp average performance and the corresponding parameter values.

Furthermore, the above procedure may reveal important information about the dataset characteristics. As we will see in the experimental section, the observation of better performance provided by smaller synthetic prototypes may indicate that the data clusters are overlapping in many dimensions (i.e. vocabulary terms in common), or that there are a lot of noisy objects/terms.

4.3.6 Implementation and complexity

In the present context, where document vectors and cluster centroids are normalized with respect to L_2-norm, it is easy to show that the medoid of a cluster is the cluster object with maximum Cosine similarity (dot product) to the centroid of that cluster. Let

$u_j = \sum_{d_i \in c_j} d_i \,/\, \left\| \sum_{d_i \in c_j} d_i \right\|_2$ the normalized centroid of cluster c_j with respect to L_2-norm, then Eq. 2.13 can be expressed as:

$$m_j = \arg\max_{d \in c_j} \left\{ d^\top \sum_{d_i \in c_j} d_i \right\} = \arg\max_{d \in c_j} \left\{ d^\top u_j \right\}. \tag{4.14}$$

Hence, we can determine the medoids of all clusters with linear cost $O(N)$ to the size of the corpus. Thus, both '*spherical*' version of k-medoids and k-means method have the same asymptotic cost. It must be noted that it is possible for a cluster to have more than one '*medoid*', i.e. objects whose total similarity to the other cluster objects has exactly the same maximum value. Moreover, those objects are equally distant to the cluster centroid. None of them can be considered superior to the others, hence, we can randomly select any of them to construct our synthetic prototype.

Suppose we are given for every object d an ordered list containing the other $N-1$ objects in descending order with respect to their similarity to d. Then it is possible to determine the $K-1$ objects in a cluster that are nearest to its medoid by linearly traversing the respective list ($K-1 \leq N$). By taking advantage of the intracluster rNN-consistency property, we can precompute offline a number of K_{nn} ($K-1 \leq K_{nn} \leq N$) nearest neighbors for each document in the dataset. If a list has less than $K-1$ objects that are assigned to the same cluster with the medoid object d, we have to necessarily apply greedy search in cluster to locate the rest nearest neighbors to d, up to the desired $K-1$. Supposing that we have set a proper K_{nn} value that eliminates the previously mentioned greedy search, then the non-incremental (λ=1) construction of a MedoidKNN$^{(r)}$ prototype costs $O(n_j+K_{nn}+KV)$. This includes the cost: i) to determine the medoid document: $O(n_j)$, ii) to locate medoid's $K-1$ nearest neighbors in the cluster: $O(K_{nn})$, and iii) to compute the centroid of the K objects: $O(KV)$. The latter is the first step of the incremental MedoidKNN^r construction (λ>1). For the steps other than the first we have to seek the nearest documents to the partial centroid (synthetic prototype) computed so far. For the j-th cluster, this can be done by computing and then sorting the pairwise similarities between the $n_j^{(i)}$ data objects and its synthetic prototype in step i, where i=2,...,λ. Thus,

Table 4.1: Datasets used in the experimental evaluation

Dataset	Source Docs/Topic	Classes	Docs	Class Balance	V	consistency 1NN	consistency 10NN	OS	CS
Talk_3	20-NGs: *guns*, *mideast*, *religion.misc*	3	900	1.0	7051	.952	.854	98.8	98.2
$\text{RS}_4^{(S)}$	20-NGs: *autos*, *motorcycles*, *crypt*,	4	800	1.0	3451	.853	.694	98.5	97.2
$\text{RS}_4^{(M)}$	*electronics*		1600	1.0	7818	.939	.807	99.3	98.7
$\text{RS}_4^{(L)}$			3928	.980	12708	.963	.872	99.6	99.2
$\text{M}_6^{(S)}$	20-NGs: *pc.hardware*, *autos*, *baseball*,	6	1200	1.0	7154	.885	.767	99.3	98.2
$\text{M}_6^{(M)}$	*hockey*, *electronics*, *med*		3000	1.0	12082	.932	.816	99.6	98.9
$\text{M}_6^{(L)}$			5891	.980	17955	.953	.862	99.7	99.2
$\text{M}_8^{(S)}$	20-NGs: *atheism*(50,795), *hockey*(100,989),	8	600	.500	4350	.767	.578	98.9	96.9
$\text{M}_8^{(M)}$	*windows.x*(100,959), *forsale*(100,957),		2000	1.0	9608	.824	.690	99.4	98.4
$\text{M}_8^{(L)}$	*electronics*(100,975), *politics.misc*(100,770) *mac.hardware*(50,955), *graphics*(50,955)		7355	.780	20592	.912	.783	99.7	99.2
NG_4	20-NGs: *comp.**, *rec.**, *sci.**, *talk.**	4	12000	.985	31498	.954	.877	99.8	99.6
Mini_{20}	20-NGs: *from all of the 20 newsgroups*	20	1870	.970	10463	.666	.494	99.4	97.5
Wap_{20}	WebACE	20	1560	.015	8460	.696	.636	98.6	95.8
K1_6	WebACE	6	2340	.043	13879	.954	.909	99.1	98.1
Rev_5	TREC	5	4069	.043	23220	.878	.834	99.2	98.
$\text{A}_4^{(1)}$	Artificial dataset generator	4	4000	1.0	9401	.951	.916	99.7	99.5
$\text{A}_4^{(2)}$			4000	1.0	9461	.922	.875	99.7	99.5
$\text{A}_4^{(3)}$			4000	1.0	9437	.849	.792	99.6	99.5
$\text{A}_4^{(4)}$			4000	1.0	9469	.693	.630	99.6	99.3

if a subset of $K^{(i)}$ cluster objects are used to construct the MedoidKNN^r for cluster c_j at step i>1, then the construction complexity is $\text{O}(n_j^{(i)}V + n_j^{(i)}log(n_j^{(i)}) + K^{(i)}V)$.

4.4 Experimental evaluation

4.4.1 Clustering methods

To provide a comparison of k-sp performance to other clustering methods, we implemented sp$k$-means, k-medoids, hierarchical agglomerative clustering (HAC), and spectral clustering. For HAC we have used the *average-link* cluster merging criterion based on the Cosine similarity [127]. In addition, we compare k-sp with feature weighting k-means (fw$k$-means) [120] and entropy weighting k-means (ew$k$-means) [122] which, according to the comparative result in the latter work, performs better than a series of other soft and hard subspace clustering methods. It is noteworthy that these two methods use the Euclidean distance measure instead of the Cosine similarity, whereas for normalized doc-

ument vectors with respect to the L_2-norm, euclidean and Cosine measures determine the same proximity ordering between data objects. The parameters h and γ, respectively, were both set to 1.5 for all datasets. This value was used as well in [120] to apply fw$k$-means on the 20-Newsgroups dataset that we also use in our experiments. In addition, in [121] it is also reported ew$k$-means to perform well on the same dataset with γ=1.5. Besides, it is also illustrated that ew$k$-means is not sensitive to the setting of γ value. Actually, we conducted a number of preliminary tests for these algorithms using parameter values within a wide range, but the observed differences in clustering performance was insignificant.

The sp$k$-means [41] is the baseline approach, the same algorithm is also utilized to refine the solution produced by HAC and k-medoids. In order to show that in the HDS feature space marginal clustering improvement should be expected by the careful selection of objects as initial seeds for sp$k$-means, since as explained single objects are inappropriate for representing groups of many objects, some sp$k$-means initialization techniques were also tested:

i) the *random clusters* where each object is randomly assigned to one cluster,

iii) the effective *k-means++* method [92] that try to spread the initial centroids away from each other.

As for *spectral clustering*, it is based on spectral analysis of the similarity matrix of the dataset. We have used the standard algorithm described in [85] (see Sec. 2.6.1).

Generally, the k-sp variants are denoted by the respective synthetic prototypes they consider, e.g. Centroid-P(p_{feat}), MedoidK(p_{obj})NN-P(p_{feat})[2]. The set of values considered for p_{obj} are: $S_{p_{\text{obj}}}$={.90, .80, .60, .40}, and for p_{feat}: $S_{p_{\text{feat}}}$={.98, .95, .90, .80, .60, .40}. In all cases, MedoidKNN$^{(r)}$ has been constructed incrementally in three steps (λ=3) with β_1=0.2, β_2=0.6, β_3=1 (see Sec. 4.3.2). In Tab. 4.2, we provide the percentage of the original features retained after computing various synthetic prototypes for a specific cluster example to provide a notion of the feature selection that is caused by object selection in a HDS feature space.

[2]MedoidK(·)NN is also denoted as K(·)NN for brevity

Since we are given the ground truth labeling of the documents in all datasets, clustering evaluation is based on the two popular supervised measures *normalized mutual information* (NMI) and *Purity*. Higher values indicate better results (see Sec. 2.6.2 for details).

4.4.2 Datasets

Real data

In order to conduct controlled experiments with respect to the corpus size, cluster sizes and overlap, both real and artificial datasets were used (see Tab. 4.1). We constructed a series of clustering problems from real collections, by first selecting certain topics from a collection and then by producing different instances of these problems. In particular, we considered several subsets of the popular 20-Newsgroups[3] collection using as ground truth the provided class label of each document. As an example, $M_6^{(S)}$, $M_6^{(M)}$, $M_6^{(L)}$ are three datasets generated from same topics but with increasing cluster sizes: small, medium, and large that includes all the documents of the selected topics. $Mini_{20}$[4] contains 100 documents from each one of the twenty newsgroups, while NG_4 is a subset containing all the four largest subjects in collection, namely *computer*, *records*, *science* and *talk*. Moreover, we used three datasets from the Cluto package[5]: $K1_6$ and Wap_{20} are from the WebACE project and contain web pages from different directories of Yahoo!, Rev5 is derived from the San Jose Mercury newspaper articles that are distributed as part of the TREC collection (TIPSTER Vol. 3).

In brief, in the preprocessing of each dataset, we eliminated trivial terms (stopwords), headers and special tags, we applied *Porter's stemming transformation* [58] and *document frequency thresholding* (DF) [64] to discard terms that appear in only one document (dft=1). Thus, all rare terms that have high *discriminating power* were maintained. Finally, we used only documents having more than five terms. In Tab. 4.1, we report

[3]Available at: http://people.csail.mit.edu/jrennie/20Newsgroups/
[4]Available at: http://kdd.ics.uci.edu/databases/20newsgroups/
[5]Available at: http://www.cs.umn.edu/~karypis/cluto

for each dataset the balance of class sizes, the 1NN and 10NN-consistency (*leave one out* classification accuracy), the overall sparsity (OS) of each dataset which is the average number of zero dimensions that a data vector presents, and the sparsity of each class (CS) when considering only the vocabulary used by the class members (note that OS $\geq$ CS). We also report for the datasets we constructed the number of documents per class that were used in cases of sensible imbalance of class sizes (Docs/Topic).

Artificial data

In order to construct the artificial text collections we implemented a corpus generator. To generate a corpus with k clusters, our algorithm assumes that the terms (the feature space) are partitioned into k+1 disjoint *topic vocabulary bags*. Each bag B_i, i=1,...,k contains the terms related to i-th topic, while an additional bag B_{k+1} contains general terms that could be used in the documents of any cluster.

Each text document is considered to be a sequence of terms and each term of a sequence is generated in two steps: 1) selecting a vocabulary bag, and then 2) selecting a term from that bag. The correlation between a data cluster and the vocabulary bags is user-defined in a $k\times(k+1)$ matrix W, where each element W_{ji} is the probability of selecting the bag B_i when producing a term for a document of the j-th cluster. To sample a term from an already selected bag (step 2) we used the *Zanette-Montemurro stochastic process* (ZM) [128] that has been proposed for generating a single long artificial text that has similar statistical characteristics to real texts, such as the *Zipf's power-law* [53] of term frequencies and the sublinear increase of the vocabulary length as the text becomes longer. To achieve these goals the ZM process considers a time decreasing probability controlled by a parameter v of inserting a previously unseen term in the text, i.e. $p_t = \alpha t^{v-1}$. Otherwise an already selected term of a bag is reselected with a probability proportional to the number of times that has already been used in the created sequence. This property of the process (called '*memory*') permits high frequencies for some terms, while the majority of terms present low frequency. In our algorithm, the generation of documents is conducted in cluster order, i.e. the documents of the first cluster then that

Table 4.2: The percentage of features retained in the synthetic cluster prototypes for a cluster containing 300 documents from the first topic of Talk_3 dataset. The centroid contains all the 4264 non-zero dimensions of the cluster.

Reference Prototype	p_{feat} 1.0	.98	.95	.90	.80	.60	.40
Centroid	100	84.0	72.5	59.3	42.0	21.5	9.8
Medoid	4.6	–	–	–	–	–	–
MedoidK(.9)NN	98.0	82.2	71.1	58.0	40.8	20.7	9.4
MedoidK(.8)NN	95.7	80.6	69.5	56.7	39.6	20.0	9.1
MedoidK(.6)NN	89.5	75.8	65.4	53.1	36.7	18.5	8.5
MedoidK(.4)NN	76.7	65.5	56.4	45.8	31.9	16.2	7.3

of the second etc. The memory of the general bag B_{k+1} is maintained during the whole procedure, but the memory of all the other bags is reset when starting the generation of the documents of a new cluster. Using this strategy, in the documents of each cluster a (generally) different set of terms from all the bags would present high frequencies.

To demonstrate the superiority of k-sp performance under situations of clusters that overlap in many dimensions we constructed four artificial datasets called $\text{A}_4^{(i)}$, i=1,...,4 using the above algorithm. All datasets have four clusters (k=4), each of them containing 1000 documents and five topic vocabulary bags were considered with 2000 terms each. The datasets exhibit increasing cluster overlap (from $\text{A}_4^{(1)}$ to $\text{A}_4^{(4)}$), by lowering the probabilities W_{ii} (i=1,...,4) and increasing the probabilities W_{ij} ($j \neq i$) of selecting a term from the rest of the bags. The probability matrices W are presented in Fig. 4.5 (the fifth bag contains the general vocabulary). The length of each document was randomly set by an exponential distribution with mean value λ_{exp}=1/100. The parameter values that we used for the ZM process are α=0.3 and v=0.9.

Generally, we seek to find a clustering solution that maximizes both NMI and Purity to values close to unit. For each dataset and method we report the values of these indexes. For the methods depending on initialization we also report the average value of each index over the runs on a dataset, while we also report (denoted as '*best*') the value of each index (NMI or purity) corresponding to the solution with the highest clustering objective function Φ_{COH} among the 50 runs.

Moreover, in order to evaluate a method's behavior during iterations, we introduce

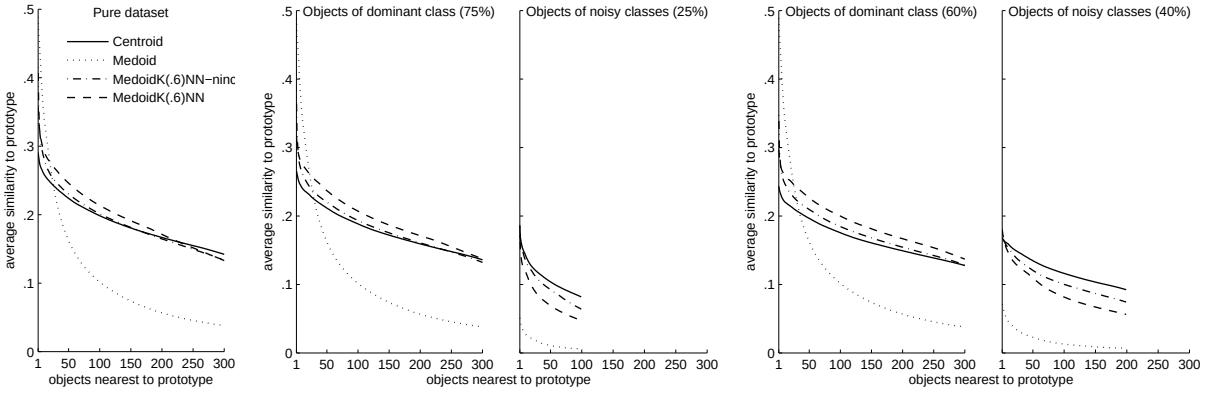


Figure 4.3: The decrease of average similarity between different types of cluster prototypes and the nearest objects around them as the number of neighbors increase. The datasets consist of objects belonging to a dominant class and two other classes corresponding to noise. We considered three percentages for the objects of the noisy classes: (a) a pure dataset (0%), (b) 25%, and (c) 40%. MedoidK(.6)NN-nincr denotes the reference prototype constructed non-incrementally using the 60% of the objects of each dataset.

the Q-*index*:

$$Q_t = 1 - \frac{\Phi_{ics}^{(t)}(\mathrm{C})}{\Phi_{ics}^{(t-1)}(\mathrm{C})},\ t > 0, \tag{4.15}$$

where $\Phi_{ics}^{(t)}(\mathrm{C})$ is the *intracluster similarity measure* defined as the sum of pairwise Cosine similarities between objects in the same cluster at iteration t:

$$\Phi_{ics}^{(t)}(\mathrm{C}) = \sum_{j=1}^{k} \left[\frac{2}{N\,(n_j - 1)} \sum_{d_i \in c_j} \sum_{d_r \in c_j,\ i<j} d_i^{\top} \cdot d_r \right], \tag{4.16}$$

where n_j the size of cluster c_j. Initially, we assume that Q_0=0 holds. Higher values of Q-index indicate greater relative improvement of the clustering quality after one iteration.

Finally, the *statistical t-test* was applied to estimate the significance of the average performance difference between k-sp and the methods under comparison for each dataset, except for HAC that is deterministic. Within a confidence interval of 95% and for the value of degrees of freedom equal to $2 \cdot number\_of\_runs - 2$ we can test if our method is significantly superior, otherwise the *null hypothesis* is accepted.

4.4.3 Experimental results

Robust cluster representation

Our first intention in the experiments is to demonstrate the robustness and effectiveness of synthetic prototypes in favoring the representation of the dominant class in a cluster that contains documents from more than one class. To this end we constructed three sets of documents from the topics of Talk_3 dataset: a) a pure set of 300 documents from the first topic (0% noisy objects), b) the previous set along with 50 documents from each of the other two topics (25% noisy objects), c) a set of 300, 130, and 70 documents from each topic (40% noisy objects). In all three cases the medoid of the complete dataset belongs to the dominant class (i.e. the first topic). Fig. 4.3 demonstrates the decrease of average similarity between different types of cluster prototypes considered for the above cases and the nearest objects around them as the number of neighbors increase. We can observe the high average similarity of the medoid with its very close neighbors that decreases rapidly as we consider wider neighborhoods. This indicates that the medoid exhibits high intracluster rNN-consistency (see Sec. 4.3.2) and empirically explains why the medoid-based construction of synthetic prototype is more class-discriminative than the centroid-based. The result is the higher average similarity to the members of the dominant class, and the lower similarity values to the documents of other classes (considered as noisy). Furthermore, the incremental construction of MedoidKNN performs better than the direct construction based on the K nearest neighbors of the medoid. Tab. 4.2 reports the percentage of features that have non-zero weights after the implicit (i.e. features retained in the reference prototype) and explicit (i.e. additional feature selection on reference prototype) feature selection. We can see the extent to which synthetic prototypes can summarize the characteristics of the document clusters, as well as that synthetic prototypes can discover feature subspaces to represent data clusters.

In another experiment we intend to demonstrate the robustness of k-sp under adverse initial conditions. We considered the $\text{M}_6^{(S)}$ dataset and examined the case where clusters are initialized by *randomly assigning each document to a cluster*. Tab. 4.3 reports the

Table 4.3: Clustering results on the $M_6^{(S)}$ dataset using k-sp variants.

Reference Prototype	$p_{\mathbf{feat}}$	**NMI** *avg.*	*best*	**Purity** *avg.*	*best*	$\bar{t}$
Centroid	1.0	.480	.564	.630	.751	17.1
Centroid	0.8	.484	.644	.632	.798	16.1
Centroid	0.4	.528	.679	.655	.807	16.3
Medoid	1.0	.286	.424	.504	.648	2.5
MedoidK(.4)NN	1.0	.564	.681	.688	.833	6.9
MedoidK(.8)NN	1.0	.686	.792	.777	.899	13.9

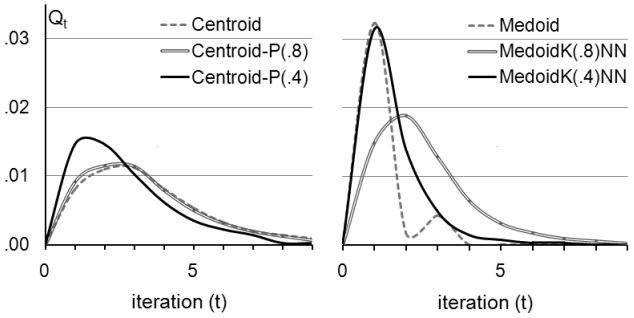


Figure 4.4: The evolution of the average Q-index with clustering iterations for 50 randomly initialized runs using the $M_6^{(S)}$ dataset.

average and best values of the evaluation measures, and the average number of iterations until convergence ($\bar{t}$) for 50 random restarts without refinement. Fig. 4.3 illustrates how the average Q-index value evolves with iterations for each method. An efficient approach should maximize the area under its corresponding curve, either by executing many iterations or by making larger improvements in shorter time. Fig. 4.3 indicates the weakness of centroid representation: it defines an optimal cluster representative assuming that all its documents should stay in that cluster. This constrains to a great extent the representation flexibility and forces the procedure to reach poor locally optimal solutions not far from the bad initial clusters. As k-sp becomes more selective on the cluster's features, as in the case of Centroids$^{(r)}$ (e.g. with P(.4)), we observe immediate clustering improvement in the first iterations. However, the main problem remains: the features are selected from the centroids of impure clusters. Despite the fact that medoids lead to a major initial improvement related to a sharper preference to represent one class out of many others in a cluster, subsequently, the procedure converges too early (2.5 iterations on average). On the other hand, the k-sp with MedoidK(.8)NN is a more balanced choice that com-

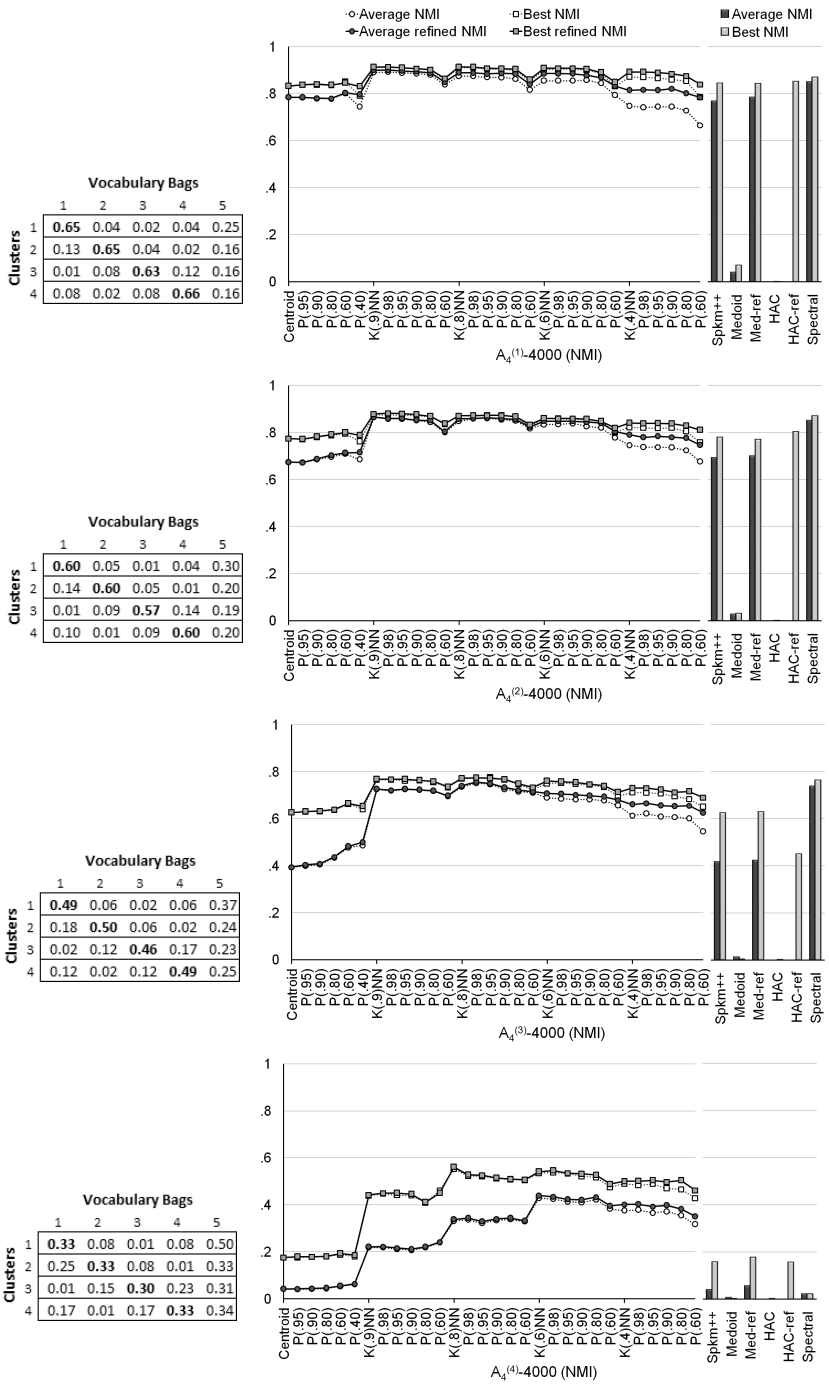


Figure 4.5: Experimental results on four artificial datasets of increasing cluster overlap, from $A_4^{(1)}$ to $A_4^{(4)}$, where the line-plots indicate the solutions of k-sp method with different parameter values. The respective results for the refined solutions are also reported.

bines efficiently the advantages of keeping a compact cluster representation and that of considering a wider set of objects around medoid for computing cluster representatives.

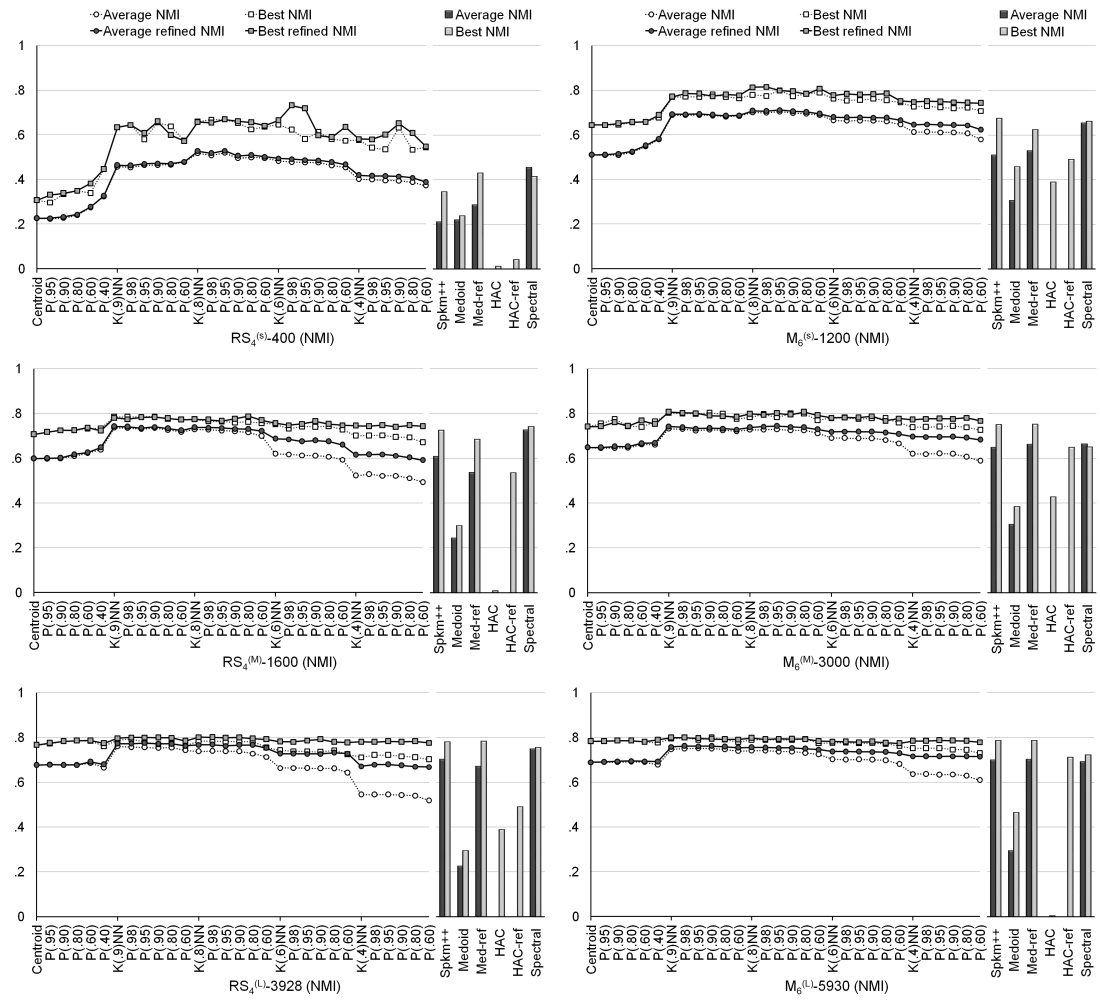


Figure 4.6: Experimental results for instances of the RS_4 and M_6 problems with different cluster sizes.

Clustering performance results

In this section, we provide experimental results using the procedure described in Sec. 4.3.5 for the datasets of Tab. 4.1 for the two sets of values $\mathrm{S}_{p_{\mathrm{obj}}}$ and $\mathrm{S}_{p_{\mathrm{feat}}}$ mentioned in Sec. 4.4.1. The results are displayed using the line-plots presented in Fig. 4.5, 4.6, 4.7. The reported '*refined*' solutions are obtained by k-sp refinement phase using centroids as cluster prototypes (see Sec. 4.3.4) on the final clusters of each of the 50 runs of basic k-sp. The bar-graphs in each row of plots present the results for sp$k$-means initialized with the k-means++ heuristic (Spkm++), k-medoids (Medoid), the refined k-medoids (Med-ref), HAC, refined HAC (HAC-ref) using sp$k$-means, and finally the spectral clustering method.

The results on artificial datasets are presented in Fig. 4.5. For a dataset of small cluster overlap, such as the $\mathrm{A}_4^{(1)}$, the performance of k-sp and spectral clustering are

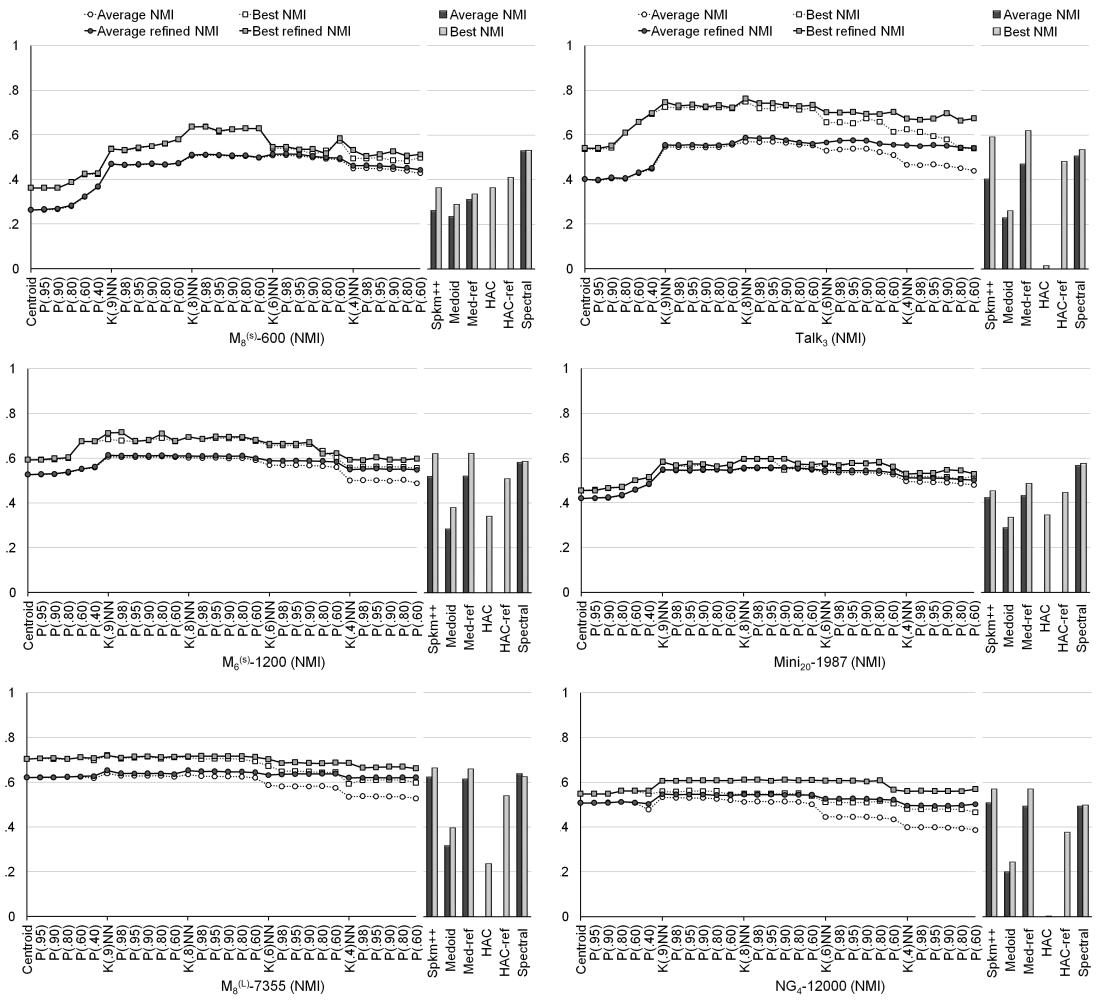


Figure 4.7: Experimental results for instances of the M_8 problem with different cluster sizes, $Talk_3$, $Mini_{20}$ and NG_4 datasets.

quite similar. However, in a more confused setting, such as the $A_4^{(3)}$ and $A_4^{(4)}$ datasets the superiority of k-sp becomes more clear. Moreover, as the overlap between clusters increases, k-sp performs significantly better than the other methods even with lower values of p_{obj} parameter (e.g. 0.6 or 0.4) where the best result is closer to the average performance of the method.

The results on real datasets that are displayed in Fig. 4.6 and 4.7 support as well the main idea of this paper. In all cases the k-sp method produced much better results than sp$k$-means. Using MedoidKNN$^{(s)}$ prototypes, the best results for larger datasets are obtained for the K(.9)NN and K(.8)NN cases. Especially for the experiments where we considered three instances of the same problem with increasing size of clusters from small to large (datasets RS_4, M_6, and M_8), it is clear that k-sp using synthetic prototypes

manages to overcome the issues arising in the case of small datasets where the number of objects per cluster is not sufficient, such as self-similarity and feature over-aggregation. The proposed refinement phase leads to even better results, while reducing the sensitivity of setting improper values for k-sp parameters. All the experimentally compared clustering methods performed better when more data objects became available for a specific problem, but the proposed k-sp remained the best among them.

By observing both curves of average and best values of the evaluation measures, we can realize the trade-off in setting k-sp parameters. When limiting the size of synthetic prototypes, k-sp avoids the bad solutions and produces much better clusterings. On the other hand, as synthetic prototypes discard too much information '*detail*' from clusters, the basic k-sp procedure becomes unable to identify the fine differences between data classes. This explains the sudden drop of the performance of K(.6) and K(.4) synthetic prototypes for medium and large datasets (e.g. $\mathrm{RS}_4^{(M)}$, $\mathrm{RS}_4^{(L)}$, $\mathrm{RS}_4^{(M)}$, $\mathrm{RS}_4^{(L)}$, $\mathrm{M}_6^{(M)}$, $\mathrm{M}_6^{(L)}$, and $\mathrm{M}_8^{(L)}$) when no refinement is applied. The information of the formed clusters can be further exploited by larger synthetic representatives in the refinement phase (where the centroids are used). Apparently, when larger synthetic prototypes are used in the main phase, the contribution of refinement turns out to be much smaller.

Tab. 4.4 summarizes the best and average performance of each method focusing on the refined solutions of k-sp, HAC, and k-medoids. Regarding k-sp, its refinement phase uses the complete feature set and centroids which, as explained in Sec. 4.3.5, enables the direct comparison of the solutions corresponding to different parameter values. The supervised evaluation measures that are presented in Tab. 4.4 correspond to the set of experiments with the maximum average value of the refined objective function determined by the procedure described in Sec. 4.3.5. The k-sp setting that provided this result in each dataset is indicated near the dataset name. The reported best refined k-sp clustering is the best solution using the latter setting of parameter values, whereas it is possible that a different parameter setting may have produced a better solution. The column *t-val* presents the t-value of the significance t-tests between the best k-sp average performance and the average performance of the other methods. For two sets of 50 experiments each,

Table 4.4: The NMI, Purity measures for the refined solutions found for each dataset. Bold values indicate the best result per column. The underlined t-values denote the cases where according to the statistical t-test k-sp appears not to be significantly better ($0<t$-val<1.999), or appears to be worse than the compared method (t-val<0).

	$\mathbf{A}_4^{(1)}$ – **k-sp: KNN(.90)-P(.98)**						$\mathbf{A}_4^{(2)}$ – **k-sp: KNN(.90)-P(1.0)**						$\mathbf{A}_4^{(3)}$ – **k-sp: KNN(.80)-P(.98)**						$\mathbf{A}_4^{(4)}$ – **k-sp: KNN(.60)-P(.98)**					
	NMI			**Purity**			**NMI**			**Purity**			**NMI**			**Purity**			**NMI**			**Purity**		
Method	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*
k-sp	**.901**	**.914**		**.968**	**.978**		**.866**	**.880**		**.960**	**.968**		**.756**	**.774**		**.916**	**.930**		**.433**	**.527**		**.740**	**.816**	
Centroid-P(.6)	.803	.846	06.46	.917	.958	03.49	.714	.801	07.25	.866	.941	05.10	.483	.665	11.84	.730	.886	08.89	.056	.193	25.05	.376	.518	24.29
spk-means	.785	.832	07.51	.909	.950	11.66	.674	.775	09.12	.847	.928	06.48	.394	.626	16.00	.668	.868	12.07	.042	.176	26.34	.357	.512	25.94
spk-means++	.768	.843	07.81	.894	.955	04.55	.692	.779	08.63	.860	.933	05.77	.416	.624	13.98	.691	.862	10.37	.038	.157	27.05	.350	.491	27.19
Medoid-ref	.784	.843	08.08	.911	.955	04.11	.699	.769	08.97	.868	.930	05.86	.423	.628	14.52	.690	.865	10.72	.055	.176	24.96	.373	.512	24.18
fwk-means	.051	.262	80.48	.366	.574	58.60	.289	.160	97.58	.334	.311	81.48	.016	.032	99.83	.314	.360	81.96	.006	.007	30.38	.286	.290	34.75
ewk-means	.131	.302	43.50	.400	.460	34.62	.073	.274	63.02	.372	.540	45.02	.032	.003	57.87	.336	.266	64.44	.009	.006	30.10	.296	.283	33.50
HAC-ref		.851			.936			.802			.936			.450			.659			.156			.418	
Spectral	.850	.869	04.86	.942	.965	02.15	.849	.869	02.05	.941	.965	02.47	.738	.763	02.41	.891	.926	02.55	.021	.021	29.01	.230	.305	32.84

Table 4.4 (continued): The NMI, Purity measures for the refined solutions found for each dataset. Bold values indicate the best result per column. The underlined t-values denote the cases where according to the statistical t-test k-sp appears not to be significantly better ($0<t$-val<1.999), or appears to be worse than the compared method (t-val<0).

Method	$\mathbf{RS}_4^{(S)}$ – k-sp: KNN(.80)-P(.95)						$\mathbf{RS}_4^{(M)}$ – k-sp: KNN(.80)-P(1.0)						$\mathbf{RS}_4^{(L)}$ – k-sp: KNN(.90)-P(.80)						$\mathbf{Talk}_3$ – k-sp: KNN(.80)-P(1.0)					
	NMI			**Purity**			**NMI**			**Purity**			**NMI**			**Purity**			**NMI**			**Purity**		
	avg	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*
k-sp	**.529**	.689		**.760**	**.875**		**.738**	**.773**		**.900**	**.926**		**.771**	**.798**		**.916**	**.935**		**.587**	**.762**		**.816**	**.935**	
Centroid-P(.6)	.277	.383	14.92	.565	.695	11.65	.625	.737	06.65	.813	.910	05.12	.691	.786	05.23	.851	.931	04.12	.431	.657	05.78	.728	.900	04.23
spk-means	.226	.307	17.97	.532	.605	13.78	.598	.706	07.93	.798	.892	05.80	.677	.766	07.18	.838	.921	04.79	.401	.540	06.97	.715	.875	04.88
spk-means++	.209	.343	18.35	.508	.623	15.08	.606	.723	08.11	.801	.899	05.97	.700	.778	04.54	.864	.926	03.36	.400	.588	06.68	.717	.823	04.54
Medoid-ref	.285	.427	15.59	.550	.675	13.77	.535	.682	12.91	.730	.876	10.88	.468	.617	04.58	.751	.916	03.34						
fwk-means	.095	.153	27.86	.420	.493	21.57	.116	.196	53.89	.448	.548	37.55	.140	.257	56.67	.470	.610	39.73	.082	.119	24.20	.510	.551	17.77
ewk-means	.134	.219	24.34	.457	.498	18.86	.219	.357	39.59	.519	.619	29.64	.248	.020	23.85	.499	.288	21.76	.174	.197	17.52	.589	.689	11.90
HAC-ref		.022			.285			.533			.680			.489			.492			.480			.734	
Spectral	.453	.413	04.99	.647	.628	07.91	.725	.740	01.19	.896	.913	00.34	.747	.754	02.77	.911	.919	00.53	.504	.533	04.18	.785	.790	02.05

Method	$\mathbf{M}_6^{(S)}$ – k-sp: KNN(.80)-P(.95)						$\mathbf{M}_6^{(M)}$ – k-sp: KNN(.90)-P(1.0)						$\mathbf{M}_6^{(L)}$ – k-sp: KNN(.90)-P(.98)						$\mathbf{Wap}_{20}$ – k-sp: KNN(.80)-P(1.0)					
	NMI			**Purity**			**NMI**			**Purity**			**NMI**			**Purity**			**NMI**			**Purity**		
	avg	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*
k-sp	**.711**	**.798**		**.808**	**.904**		**.741**	**.807**		**.835**	**.907**		**.761**	**.799**		**.861**	**.905**		.592	**622**		.658	**.696**	
Centroid-P(.6)	.552	.657	13.12	.670	.814	09.57	.667	.768	05.86	.755	.880	04.97	.693	.780	06.36	.773	.893	06.31	.556	.574	07.56	.621	.637	06.38
spk-means	.510	.644	17.05	.647	.803	11.56	.648	.741	07.36	.742	.870	12.87	.689	.782	06.85	.769	.895	06.74	.538	.544	11.35	.609	.624	08.48
spk-means++	.509	.673	15.85	.641	.831	11.34	.647	.750	07.77	.743	.876	06.12	.698	.785	06.27	.783	.899	05.92	.545	.547	11.15	.616	.609	08.00
Medoid-ref	.527	.622	14.30	.648	.759	10.55	.660	.751	06.28	.753	.876	05.00	.701	.784	05.86	.781	.887	06.17	.548	.576	11.08	.628	.643	06.06
fwk-means	.133	.186	54.59	.372	.443	37.00	.148	.188	53.78	.390	.440	36.52	.160	.241	75.31	.398	.484	51.23	.369	.357	45.34	.486	.487	31.71
ewk-means	.245	.313	49.58	.456	.475	33.78	.323	.295	40.21	.470	.377	29.86	.352	.097	54.27	.461	.271	39.68	.439	.433	09.37	.531	.535	08.52
HAC-ref		.489			.492			.647			.648			.709			.793			.527			.573	
Spectral	.652	.659	06.76	.726	.754	07.03	.662	.649	08.33	.754	.729	06.34	.690	.720	09.37	.771	.821	08.52	**.596**	.602	-1.21	**.664**	.665	-1.43

Table 4.4 (continued): The NMI, Purity measures for the refined solutions found for each dataset.

Method	$M_8^{(S)}$ – k-sp: KNN(.60)-P(.98) NMI			Purity			$M_8^{(M)}$ – k-sp: KNN(.90)-P(1.0) NMI			Purity			$M_8^{(L)}$ – k-sp: KNN(.90)-P(.98) NMI			Purity			Rev_5 – k-sp: KNN(.80)-P(1.0) NMI			Purity		
	avg	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*
k-sp	**.615**	**.642**		**.706**	**.738**		.692	**.796**		.786	**.904**		**.786**	**.839**		**.854**	**.928**		**.577**	**.676**		**.767**	**.833**	
Centroid-P(.6)	.331	.459	28.81	.511	.582	19.59	.610	.709	08.92	.704	.839	07.45	.734	.828	05.22	.795	.919	04.44	.542	.659	02.67	.745	.828	02.26
spk-means	.275	.367	33.20	.473	.528	23.71	.578	.667	12.14	.688	.812	08.88	.733	.826	05.33	.791	.919	04.74	.535	.651	02.41	.733	.822	02.31
spk-means++	.261	.361	40.56	.450	.537	30.60	.517	.619	20.07	.610	.735	15.66	.622	.661	31.82	.711	.751	20.85	.540	.663	02.42	.739	.819	02.18
Medoid-ref	.332	.445	27.61	.510	.635	18.99	.565	.697	14.13	.668	.833	10.45	.733	.817	05.75	.788	.911	05.14	.526	.653	03.00	.717	.819	03.33
fwk-means	.152	.197	56.08	.360	.400	39.85	.145	.195	59.75	.340	.420	40.94	.156	.246	72.02	.353	.473	45.20	.234	.367	20.82	.566	.700	15.16
ewk-means	.188	.288	49.45	.400	.442	35.13	.281	.279	42.86	.418	.388	32.34	.316	.279	53.99	.418	.364	36.85	.278	.078	05.57	.561	.388	02.27
HAC-ref		.302			.335			.607			.640			.664			.706			.237			.515	
Spectral	**.615**	.620	00.00	.645	.650	08.86	**.733**	.733	-5.48	**.817**	.818	-3.24	.741	.774	05.57	.832	.886	02.26	.406	.411	14.46	.664	.671	11.95

Method	NG_4 – k-sp: KNN(.90)-P(1.0) NMI			Purity			$Mini_{20}$ – k-sp: KNN(.80)-P(.90) NMI			Purity			$K1_6$ – k-sp: KNN(.90)-P(1.0) NMI			Purity		
	avg	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*	*avg*	*best*	*t-val*
k-sp	**.547**	**.607**		**.734**	**.799**		.557	**.597**		**.546**	**.603**		.701	**.802**		.834	**.897**	
Centroid-P(.6)	.510	.561	02.62	.702	.751	02.31	.459	.501	12.59	.442	.484	18.73	.690	.785	01.00	.837	.887	00.37
spk-means	.507	.548	02.73	.699	.748	02.45	.420	.454	30.07	.418	.455	22.24	.675	.725	02.52	.831	.838	01.59
spk-means++	.506	.568	02.70	.696	.755	02.69	.422	.451	30.81	.425	.446	21.99	.680	.770	02.09	.833	.883	01.30
Medoid-ref	.492	.568	03.66	.694	.756	03.00	.431	.484	28.47	.424	.484	20.94	.685	.767	01.59	.829	.885	01.90
fwk-means	.081	.152	42.94	.412	.494	28.20	.081	.152	91.60	.412	.494	64.11	.303	.454	28.07	.715	.756	13.18
ewk-means	.063	.001	27.76	.316	.253	24.93	.286	.312	08.78	.314	.308	46.82	.417	.537	16.36	.762	.827	08.00
HAC-ref		.375			.591			.444			.348			.582			.860	
Spectral	.492	.497	05.43	.711	.714	02.48	**.566**	.573	-04.04	.522	.539	04.66	**.741**	.763	-5.16	**.847**	.860	-1.87

the critical t-value is t_c=1.999 (p_c=5% for p value). This means that if the computed t-value$\geq t_c$, then the null hypothesis is rejected ($p\geq$5%, respectively), i.e. our method is superior, otherwise the *null hypothesis* is accepted indicating a marginal improvement achieved by k-sp. If the t-value is negative, k-sp performs worse than the compared method. In Tab. 4.4 the t-values ≤ 1.999 are underlined.

According to the significance t-tests, k-sp is clearly superior to the baseline methods such as sp$k$-means, sp$k$-means initialized with the k-means++ technique, k-medoids and HAC as well as their refined solutions using sp$k$-means, and the soft subspace clustering methods fw$k$-means, ew$k$-means. Compared to spectral clustering k-sp is superior in most datasets, in terms of both NMI and Purity. Spectral clustering seems to be clearly superior only for datasets $\mathrm{M}_8^{(M)}$ and $\mathrm{K1}_6$. It is also worth mentioning that the computational complexity of spectral clustering is $\mathrm{O}(N^3)$ which is significantly higher than that of k-sp. It must be also emphasized that for all datasets the best solutions were provided by the k-sp method.

Discussion

As a general conclusion about the experimental study, it turns out that the refined k-sp approach using medoidKNN with p_{obj}=.9 or .8 seems to be the best method exhibiting superior clustering performance as well as robustness in the case of small, or noisy datasets where the clusters overlap in many dimensions. High values of p_{feat} (e.g. .98 or .95) may also help in some cases. However, we explained in Sec. 4.3.5 that the user specifies only the two sets of parameter values $\mathrm{S}_{p_{\mathrm{obj}}}$, $\mathrm{S}_{p_{\mathrm{feat}}}$, and the best result can then be identified automatically by examining the values of the objective function of the refined k-sp clusterings. We should also remark that k-sp's feature selection on reference prototypes can efficiently summarize to a great extent the characteristics of the document clusters, since in most cases its application does not deteriorate the clustering performance. When p_{obj} value is kept fixed and small number of features is considered (e.g. p_{feat}=.6 that is expected to be about 20% of cluster's features, see Tab. 4.2) then, in most cases, the quality of the clusters produced using MedoidK(p_{obj})NN$^{(s)}$ is comparable to the respective results of the

respective unfiltered reference prototypes. As for the Centroid$^{(s)}$, it is the k-sp variant that mostly profits by the prototype filtering. These findings indicate the straightforward applicability of k-sp method to corpus summarization problems or offline term selection.

In both artificial and real document datasets neither the sophisticated k-means++ initialization, nor the refined k-medoids helped the sp$k$-means to discover much better clusterings. There are also cases where these methods perform equally or worse than typical sp$k$-means. For the refined k-medoids the reason for this observation is explained in Sec. 4.3.1 and is related to the inability of any data object to represent a large group of objects in HDS feature space. Thus, sp$k$-means is seeded in a little better way than Forgy's random selection. The fact that sp$k$-means++ and the refined k-medoids perform similarly implies that the probability introduced by the former in order to select objects that are far from each other may not reflect their respective *semantic distance*, since it does not take into account the special properties of text feature space, such as sparsity.

An interesting remark is that the soft subspace clustering methods tested, fw$k$-means and ew$k$-means, did not manage to provide satisfactory solutions. In Sec. 4.2.2 and Sec. 4.3.1, we reported as one of their disadvantages the fact that, by introducing explicit feature weights per cluster, the parameters to be estimated are doubled. This becomes more problematic for the very high dimensional datasets used in our experiments. It is worth mentioning that in the experiments in [120] and [121] at most 2000 features were used to represent the documents of datasets containing 2000 to 15905 objects. Apparently, this experimental setting focuses on high dimensional data but of lower scale. The very large-scale of dimensionality in our experiments seems to reveal their weakness regarding the number of parameters they use. In most cases, ew$k$-means presented better results to that of fw$k$-means with respect to the average evaluation measures. At the same time for many datasets, e.g. $\mathrm{A}_4^{(3)}$, $\mathrm{A}_4^{(4)}$, and $\mathrm{RS}_4^{(L)}$, the best clustering of ew$k$-means is evaluated to be of lower quality than the average clustering found by the algorithm. This observation indicates that the feature weight entropy term e_j introduced in Eq. 4.6 may dominate the value of the objective function. We tried to lower down the γ value without observing any improvement. This implies that the feature weight entropy may not always capture the

quality of a cluster, whereas numerical issues may also arise for the entropy computation in a HDS feature space.

4.5 Conclusions

We have proposed the k-synthetic prototypes (k-sp) clustering method that incorporates the synthetic prototypes into the spherical k-means (sp$k$-means) procedure for document clustering. Through the computation of synthetic prototypes (such as MedoidKNN) cluster-based dynamic feature selection is achieved that favors the representation of the dominant class of a cluster and enables the reassignment of the improperly clustered documents to other clusters. The proposed method is general, simple and effective and includes spherical k-means as a special case. As indicated by extensive experimental results using several datasets, the method provides robust clustering performance especially in cases of small datasets, or noisy clusters that overlap in many dimensions, and compares favorably against sp$k$-means (with Forgy's and k-means++ initialization), k-medoids, HAC, spectral clustering, and the subspace clustering methods fw$k$-means and ew$k$-means. It is remarkable that in the HDS feature spaces of the datasets we used, state of the art soft subspace clustering methods did not manage to achieve better solutions even than baseline methods such as sp$k$-means.

The proposed k-sp approach exhibits similarity to subspace clustering methods, since the introduced synthetic prototypes define different subspaces in which data classes are more distinguishable. Therefore, one could argue that k-sp in high dimensional and sparse spaces is also a subspace clustering method. To clarify the differences, we remark that many of the subspace clustering methods [120–122] construct each cluster prototype by explicitly computing weights for each dimension using all cluster objects. On the other hand, k-sp first applies object selection to construct a reference prototype (resulting in implicit feature selection), and then proceeds with optional explicit feature selection on the reference prototype. Moreover, the motivation of k-sp is to address the self-similarity

and feature over-aggregation phenomena that are very intense in the HDS feature spaces. We have also shown that the solutions obtained from the *basic k-sp phase* can be refined by the *refinement k-sp phase* using the whole feature set, which is in contrast with the traditional idea of subspace clustering.

Chapter 5

A Framework for Incremental Clustering Using Synthetic Cluster Prototypes

5.1 Introduction

In this chapter we present a framework for incremental prototype-based clustering that is based on *partial updates* on a given solution. In a partial update phase only a subset of the cluster prototypes, clusters, and data objects participate in the clustering process. Two widely known incremental clustering approaches, *global k-means* and *divisive k-means*, are revisited and unified according to this analysis. Focusing on HDS spherical data, we discuss in detail the problem of increasing the order of a current k-clustering solution by adding one new component. The idea of *synthetic cluster prototypes* presented in the previous chapter is exploited for incremental prototype-based clustering. To this end, we propose the *global k-synthetic prototypes* (*gk-sp*) clustering algorithm, which is a modification of the global k-means algorithm for HDS data. The gk-sp method uses the *k-synthetic prototypes* method for fine-tunning the k-solution and introduces a partial update scheme to setup the initializing k+1 prototypes for the refining phase. Similarly, the *global bisecting k-sp* (*gbk-sp*) method is also proposed based on bisecting k-means which, additionally to the traditional bisecting k-means, tries all splits in all clusters. Experiments on real and artificial document datasets illustrate that the proposed gk-sp method outperforms other competitive incremental and flat methods of the k-means family, in terms of clustering error and external clustering evaluation measures.

5.2 Data clustering as optimization

5.2.1 General formulation

Formally, hard clustering aims to partition the input dataset of N vectors $\mathrm{X}=\{x_i\}_{i=1}^{N}$, $x_i=(x_{i1},\ldots,x_{id})^\top\in\mathbb{R}^d$, in k disjoint sets of similar objects $\mathrm{C}=\{c_j\}_{j=1}^{k}$ called *clusters*, where c_j is a set that includes all objects assigned to j-th cluster. Typically, clustering methods try to describe the data by training a data model $\Theta^k=\{\theta_j\}_{j=1}^{k}$ consisting of k components, one for each cluster. It can be a *probabilistic model* (generative), or a *representation*

model consisting of a set of *prototypes* in the feature space. The *model complexity* refers to the number of parameters that are '*learned*' and describes the computational effort and memory space required by the clustering procedure. In general, an identical model θ_j is employed for each individual component, while the number k may differ from the number of underlying data classes denoted as κ. Thus, the *overall model complexity* can be expressed as:

$$complexity(\Theta^k) = k\, complexity(\theta). \tag{5.1}$$

Assuming a decided number of parameters for each component, the model complexity is then solely determined by the number of clusters k and referred to as *model order*.

The clustering problem can be formulated as an optimization problem; more specifically as a minimization of a given objective function $\Phi(\Theta^k|\mathrm{X})$ that measures the clustering error:

$$\hat{\Theta}^k = \underset{\theta_1,\ldots,\theta_k}{\arg\min}\, \Phi(\Theta^k|\mathrm{X}). \tag{5.2}$$

The notations $\Phi(\Theta^k|\mathrm{X})$ and $\Phi(\mathrm{C})$ describe the same objective: Θ^k can be derived from C which contains the partition of data objects, and vise-versa, while in most situations all data objects in X are involved in the computations.

5.3 Prototypes and objective functions

Without loss of generality, in this paper we primarily refer to prototype-based clustering methods. The arithmetic mean is a widely-used prototype is, also center or centroid, of the n_j cluster members: $\mu_j = \frac{1}{n_j}\sum_{x_i \in c_j} x_i$. When robustness to noise is a key requirement, the *medoid* object can be used instead which minimizes the sum of distances from all objects in a cluster: $m_j = \arg\min_{x_i \in c_j} \left\{\sum_{x_q \in c_j} \mathrm{dist}(x_i, x_q)\right\}$, where $\mathrm{dist}(\cdot,\cdot)$ is a distance measure between two vectors in feature space.

In both cases, the components of the training model $\theta_j = (\theta_{j1}, \ldots, \theta_{jd})$, $j = 1, \ldots, k$, are d-dimensional vectors and the overall complexity of such representation is $k\,d$. Moreover,

the respective objective function that is minimized is the *distortion error*, or *clustering error*:

$$\Phi(\Theta^k) = \sum_{j=1}^{k} \Phi(\theta_j) = \sum_{j=1}^{k}\sum_{i=1}^{N} w_{ij}\ \mathrm{dist}(\theta_j, x_i), \tag{5.3}$$

subject to

$$\sum_{j=1}^{k} w_{ij} = 1,$$

where w_{ij} are the object assignment variables that depend on Θ^k. Especially for hard clustering, it is w_{ij}=1 iff x_i is assigned to cluster c_j. If the sum of squared Euclidean distances (SSE) is used between the objects of the clusters and their respective prototypes, $\mathrm{dist}(\theta, x)$=$\|\theta - x\|_2^2$, the resulting Φ_{SSE} is called *squared Euclidean error* or *distortion* (see Sec. 2.6.1).

When the $1 - Cosine$ is considered as distance function and if we use the centroid[1] $\bar{\theta}_j$=$\theta_j / \|\theta_j\|_2$, then the respective clustering objective is the complementary of Cohesion (CC):

$$\Phi_{\mathrm{CC}}(\bar{\Theta}^k) = N - \sum_{j=1}^{k}\sum_{i=1}^{N} w_{ij}\ \bar{\theta}_j^{\top} x_i. \tag{5.4}$$

This reduces to $N - \sum_{j=1}^{k} \|s_j\|_2$, where s_j=$\sum_{x_i \in c_j} x_i$.

It must be noted that the hard clustering problem may also be reformulated in terms of non-convex, non-smooth optimization [129, 130]:

$$\Phi(\Theta^k) = \sum_{i=1}^{N} \min_{j=1,\ldots,k} \mathrm{dist}(\theta_j, x_i). \tag{5.5}$$

Instead of a sum of all individual cluster errors $\Phi(\theta_j)$ as in Eq. 5.3, this function considers a sum of minima functions of the representation error of each data object which makes it non-smooth (not differentiable everywhere). Eq. 5.5 is the continuous analog of Eq. 5.3, hence, they are equivalent in the sense that one's local minimizer is also a minimizer for the other as well as they both have the same global minimizers.

In the previous chapter, the *synthetic cluster prototypes* (*sp*) have been proposed for

[1]The term *centroid* may imply the mean or the L_2-normalized mean depending on the context, and similarly for the notations θ, Θ when denoting prototypes.

Algorithm 3 – The generic k-means clustering algorithm

input: dataset X, the number of clusters k
1. *Initialize prototypes*: usually at random, or using more complex procedures.
2. *Assignments update*: each object is assigned to the cluster represented by the closest prototype to that object
3. *Prototypes update*: the prototypes are recomputed
4. *Stopping criterion*: if (important) changes in prototypes are observed then goto step 2.

output: (the prototypes Θ^k, and the partition C^k)

representing clusters of spherical data, such as text documents. As already mentioned in Chapter 4, centroid and medoid constitute special cases of *sp*. Moreover, considering θ_j=sp_j in Eq. 5.3 or Eq. 5.4, we get a general expression of the objective function that describes the representation quality of *sp*.

5.4 Flat prototype-based clustering algorithms

k-means and k-medoids are flat clustering algorithms which originally minimize the sum of squared Euclidean distances between the objects of the clusters and their respective prototypes, denoted as Φ_{SSE} (Eq. 5.3). Spherical k-means (sp$k$-means) [41] is a modification designed for spherical data where it has been shown to be more effective than original k-means version. Spk-means uses L_2-normalized centroid prototypes and minimizes Φ_{CC} (Eq. 5.4). The number of clusters k is provided in advance. The generic k-means clustering procedure presented in Algorithm 3 improves iteratively the solution.

This procedure performs in a gradient descend fashion wrt the minimization of the objective function [131] since both update steps are optimal: i) the assignment step follows the *nearest neighbor rule* and ii) the prototype used is the arithmetic mean, or the medoid respectively for k-medoids, of the cluster members which are the optimal representatives under the respective constraints. In this way, the iterative reduction of the *representation error* for each data object is achieved. From an optimization point of view, the partitional clustering approach is formulated by Eq. 5.2 with the additional remark that it seeks for all components $\theta_1, \ldots, \theta_k$ simultaneously until convergence.

The competition between prototypes for representing the data results in a locally

optimal solution where the prototypes are set at positions with high data density. However, the quality of the final solution depends heavily on the initial prototypes. Common deficiencies are the data under-representation where no prototype is assigned to every underlying data class. At the same time, another data class may be over-represented by more than one prototype representing different subsets of its objects.

The problem of trapping into poor local solutions is tackled with the careful selection of the initial prototypes (see Sec. 2.6.1). The most simple initialization approach is the random selection of k objects from the dataset (Forgy's approach). However there exist more efficient options [80] and some of them have linear complexity cost to the number of data O(N). Initialization using density estimation, with kd-tree for low-dimensional feature spaces [132] or based on neighborhoods around objects [133, 134], have also been proposed. In addition, other clustering methods may also be used in order to produce an initial partition that is further refined by k-means. Such examples are agglomerative clustering [42], genetic algorithms [94, 95], and simulated annealing [96].

Improving the convergence monitoring of k-sp. The k-synthetic prototypes clustering method (k-sp), which has been presented in Chapter 4, incorporates the synthetic prototypes as cluster representatives in the above iterative local search strategy. The algorithm mainly uses two parameters for object and feature filtering ($p_{\text{obj}}, p_{\text{feat}}$), however without loss of generality, we use only object filtering herein. The *sp* construction reduces the representation error in clusters during iterations, whereas it does not guarantee the monotonic convergence. In Sec. 4.3 we used monitoring of the overall clustering error to stop the procedure when an increase occurs. Here we propose a relaxed monitoring mechanism which lets k-sp to continue iterating, when error increase is observed, for a number of *deteriorating steps* (ds) seeking for a state that has lower error than the best found so far. If during the ds steps no improvement is achieved, the algorithm rolls back to the best solution found and terminates. If, however, a better solution is found, then the algorithm proceeds from that point on, and resets the counter of the deterioration steps (ds iterations will be permitted in case another error deterioration will be observed).

Algorithm 4 – An abstract incremental clustering procedure

input: dataset X, a partition $C^{k_{\text{init}}}$ of k_{init} clusters, and the desired number of clusters κ
1. *Initialize* with model $\Theta^{k_{\text{init}}}$ computed for the input data partition $C^{k_{\text{init}}}$ and set $k{=}k_{\text{init}}$
2. *Improve model structure* by increasing the number of components (incremental step)
3. *Improve model parameters* using a clustering algorithm for k clusters (fine-tuning step)
4. Stopping criterion: if $k{\neq}\kappa$ then goto step 2

output: $(\Theta^{\kappa}, C^{\kappa})$

The main consideration of this modified strategy is that a temporary deterioration of the clustering may avoid a bad local optimum and lead to some better area of the search space. Notably, this search approach resembles, in some sense, to discrete optimization algorithms such as simulated annealing that uses the concept of '*temperature*' for probabilistic decisions. The proposed mechanism is simpler to that approach, but in both cases the roles of control parameters ds and temperature are similar. In addition, they provide a way to avoid trapping in oscillation between solutions. In empirical experiments $ds{=}5$ was shown to be a good setup value.

After termination, the refinement step of k-sp takes place where the cluster centroids are updated to further improve the clustering result. In fact the improvement of this step may empirically confirm the rationale behind k-sp: once the important '*coarse*' information has been extracted, fine-grained details can be used to improve clustering (e.g. objects far from prototypes, or features without much discriminating power).

5.5 Incremental prototype-based clustering algorithms

Incremental clustering starts with k_{init} given clusters and works in a top-down manner until the desirable $\kappa{>}k_{\text{init}}$ clusters have been formed. This is outlined in Algorithm 4. If $k_{\text{init}}{=}1$ then $C^{k_{\text{init}}}{=}X$ containing all data objects. Nevertheless, it is implied that the initial model is computed for the provided partition $C^{k_{\text{init}}}$ that could have been produced by a different clustering procedure. Incremental strategy has higher computational cost comparing to flat clustering, but important advantages as well:

i) it reduces the deficiencies of flat clustering that tries to locate good starting pro-

totypes for all clusters at once, which is inefficient especially in the case where the number of clusters is large,

ii) is able to extract the cluster structure information of the dataset in different refining levels, and

iii) is convenient to be combined with cluster-based criteria that enable the estimation of the optimal number of clusters (for example the methods [32, 45, 135–137] discussed in Chapter 6).

This quite generic methodology is the backbone of popular incremental algorithms that we present in what follows.

5.5.1 Divisive prototype-based clustering

In the *incremental step* of divisive prototype-based clustering (DPC) [83], one of the model components, let θ_s, is selected based on a criterion (e.g. cluster variance, or size, etc.) and is replaced by two new components θ_{s_1}, θ_{s_2}. The new components aim to better represent the objects that were previously represented solely by θ_s. Following an abstract notation, if $\hat{\Theta}^{k-1}$ is the locally optimal solution already computed, then the fine-tunning of θ_{s_1}, θ_{s_2} which is carried out in the bisection (split) step of c_s is:

$$\Theta^k = \underset{\theta_{s_1},\ \theta_{s_2}}{\arg\min}\, \Phi(\hat{\Theta}^{k-1}\backslash\theta_s,\ \theta_{s_1}, \theta_{s_2} \mid \mathrm{X}_s) \tag{5.6}$$

$$= \left\{ \hat{\Theta}^{k-1}\backslash\theta_s,\ \underset{\theta_{s_1},\ \theta_{s_2}}{\arg\min}\, \Phi(\theta_{s_1}, \theta_{s_2} \mid \mathrm{X}_s) \right\}, \tag{5.7}$$

where $\hat{\Theta}^{k-1}\backslash\theta_s$ is the set-theoretic subtraction of θ_s from $\hat{\Theta}^{k-1}$, and $\mathrm{X}_s{=}\{x_i : x_i{\in}c_s\}$ is the subset of data objects assigned to the selected cluster which is split. Notably, the resulting solution of Eq. 5.7 is not locally optimal as a whole, for all k components. Intuitively, this implies that an application of k-means on that partition would generally update those components and clusters. Contrary, $\hat{\Theta}^{k-1}$ is optimal, which implies that a refinement has already been applied on all components. Furthermore, the factor

$[\arg\min_{\theta_{s_1},\,\theta_{s_2}} \Phi(\theta_{s_1}, \theta_{s_2} \mid \mathrm{X}_{c_s})]$ denotes the local search involving the two components and the objects of cluster c_s. Usually, 2-means is used to split a cluster, starting from two adequately diverse positions in cluster. More specifically, given that θ_{s_1} is seeded by an arbitrary data object chosen at random from the cluster, then θ_{s_2} can be obtained with one of the following approaches:

1) select randomly,
2) use of the previous cluster prototype, i.e. θ_{s_2}=θ_s,
3) find a position at opposite direction with respect to the current prototype,

$$\theta_{s_2} = \begin{cases} \theta_s - (\theta_{s_1} - \theta_s), & \text{for Euclidean distance} \\ 2\psi_{(\theta_{s_1} \perp \bar{\theta}_s)} - \theta_{s_1}, & \text{for Cosine similarity} \end{cases} \tag{5.8}$$

where $\psi_{(\theta_{s_1} \perp \bar{\theta}_s)} = \bar{\theta}_s \left[1 + (\theta_{s_2} - \bar{\theta}_s)\, \bar{\theta}_s^{\top}\right]$ the intersection of the perpendicular vector from θ_{s_1} onto the normalized $\bar{\theta}_s$.

A number of trials may be required to determine a good split. The first approach is the most naive, while the third one is expected to speed up 2-means convergence. However, the result of (3) would be quite similar to that of setting θ_{s_2} to be the previous prototype θ_s, because both points would lay on the same direction in space wrt θ_{s_2}. Another interesting remark, especially for HDS data, is that even though θ_{s_1} might be seeded with a sparse data vector, however, the initialization cases (2) and (3) do compute a non-sparse θ_{s_2}.

Alternatively, principal direction divisive partitioning (PDDP) [82] tries to split a cluster along the direction of higher data variance. Originally, it was proposed as a deterministic procedure which splits according to the positions of objects wrt the line perpendicular to the principal direction of the cluster, which also passes by the cluster mean (i.e. the sign of the projection). However, it is easy to realize that this could be used in a non-deterministic way as well: i) consider the object projections on the principal direction and then ii) follow one of the aforementioned approaches (2) or (3). This would avoid the arbitrary choice of the mean as cluster split point. In [138] k-means was used as a steering procedure aiming to refine the 2-clustering of PDDP split. Other works have

proposed to consider projection directions other than the principal, while in [139] the distribution of the data projections on the principal direction is further studied in order to determine a good cluster split.

5.5.2 Global k-means

In the incremental step, gk-means algorithm [84] only adds one component to the pre-existing $\hat{\Theta}^{k-1}$. Specifically, it makes N trials to add the new component, where, in each of the trials, the component is initialized at a position coinciding with a different *object seed*[2]. Let x_i be the object seed in a trial, then the refinement consists of a typical k-means run on all clusters:

$$\hat{\Theta}^k = \underset{\theta_1,\ldots,\theta_k}{\arg\min}\, \Phi(\hat{\Theta}^{k-1}, \theta_k = x_i \mid \mathrm{X}). \tag{5.9}$$

Among the N solutions corresponding to different seeds, that one with minimum objective function value is kept as the optimal k-clustering. For the k=1 case, the algorithm uses the arithmetic mean of the dataset. Global k-medoids is similar and only differs in an additional constraint that is considered: the prototypes to be cluster medoids. The drawback of gk-means is the heavy computational cost, since N runs of k-means (or k-medoids, respectively) are required to add one new cluster.

A variation that reduces the computational burden of gk-means exhaustive search is the fast global k-means (fg$k$-means) [84]. It introduces an estimation of the improvement in clustering error which is computed for all object seeds, but without involving any prototype re-computation. Given the solution $\hat{\Theta}^{k-1}$ and the respective clustering error value $\Phi(\hat{\Theta}^{k-1})$, then an upper bound of the error $\Phi(\hat{\Theta}^k)$ is computed for each seed such that:

$$\Phi(\hat{\Theta}^k(x_i)) \;\leq\; \Phi(\hat{\Theta}^{k-1}) - b_i^{k-1}, \tag{5.10}$$

where $\hat{\Theta}^k(x_i)$ is the local optimum discovered when $\{\hat{\Theta}^{k-1}, \theta_k = x_i\}$ was set as initial con-

[2] '*Seed*' is a more general term and may refer to any starting position in $\mathbb{R}^d$, including those points coinciding with a data object which are called *object seeds*.

Algorithm 5 – Find the starting state for k-th cluster prototype (mfg$k$-means)

input: dataset X, the locally optimal prototypes $\hat{\Theta}^{k-1}=\theta_1,\ldots,\theta_{k-1}$
1. *Seed initialization*: for each object seed x_i, let $\theta_k(x_i)=x_i$
2. *Prototype computation*: find $c_k=\mathrm{S}(x_i)$ (see Eq. 5.13) and recompute $\theta_k(x_i)=Centroid(c_k)$
3. *Find the best case*: $\theta_k=\arg\min_{i=1,\ldots,N}\ \phi^k(\theta_k(x_i))$
4. *Fine-tuning*: repeat until no change occurs in cluster c_k
   a. $c_k=\mathrm{S}(\theta_k)$
   b. $\theta_k=Centroid(c_k)$

output: θ_k

ditions. Formally, for $k>1$, the quantity b_i^{k-1} is computed as:

$$b_i^{k-1}=\sum_{q=1}^{N}\max\Big\{0,\ \delta_q^{k-1}-\mathrm{dist}(x_q,x_i)\Big\}, \tag{5.11}$$

where $\delta_q^{k-1}=\min_j\{\mathrm{dist}(\hat{\theta}_j^{k-1},x_q)\}$ denotes the distance between x_q and its nearest prototype. The object with maximum b_i is expected to provide the largest decrease in error and, thus, is selected to seed the k-th cluster prototype.

Another perspective to this approach is to reformulate Eq.5.11 as an optimization of an *auxiliary objective function*, in the spirit of Eq. 5.5 but with components $\theta_1,\ldots,\theta_{k-1}$ at fixed positions [140] and constrained θ_k to be a data object:

$$\phi^k(\theta_k)=\sum_{q=1}^{N}\min\Big\{\delta_q^{k-1},\mathrm{dist}(\theta_k,x_q)\Big\}, \tag{5.12}$$

$$\text{subject to } \theta_k\in\mathrm{X}.$$

The modified fast global k-means (mfg$k$-means) [140] is a more efficient alternative compared to fast gk-means, with the disadvantage of increased computational cost. The difference lies in the way the starting point of the new cluster prototype is determined. All objects are tested as seeds at each incremental step, but an intermediate procedure has been included which computes the final initial state for the added component. This approach aims to minimize the auxiliary objective of Eq. 5.12 without putting constraints on the input, hence $\theta_k\in\mathbb{R}^d$. Given an arbitrary point $y\in\mathbb{R}^d$, let us denote the set of objects that are nearest to y than to the prototypes of the clusters to which they are currently

assigned:

$$\mathrm{S}(y) = \left\{ x_q : \mathrm{dist}(x_q, y) < \min_{j=1,\ldots,k} \mathrm{dist}(x_q, \theta_j) \right\}. \tag{5.13}$$

The respective algorithm to find the starting state of k-th component is presented in Algorithm 5. Thinking in terms of the k-means procedure, it is clear that fg$k$-means first lets the objects move to c_k and then computes only once the new value of the objective function (step 2 of Algorithm 4). Mfg$k$-means proceeds further and recomputes the prototype θ_k by taking into account the newcomer objects (step 3 of Algorithm 4). Next, the best case is further refined by updating only the prototype of the new cluster. At the end of this procedure, all objects are assigned to the cluster whose prototype is the nearest to them and, since the centroid θ_k is the optimal representative for the objects in c_k, the returned θ_k is a local minimizer of ϕ^k defined in Eq. 5.12.

If short running time is not the primary concern, we propose a straightforward alteration denoted as mg$k$-means (i.e. not fast) that could lead to a better starting point by just inverting the order of steps 3 and 4 in Algorithm 5. This approach selects the best starting point after refining all $\theta_k(x_i)$ produced in step 2 of Algorithm 5. To the best of our knowledge, this modification has not been proposed in the related literature. Note that this can also contribute in the experimental evaluation of the algorithms seeking for a good starting position for the new component, because all starting states are refined.

The speed up of global k-means procedure and its variations may be achieved by:

1) the improvement of the computation of the starting point, given an object seed. This could also lead to faster refinement of all k components that follows.
2) the utilization of early stopping criteria that would identify the non-interesting candidates during their examination in the clustering procedure. Since all variations start from an object seed, we could focus on those seeds that seem more promising to drastically reduce the clustering error.
3) the pruning of the candidates set. Similarly to the previous case, this could discard candidates that lay at non-interesting areas of the data space (e.g. close to existing components). Random pruning is also an option, although a naive one.

4) the introduction of $r>1$ components at each incremental step. In practice, this is prohibitive to be applied in a greedy fashion because the object combinations to be tested would be $\binom{N}{r}=\frac{N!}{r!\,(N-r)!}$.

The variations discussed so far belong to the categories (1) and (2). There have been proposed other modifications of mfg$k$-means that mainly aim to reduce the required computational resources by adopting the directions (3) and (4) of the above. In [141] the projection of data in the eigenspace is proposed, which enables the efficient identification of the nearest neighbors of an object without storing the whole similarity matrix. Moreover, an algorithm is presented that introduces multiple new components in each step. On the other hand, the approach of [142] reduces the computational complexity by examining only a subset of object seeds laying at different areas of the dataset. This is achieved by considering a finite set of weights $\mathrm{U}=\{\upsilon_i\}_{i=1}^{j}$, $\upsilon_i\in\mathbb{R}^+$, and by altering Eq. 5.13 to:

$$\mathrm{S}^{\upsilon_i}(y)=\left\{x_q : \upsilon_i \operatorname{dist}(x_q,y) < \min_{j=1,\ldots,k} \operatorname{dist}(x_q,\theta_j)\right\}. \tag{5.14}$$

Similarly, Eq. 5.12 is rewritten to consider the weight υ_i that forces the minimization to focus on different parts of the dataset (although that, according to the above definition, these parts are nested). For instance, for a small υ_i value the examined seeds would be objects near the existing prototypes. The best solution found using the different υ_i weights is the final starting point for the k-th component.

5.6 A framework for incremental clustering

This section presents a framework for incremental prototype-based clustering. The proposed framework is generic and exploits the idea of *partial updates* described next.

Definition 1 – Partial update (PU): *A partial update is a local search clustering procedure which starts from a solution* (Θ^k,C^k) *and is constrained to perform at most* t *iterations involving the following three subsets of entities:*

Algorithm 6 – The incremental clustering framework

input: dataset X, a partition $\mathrm{C}^{k_{\text{init}}}$ of k_{init} clusters, the number of desired clusters κ

1. *Initialization* with model $\Theta^{k_{\text{init}}}$ computed for the input data partition $\mathrm{C}^{k_{\text{init}}}$ and set $k{=}k_{\text{init}}$.
2. *Incremental step*: the model complexity is increased ($k{=}k{+}1$) and a proper initialization is determined for the new component(s):

$$\Theta^{k,\text{INC}} = \left\{ \hat{\Theta}^{k-1} \backslash \theta^{k}_{(-)},\ \theta^{k}_{(+)} \right\}, \tag{5.15}$$

where $\theta^{k}_{(-)} \subseteq \hat{\Theta}^{k-1}$ a set of components to be removed from the provided previous (k-1)-solution, and $\theta^{k}_{(+)}$ the set of components to be introduced, each one initialized at a starting position of interest.

3. *Partial updates*: a series of $m{-}1$ successive intermediate steps, where each PU_i is seeking for a good starting state for the next PU_{i+1}, based on the solution of the previous PU_{i-1} and by performing at most t iterations. Let $\mathrm{PU}_0 \rightarrow \Theta^{k,0}{=}\Theta^{k,\text{INC}}$ (the output of step 1), then the general expression for each PU_i, $i{=}1,\ldots,m{-}1$, is:

$$\Theta^{k,i} = \underset{\theta_j \in \partial^{k,i}_{\text{act}}}{\arg\min}\, \Phi(\Theta^{k,i-1} | \, \mathrm{X}^{k,i}_{\text{act}}) \tag{5.16}$$

$$= \left\{ \Theta^{k,i-1} \backslash \partial^{k,i}_{\text{act}},\ \underset{\theta_j \in \partial^{k,i}_{\text{act}}}{\arg\min}\, \Phi(\partial^{k,i}_{\text{act}} \,|\, \mathrm{X}^{k,i}_{\text{act}}) \right\}, \quad \text{subject to } \mathcal{C}^{k}_{\text{act}}, \tag{5.17}$$

where $\partial^{k,i}_{\text{act}} \subseteq \Theta^{k,i}$ the active components, $\mathrm{X}^{k,i}_{\text{act}}$ the objects that are let to change cluster assignment, and $\mathcal{C}^{k}_{\text{act}}$ the active clusters. PU_i should be less constrained comparing to any PU_j, $j{<}i$, wrt the number of active components, clusters, and data objects.

4. *Full update* (FU): k-clustering refinement is used as FU which can also be considered as PU_m with no constraints ($\mathcal{C}^{k,m}_{\text{act}}{=}\mathrm{C}^{k,m-1}$, $\partial^{k,m}_{\text{act}}{=}\Theta^{k,m-1}$, and $\mathrm{X}^{k,m}_{\text{act}}{=}\mathrm{X}$). It finds a locally optimal solution starting from the output of the last PU_{m-1}, $\Theta^{k,m-1}$, and by updating all k clusters and their prototypes:

$$\hat{\Theta}^{k} = \underset{\theta_1,\ldots,\theta_k}{\arg\min}\, \Phi(\Theta^{k,m-1} | \mathrm{X}). \tag{5.18}$$

5. Stopping criterion: if $k{\neq}\kappa$ then goto step 2.

output: $(\Theta^{\kappa}, \mathrm{C}^{\kappa})$

i) *the active clusters* $\mathcal{C}^{k}_{\text{act}} \subseteq \mathrm{C}^{k}$ *that compete to each other to gain new object members. The rest* $\mathrm{C}^{k} \backslash \mathcal{C}^{k}_{\text{act}}$ *clusters may only lose or take back objects they had prior to PU.*

ii) *the active objects* $\mathrm{X}^{k}_{\text{act}} \subseteq X$ *that are let to move to and among the active clusters, or return back to the cluster they were assigned before PU.*

iii) *the active components* $\partial^{k}_{\text{act}} \subseteq \Theta^{k}$ *corresponding to a subset of active clusters and they are updated when the respective cluster changes.* ∎

Let us first define a function $\mathrm{J}(\cdot)$ that returns the cluster(s) related to the input parameter, thus, $\mathrm{J}(x_i, x_q)$ would return the clusters to which these objects are assigned, $\mathrm{J}(c_i)$ would return the index of the cluster i and the same for $\mathrm{J}(\theta_i)$. Some remarks follow, regarding the PU definition:

- If the similarity of all objects to the cluster they belong has been stored before

the PU, then the only prototypes that are involved in calculations of similarities to objects are those corresponding to the active clusters that aim to attract new objects, i.e. $\mathrm{J}(\mathcal{C}^k_{\mathrm{act}})$. Otherwise, the involving prototypes will be those corresponding to the clusters indexed by $\mathrm{J}(\mathrm{X}^k_{\mathrm{act}}) \supseteq \mathcal{C}^k_{\mathrm{act}}$.

- Generally $\mathrm{J}(\partial^k_{\mathrm{act}}) \subseteq \mathcal{C}^k_{\mathrm{act}}$ holds, which means that the prototypes of some of the competing clusters might be adaptive to the changes in object members, while others might be static. A cluster c will not participate in the update procedure at all, only if $\mathrm{J}(c) \notin \mathcal{C}^k_{\mathrm{act}}$ and $c \notin \mathrm{J}(\mathrm{X}^k_{\mathrm{act}})$.
- The resulting PU solution is also locally non-optimal in the sense that a k-means run would further update both clusters and prototypes.

Algorithm 6 provides the formulation of the framework which summarizes effectively the popular incremental clustering algorithms and most of their variations mentioned in Sec. 5.5. The initialization is identical to that discussed for Algorithm 4. The incremental step is specified by the two sets $\Theta^k_{(-)}$, $\Theta^k_{(+)}$ and the initialization of the introduced components in set $\Theta^k_{(+)}$. By definition, a PU is specified by the respective sets $\mathcal{C}^k_{\mathrm{act}}$, $\mathrm{X}^k_{\mathrm{act}}$, and $\partial^k_{\mathrm{act}}$ that can be computed based on the provided partition and the respective prototypes that should be partially updated.

Table 5.1 presents the parameter setup that reduces the generic framework procedure to each of the incremental algorithms discussed earlier in Sec. 5.5. The first row refers to DPC that in the incremental step removes the θ_s component and adds two new components, θ_{s_1}, θ_{s_2}. Those two are seeded using objects belonging to the cluster that is split, i.e. $x_i \in c_s$. Then in the PU step, it is noted that it applies only PU_1 where the only updated prototypes are the two newly introduced θ_{s_1}, θ_{s_2}, and that the clusters that are changing their members are those corresponding to the updated prototypes. Finally, the data objects that participate in the PU are the members that initially belong to c_s that is split. The second row is the DPC initialization where only one component is added while the existing prototype is used to perform the cluster split. Next we have the gk-means variations, which do let all the objects of X to be reassigned among the clusters. As

Table 5.1: Different parameter setups that reduce the generic clustering framework procedure to popular incremental algorithms.

	Incremental step			**Partial update step**				
Algorithm	$\Theta^k_{(-)}$	$\Theta^k_{(+)}$	init $\Theta^k_{(+)}$	PU	∂^k_{act}	$\mathcal{C}^k_{\text{act}}$	X^k_{act}	t
DPC	θ_s	$\{\theta_{s_1}, \theta_{s_2}\}$	x_i,$x_q \in c_s$	PU_1	$\{\theta_{s_1}, \theta_{s_2}\}$	$\{c_{s_1}, c_{s_2}\}$	$\text{X} \cap c_s$	max
DPC ($\theta_{s_2} = \theta_c$)	$\emptyset$	θ_{s_1}	$x_i \in c_s$	PU_1	$\{\theta_{s_1}, \theta_{s_2} = \theta_s\}$	$\{c_{s_1}, c_{s_2}\}$	$\text{X} \cap c_s$	max
gk-means	$\emptyset$	θ_k	$x_i \in \text{X}$	FU	Θ^k	C^k	X	max
fg$k$-means	$\emptyset$	θ_k	$x_i \in \text{X}$	PU_1	$\emptyset$	c_k	X	1
mg$k$-means	$\emptyset$	θ_k	$x_i \in \text{X}$	PU_1	θ_k	c_k	X	max
mfg$k$-means	$\emptyset$	θ_k	$x_i \in \text{X}$	PU_1	θ_k	c_k	X	1
				PU_2	θ_k	c_k	X	max
gk-sp	$\emptyset$	θ_k	$x_i \in \text{X}$	FU	Θ^k	C^k	X	max
fg$k$-sp	$\emptyset$	θ_k	$x_i \in \text{X}$	PU_1	$\emptyset$	c_k	X	1
gk-sp-mPU	$\emptyset$	θ_k	$x_i \in \text{X}$	PU_1	$\{\theta_j\text{: } \theta_j \sim x_i\}$	$\{c_j\text{: } \theta_j \sim x_i\}$	X	max
fg$k$-sp-mPU	$\emptyset$	θ_k	$x_i \in \text{X}$	PU_1	θ_k	$\{c_j\text{: } \theta_j \sim x_i\}$	X	1
				PU_2	$\{\theta_j\text{: } \theta_j \sim x_i\}$	$\{c_j\text{: } \theta_j \sim x_i\}$	X	max

(1) t=max implies no constraint on the number of iterations; the PU terminates upon convergence.
(2) $\{\theta_j\text{: } \theta_j \sim x_i\}$ denotes the set of prototypes that have similarity with the object seed x_i to some required extend. Respectively for active clusters $\{c_j\text{: } \theta_j \sim x_i\}$.

indicated, the original algorithm applies directly the full update (FU), i.e. unconstrained PU where all clusters are competitive and all prototypes are updated (FU is applied by all algorithms but we mention it only for the methods that do not use a constrained PU). For fg$k$-means the constraint of t=1 iteration is noted for the PU step. However, this is not an direct constraint but rather because its PU does not permit the update of any prototype, and therefore only one iteration is possible. On the contrary, t=1 constitutes an explicit constraint of mfg$k$-means. The last four lines refer to the proposed global k-synthetic prototypes clustering method that is discussed in Sec. 5.8 that follows.

For clarity of presentation, not all algorithms have been included in the table because they are similar to those mentioned. PDDP-based algorithms work with data projection instead of the original space, while in all other aspects would use the same parameters as DPC. Regarding the global k-means variations [141] and [142], they use careful selection of the seeds of $\Theta^k_{(+)}$ from a subset of X without focusing on any cluster. Besides, the adopted PU approach is the same with that of mfg$k$-means. As for the variations that introduce r new components at each step [141], they can be directly derived by setting $\Theta^k_{(+)} = \{\theta_k, ..., \theta_{k+r}\}$.

5.7 Principles for efficient incremental clustering for HDS data

The design of an effective incremental clustering algorithm for HDS feature spaces can be based on the proposed framework of Sec. 5.6. The parameter setup and the local search of the PU step should take into account the special properties of such spaces.

Self-similarity and feature over-aggregation. It has already been reported that these two phenomena affect negatively the flat prototype-based clustering procedure (specifically, centroids or medoids, see also Sec. 4.3.1). The first implies that there is a tendency for an object to remain in the cluster where it already belongs to, because the self-similarity may dominate the nearest cluster prototype calculation. The second phenomenon implies that, due to the large number of dimensions, the prototypes encounter difficulties in becoming specialized in one data class when the initialization provides impure clusters. In contrast, they aggregate information (non-zero weighted features) from many classes. Both phenomena play more important role when the clusters become smaller in size, which is the case in incremental clustering as k grows. In the flat clustering case, it has been shown that such phenomena can be addressed to a satisfactory extend using the synthetic prototypes approach for cluster representation.

Unfair prototype competition. Prototype-based clustering is a competitive learning process where the prototypes compete with each other for representing the data. In what concerns incremental clustering in HDS space, where a new component is added to a model of lower order, a special problem emerges at the beginning of a PU regarding the '*unfairness*' of the competition between the already formed $k-1$ components, and the new one. This is due to the large difference in sparsity between the object seed, and the existing prototypes that have aggregated a lot of information from their clusters. Self-similarity makes more difficult for arbitrary objects to join the new cluster. Additionally, since a single sparse vector object (or a very small set of objects) is usually inappropriate to represent a data group (see Sec. 4.3.1), it is reasonable to be also inappropriate to attract a coherent group of objects from the other clusters, if chosen to seed the new prototype. Note that, as k grows incrementally, the formed clusters become smaller and

less impure. Consequently, the respective prototypes overfit to current partition and make more difficult for the new cluster to drastically change the solution found so far.

Intuitively, an approach that would mitigate this *unfair prototype competition problem* should try to reduce the *information imbalance* between the formed prototypes θ_i, i=1, ...$k-1$ and the new one θ_k. And this could be achieved by:

1) a *reduction* of the representation quality/accuracy of the formed prototypes for their clusters which could *increase its sparsity*, and/or
2) an *enrichment* of the new starting point θ_k which would necessarily *decrease the sparsity* of the considered object seed.

To the best of our knowledge, it is the first time such analysis and discussion is provided in the context of prototype-based incremental clustering and, thus, the related approaches of literature do not tackle the above issues. Only the initialization case (1) of DPC (see Sec. 5.5.1) seems to avoid such problems: initially, two object seeds fairly compete to split one cluster. Next, the two fine-tuned centroids, which are much less sparse than the object seeds, are used in refinement of all clusters. In initialization cases (2) and (3), θ_{s_1} is an object seed but the computed θ_{s_2} is much less sparse. In fact, θ_{s_2} is always as sparse as the cluster centroid θ_s used for its computation. As for gk-means, it lets a sparse object seed to compete directly with the k-1 cluster prototypes, which has the previously discussed disadvantages.

On the other hand, mg$k$-means is the only algorithm that enhances the competitiveness of the new prototype θ_k, even though indirectly. While updating θ_k in the PU step, the rest of prototypes do not adapt to represent better the remaining members of their clusters and hence the probability to lose more members during next PU iterations increases. However we should note that this is not an intentional property of the method, because mg$k$-means has not been designed for HDS data spaces that we discuss here. Therefore, the aim of its PU is rather to speed up the original gk-means algorithm.

Algorithm 7 – Initialization and modified partial update of the fg$k$sp-mPU method.

input: dataset X, the locally optimal $\hat{\Theta}^{k-1}$=$\theta_1,\ldots,\theta_{k-1}$, the k-sp parameters p_{obj}, $p_{\text{feat}}, \lambda, \beta, ds$

1. *Seed initialization*: for each object seed x_i, let $\theta_k(x_i)$=x_i
2. *Reduction of old prototypes*: for each cluster j=1, ..., k−1, θ_j=$ConstructSP\,(c_k,\, p_{\text{obj}},\, p_{\text{feat}},\, \lambda, \beta)$
3. *New prototype computation*: find c_k=S(x_i) (see Eq. 5.13) and let J(S(x_i)) the set of clusters to which the objects of S(x_i) previously belonged according to $\hat{\Theta}^{k-1}$, then recompute $\theta_k(x_i)$=$ConstructSP\,(c_j,\, p_{\text{obj}},\, p_{\text{feat}},\lambda,\beta)$
4. *Fine-tunning with mPU*: apply $ksp(\text{X}, k, p_{\text{obj}},\, p_{\text{feat}},\, \lambda,\, \beta,\, ds)$ with active prototypes ∂^k_{act}=$\{\theta_j\colon\ j\in$J(S(x_i))$\}$, active clusters $\mathcal{C}^k_{\text{act}}$=$\{c_j\colon\ j\in$J(S($x_i$))$\}$, all objects active X^k_{act}=X, and the initial prototypes found in steps 2, 3
5. *Find the best case*: where the mPU of step 4 has provided the solution of lower clustering error

output: ∂^k_{act}

5.8 The global k-synthetic prototypes clustering method

In this section, we present a method for incremental prototype-based clustering specially designed for HDS data. The method is called *global k-synthetic prototypes* (gk-sp). The basic gk-sp and the fast version fg$k$-sp apply the same incremental steps to those of gk-means and fg$k$-means, respectively, as mentioned in Tab. 5.1: it introduces one component each time which is initially seeded with a data object. The novelty lays i) in a mechanism that helps the new prototype become a strong attractor for the data objects, and ii) an improved PU procedure which updates prototypes close to the considered object seed. These are designed in respect to the remarks and conclusion of the Sec. 5.7

The primary aim of this PU is to reduce the information imbalance between the formed prototypes and the newly introduced one. We propose a novel *reduction-enrichment mechanism* (REM) for this purpose that *reduces* the already formed prototypes $\hat{\Theta}^{k-1}$ while, at the same time, *enriches* the newly introduced prototype θ_k to help it become a strong attractor. This is achieved:

1) using the synthetic prototypes (sp) that constitute a reduced representation (more sparse) comparing to the respective cluster centroids provided in $\hat{\Theta}^{k-1}$,
2) by constructing a larger *sp* for the new cluster c_k:

$$K_j = \begin{cases} \lceil p_{\text{obj}}\ n_j \rceil & ,j \leq k-1 \\ \min\Big\{ n_j,\ \ \max\big\{ \lceil p_{\text{obj}}\ n_j \rceil\,,\ \max\limits_{q=1,\ldots,k-1} K_q \big\} \Big\} & ,j = k. \end{cases} \tag{5.19}$$

The above formula implies that as long as the new cluster is smaller than the largest of the other clusters, then practically the centroid will be computed as prototype. Note that this special treatment is recalled in case we observe:

$$\lceil p_{\text{obj}}\ n_j \rceil \geq \max_{q=1,\ldots,k-1} K_q. \tag{5.20}$$

After that point, the computation will be based on the upper branch of Eq. 5.19.

In addition, we propose the improved g$k$sp-mPU version that employs a more sophisticated PU approach. The *modified partial update* (mPU) considers as active all the clusters from which the new cluster detaches objects at the first iteration. Furthermore, all data objects are considered to be active and can move among the active clusters or return back to their initial cluster. Formally, we set $\partial^k_{\text{act}}=\{\theta_j\colon\ j\in \text{J}(\text{S}(x_i))\}$, $\mathcal{C}^k_{\text{act}}=\{c_j\colon\ j\in \text{J}(\text{S}(x_i))\}$, and $\text{X}^k_{\text{act}}=\text{X}$. In this way, the competition is set among the clusters that are close to the new seed. The intuition is to help the prototypes make more drastic changes to the solution so far. It is remarkable that an inactive prototype implies that it will not adapt to the changes of object members and eventually will become '*outdated*' to some extend. Consequently, this will permit easier departures from the cluster to an active clusters around the seed.

This approach is a better trade-off between computational cost and effectiveness, than the other PUs. The respective fast g$k$sp-mPU version (fg$k$sp-mPU) proceeds for all object seeds up to the PU step of Algorithm 6 but applies the FU only on the best solution found by PU_{m-1} in each case. This procedure is described in Algorithm 7.

5.9 Experimental study

5.9.1 Setup

In the experimental evaluation we compare some of the well-known incremental algorithms, such as gk-means [84], the fast modified gk-means [140] (fmg$k$m) and the divisive k-means, with those algorithms derived by the proposed framework. In particular, in this chapter we have proposed the global k-sp (g$k$sp), the global bisecting k-means (gb$k$m), and the modified gk-means (mg$k$m) (which is the *slow* version of fmg$k$m proposed in [140]). We did not include the experimental results for the traditional bisecting k-means (bkm); instead we used the proposed gb$k$m that examines the division of every cluster in order to select the best split, and thus performs better. To setup a fair testbed, the divisive algorithms initialize the two new prototypes by selecting one random data object from the cluster that is split, and the previous centroid of that cluster.

Moreover we have tested the respective *fast* versions of all the *slow* clustering methods we considered (they are denoted with an 'f' as initial letter, e.g. fgb$k$m). The difference lays on the fact that a *fast* version first computes the result of the partial updates (PUs: PU_j, j=1,...,m-1) for all initializations (the different object seeds) and then performs the update of the full model (FU). Contrary, the *slow* versions apply FU for all the objects, right after the PUs, and select the best solution out of them as the $\hat{\Theta}^k$.

The use of synthetic prototypes (sps) was also proposed in this chapter to be incorporated into the incremental prototype-based clustering methods (these variations are denoted with the suffix 'sp', e.g. gb$k$sp). The parameters of the sp construction are set to be the same as in the experimental setup of Chapter 4, however we have fixed the value of p_{obj} parameter to 80% of the cluster objects. Moreover we used the slightly altered k-sp version that permits some error deterioration steps (we set ds=5, see Chapter 5.4).

Note that all the sp-based variations employ the *reduction-enrichment mechanism* (REM) presented in Sec. 5.8. This mechanism helps the new prototype to be more competitive to the prototypes of the already formed clusters in order to attract data objects. The g$k$sp-mPU is a proposed variation that uses the *modified partial update*

(mpU). The main idea of mPU is to first set a competition between the new prototype and the prototypes that are close to it. After convergence a FU is applied. Note also that in this case all objects of the dataset can move to the active clusters according to the mPU definition. The intuition is to help the prototypes make more drastic changes to the solution so far. It must be noted that the inactivity of a prototype implies that it does not adapt to changes of the object members of its cluster. This behavior will eventually make the prototype '*outdated*' to some extent, and hence will permit cluster objects to easier move to other clusters.

Most of the datasets on which we tested the algorithms have been also used in Chapter 4: the artificial dataset $\mathrm{A}_4^{(3)}$, and the real datasets $\mathrm{RS}_4^{(S)}$, $\mathrm{M}_6^{(S)}$, $\mathrm{M}_6^{(M)}$, $\mathrm{M}_8^{(S)}$ and Mini_{20}. More specifically, in this study we used small and medium-sized datasets. The only new is the artificial dataset A_{30}, which was created as $\mathrm{A}_4^{(3)}$ with the process presented in Sec. 4.4.2. The notation we follow for the dataset names is the same to Chapter 4: the subscript denotes the number of clusters and the superscript denotes (if any) a general characterization whether the dataset is (S)mall, or (M)edium regarding the number of the contained objects. We tried to cluster the small datasets in the whole range from 2 up to $3k$ number of clusters, while for the larger datasets we applied clustering from 1 up to k clusters.

Internal and external clustering evaluation has been conducted. Our primary consideration is that data clustering problem is formulated as an optimization procedure, therefor we have used minimum attained value of the objective function as the main measure for method comparison. The same vector representations (BOW) of the documents are given as input to the algorithms, and all of them employ the complementary of clustering Cohesion (CC) Eq. 5.4 as objective error function. In order to demonstrate the reduction of error as more clusters are added into the solution, we normalize the error values wrt the error of the single cluster case where all data object are in one cluster:

$$\bar{\Phi}(\hat{\Theta}^k) = \frac{\Phi_{\mathrm{CC}}(\hat{\Theta}^k)}{\Phi_{\mathrm{CC}}(\hat{\Theta}^1)} \in [0,1]. \tag{5.21}$$

For the external evaluation we used the *normalized mutual information* (NMI) that measures with a value in $[0, 1]$ the agreement between the achieved clustering and the provided ground truth labeling of the datasets. Lower values of $\bar{\Phi}(\hat{\Theta}^k)$ and higher values of NMI indicate a better clustering result. We should note here that both internal and external evaluation are useful for the experimental study. However, internal and external measures may not always agree about which of two clustering solutions are better. For example we may achieve a decrease of the objective error value but at the same time observe a lower NMI value. This might happen because the object labeling has been created by humans and may not reflect the underlying properties of the feature space in which the data vectors are represented. From the optimization point of view, and given the data representations, the algorithm which is more efficient in minimizing the objective value for a dataset should generally be considered as the best method.

5.9.2 Experimental results

At the end of this chapter, we provide for each dataset two figures with graphic plots that demonstrate the behavior and performance of the compared clustering methods: the first for the *slow* incremental clustering methods and the second for their respective *fast* versions (e.g. Fig. 5.1 and Fig. 5.2, respectively, refer to dataset $\mathrm{RS}_4^{(S)}$). Each figure presents: i) plots illustrating the the size of each cluster (y-axis) as lines of the same color wrt the number of clusters k (x-axis), and ii) the plots of the cluster evaluation measures (e.g. Fig. 5.1b) as the number of clusters increases[3]. These values correspond to the solution with the minimum objective value found in each incremental step.

One key observation concerns the significant influence of the small number of objects in a dataset (size of dataset) on the performance of the clustering algorithms. For small datasets the algorithms demonstrate higher performance variation and, it can be observed more clearly the effectiveness of the proposed methodologies that help the newly added cluster to gain more objects. The plots of the size of clusters provide empirical evidence

[3]The legend shown in the lower right corner of a figure refers to the evaluation measures only. The colors of the left side of each figure have been arbitrarily selected, however, each color indicates the relative number of members of a particular cluster

about the *unfair competition between prototypes* that we have discussed in this chapter. For example, see Fig. 5.3 and Fig. 5.4 for $M_8^{(S)}$, and Fig. 5.5 and Fig. 5.6 for $M_6^{(S)}$. Speaking about Fig. 5.3, we can see that gkm finds a clustering for k=2 that splits the dataset with 80%-20% ratio (the red line). This is fair enough as a first step, since the eight data classes we seek are almost of the same size. Then the third cluster (green line) comes up at the third incremental step and takes another part of about 20% of the data. However, all the clusters that are introduced in the solution after that moment fail to attract a significant number of data objects and to form interesting clusters. The initial cluster that is the largest continues to lose objects towards the new cluster as k grows, but even up to the end of the experiment it retains over 20% of the data even though 23 other clusters were competing to each other.

The clusters formed in the initial incremental steps may not let the new clusters added later become part of the clustering solution. It is natural for such a behavior to be more intense when we attempt to partition a dataset in more clusters than the groups of the underlying structure. There, we could assume that the structure of the data is strong and cannot be easily split into small pieces (clusters). The most characteristic such case is the artificial dataset $A_4^{(2)}$ (see Fig. 5.13 and Fig. 5.14), where the centroid-based algorithms initially find four large clusters, but then they fail to seek further for homogeneous subclusters. In the larger datasets the methods seem to behave differently in the sense that there is more '*action*' and the clusters compete more strongly to each other (for instance see Fig. 5.7, Fig. 5.8). But it is also clear that the phenomenon we talk about is still present in every instance of our experiments. We can see in the latter case that the clusters are separated in two groups: those that compete to each other and the rest of them that contain probably a small set of closely related objects without being able to attract new data. Another example is Mini_{20} (see Fig. 5.9 and Fig. 5.10) which contains 20 clusters with 2000 data objects in total, but it is clear that, in a analogous extend, the first two clusters have a strong advantage in the clustering competition. Fig. 5.14 and Fig. 5.13 provide the only example where the centroid-based approaches create competitive clusters and prototypes during all incremental steps. The

respective evaluation measures indicate that again the proposed approaches achieve better clusterings and the difference in quality seems to increase with the number of clusters.

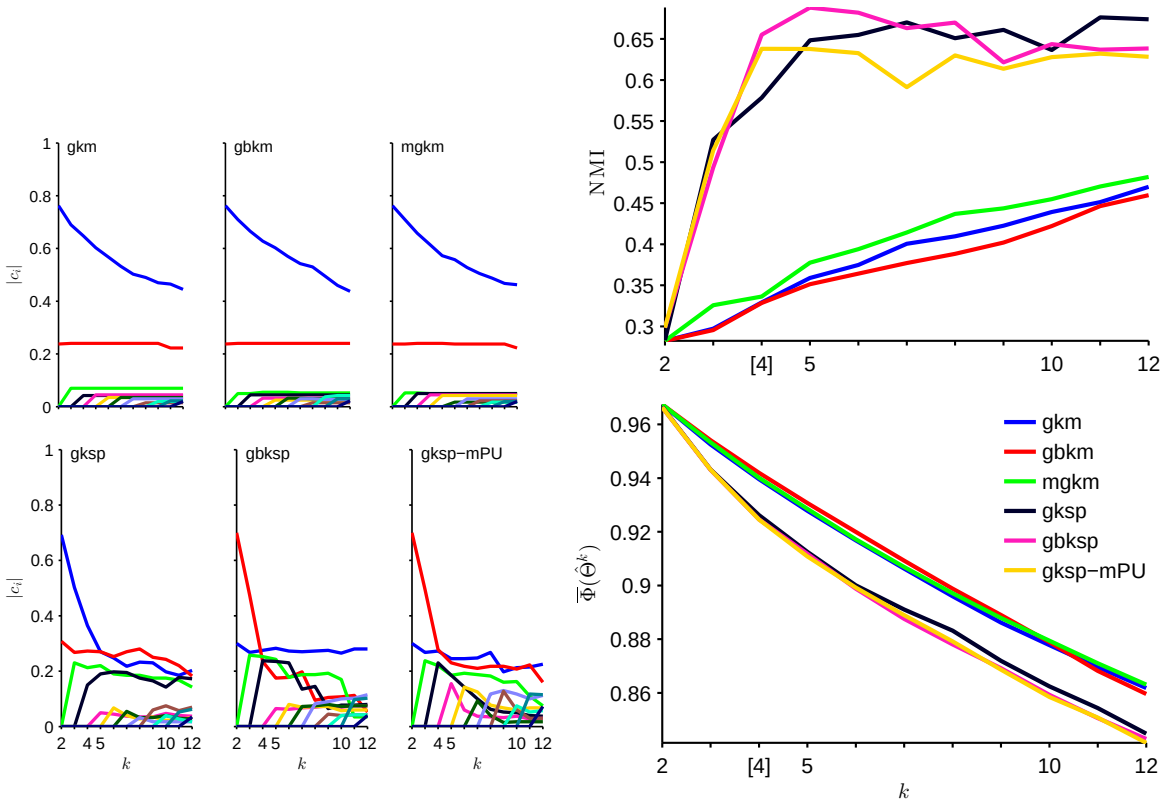
All the previous observations hold in both the *fast* and *slow* incremental clustering approaches. Also in both *slow* and *fast* variations, in real or artificial HDS data, the incorporation of synthetic prototypes had a positive impact. The methods are among the most competitive optimizers. Among them, in most cases the g$k$sp-mPU and fg$k$sp-mPU seem to be one of the best choices.

In what concerns the *reduction-enrichment mechanism* (REM), this experimental study provides empirical evidence supporting the claims of our analysis, the clustering methodologies, and the optimization heuristics proposed in this chapter. In addition, the experiments also indicated directions to improve the proposed techniques. For instance, the REM mechanism, and in particular the approach to enrich the new cluster in order to become competitive, in most cases was helpful to find a good solution in adverse settings where even the synthetic prototypes seem not to help the g$k$sp to effectively discover new clusters (e.g. in Fig. 5.12 and Fig. 5.12). On the other hand if we observe the left part of the figures, there are cases where the new cluster enjoys a '*sudden popularity*' when it is introduced and right after it returns to deprecation. This provocation might be necessary to achieve a drastic change in the competition between the clusters and prototypes, however it is clear that issues arise regarding the stability of this approach. One possible direction that this issue should be investigated is to consider the median of the cluster sizes as the limit after which we cancel the favoring of the new cluster. Alternatively, the enrichment may be defined to have larger duration in time, since in the present approaches it is applied only when the cluster is introduced to the solution.

5.10 Conclusions

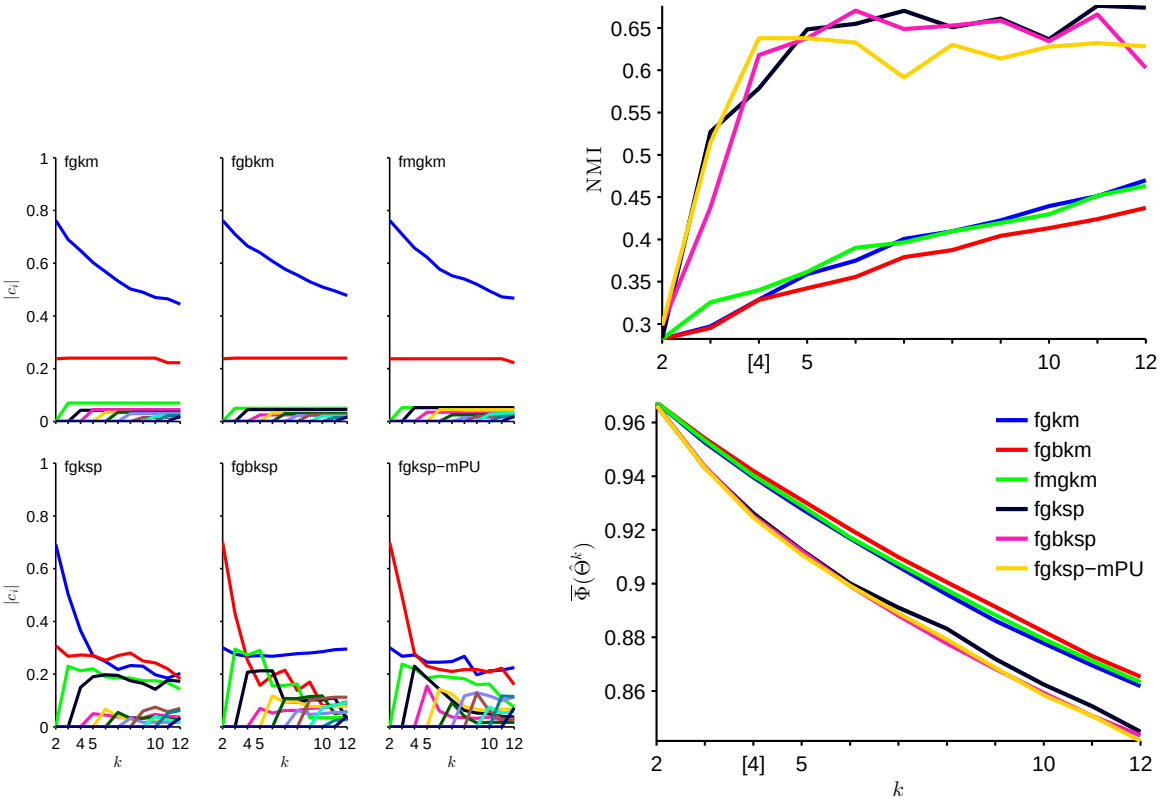
In this chapter we have presented a framework for prototype-based incremental k-means clustering that is based on *partial updates* on a given solution. In a partial update phase

only a subset of the cluster prototypes, clusters, and data objects participate in the clustering process. According to our provided analysis we have also revisited and unified two widely known incremental clustering approaches: the *global k-means* and *divisive k-means*. In this chapter we focused on HDS spherical data and discussed the problem of increasing the order of a current k-clustering solution by adding one new component. We proposed the incorporation of synthetic cluster prototypes presented in the previous chapter into incremental prototype-based clustering. Accordingly, we proposed the *global k-synthetic prototypes* (*gk-sp*) clustering algorithm, which is a modification of the global k-means algorithm for HDS data. The gk-sp method uses the *k-synthetic prototypes* method for fine-tunning the k-solution and introduces a partial update scheme to initialize the $k+1$ prototypes for the refining phase. Experiments on real and artificial document datasets illustrate that the proposed gk-sp method outperforms other incremental methods of the k-means family.



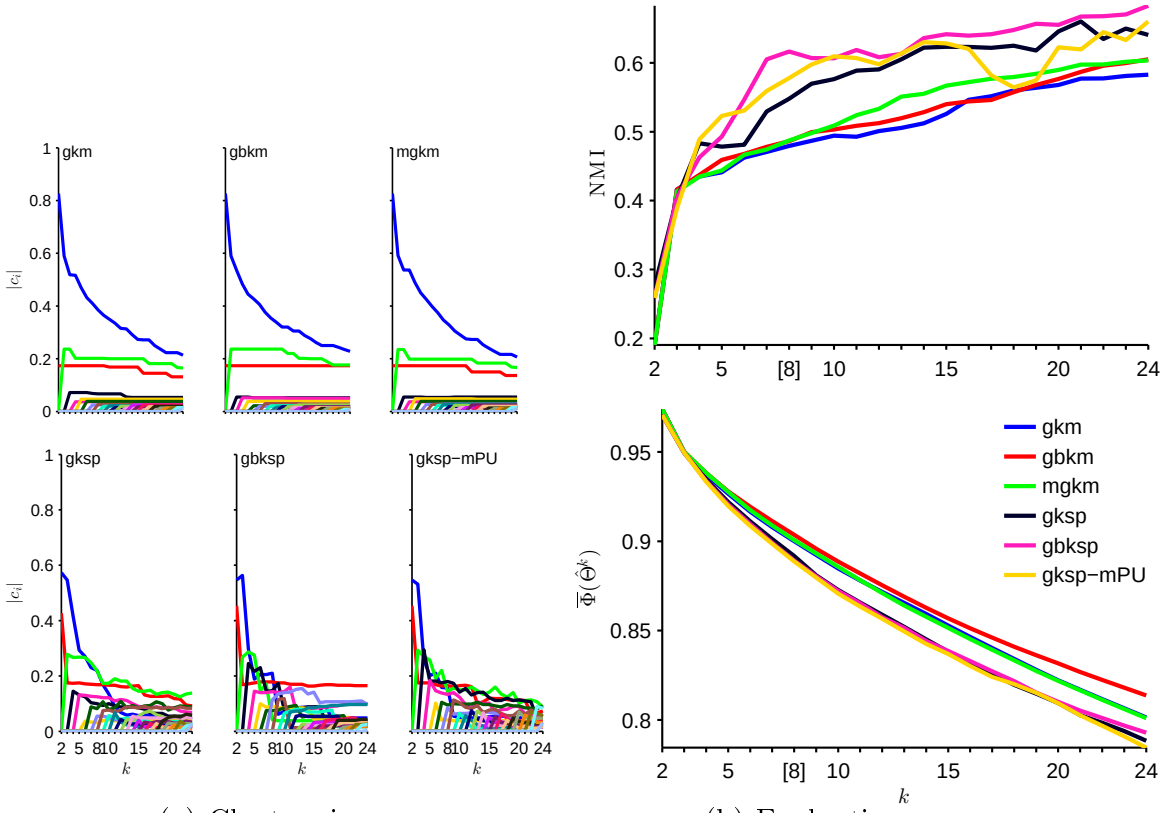
(a) Cluster sizes (b) Evaluation measures

Figure 5.1: Slow incremental clustering versions for $\mathrm{RS}_4^{(S)}$.



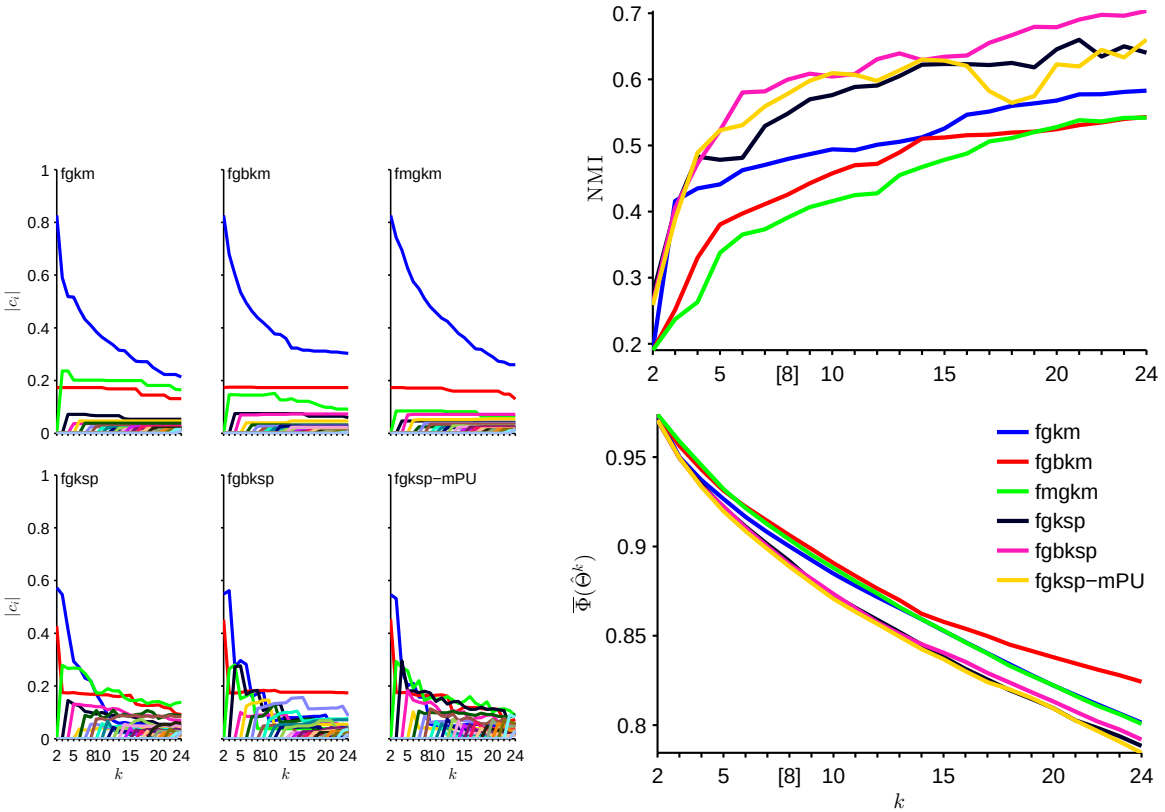
(a) Cluster sizes (b) Evaluation measures

Figure 5.2: Fast incremental clustering versions for $\mathrm{RS}_4^{(S)}$.



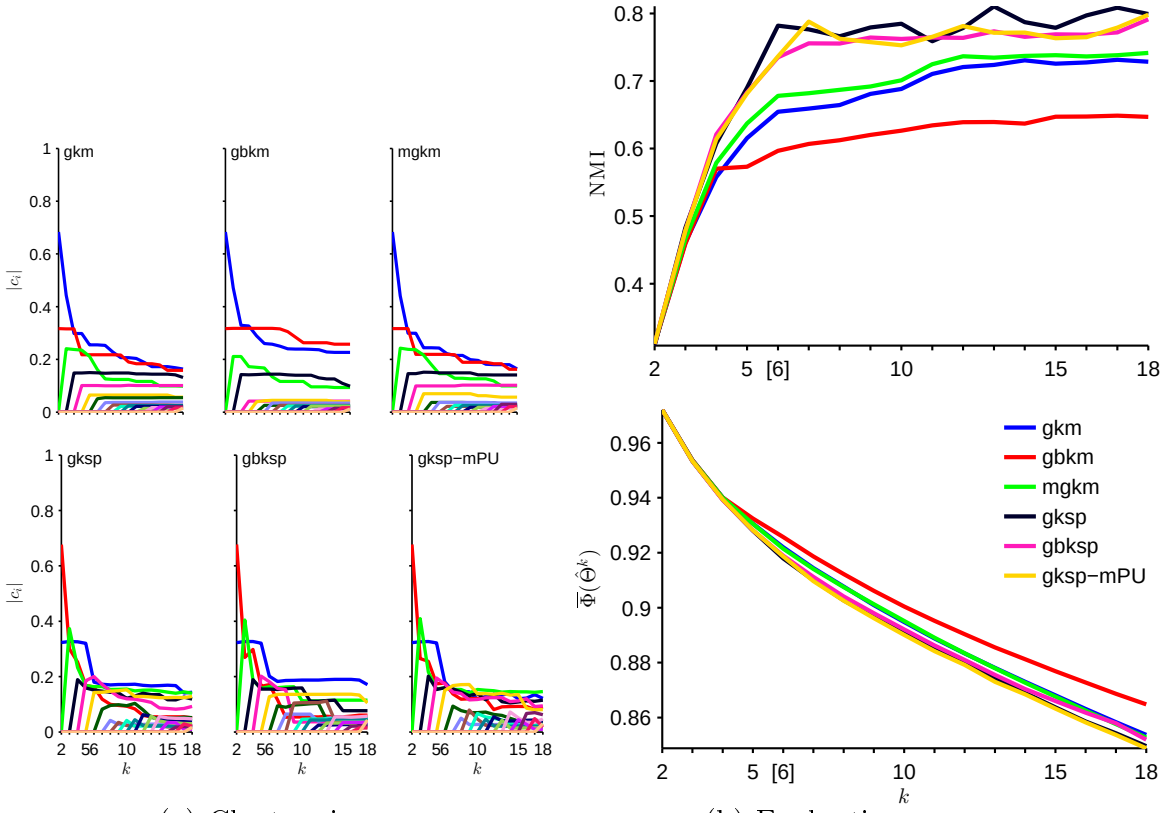
(a) Cluster sizes (b) Evaluation measures

Figure 5.3: Slow incremental clustering versions for $\mathrm{M}_8^{(S)}$.



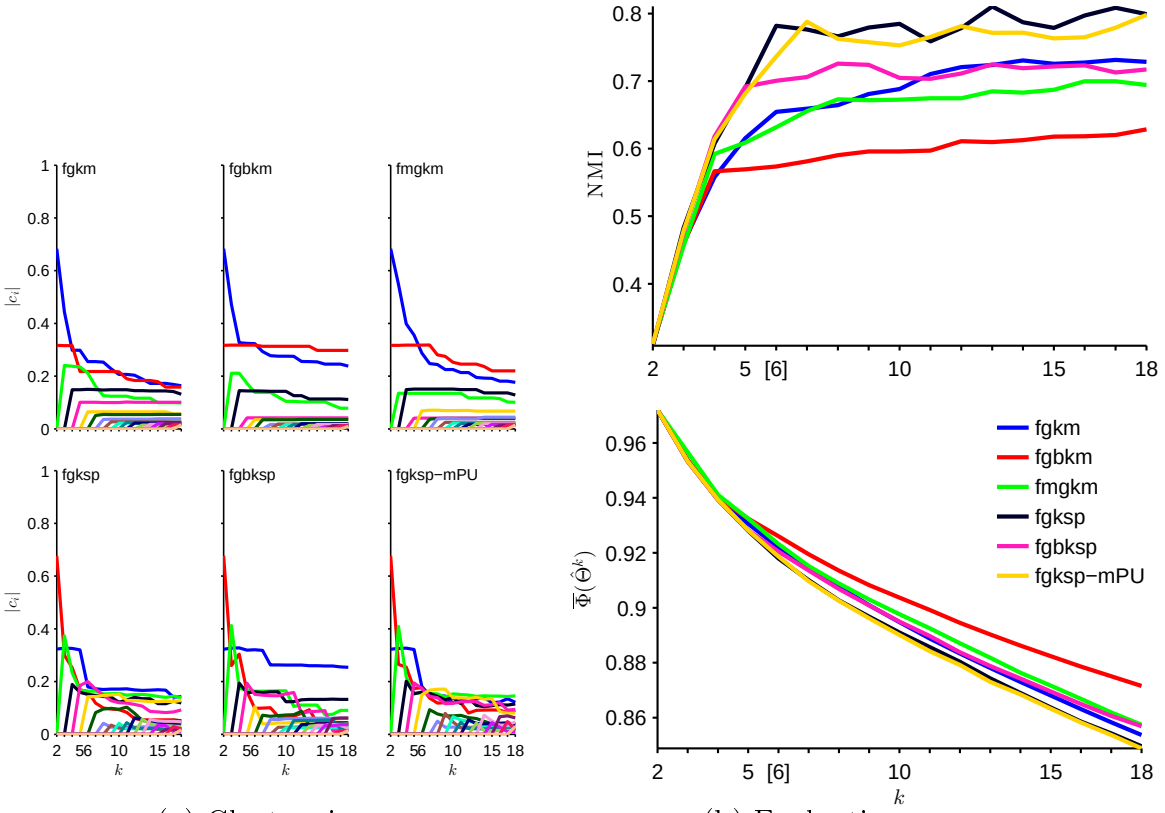
(a) Cluster sizes (b) Evaluation measures

Figure 5.4: Fast incremental clustering versions for $\mathrm{M}_8^{(S)}$.



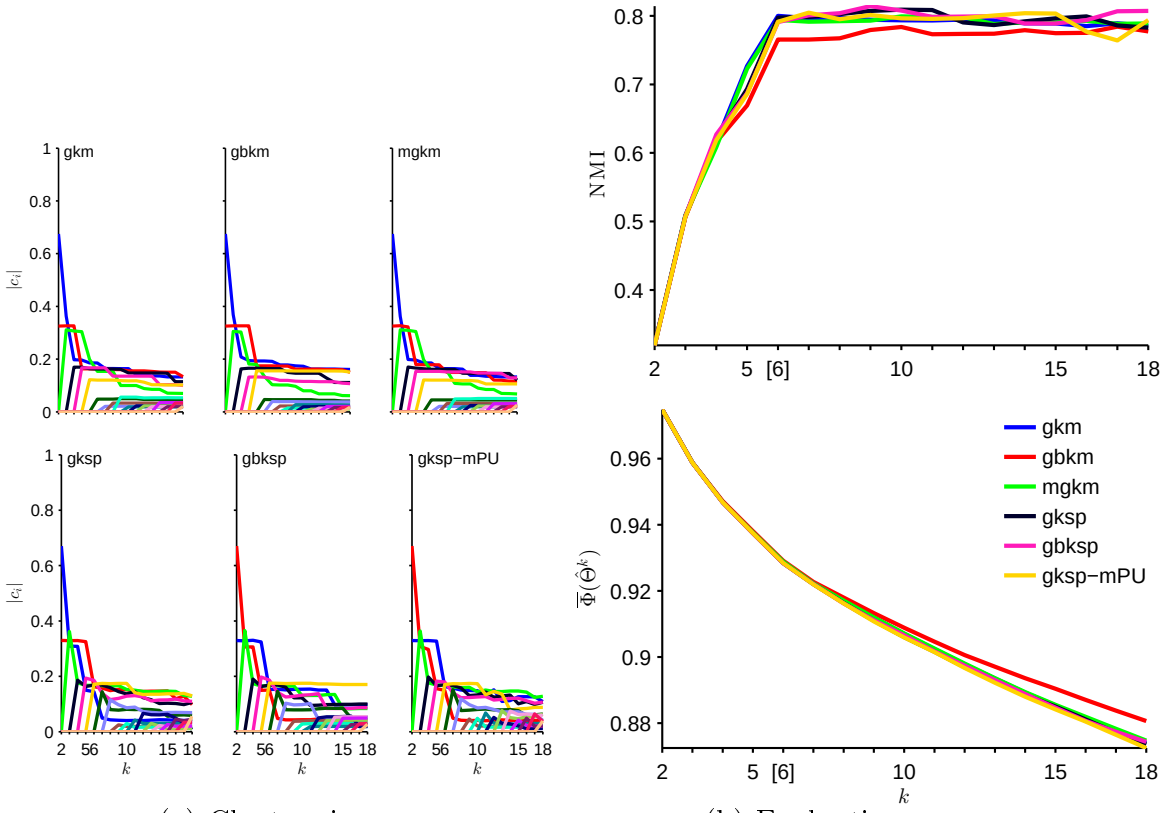
(a) Cluster sizes (b) Evaluation measures

Figure 5.5: Slow incremental clustering versions for $M_6^{(S)}$.



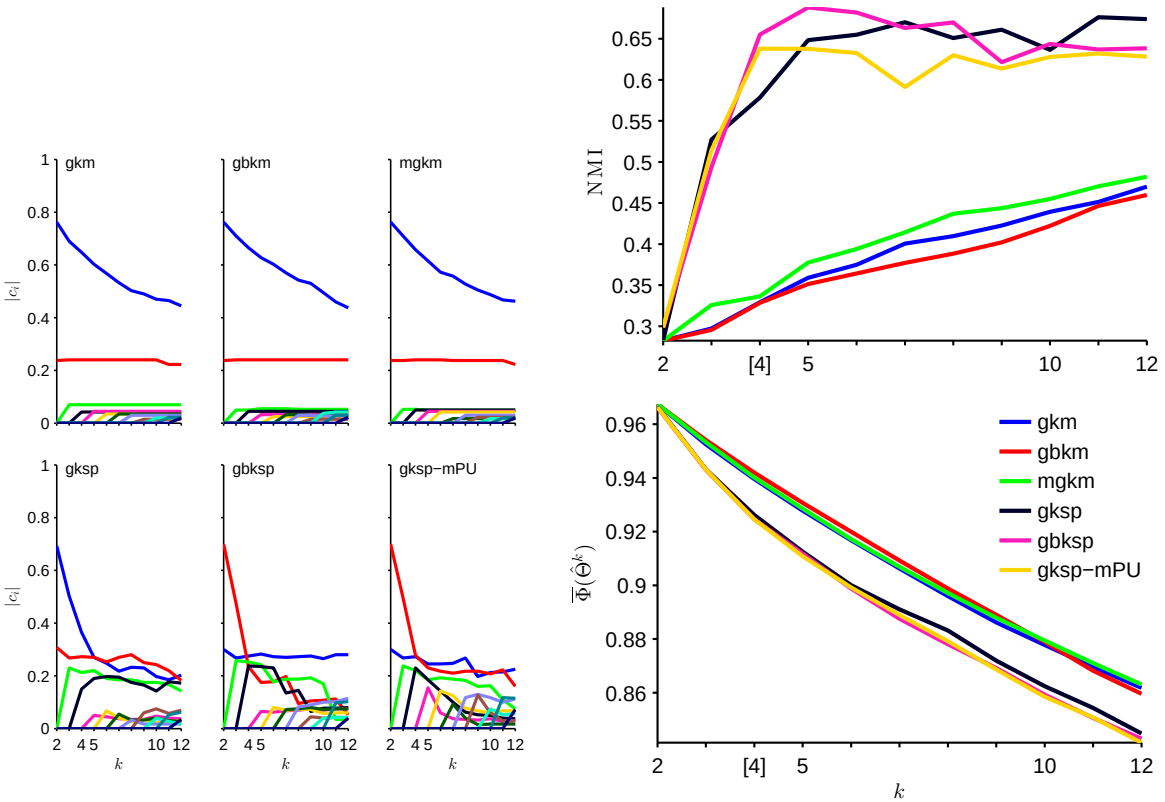
(a) Cluster sizes (b) Evaluation measures

Figure 5.6: Fast incremental clustering versions for $M_6^{(S)}$.



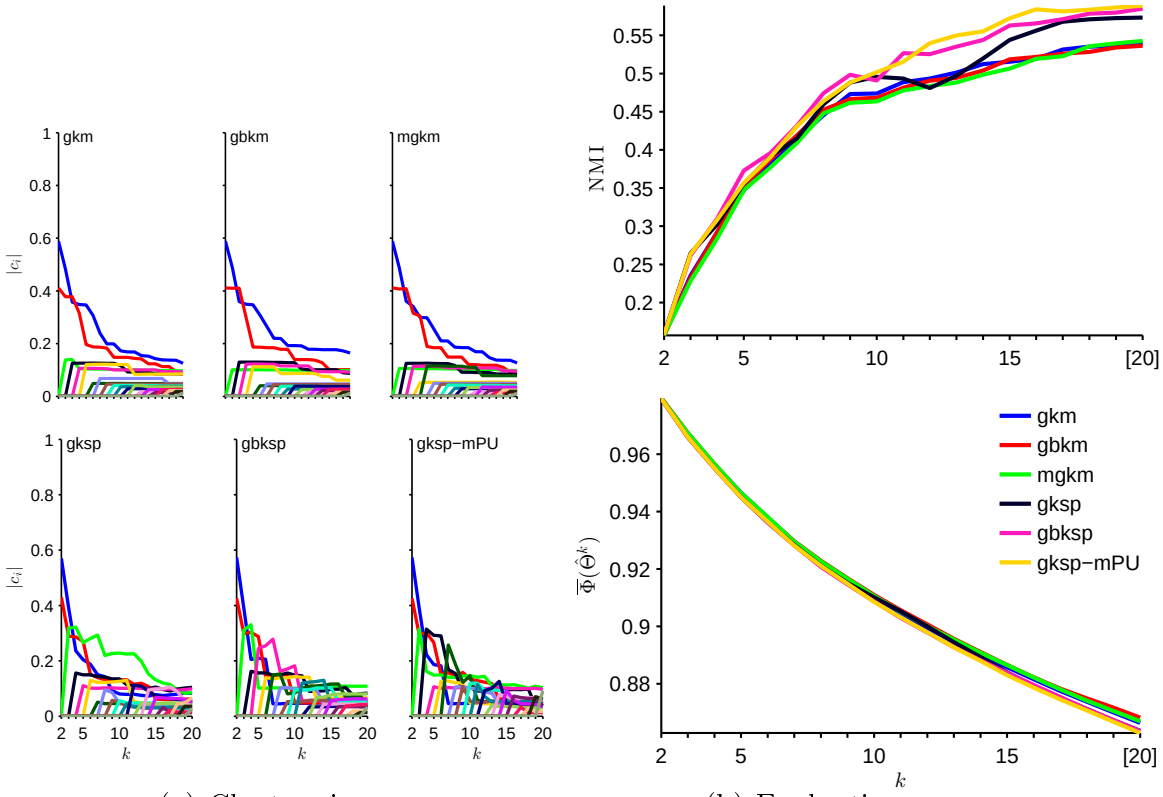
(a) Cluster sizes (b) Evaluation measures

Figure 5.7: Slow incremental clustering versions for $\mathrm{M}_6^{(M)}$.



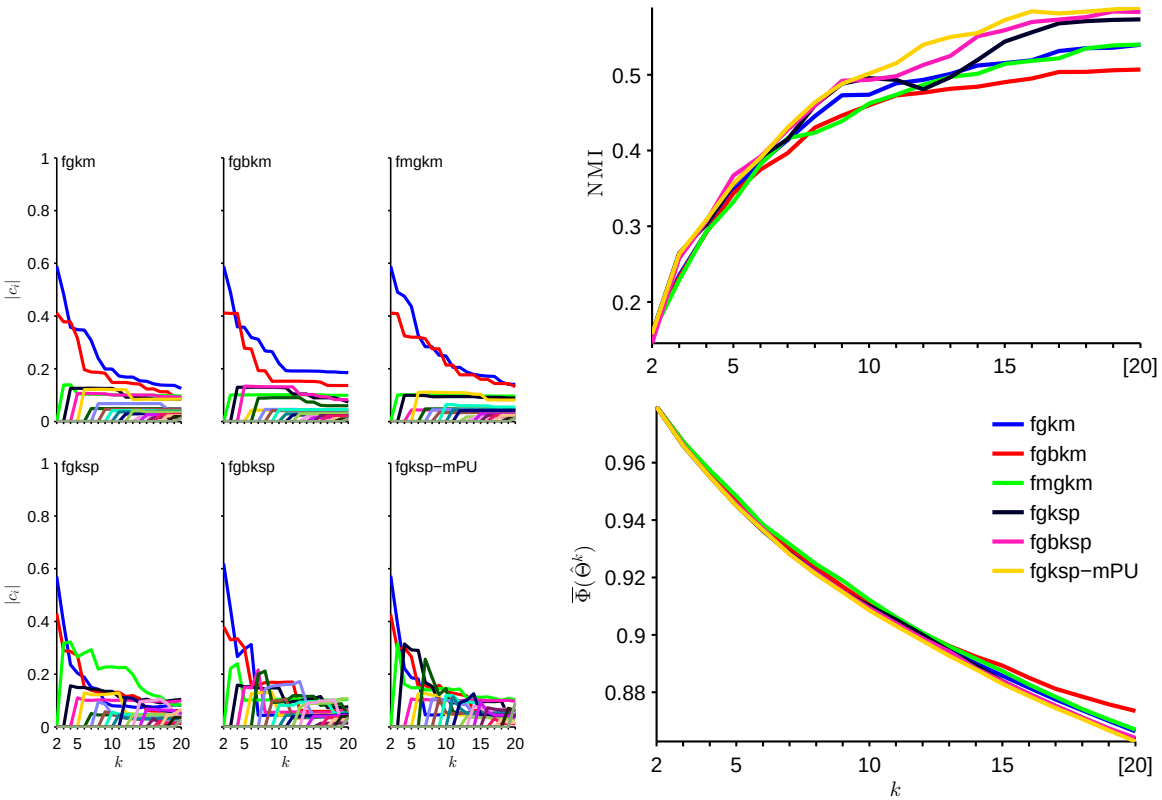
(a) Cluster sizes (b) Evaluation measures

Figure 5.8: Fast incremental clustering versions for $\mathrm{M}_6^{(M)}$.



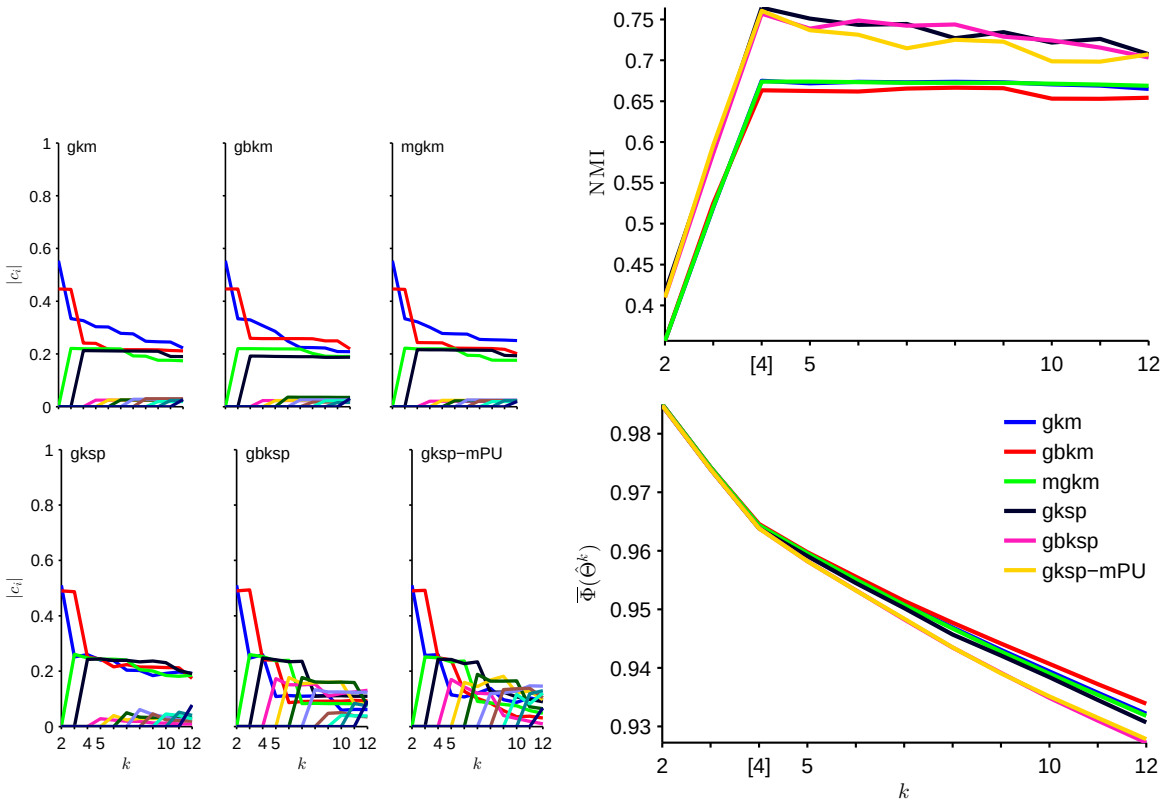
(a) Cluster sizes (b) Evaluation measures

Figure 5.9: Slow incremental clustering versions for Mini_{20}.



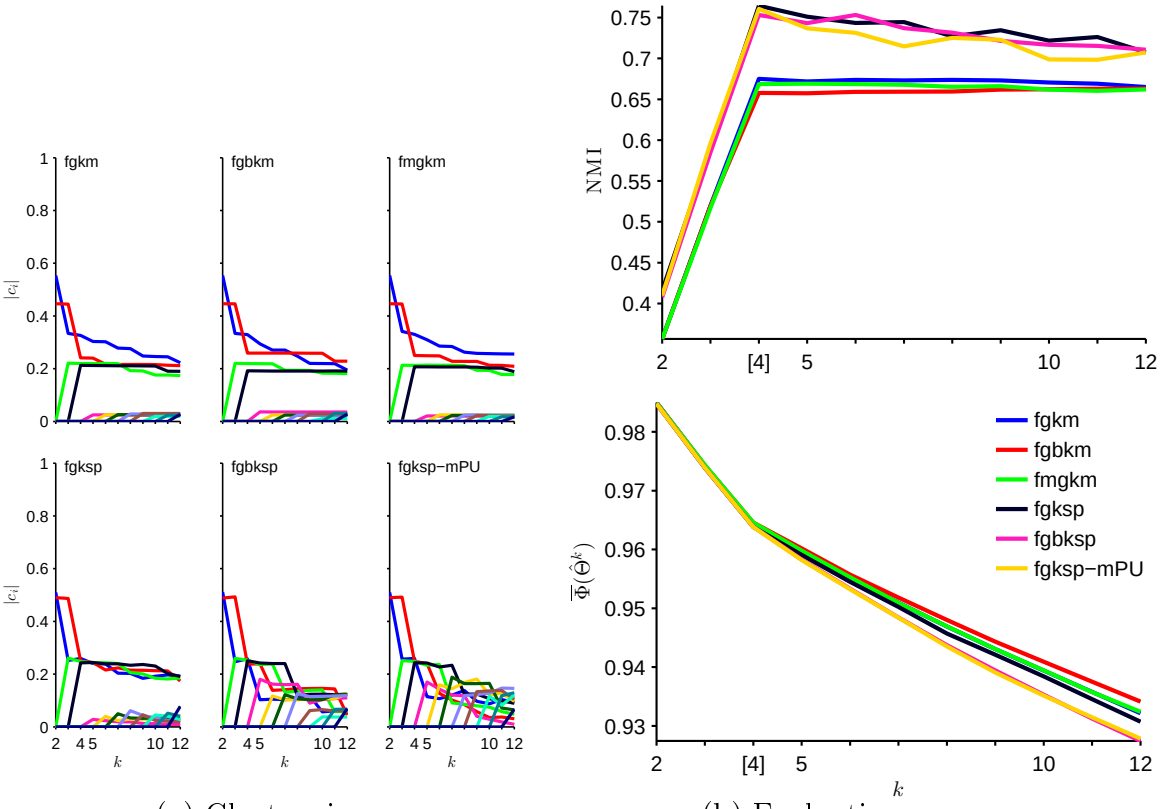
(a) Cluster sizes (b) Evaluation measures

Figure 5.10: Fast incremental clustering versions for the Mini_{20}.



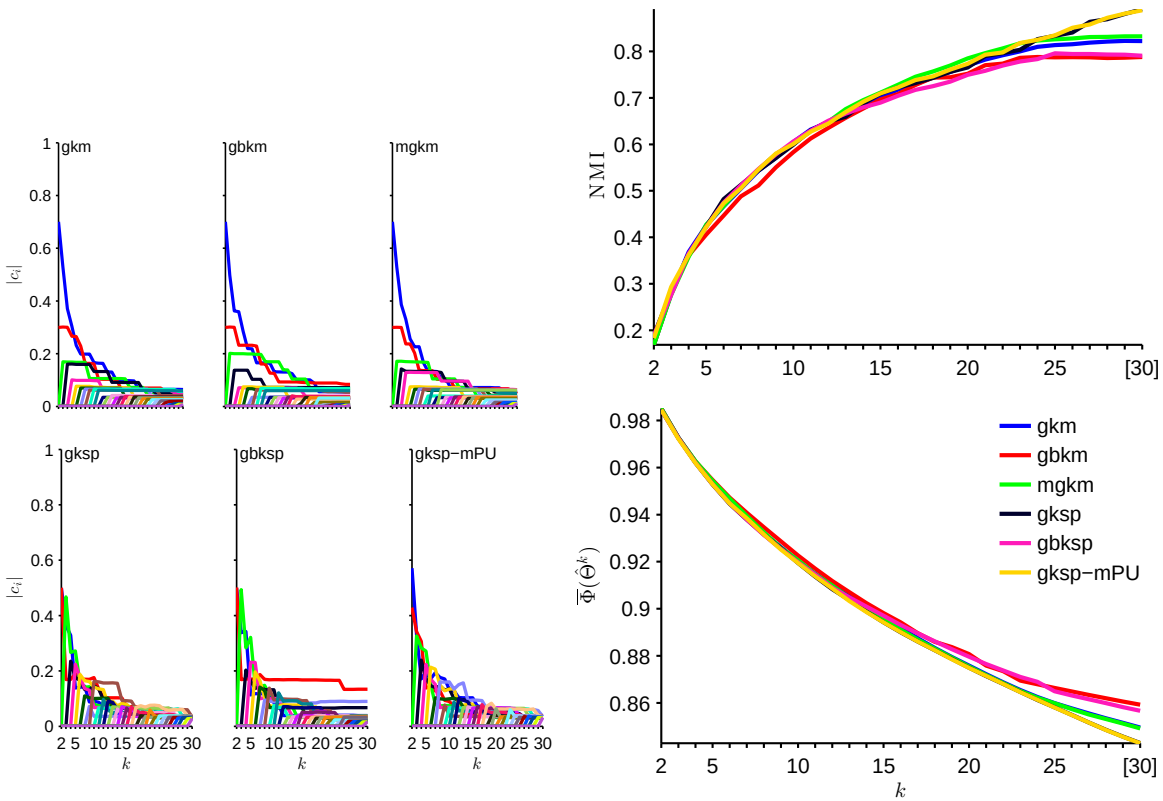
(a) Cluster sizes (b) Evaluation measures

Figure 5.11: Slow incremental clustering versions for the artificial dataset $A_4^{(3)}$.



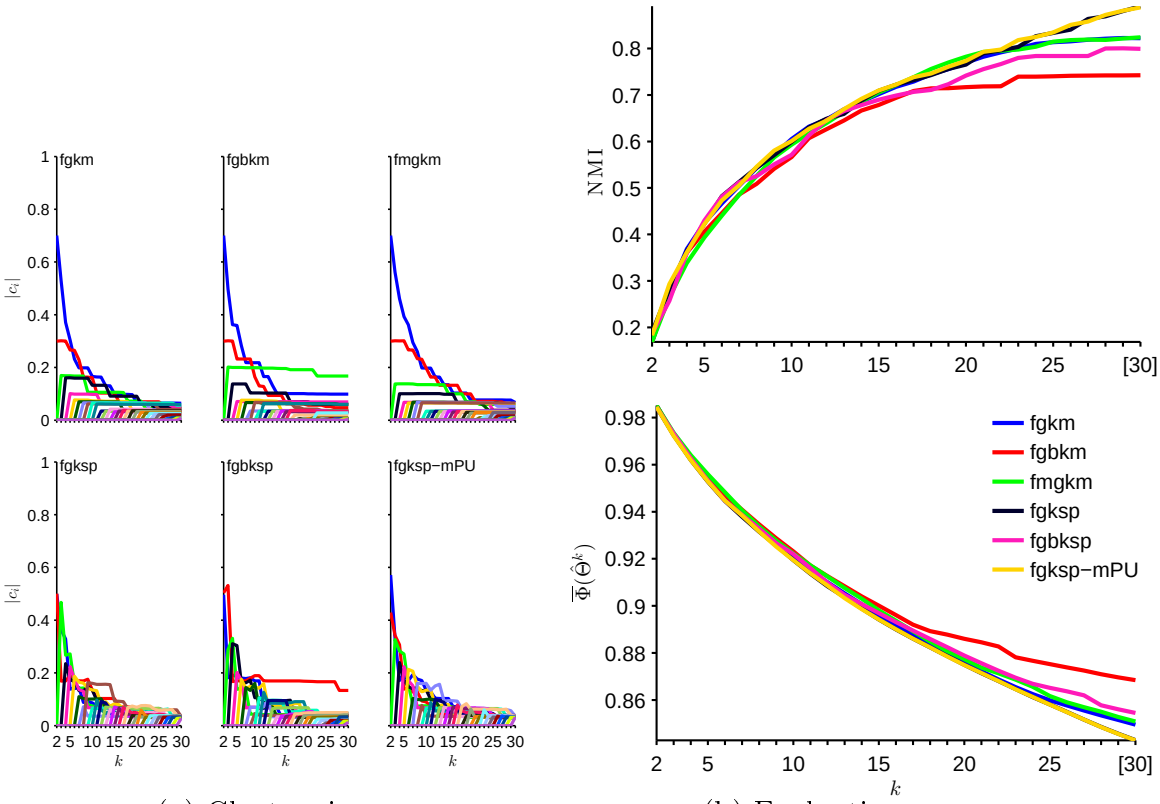
(a) Cluster sizes (b) Evaluation measures

Figure 5.12: Fast incremental clustering versions for the artificial dataset $A_4^{(3)}$.



(a) Cluster sizes (b) Evaluation measures

Figure 5.13: Slow incremental clustering versions for for the artificial dataset A_{30}.



(a) Cluster sizes (b) Evaluation measures

Figure 5.14: Fast incremental clustering versions for the artificial dataset A_{30}.

Chapter 6

Dip-means: An Incremental Clustering Method for Estimating the Number of Clusters

6.1 Introduction

There are various algorithms that can find reasonable clusterings. Most clustering methods consider the number of clusters k as a required input, and then they apply an optimization procedure to adjust the parameters of the assumed cluster model. As a consequence,

in exploratory analysis, where the data characteristics are not known in advance, an appropriate k value must be chosen. This is a rather difficult problem, but at the same time very fundamental in order to apply data clustering in practice.

Several algorithms have been proposed to determine a proper k value, most of which wrap around an iterative model-based clustering framework, such as the k-means or the more general *expectation-maximization* (EM). In a top-down (incremental) strategy they start with one cluster and proceed to splitting as long as a certain criterion is satisfied. At each phase, they evaluate the clustering produced with a fixed k and they decide whether to increase the number of clusters as follows:

Repeat until no changes occur in the model structure

1. Improve model parameters by running a conventional clustering algorithm for a fixed k value.
2. Improve model structure, usually through cluster splitting.

One of the first attempts in extending k-means in this direction was x-means [143] which uses a regularization penalty based on model's complexity. To this end, *Bayesian information criterion* (BIC) [144] was used, and among many models the one with highest BIC is selected. This criterion works well only in cases where there are plenty of data and well-separated spherical clusters. Alternative selection criteria have also been examined in literature [136].

G-means [145] is another extension to k-means that uses a statistical test for the hypothesis that each cluster has been generated from Gaussian distribution. Since statistical tests become weaker in high dimensions, the algorithm first projects the datapoints of a cluster on an axis of high variance and then applies Anderson-Darling statistic with a fixed significance level α. Clusters that are not accepted are split repeatedly until the entire assumed mixture of Gaussians is discovered. *Projected g-means* (pg-means) [146] again assumes that the dataset has been generated from a Gaussian mixture, but it tests the overall model at once and not each cluster separately. Pg-means bases on the EM algorithm. Using a series of random linear projections, it constructs a one-dimensional projection of the dataset and the learned model and then tests the model fitness in the

projected space with Kolmogorov-Smirnov (KS) test [147]. The advantage of this method is the ability to discover Gaussian clusters of various scales and different covariances, that may overlap. *Bayesian k-means* [148] introduces *maximization-expectation* (ME) to learn a mixture model by maximizing over hidden variables (datapoint assignments to clusters) and computing expectation over random model parameters (centers and covariances). If the data come from a mixture of Gaussian components, this method can be used to find the correct number of clusters and is competitive to the aforementioned approaches. Other alternatives have also been proposed, such as *gap statistic* [149], *self-tuning spectral clustering* [150], *data spectroscopic clustering* [151], and *stability-based model validation* [152–154], however they are not closely related to the proposed method.

The work in this chapter is primarily motivated by the non generality of the approaches in [145] and [146], as they make *Gausssianity assumptions* about the underlying data distribution. As a consequence, they tend to overfit for clusters that are uniformly distributed, or have a non-Gaussian unimodal distribution. Additional limitations are that they are designed to handle numerical vectors only and require the data in the original dataspace. The contribution of our work is two-fold. Firstly, we propose a statistical test for unimodality, called *dip-dist*, to be applied into a data subset in order to determine if it contains a single or multiple cluster structures. Thus, we make a more general assumption about what is an acceptable cluster. Moreover, the test involves pairwise distances or similarities and not the original data vectors. Secondly, we propose the *dip-means* incremental clustering method [45] which is a wrapper around k-means. We experimentally show that dip-means is able to cope with datasets containing clusters of arbitrary density distributions. This is tested using artificial dataset, while we also use real-wold data such as images from handwritten digits and objects, and text document. The object images and the text data represented in a very high dimensional and sparse space, where additional challenges arise for any statistic test. Moreover, it can be easily extended in kernel space by using the kernel k-means [155] and modifying appropriately the cluster splitting procedure.

6.2 Dip-dist criterion for cluster structure evaluation

In cluster analysis, the detection of multiple cluster structures in a dataset requires assumptions about what the clusters we seek look like. The assumptions about the presence of certain data characteristics along with the tests employed for verification, considerably influence the performance of various methods. It is highly desirable for the assumptions to be general in order not to restrict the applicability of the method to certain types of clusters only (e.g. Gaussian). Moreover, it is of great value for a method to be able to verify the assumed cluster hypothesis with well designed statistical hypothesis tests that are theoretically sound, in contrast to various alternative ad hoc criteria.

We propose the novel *dip-dist criterion* for evaluating the cluster structure of a dataset that is based on testing the empirical density distribution of the data for unimodality. The *unimodality assumption* implies that the empirical density of an acceptable cluster should have a single mode; a region where the density becomes maximum, while non-increasing density is observed when moving away from the mode. There are no other underlying assumptions about the shape of a cluster and the distribution that generated the empirically observed unimodal property. Under this assumption, it is possible to identify clusters generated by various unimodal distributions, such as Gaussian, Student-t, etc. The Uniform distribution can also be identified, since it is an extreme single mode case where the mode covers all the region with non-zero density.

A convenient issue is that unimodality can be verified using powerful statistical hypothesis tests (especially for one-dimensional data), such as Silverman's method which uses fixed-width kernel density estimates [156] or the widely-used *Hartigan's dip statistic* [157]. As the dimensionality of the data increases, the tests require a sufficient number of data points in order to be reliable. Thus, although the data may be of arbitrary dimensionality, it is important to apply unimodality tests on one-dimensional data values. Furthermore, it would be desirable, if the test could also be applied in cases where the distance (or similarity) matrix is given and not the original datapoints.

To meet the above requirements we propose the *dip-dist* criterion for determining uni-

modality in a set of datapoints using only their pairwise distances (or similarities). More specifically, if we con-sider an arbitrary datapoint as a *viewer* and form a vector whose components are the distances of the viewer from all the datapoints, then the distribution of the values in this distance vector could reveal information about the cluster structure. In presence of a single cluster, the distribution of distances is expected to be unimodal. In the case of two distinct clusters, the distribution of distances should exhibit two distinct modes, with each mode containing the distances to the datapoints of each cluster. Consequently, a unimodality test on the distribution of the values of the distance vector would provide indication about the unimodality of the cluster structure. However, there is a dependence of the results on the selected viewer. Intuitively, viewers at the boundaries of the set are expected to form distance vectors whose density modes are more distinct in case of more than one clusters. To tackle the viewer selection problem, we consider all the datapoints of the set as individual viewers and perform the unimodality test on the distance vector of each viewer. If there exist viewers that reject unimodality (called *split viewers*), we conclude that the examined cluster includes multiple cluster structures.

For testing unimodality we use *Hartigans' dip test* [157]. A function $F(t)$ is unimodal with mode the region s_m=$\{(t_L, t_U) : t_L \leq t_U\}$ if it is convex in s_L=$(-\infty, t_L]$, constant in $[t_L,\ t_U]$, and concave in s_U=$[t_U, \infty)$. This implies the non-increasing probability density behavior when moving away from the mode. For bounded input functions F, G, let $\rho(F,G)$=$\max_t |F(t) - G(t)|$, and let $\mathcal{U}$ be the class of all unimodal distributions. Then the *dip statistic* of a distribution function F is given by:

$$dip(F) = \min_{G \in \mathcal{U}} \rho(F, G). \tag{6.1}$$

In other words, the dip statistic computes the minimum among the maximum deviations observed between the cdf F and the cdfs from the class of unimodal distributions. A nice property of dip is that, if F_n is a sample distribution of n observations from F, then $lim_{n\to\infty} dip(F_n)$=$dip(F)$. In [157] it is argued that the class of uniform distributions U is the most appropriate for the null hypothesis, since its dip values are stochastically larger

than other unimodal distributions, such as those having exponentially decreasing tails.

Given a vector of observations $f=\{f_i : f_i \in \mathbb{R}\}_{i=1}^{n}$, then the algorithm for performing the dip test [157] is applied on the respective empirical cdf $F_n(t)=\frac{1}{n}\sum_n I(f_i \leq t)$. It examines the $n(n\text{-}1)/2$ possible modal intervals $[t_L, t_U]$ between the sorted n individual observations. For all these combinations it computes in $\mathrm{O}(n)$ time the respective greatest convex minorant and the least concave majorant curves in $(min_t F_n, t_L)$ and $(t_U, max_t F_n)$, respectively. Fortunately, for a given F_n, the complexity of one dip computation is $\mathrm{O}(n)$ [157]. The computation of the p-value for a unimodality test uses bootstrap samples and expresses the probability of $dip(F_n)$ being less than the dip value of a cdf U_n^r of n observations sampled from the U[0,1] Uniform distribution:

$$\mathrm{P} = \frac{\#\left[dip(F_n) \leq dip(U_n^r)\right]}{b},\ r = 1, ..., b. \tag{6.2}$$

The null hypothesis H_0 that F_n is unimodal, is accepted at significance level α if p-value $> \alpha$, otherwise H_0 is rejected in favor of the alternative hypothesis H_1 which suggests multimodality.

Let a dataset $X=\{x_i : x_i \in \mathbb{R}^d\}_{i=1}^{N}$ then, in the present context, the dip test can be applied on any subset c, e.g. a data cluster, and more specifically on the ecdf ${F_n}^{(x_i)}(t)$ of the distances between a reference viewer x_i of c and the n members of the set:

$$F_n^{(x_i)}(t) = \frac{1}{n}\sum_{x_j \in c}\{Dist(x_i, x_j) \leq t\}. \tag{6.3}$$

We call the viewers that identify multimodality and vote for the set to split as *split viewers*. The dip-dist computation for a set c with n datapoint members is summarized as follows:

1. Compute U_n^r and the respective $dip(U_n^r)$, r=1, ..., b, for the Uniform sample distributions.
2. Compute $F_n^{(x_i)}$ and $dip(F_n^{(x_i)})$, i=1, ..., n, for datapoint viewers using the sorted matrix $Dist$.
3. Estimate the p-values $\mathrm{P}^{(x_i)}$, i=1, ..., n, based on Eq. 6.2 using a significance level α

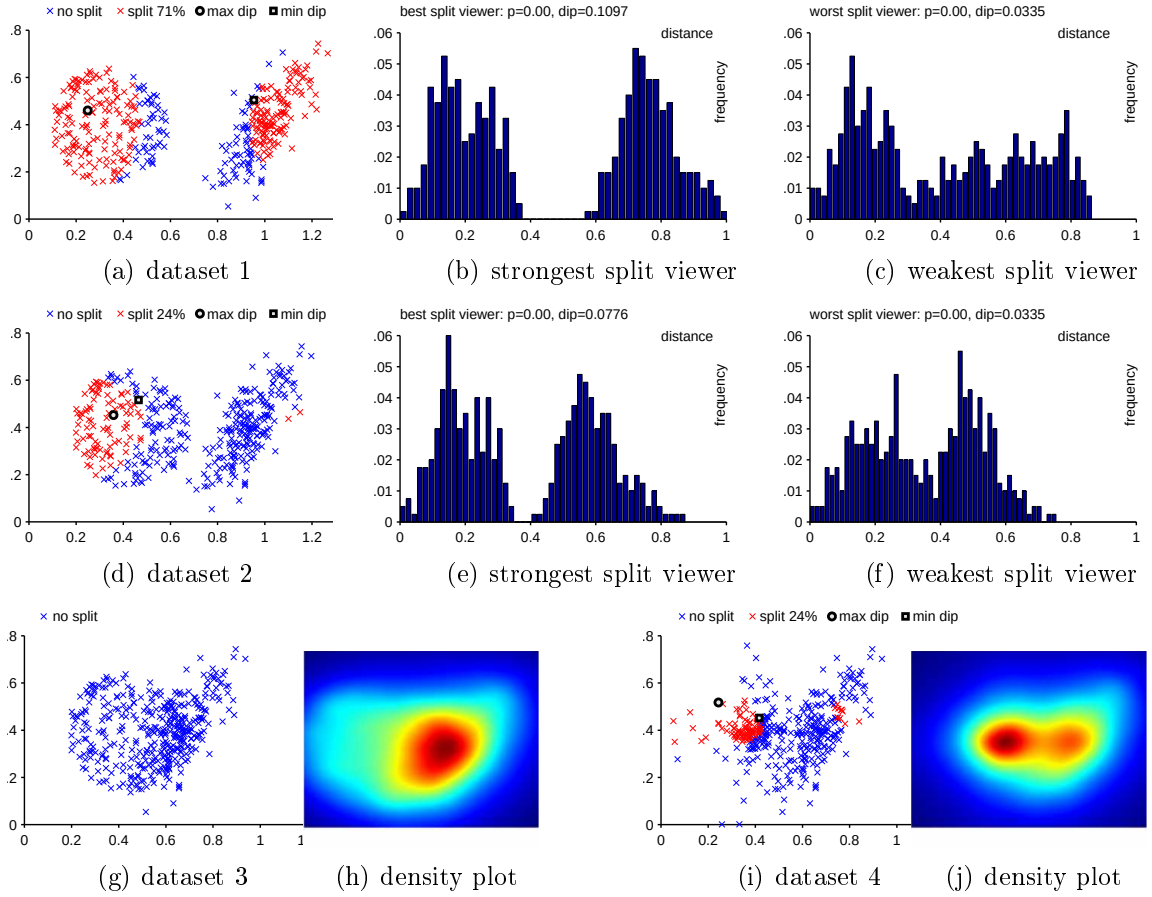


Figure 6.1: Application of dip-dist criterion on 2d synthetic data with two structures of 200 datapoints each. The split viewers are denoted in red color. (a) One Uniform spherical and one elliptic Gaussian structure. (b)(c) The histograms of pairwise distances corresponding to the strongest and weakest split viewer. (d) The two structures come closer; the split viewers are reduced, so does the dip value for the split viewer. (g) The two structures are no longer distinguishable as the density map in (h) shows one mode. (i) The Uniform spherical is replaced with a structure generated from a Student-t distribution.

and compute the percentage of viewers identifying multimodality.

Since the ascending ordering of the rows of $Dist$, required for computing $F_n^{(x_i)}$, can be done once during offline preprocessing, and that the same b samples of Uniform distribution can be used for testing all viewers, the dip-dist computation for a set with n datapoints has $\mathrm{O}(bn \log n + n^2)$ complexity.

Fig. 6.1 illustrates an example of applying the dip-dist criterion on synthetic data. We generated a Uniform spherical and a Gaussian elliptic structure, and then constructed three different two-dimensional datasets by decreasing the distance between them. The dip test parameters are set α=0 and b=1000. The histograms in each row indicate the

result of the dip test. As the structures come closer, the number of viewers that observe multimodality decreases. Eventually, the structures form a unimodal distribution (Fig. 6.1(g)), which may be visually verified from the presented density map. The fourth dataset of Fig. 6.1(j) was created by including a structure generated by a Student-t distribution centered at the same location where the sphere is located in Fig. 6.1(g). The respective density map shows clearly two modes, evidence that justifies why the dip-dist criterion determines multimodality with 24% of the viewers suggesting the split. More generally, if the percentage of split viewers is greater than a small threshold, e.g. 1%, we may decide that the cluster is multimodal.

6.3 The dip-means algorithm

Dip-means is an incremental clustering algorithm that combines three individual components. The first is a local search clustering technique that takes as input a model of k clusters and optimizes the model parameters. For this purpose k-means is used where the cluster models are their centroids. The second, and most important, decides whether a data subset contains multiple cluster structures using the dip-dist presented in Sec. 6.2. The third component is a divisive procedure (bisecting) that, given a data subset, performs the splitting into two clusters and provides the two centers.

Dip-means methodology takes as input the dataset X and two parameters for the dip-dist criterion: the significance level α and the percentage threshold v_{thd} of cluster members that should be split viewers to decide for a division (Algorithm 8). For the sake of generality, we assume that dip-means may start from any initial partition with $k_{init} \geq 1$ clusters. In each iteration, all k clusters are examined for unimodality, the set of split viewers v_j is found, and the respective cluster c_j is characterized as *split candidate* if $|v_j|/n_j \geq v_{thd}$. In this case, a non-zero score value is assigned to each cluster being a split candidate, while zero score is assigned to clusters that do not have sufficient split viewers. Various alternatives can be employed in order to compute a score for a split

candidate based on the percentage of split viewers, or even the size of clusters. In our implementation $score_j$ of a split candidate cluster c_j is computed as the average value of the dip statistic of its split viewers:

$$score_j = \begin{cases} \frac{1}{|v_j|} \sum_{x_i \in v_j} dip(F^{(x_i)}), & \frac{|v_j|}{n_j} \geq v_{thd} \\ 0 \qquad\qquad\qquad\qquad , & \text{otherwise.} \end{cases} \tag{6.4}$$

In order to avoid the overestimation of the real number of clusters, only the candidate with maximum score is split in each iteration. A cluster is split into two clusters using a 2-means local search approach starting from a pair of sufficiently diverse centroids m_L, m_R inside the cluster and concerning only the datapoints of that cluster. We use a simple way to set up the initial centroids $\{m_L, m_R\} \leftarrow \{x,\ \mu - (x - \mu)\}$, where x a cluster member selected at random and m the cluster centroid. In this way m_L, m_R lay at equal distances from m, though in opposite directions. The 2-means procedure can be repeated starting from different μ_L, μ_R initializations in order to discover a good split. A computationally more expensive alternative could be the deterministic *principal direction divisive partitioning* (PDDP) [82] that splits the cluster based on the principal component. We refine the solution at the end of each iteration using k-means, which fine-tunes the model of $k+1$ clusters. The procedure terminates when no split candidates are identified among the already formed clusters.

6.4 Extending dip-means in kernel space

The proposed dip-dist criterion uses only the pairwise distances, or similarities, between datapoints and not the vector representations themselves. This enables its application in kernel space Φ, provided a kernel matrix K with the $N \times N$ pairwise datapoint inner products, $K_{ij} = \phi(x_i)^T \phi(x_j)$. Algorithm 8 can be modified appropriately for this purpose. More specifically, *kernel dip-means* uses kernel k-means [155] as local search technique,

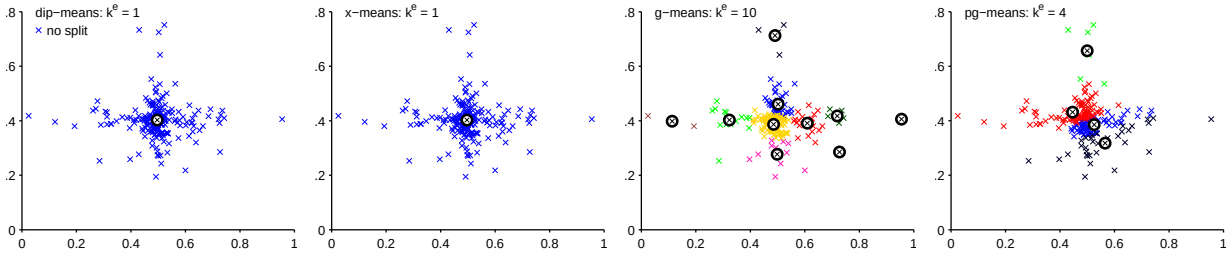
Algorithm 8 Dip-means (X, k_{init}, α, v_{thd})

input: dataset $X=\{x_i\}_{i=1}^{N}$, the initial number of clusters k_{init}, a statistic significance level α for the unimodality test, percentage v_{thd} of split viewers required for a cluster to be considered as a split candidate.
output: the sets of cluster members C=$\{c_j\}_{j=1}^{k}$, the models $M=\{\mu_j\}_{j=1}^{k}$ with the centroid of each c_j set.
let: $score$=unimodalityTest(c, α, v_{thd}) returns a score value for the cluster c,
{C, M}=kmeans(X, k) the k-means clustering,
{C, M}=kmeans(X, M) when initialized with model M,
$\{m_L, m_R\}$=splitCluster(c) that splits a cluster c and returns two centers μ_L, μ_R.

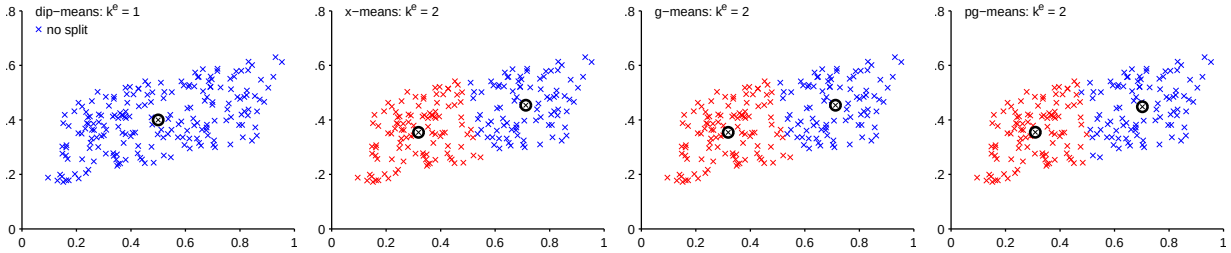
$k \leftarrow k_{init}$
{C, M} $\leftarrow$ kmeans(X, k)
do while changes in cluster number occur
  for j=1,...,k % *for each cluster j*
    $score_j \leftarrow$ unimodalityTest(c_j, α, v_{thd}) % *compute the score for unimodality test*
  end for
  if $\max_j(score_j) > 0$ % *there exist split candidates*
    target $\leftarrow$ argmax$_j$ ($score_j$) % *index of cluster to be split*
    $\{\mu_L, \mu_R\} \leftarrow$ splitCluster(c_{target})
    $M \leftarrow \{M\text{-}\mu_{target}, \mu_L, \mu_R\}$ % *replace the old centroid with the two new ones*
    {C, M} $\leftarrow$ kmeans(X, M) % *refine solution*
  end if
end do
return {C, M}

which also implies that centroids cannot be computed in kernel space, thus each cluster is now described explicitly by the set of its members c_j.

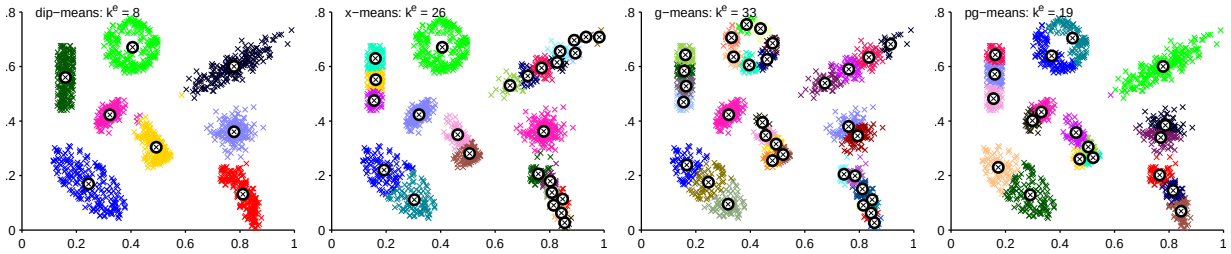
In this case, since the transformed data vectors $\phi(x)$ are not available, the cluster splitting procedure could be seeded by two arbitrary cluster members. However, we propose a more efficient approach. As discussed in Sec. 6.2, the distribution of pairwise distances between a reference viewer and the members of a cluster reveals information about the multimodality of data distribution in the original space. This implies that a split of the cluster members based on their distance to a reference viewer constitutes a reasonable split in the original space, as well. To this end, we may use 2-means to split the elements of the one-dimensional similarity vector. We consider as reference split viewer the cluster member with the maximum dip value. Here, 2-means is seeded using two values located at opposite positions with respect to the distribution's mean. After convergence, the resulting two-way partition of the datapoints, derived by the partition of the corresponding similarity values to the selected reference split viewer, initializes a local search with kernel 2-means.



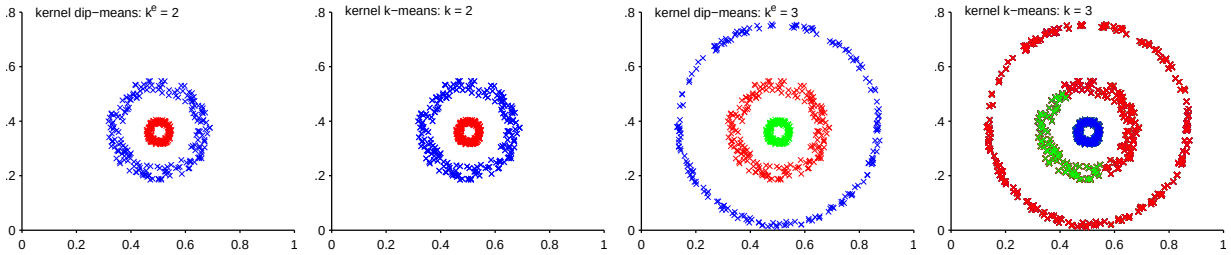
(a) Single structure generated by a Student-t distribution



(b) Single Uniform rectangle structure



(c) Eight clusters of various density and shape



(d) Two Uniform ring structures

(e) Three Uniform ring structures

Figure 6.2: Clustering results on 2d synthetic unimodal cluster structures with 200 datapoints each (the centroids are marked with ⊗). (a)(b) Single cluster structures. (c) Various structure types. Based on the leftmost subfigure, it contains a Uniform rectangle (green), a sphere with increasing density at its periphery (light green), two Gaussian structures (black, pink), a Uniform ellipse (blue), a triangle denser at a corner (yellow), a Student-t (light blue), and a Uniform arbitrary shape (red). (d)(e) Non-linearly separable ring clusters (kernel-based clustering with an RBF kernel).

6.5 Experiments

In our evaluation we compare the proposed dip-means method with x-means [143], g-means [145] and pg-means [146] that are closely related to present work. In all compared

Table 6.1: Results for synthetic datasets with fixed k^*=20 clusters with 200 datapoints in each cluster.

	Case 1, d=4			Case 1, d=16			Case 1, d=32		
Methods	k^e	ARI	VI	k^e	ARI	VI	k^e	ARI	VI
dip-means	20.0±0.0	1.00±0.0	0.00±0.0	20.0±0.0	1.00±0.0	0.00±0.0	20.0±0.0	1.00±0.0	0.00±0.0
x-means	7.3±9.3	0.30±0.5	2.07±1.3	28.6±7.8	0.88±0.1	0.27±0.2	31.3±5.6	0.84±0.1	0.36±0.2
g-means	20.3±0.5	0.99±0.0	0.01±0.0	20.3±0.5	0.99±0.0	0.01±0.0	20.5±0.6	0.99±0.0	0.02±0.0
pg-means	19.2±2.5	0.90±0.1	0.16±0.2	19.0±0.9	0.95±0.1	0.07±0.1	3.2±5.1	0.09±0.2	2.62±0.9
	Case 2, d=4			Case 2, d=16			Case 2, d=32		
Methods	k^e	ARI	VI	k^e	ARI	VI	k^e	ARI	VI
dip-means	20.0±0.0	0.99±0.0	0.05±0.0	20.0±0.0	0.99±0.0	0.02±0.0	20.0±0.0	0.99±0.0	0.01±0.0
x-means	24.8±39.	0.26±0.4	2.26±1.1	80.1±15.	0.75±0.1	0.75±0.2	71.6±14.	0.75±0.1	0.66±0.2
g-means	79.2±22.	0.77±0.1	0.70±0.2	105.9±30.	0.83±0.1	0.66±0.2	133.6±42.	0.83±0.1	0.72±0.2
pg-means	14.2±4.7	0.67±0.2	0.65±0.5	10.4±3.4	0.30±0.2	1.26±0.5	4.0±1.5	0.06±0.1	2.40±0.2

methods we use the same incremental cluster split and local searching strategy as adopted in Algorithm 8 that starts with a single cluster (k_{init}=1) and:

i) at each iteration one cluster is selected for a bisecting split,

ii) 10 split trials are performed with 2-means initialized with the simple technique described in Sec. 6.3, and the split with lower clustering error (the sum of squared differences between cluster centers and their assigned datapoints) is kept,

iii) the refinement is applied after each iteration on all k+1 clusters.

Hence, only the statistical test that decides whether to stop splitting differs in each case. Exception is the pg-means method that uses EM for local search and does not rely on cluster splitting to add a new cluster. We use the method exactly as presented in [146]. For the kernel-based experiments we use the necessary modifications described at the end of Sec. 6.3 and compare with kernel k-means [155]. The parameters of the dip-dist criterion are set as α=0 for significance level of dip test and b=1000 for the number of bootstraps. We consider as split candidates the clusters having at least v_{thd}=1% split viewers. These values were fixed in all experiments. For both g-means and pg-means we set the significance level α=0.001, while we use 12 random projections for the latter. In order to compare the ground truth labeling and the grouping produced by clustering, we utilize the *variation of information* (VI) metric [100] and the *adjusted rand index* (ARI) [101]. Better clustering is indicated by lower values of VI and higher for ARI.

We first provide clustering results for synthetic 2d datasets in Fig. 6.2 (k^e denotes the estimated number of clusters). In Fig. 6.2(a)(b) we provide two indicative examples of single cluster structures. X-means decides correctly for the structure generated from Student-t distribution, but overfits in the Uniform rectangle case, while the other two methods overfit in both cases. In the multicluster dataset of Fig. 6.2(c) dip-means successfully discovers all clusters, in contrast to the other methods that significantly overestimate. To test the kernel dip-means extension, we created two 2d synthetic dataset containing two and three Uniform ring structures and we used an RBF kernel to construct the kernel matrix K. It is clear that x-means, g-means, and pg-means are not applicable in this case. Thus we present in Fig. 6.2(d)(e) the results using kernel dip-means and also the best solution from 50 randomly initialized runs of kernel k-means with the true number of clusters. As we may observe, dip-means estimates the true number of clusters and finds the optimal grouping of datapoints in both cases, whereas kernel k-means fails in the three ring case. Furthermore, we created synthetic datasets with true number k^*=20 clusters, with 200 datapoints each, in d= 4, 16, 32 dimensions with low separation [158]. Two cases were considered:

1. Gaussian mixtures of varying eccentricity, and
2. datasets with various cluster structures, i.e. Gaussian (40%), Student-t (20%), Uniform ellipses (20%) or Uniform rectangles (20%). For each case and dimensions, we generated 30 datasets to test the methods.

As the results in Tab. 6.1 indicate, dip-means provides excellent clustering performance in all cases and estimates accurately the true number of clusters. Moreover, it performs remarkably better than the other methods, especially for the datasets of Case 2.

Real-world datasets were also used, where the provided class labels were considered as ground truth. Handwritten Pendigits (UCI) [101] contains 16 dimensional vectors, each one representing a digit from 0-9 written by a human subject. The data provide a training PD_{tr} and a testing set PD_{te} with 7494 and 3498 instances, respectively. We also consider two subsets that contain the digits $\{0, 2, 4\}$ (PD3_{tr} and PD3_{te}) and $\{3, 6, 8, 9\}$ (PD4_{tr} and PD4_{te}). We do not apply any preprocessing. Coil-100 is the second dataset [159],

Table 6.2: Clustering results for real-world data. Bold indicates best values.

	$PD3_{te}$ (k^*=3)			$PD4_{te}$ (k^*=4)			$PD10_{te}$ (k^*=10)		
Methods	k^e	ARI	VI	k^e	ARI	VI	k^e	ARI	VI
dip-means	3	**0.879**	**0.332**	4	**0.626**	**0.545**	7	0.343	**1.587**
x-means	155	0.031	3.792	194	0.039	3.723	515	0.041	3.825
g-means	21	0.226	1.800	36	0.209	2.049	73	0.295	1.961
pg-means	4	0.835	0.359	10	0.576	0.954	13	**0.447**	1.660
	$PD3_{tr}$ (k^*=3)			$PD4_{tr}$ (k^*=4)			$PD10_{tr}$ (k^*=10)		
Methods	k^e	ARI	VI	k^e	ARI	VI	k^e	ARI	VI
dip-means	3	**0.963**	**0.116**	4	**0.522**	**0.841**	9	0.435	**1.452**
x-means	288	0.018	4.378	381	0.020	4.372	942	0.024	4.387
g-means	52	0.106	2.641	58	0.143	2.464	149	0.160	2.605
pg-means	5	0.655	0.740	8	0.439	1.320	14	**0.494**	1.504
	Coil3 (k^*=3)			Coil4 (k^*=4)			Coil5 (k^*=5)		
Methods	k^e	ARI	VI	k^e	ARI	VI	k^e	ARI	VI
dip-means	3	**1.000**	**0.000**	5	**0.912**	**0.173**	4	**0.772**	**0.308**
x-means	8	0.499	0.899	11	0.499	0.951	15	0.601	0.907
g-means	7	0.669	0.650	12	0.502	0.977	18	0.434	1.204

which contains 72 images taken from different angles for each one of the 100 included objects. We used tree subsets Coil3, Coil4, Coil5, with images from 3, 4 and 5 objects, respectively[1]. *SIFT descriptors* [5] are first extracted from the greyscale images that are finally represented by the *bag of visual words* model using 1000 visual words.

We also considered three subsets of text document data taken from the 20-Newsgroups collection[2]. All datasets have 200 document vectors per included category. Each document is encoded using the *bag of words* representation. In particular, TD_1 contains documents from each of the first three categories of the collection with 4271 term features. TD_2 contains documents from categories 6, 7, 11, 13, and 19 with 5492 features and, finally, TD_3 includes TD_2 while additionally contains documents from the categories 1, 2 that altogether have 8280 term features.

As reported in Tab. 6.2, dip-means correctly discovers the number of clusters for the subsets of Pendigits, while providing a reasonable underestimation k^e near the optimal for the full datasets $PD10_{tr}$ and $PD10_{te}$. Apart from the excessive overfitting of x-means and g-means, pg-means seems to concludes in overestimated k^e. In the high dimensional and

[1]These objects are also included in the Coil20 subset [160].
[2]Available at: http://people.csail.mit.edu/jrennie/20Newsgroups/

Table 6.3: Clustering results for text data. Bold indicates best values.

	TD_1 (k^*=3)			TD_2 (k^*=5)			TD_3 (k^*=8)		
Methods	k^e	ARI	VI	k^e	ARI	VI	k^e	ARI	VI
dip-means	3	**0.464**	**0.793**	4	**0.561**	**0.670**	6	**0.333**	**1.184**
x-means	1	0.000	1.099	2	0.192	1.972	3	0.249	1.384
pg-means	1	0.000	1.099	10	0.271	2.243	–	–	–

sparse space of the considered Coil subsets, x-means and g-means provide more reasonable k^e estimations, but still overestimations. An explanation for this behavior is that they discover smaller groups of similar images, i.e. images taken from close angles to the same object, but fail to unify the subclusters at higher level. Note also that we did not manage to test pg-means in Coil-100 subsets, since covariance matrices were not positive definite.

In what concerns text data, the experimental results are reported in Tab. 6.3. We did not include results for g-means that overfitted in all cases and required too much running time to terminate. In fact, in all cases g-means stopped due to the empirical stopping criterion that we set in order to stop splitting very small clusters. Dip-means seems to provide better estimates for the number of clusters, however, in two cases it provided an underestimation and only for the smaller of the datasets it gave a correct estimation. We should note that, these are high dimensional and sparse datasets, and we provided them as input to the methods without any projection (dimensionality reduction) that could possibly make clearer the cluster structure. Pg-means failed for the last dataset, since the covariance matrices were not positive definite. The superiority of dip-means is also indicated by the reported values for ARI and VI measures.

6.6 Conclusions

We have presented a novel approach for testing whether multiple cluster structures are present in a set of data objects (e.g. a data cluster). The proposed *dip-dist criterion* checks for unimodality of the empirical data density distribution, thus it is much more general

compared to alternatives that test for Gaussianity. Dip-dist uses a statistical hypothesis test, namely Hartigans' dip test, in order to verify unimodality. If a data object of the set is considered as a *viewer*, then the dip test can be applied on the one-dimensional distance (or similarity) vector with components the distances between the viewer and the members of the same set. We exploit the idea that the observation of multimodality in the distribution of distances indicates multimodality of the original data distribution. By considering all the data objects of the set as individual viewers and by combining the respective results of the test, the presence of multiple cluster structures in the set can be determined.

We have also proposed a new incremental clustering algorithm called *dip-means*, that incorporates dip-dist criterion in order to decide for cluster splitting. The procedure starts with one cluster, it iteratively splits the cluster indicated by dip-dist as more probable to contain multiple cluster structures, and terminates when no new cluster split is suggested. By taking advantage of the fact that dip-dist utilizes only information about the distances between data objects, we have modified appropriately the main algorithm to propose *kernel dip-means* which can be applied in kernel space.

The proposed method is fast, easy to implement, and works very well under a fixed parameter setting. The reported clustering results indicate that dip-means can provide reasonable estimates of the number of clusters, and produce meaningful clusterings in both dataset types in a variety of artificial and real datasets.

Chapter 7

Conclusions

7.1 Conclusions and future work

In this thesis we have studied and developed machine learning and data mining methods for extracting knowledge from document collections. More specifically, we focused on the problem of document clustering, which is an unsupervised approach for the extraction of information regarding the cluster structure of a dataset. The motivation of this dissertation was to design novel and efficient methodologies for document representation and clustering that take into account the particular characteristics of tet documents.

First, in Chapter 2 an extensive discussion has been included on the special properties of the natural languages and the ways that text documents are transformed and represented as feature vectors. Such feature spaces are characterized by the *high dimensionality and sparsity* (HDS) which in turn impose difficulties when typical clustering methods are applied.

In Chapter 3 we revisited the oversimplistic *term independence assumption* that is considered in most vector space models (VSM) used for document representation. Specif-

ically, we presented an extension to VSM approach for text document representation called *global term context vector model* (GTCVM). The main contribution of the method is that it proposes a way to compute the similarity between two term features based on the local context in which those terms appear in the term sequences of documents. In this way, the bag of words (BOW) document vectors were mapped onto a new feature space spanned by *term similarity vectors*. The method proceeds as follows: i) it captures local contextual information for each term occurrence in the term sequences of documents; ii) the local contexts for the occurrences of a term are combined to define the global context of that term; iii) using the global context of all terms a proper semantic matrix is constructed; iv) this matrix is further used to linearly map traditional BOW document vectors onto a '*semantically smoothed*' feature space. In the experimental study, we employed this vector mapping to verify the impact of the smoothed feature space in the text document clustering problem using standard algorithms such as k-means and spectral clustering. The results demonstrated the improvement of clustering solutions when the proposed GTCVM representation was used compared to traditional VSM-based approaches like BOW, or other techniques that also try to exploit the contextual information of terms to define dependencies between them.

In Chapter 4, we investigated the centroid-based cluster representation for HDS data. *Synthetic cluster prototypes* were proposed for representing a cluster of HDS vectors, such as document representations. This novel prototype is computed by i) first selecting a subset of the objects in the cluster, then ii) computing the representative of these objects and, finally, iii) selecting important features. We also proposed the *MedoidKNN* synthetic prototype that favors the representation of the dominant class in a cluster. In the proposed robust clustering method called *k-synthetic prototypes* (*k-sp*), we incorporated the aforementioned cluster prototypes. This algorithm extends the spherical k-means algorithm. We also discussed the k-sp property of searching a proper subspace for each cluster. This property derives from the fact that a set of HDS vectors is necessarily described by an equal or larger number of non-zero dimensions compared to any of its subsets. In this sense, the proposed method exhibits some similar characteristics to subspace clustering.

This approach was experimentally tested against various widely-used clustering methods such as spherical k-means (using a series of initialization schemes), spectral cluster, and subspace clustering algorithms. The clustering results indicated the robustness of k-sp, especially for small datasets and clusters overlapping in many dimensions, while superior performance was observed to that of the compared clustering approaches.

In Chapter 5, we studied the document clustering problem using the incremental approach to build the final partition and the respective representatives. This approach begins from a small number of clusters and, at each incremental step, it introduces one new cluster in the solution computed so far until a desired number of clusters have been formed. Our first contribution in this chapter was to provide a detailed presentation of the related significant advances, and then to present a *framework for prototype-based incremental clustering*. Our framework introduces the idea of *partial updates* (PU) on a given solution. A partial update is defined by the *activity state* (active or inactive) of subsets of clusters, prototypes, and data objects. An *active* entity could participate in the iterations of the clustering procedure. For instance, an active data object may move among the active clusters that compete to each other in order to attract more objects. Moreover, an active prototype may be updated when the members of its cluster are modified during iterations, however, an active cluster does not always have an active prototype. We have shown that several known prototype-based incremental clustering approaches can be considerer as special cases of the proposed framework which offers a good summarization of such approaches. Next, we stressed the '*unfair prototype competition*' problem, which implies that the new prototype which is introduced at each incremental step is difficult to be competitive to the already formed prototypes. To address this issue, which is due to the HDS property of the data, we proposed a *reduction-enrichment mechanism* (REM) aiming to make more sparse the formed cluster prototypes and, at the same time, help the new prototype to become a strong attractor for the active objects. These new ideas have been incorporated in the incremental *global k-synthetic prototypes* (g$k$sp) clustering method with good results.

In Chapter 6, we considered a key problem in data clustering: the estimation of the

number of clusters in a dataset. To this end, we presented *dip-means*, a novel robust incremental method that learns the number of data clusters. The only assumption of dip-means is the fundamental cluster property that each cluster should admit a unimodal distribution. This is much more general than the assumptions made by other methods, such as the Gaussianity assumption. Dip-means employs the novel *dip-test* which is a univariate statistic hypothesis test for unimodality. This test is applied on the distances between an object of the cluster, considered as '*viewer*', and the cluster members. If there exist some viewers that give evidence for multimodality according to the dip-test, then the cluster is considered multimodal and should be further split. From the statistical and computational point of view, it is important that the unimodality test is applied on univariate distance vectors, and that if the similarity matrix is provided, then the actual data vectors are not necessarily required. These characteristics make dip-test applicable in combination with many standard incremental, or kernel-based methods since only the pairwise distances are involved in the computations. We experimentally compared dip-means with several other methods and the results indicated that dip-means provides better estimations of the number of clusters, especially for non-Gaussian data.

7.2 Directions for future work

For the document representation problem, a possible research direction is to investigate the potential of combining the local and global term contextual information in order to build compact concept vectors and hence to efficiently project the transformed document vectors in feature spaces of lower dimensionality. We also intend to perform a systematic study for procedures that could efficiently compute a different parameter value for each individual vocabulary term which, in term, could improve the global term context vectors. Additionally, document-term co-clustering is another interesting problem where term similarities may apply. Finally, we aim at examining the proposed representation for document classification.

Regarding the k-sp method, a direction for future work is to develop techniques for the automated specification of p_{obj}. Alternatively, we could extend the feature selection procedure to a continuous weighting scheme, instead of the current binary weighting. It is interesting to investigate the possibility of developing a gradual adjustment of the k-sp parameters aiming to achieve a gradual change of the prototype behavior from medoid-like to centroid-like. This would also eliminate the separate refinement phase. Moreover, the proposed method could be tested in other related problems, such as term selection for cluster summarization, organization of noisy document collections, online document clustering, and semi-supervised document clustering.

In what concerns incremental prototype-based document clustering, it is possible to further exploit the proposed framework in order to develop more efficient clustering methods based on the presented design principles. The global k-sp could also take advantage of any future improvement of the original k-sp algorithm. Worth to note that it is of great value to investigate ways to reduce the computational cost of such incremental clustering approaches so that they become applicable in large-scale problems.

Dip-means is a very promising methodology for estimating the number of clusters of a dataset that deserves further study and investigation. Further study could be done concerning how to adjust the values of the parameter used in the dip-test. The proposed method for unimodality cluster testing could also be used as a validation technique in cluster analysis problems. Moreover, apart from testing dip-dist in real-world applications, there are several ways to improve the implementation details, especially in the kernel-based version. We also plan to test its effectiveness in other settings such as online clustering of stream data.

As an overall remark we believe that there is also clear potential to combine the methods developed and presented in this thesis. For instance, dip-test could be used to examine the characteristics of a cluster in order to adjust properly its synthetic prototype. Such combinations could be an important direction for further study.

Bibliography

[1] G. Salton, A. Wong, C. Yang, A vector space model for automatic indexing, Communications of the ACM 18 (11) (1975) 613–620.

[2] G. Salton, Automatic text processing: the transformation, analysis, and retrieval of information by computer, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[3] F. Jurie, B. Triggs, Creating efficient codebooks for visual recognition, in: Proceedings of the 10th IEEE International Conference on Computer Vision, Vol. 1 of ICCV, IEEE Computer Society, Washington, DC, USA, 2005, pp. 604–610.

[4] A. Mikulík, M. Perdoch, O. Chum, J. Matas, Learning a fine vocabulary, in: Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III, ECCV, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 1–14.

[5] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.

[6] J. C. van Gemert, C. J. Veenman, A. W. M. Smeulders, J.-M. Geusebroek, Visual word ambiguity, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (7) (2010) 1271–1283.

[7] M. S. Lew, N. Sebe, C. Djeraba, R. Jain, Content-based multimedia information retrieval: State of the art and challenges, ACM Transactions on Multimedia Computing, Communications and Applications 2 (1) (2006) 1–19.

[8] V. Chasanis, A. Kalogeratos, A. Likas, Movie segmentation into scenes and chapters using locally weighted bag of visual words, in: Proceedings of the ACM International Conference on Image and Video Retrieval, CIVR, 2009, pp. 35:1–35:7.

[9] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: spatial pyramid matching for recognizing natural scene categories, in: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2 of CVPR, IEEE Computer Society, Washington, DC, USA, 2006, pp. 2169–2178.

[10] P. Tirilly, V. Claveau, P. Gros, Language modeling for bag-of-visual words image categorization, in: Proceedings of the 2008 International Conference on Content-based Image and Video Retrieval, CIVR, ACM, New York, NY, USA, 2008, pp. 249–258.

[11] J. Illig, A. Hotho, R. Jäschke, G. Stumme, A comparison of content-based tag recommendations in folksonomy systems, in: Proceedings of the 1st International Conference on Knowledge Processing and Data Analysis, KONT/KPP, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 136–149.

[12] Y.-T. Lu, S.-I. Yu, T.-C. Chang, J. Y.-j. Hsu, A content-based method to enhance tag recommendation, in: Proceedings of the 21st International joint Conference on Artifical Intelligence, IJCAI, Morgan Kaufmann, San Francisco, CA, USA, 2009, pp. 2064–2069.

[13] T. M. Mitchell, Machine Learning, 1st Edition, McGraw-Hill, New York, NY, USA, 1997.

[14] C. M. Bishop, Pattern recognition and machine learning (Information science and statistics), 1st Edition, Springer, 2007.

[15] C. M. Bishop, J. Lasserre, Generative or discriminative? getting the best of both worlds, Bayesian Statistics 8 (2007) 3–24.

[16] Y. Yang, J. O. Pedersen, A comparative study on feature selection in text categorization, in: Proceedings of the 14th International Conference on Machine Learning, ICML, 1997, pp. 412–420.

[17] N. M. Wanas, D. A. Said, N. H. Hegazy, N. M. Darwish, A study of local and global thresholding techniques in text categorization, in: Proceedings of the 5th Australasian Conference on Data Mining and Analystics, Vol. 61 of AusDM, 2006, pp. 91–101.

[18] Y. Zhang, Z.-H. Zhou, Multilabel dimensionality reduction via dependence maximization, ACM Transactions Knowledge Discovery from Data 4 (2010) 14:1–14:21.

[19] P. Mitra, C. A. Murthy, S. K. Pal, Unsupervised feature selection using feature similarity, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (3) (2002) 301–312.

[20] N. Wiratunga, R. Lothian, S. Massie, Unsupervised feature selection for text data, in: Proceedings of the 8th European Conference on Case-Based Reasoning, ECBR, 2006, pp. 340–354.

[21] Q. Wu, Y. Ye, M. Ng, H. Su, J. Huang, Exploiting word cluster information for unsupervised feature selection, in: Proceedings of the 11th Pacific Rim International Conference on Artificial Intelligence, PRICAI, 2010, pp. 292–303.

[22] D. Cai, C. Zhang, X. He, Unsupervised feature selection for multi-cluster data, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 333–342.

[23] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, R. Harshman, Indexing by latent semantic analysis, Journal of the American Society for Information Science 41 (1990) 391–407.

[24] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent Dirichlet allocation, Journal of Machine Learning Research 3 (2003) 993–1022.

[25] I. Steinwart, Support vector machines, Springer, New York, 2008.

[26] Y. Yang, X. Liu, A re-examination of text categorization methods, in: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR, 1999, pp. 42–49.

[27] C. C. Aggarwal, C. Zhai, A survey of text classification algorithms, in: C. C. Aggarwal, C. Zhai (Eds.), Mining Text Data, Springer, 2012, pp. 163–222.

[28] A. Genkin, D. D. Lewis, D. Madigan, Large-scale bayesian logistic regression for text categorization, Technometrics 49 (3) (2007) 291–304.

[29] J. W. Tukey, We need both exploratory and confirmatory, The American Statistician 34 (1) (1980) 23–25.

[30] M. Mahajan, P. Nimbhorkar, K. Varadarajan, The planar k-means problem is np-hard, Theoretical Computer Science 442 (2012) 13–21.

[31] D. Aloise, A. Deshpande, P. Hansen, P. Popat, Np-hardness of euclidean sum-of-squares clustering, Machine Learning 75 (2) (2009) 245–248.

[32] B. Mirkin, Choosing the number of clusters, Data Mining and Knowledge Discovery 1 (3) (2011) 252–260.

[33] N. Srebeo, G. Shakhnarovich, S. Roweis, When is clsutering hard?, in: PASCAL Workshop on Statistics and Optimization of Clustering, 2005.

[34] M. Ackerman, S. Ben-David, Clusterability: A theoretical study, Journal of Machine Learning Research - Proceedings Track 5 (2009) 1–8.

[35] U. von Luxburg, R. C. Williamson, I. Guyon, Clustering: Science or art?, Journal of Machine Learning Research - Proceedings Track 27 (2012) 65–80.

[36] J. M. Kleinberg, An impossibility theorem for clustering, in: Proceedings of the 16th Annual Conference on Neural Information Processing Systems, NIPS, 2002, pp. 446–453.

[37] L. Parsons, E. Haque, H. Liu, Subspace clustering for high dimensional data: a review, ACM SIGKDD Explorations Newsletter 6 (1) (2004) 90–105.

[38] R. Xu, D. Wunsch, II, Survey of clustering algorithms, Transaction on Neural Networks 16 (3) (2005) 645–678.

[39] P. Berkhin, A survey of clustering data mining techniques, in: J. Kogan, C. Nicholas, M. Teboulle (Eds.), Grouping Multidimensional Data, Springer, Berlin, Heidelberg, 2006, pp. 25–71.

[40] H.-P. Kriegel, P. Kröger, A. Zimek, Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering, ACM Transactions Knowledge Discovery from Data 3 (1) (2009) 1–58.

[41] I. Dhillon, D. Modha, Concept decompositions for large sparse text data using clustering, Machine Learning 42 (1) (2001) 143–175.

[42] Y. Zhao, G. Karypis, U. Fayyad, Hierarchical clustering algorithms for document datasets, Data Mining and Knowledge Discovery 10 (2) (2005) 141–168.

[43] A. Kalogeratos, A. Likas, Text document clustering using global term context vectors, Knowledge and Information Systems 31 (3) (2012) 455–474.

[44] A. Kalogeratos, A. Likas, Document clustering using synthetic cluster prototypes, Data and Knowledge Engineering 70 (3) (2011) 284–306.

[45] A. Kalogeratos, A. Likas, Dip-means: an incremental clustering method for estimating the number of clusters, in: Proceedings of the 26th Annual Conference on Neural Information Processing Systems, NIPS, 2012.

[46] A. Purandare, T. Pedersen, Discriminating among word meanings by identifying similar contexts, in: Proceedings of the 19th National Conference on Artificial Intelligence, San Jose, CA, USA, 2004, pp. 964–965.

[47] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, K. Miller, Wordnet: An on-line lexical database, International Journal of Lexicography 3 (1990) 235–244.

[48] Wikipedia, Wikipedia, the free encyclopedia (2004).

[49] J. Jing, L. Zhou, M. Ng, Z. Huang, Ontology-based distance measure for text clustering, in: Proceedings SIAM SDM Workshop on Text Mining, 2006.

[50] C. Chen, F. Tseng, T. Liang, An integration of fuzzy association rules and wordnet for document clustering, Knowledge and Information Systems 28 (3) (2010) 687–708.

[51] P. Wang, C. Domeniconi, Building semantic kernels for text classification using wikipedia, in: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, 2008, pp. 713–721.

[52] I. Dhillon, Y. Guan, J. Fan, Efficient clustering of very large document collections, in: V. Grossman, C. Kamath, R. Namburu (Eds.), Data Mining for Scientific and Engineering Applications, Kluwer, 2001, pp. 357–381.

[53] G. Zipf, The psycho-biology of language, an introduction to dynamic philology, MIT Press, Cambridge, MA, USA, 1936.

[54] H. Heaps, Information Retrieval: Computational and Theoretical Aspects, Academic Press, Inc., Orlando, FL, USA, 1978.

[55] L. Lu, Z.-K. Zhang, T. Zhou, Zipf's law leads to heaps' law: Analyzing their relation in finite-size systems, PLoS ONE 5 (12) (2010) e14139.

[56] I. I. Eliazar, M. H. Cohen, Power-law connections: From zipf to heaps and beyond, Annals of Physics 332 (0) (2012) 56–74.

[57] L. Lu, Z.-K. Zhang, T. Zhou, Deviation of zipf's and heaps' laws in human languages with limited dictionary sizes, in: Scientific Reports, 2013. doi:10.1038/srep01082.

[58] M. Porter, An algorithm for suffix stripping, in: K. Jones, P. Willett (Eds.), Readings in Information Retrieval, Morgan Kaufmann, San Francisco, CA, USA, 1997, pp. 313–316.

[59] R. Krovetz, Viewing morphology as an inference process, in: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR, ACM, New York, NY, USA, 1993, pp. 191–202.

[60] D. Sharma, Stemming algorithms: A comparative study and their analysis, International Journal of Applied Information Systems 4 (3) (2012) 7–12.

[61] W. N. Francis, H. Kucera, A. W. Mackie, Frequency analysis of english usage: Lexicon and grammar, Journal of the Dictionary Society of North America.

[62] R. Lo, B. He, I. Ounis, Automatically building a stopword list for an information retrieval system, in: Proceedings of the 5th Dutch-Belgium Retrieval Workshop, 2005, pp. 964–965.

[63] M. Makrehchi, M. S. Kamel, Automatic extraction of domain-specific stopwords from labeled documents, in: Proceedings of the 30th European Conference on Information Retrieval Research, ECIR, 2008, pp. 222–233.

[64] D. D. Lewis, Feature selection and feature extraction for text categorization, in: Proceedings of the Workshop on Speech and Natural Language, HLT, Association for Computational Linguistics, Stroudsburg, PA, USA, 1992, pp. 212–217.

[65] A. Schenker, H. Last, M. amd Bunke, A. Kandel, Clustering of web documents using a graph model, in: A. Antonacopoulos, J. Hu (Eds.), Web Document Analysis: Challenges and Opportunities, World Scientific Publishing Company, 2003, pp. 3–18.

[66] K. M. Hammouda, M. S. Kamel, Efficient phrase-based document indexing for web document clustering, IEEE Transactions on Knowledge and Data Engineering 16 (10) (2004) 1279–1296.

[67] A. Kalogeratos, A. Likas, A significance-based graph model for clustering web documents, in: Proceedings of the 4th Helenic Conference on Advances in Artificial Intelligence, SETN, 2006, pp. 516–519.

[68] C. Manning, P. Raghavan, H. Schutze, Introduction to Information Retrieval, Cambridge University Press, Cambridge, MA, USA, 2008.

[69] S. Wong, W. Ziarko, P. Wong, Generalized vector spaces model in information retrieval, in: Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR, ACM, New York, NY, USA, 1985, pp. 18–25.

[70] A. Banerjee, S. Merugu, I. S. Dhillon, J. Ghosh, Clustering with bregman divergences, Journal of Machine Learning Research 6 (2005) 1705–1749.

[71] J. Ghosh, A. Strehl, Similarity-based text clustering: A comparative study, in: J. Kogan, C. Nicholas, M. Teboulle (Eds.), Grouping Multidimensional Data, Springer, Berlin, Heidelberg, 2006, pp. 73–97.

[72] X. Wang, A fast exact k-nearest neighbors algorithm for high dimensional search using k-means clustering and triangle inequality, in: Proceedings of the 2011 International Joint Conference on Neural Networks, IJCNN, 2011, pp. 1293–1299.

[73] M. Kryszkiewicz, Determining cosine similarity neighborhoods by means of the euclidean distance, in: A. Skowron, Z. Suraj (Eds.), Rough Sets and Intelligent Systems, Vol. 43 of Intelligent Systems Reference Library, Springer, Berlin, Heidelberg, 2013, pp. 323–345.

[74] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, Communications of the ACM 51 (1) (2008) 117–122.

[75] D. Metzler, S. Dumais, C. Meek, Similarity measures for short segments of text, in: Proceedings of the 29th European Conference on Information Retrieval Research, ECIR, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 16–27.

[76] P. Bradley, U. Fayyad, Refining initial points for k-means clustering, in: Proceedings of the 15th International Conference on Machine Learning, ICML, 1998, pp. 91–99.

[77] J. M. Peña, J. A. Lozano, P. Larrañaga, An empirical comparison of four initialization methods for the k-means algorithm, Pattern Recognition Letters 20 (10) (1999) 1027–1040.

[78] D. Arthur, S. Vassilvitskii, k-means++: the advantages of careful seeding, in: Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete algorithms, SODA, 2007, pp. 1027–1035.

[79] R. Maitra, Initializing partition-optimization algorithms, IEEE/ACM Transactions on Computational Biology and Bioinformatics 6 (1) (2009) 144–157.

[80] M. E. Celebi, H. A. Kingravi, P. A. Vela, A comparative study of efficient initialization methods for the k-means clustering algorithm, Expert Systems with Applications 40 (1) (2013) 200–210.

[81] L. Kaufman, P. J. Rousseeuw, Finding groups in data: An introduction to cluster analysis, John Wiley, 1990.

[82] D. Boley, Principal direction divisive partitioning, Data Mining and Knowledge Discovery 2 (4) (1998) 325–344.

[83] S. M. Savaresi, D. L. Boley, A comparative analysis on the bisecting k-means and the pddp clustering algorithms, Intelligent Data Analysis 8 (4) (2004) 345–362.

[84] A. Likas, N. Vlassis, J. J. Verbeek, The global k-means clustering algorithm, Pattern Recognition 36 (2) (2003) 451–461.

[85] A. Ng, M. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: Proceedings of the 15th Annual Conference on Neural Information Processing Systems, Vol. 14 of NIPS, 2001, pp. 849–864.

[86] T. Hofmann, Probabilistic latent semantic indexing, in: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR, ACM, New York, NY, USA, 1999, pp. 50–57.

[87] D. M. Blei, T. L. Griffiths, M. I. Jordan, J. B. Tenenbaum, Hierarchical topic models and the nested chinese restaurant process, in: Proceedings of the 18th Annual Conference on Neural Information Processing Systems, NIPS, MIT Press, 2004.

[88] A. Perotte, N. Bartlett, N. Elhadad, F. Wood, Hierarchically supervised latent Dirichlet allocation, in: Proceedings of the 26th Annual Conference on Neural Information Processing Systems, NIPS, 2012.

[89] R. E. Madsen, D. Kauchak, C. Elkan, Modeling word burstiness using the Dirichlet distribution, in: Proceedings of the 22nd International Conference on Machine Learning, ICML, ACM, New York, NY, USA, 2005, pp. 545–552.

[90] C. Elkan, Clustering documents with an exponential-family approximation of the Dirichlet compound multinomial distribution, in: Proceedings of the 23rd International Conference on Machine Learning, ICML, ACM, New York, NY, USA, 2006, pp. 289–296.

[91] J. Reisinger, A. Waters, B. Silverthorn, R. J. Mooney, Spherical topic models, in: Proceedings of the 27th International Conference on Machine Learning, ICML, 2010, pp. 903–910.

[92] I. Dhillon, Y.Guan, J. Kogan, Refining clusters in high-dimensional text data, in: Proceedings of the Workshop on Clustering High Dimensional Data and Its Applications at the 2nd SIAM International Conference on Data Mining, ICDML, 2002, pp. 71–78.

[93] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, A. Wu, A local search approximation algorithm for k-means clustering, Computational Geometry: Theory and Applications 28 (2–3) (2004) 89–112.

[94] W. Kwedlo, P. Iwanowicz, Using genetic algorithm for selection of initial cluster centers for the k-means method, in: Proceedings of the 10th International Conference on Artifical Intelligence and Soft Computing, Part II, ICAISC (2), 2010, pp. 165–172.

[95] B. Al-shboul, S. hyon Myaeng, Initializing k-means using genetic algorithms (2009).

[96] G. T. Perim, E. D. Wandekokem, F. M. Varejão, k-means initialization methods for improving clustering by simulated annealing, in: Proceedings of the 11th Ibero-American Conference on Artificial Intelligence, IBERAMIA, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 133–142.

[97] N. Grira, M. E. Houle, Best of both: a hybridized centroid-medoid clustering heuristic, in: Proceedings of the 24th International Conference on Machine Learning, ICML, ACM, New York, NY, USA, 2007, pp. 313–320.

[98] M. Halkidi, Y. Batistakis, M. Vazirgiannis, On clustering validation techniques, Journal of Intelligent Information Systems 17 (2-3) (2001) 107–145.

[99] D. Cai, X. He, J. Han, Document clustering using locality preserving indexing, IEEE Transactions on Knowledge and Data Engineering 17 (12) (2005) 1624–1637.

[100] M. Meilă, Comparing clusterings—an information based distance, Journal of Multivariate Analysis 98 (5) (2007) 873–895.

[101] L. Hubert, P. Arabie, Comparing partitions, Journal of Classification 2 (1985) 193–218.

[102] G. Lebanon, Y. Mao, J. Dillon, The locally weighted bag of words framework for document representation, Journal of Machine Learning Research 8 (2007) 2405–2441.

[103] M. Keikha, N. Razavian, F. Oroumchian, H. Razi, Document representation and quality of text: An analysis, in: M. Berry, M. Castellanos (Eds.), Survey of Text Mining II, Springer, London, UK, 2008, pp. 219–232.

[104] D. Mladenic, Machine learning on non-homogeneous, distributed text data., Ph.D. thesis, University of Ljubljana, Faculty of Computer and Information Science (1998).

[105] D. Lewis, An evaluation of phrasal and clustered representations on a text categorization task, in: Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR, 1992, pp. 37–50.

[106] F. Beil, M. Ester, X. Xu, Frequent term-based text clustering, in: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge discovery and data mining, KDD, 2002, pp. 436–442.

[107] B. Fung, K. Wang, M. Ester, Hierarchical document clustering using frequent itemsets, in: Proceedings of the 3rd SIAM International Conference on Data Mining, SDM, 2003, pp. 59–70.

[108] W. Zhang, T. Yoshida, X. Tang, Q. Wang, Text clustering using frequent itemsets, Knowledge-Based Systems 23 (5).

[109] Y. Li, S. Chung, J. Holt, Text document clustering based on frequent word meaning sequences, Data and Knowledge Engineering 64 (1) (2008) 381–404.

[110] G. Karypis, E. Han, Concept indexing: a fast dimensionality reduction algorithm with applications to document retrieval and categorization, Tech. Rep. TR-00-0016, University of Minnesota (2000).

[111] A. Farahat, M. Kamel, Statistical semantics for enchancing document clustering, Knowledge and Information Systems 28 (2) (2010) 365–393.

[112] L. AlSumait, C. Domeniconi, Text clustering with local semantic kernels, in: M. Berry, M. Castellanos (Eds.), Survey of Text Mining II, Springer, London, UK, 2008, pp. 219–232.

[113] H. Billhardt, D. Borrajo, V. Maojo, A context vector model for information retrieval, Journal of the American Society for Information Science and Technology 53 (3) (2002) 236–249.

[114] W. Pu, N. Liu, S. Yan, J. Yan, K. Xie, Z. Chen, Local word bag model for text categorization, in: Proceedings of the 2007 7th IEEE International Conference on Data Mining, ICDM, 2007, pp. 625–630.

[115] K. Grauman, T. Darrell, The pyramid match kernel: Efficient learning with sets of features, Journal of Machine Learning Research 8 (2007) 725–760.

[116] C. Apté, F. Damerau, S. M. Weiss, Towards language independent automated learning of text categorization models, in: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR, 1994, pp. 23–30.

[117] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is "nearest neighbor" meaningful?, in: Proceedings of the 7th International Conference on Database Theory, ICDT, 1999, pp. 217–235.

[118] D. S. Modha, W. S. Spangler, Feature weighting in k-means clustering, Machine Learning 52 (3) (2003) 217–237.

[119] J. Friedman, J. Meulman, Clustering objects on subsets of attributes, Journal of the Royal Statistical Society 66 (2004) 815–849.

[120] L. Jing, M. K. Ng, J. Xu, J. Z. Huang, Subspace clustering of text documents with feature weighting k-means algorithm, in: Proceedings of the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD, 2005, pp. 802–812.

[121] C. Domeniconi, D. Gunopulos, S. Ma, B. Yan, M. Al-Razgan, D. Papadopoulos, Locally adaptive metrics for clustering high dimensional data, Data Mining and Knowledge Discovery 14 (1) (2007) 63–97.

[122] L. Jing, M. K. Ng, J. Z. Huang, An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data, IEEE Transactions on Knowledge and Data Engineering 19 (8) (2007) 1026–1041.

[123] C.-Y. Tsai, C.-C. Chiu, Developing a feature weight self-adjustment mechanism for a k-means clustering algorithm, Computational Statistics and Data Analysis 52 (10) (2008) 4658–4672.

[124] H. Cheng, K. Hua, K. Vu, Constrained locally weighted clustering, in: Proceedings VLDB International Conference on Very Large Data Bases, 2008, pp. 90–101.

[125] I. S. Dhillon, Y. Guan, J. Kogan, Iterative clustering of high dimensional text data augmented by local search, in: Proceedings of the 2002 IEEE International Conference on Data Mining, 2002, pp. 131–138.

[126] C. Ding, X. He, K-nearest-neighbor consistency in data clustering: incorporating local information into global optimization, in: Proceedings of the 2004 ACM Symposium on Applied Computing, SAC, ACM, New York, NY, USA, 2004, pp. 584–589.

[127] M. Steinbach, G. Karypis, V. Kumar, A comparison of document clustering techniques, in: Proceedings of the Workshop on Text Mining at the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000, pp. 20–23.

[128] D. H. Zanette, M. A. Montemurro, Dynamics of text generation with realistic zipf's distribution, Journal of Quantitative Linguistics 12 (1) (2005) 29–40.

[129] H. H. Bock, On some significance tests in cluster analysis, Journal of Classification 2 (1985) 77–108.

[130] H. Bock, Clustering and neural networks, in: Proceedings of the 6th Conference of the International Federation of Classification Societies on Advances in Data Science and Classification, IFCS, 1998, pp. 265–278.

[131] K. L. Du, Clustering: A neural network approach, Neural Networks 23 (1) (2010) 89–107.

[132] S. J. Redmond, C. Heneghan, A method for initialising the k-means clustering algorithm using kd-trees, Pattern Recognition Letters 28 (8) (2007) 965–973.

[133] F. Cao, J. Liang, G. Jiang, An initialization method for the k-means algorithm using neighborhood model, Computers and Mathematics with Applications 58 (3) (2009) 474–483.

[134] S. S. Khan, A. Ahmad, Cluster center initialization algorithm for k-means clustering, Pattern Recognition Letters 25 (11) (2004) 1293–1302.

[135] G. Milligan, M. Cooper, An examination of procedures for determining the number of clusters in a data set, Psychometrika 50 (1985) 159–179.

[136] X. Hu, L. Xu, A comparative study of several cluster number selection criteria, in: Proceedings of the 4th International Conference on Intelligent Data Engineering and Automated Learning, IDEAL, 2003, pp. 195–202.

[137] M. M.-T. Chiang, B. Mirkin, Intelligent choice of the number of clusters in k-means clustering: An experimental study with different cluster spreads, Journal of Classification 27 (1) (2010) 3–40.

[138] D. Zeimpekis, E. Gallopoulos, principal direction divisive partitioning with kernels and k-means steering, in: M. Berry, M. Castellanos (Eds.), Survey of Text Mining II, Springer, London, UK, 2008, pp. 45–64.

[139] S. K. Tasoulis, D. K. Tasoulis, V. P. Plagianakos, Enhancing principal direction divisive clustering, Pattern Recognition 43 (10) (2010) 3391–3411.

[140] A. M. Bagirov, Modified global k-means algorithm for minimum sum-of-squares clustering problems, Pattern Recognition 41 (10) (2008) 3192–3199.

[141] J. Z. C. Lai, T.-J. Huang, Fast global k-means clustering using cluster membership and inequality, Pattern Recognition 43 (5) (2010) 1954–1963.

[142] A. M. Bagirov, J. Ugon, D. Webb, Fast modified global k-means algorithm for incremental cluster construction, Pattern Recognition 44 (4) (2011) 866–876.

[143] D. Pelleg, A. W. Moore, X-means: Extending k-means with efficient estimation of the number of clusters, in: Proceedings of the 17th International Conference on Machine Learning, ICML, 2000, pp. 727–734.

[144] R. Kass, L. Wasserman, A reference bayesian test for nested hypotheses and its relationship to the schwarz criterion, Journal of the American Statistical Association (1995) 928–934.

[145] G. Hamerly, C. Elkan, Learning the k in k-means, in: Proceedings of the 17th Annual Conference on Neural Information Processing Systems, 2003.

[146] Y. Feng, G. Hamerly, Pg-means: learning the number of clusters in data, in: Proceedings of the 20th Annual Conference on Neural Information Processing Systems, 2006, pp. 393–400.

[147] F. J. Massey, The Kolmogorov-Smirnov test for goodness of fit, Journal of the American Statistical Association 46 (253) (1951) 68–78.

[148] K. Kurihara, M. Welling, Bayesian k-means as a "maximization-expectation" algorithm, Neural Computation 21 (4) (2009) 1145–1172.

[149] R. Tibshirani, G. Walther, T. Hastie, Estimating the number of clusters in a data set via the gap statistic, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63 (2) (2001) 411–423.

[150] C. Yang, X. Zhang, L. Jiao, G. Wang, Self-tuning semi-supervised spectral clustering, in: Proceedings of the International Conference on Computational Intelligence and Security, Vol. 1 of CIS, 2008, pp. 1–5.

[151] T. Shi, M. Belkin, B. Yu, Data spectroscopy: Eigenspaces of convolution operators and clustering, The Annals of Statistics 37 (6B) (2009) 3960–3984.

[152] E. Levine, E. Domany, Resampling method for unsupervised estimation of cluster validity, Neural Computation 13 (11) (2001) 2573–2593.

[153] T. Lange, V. Roth, M. L. Braun, J. M. Buhmann, Stability-based validation of clustering solutions, Neural Computation 16 (6) (2004) 1299–1323.

[154] R. Tibshirani, G. Walther, Cluster validation by prediction strength, Journal of Computational and Graphical Statistics 14 (3) (2005) 511–528.

[155] I. S. Dhillon, Y. Guan, B. Kulis, Kernel k-means: spectral clustering and normalized cuts, in: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, ACM, New York, NY, USA, 2004, pp. 551–556.

[156] B. W. Silverman, Using kernel density estimates to investigate multimodality, Journal of the Royal Statistical Society. Series B (Methodological) 43 (1) (1981) 97–99.

[157] P. H. J.A. Hartigan, The dip test of unimodality, The Annals of Statistics 13 (1) (1985) 70–84.

[158] J. J. Verbeek, N. Vlassis, B. Kröse, Efficient greedy learning of Gaussian mixture models, Neural Computation 15 (2) (2003) 469–485.

[159] S. N. S.A. Nene, H. Murase, Columbia object image library (coil-100), Tech. Rep. CUCS-006-96, Columbia University (February 1996).

[160] S. N. S.A. Nene, H. Murase, Columbia object image library (coil-100), Tech. Rep. CUCS-005-96, Columbia University (February 1996).

Author's Publications

1. **A. Kalogeratos** and A. Likas, Dip-means: an incremental clustering method for estimating the number of clusters, Advances in Neural Information Processing Systems (NIPS), (2012).
2. **A. Kalogeratos** and A. Likas, Text Document Clustering Using Global Term Context Vectors, Knowledge and Information Systems (KAIS), Springer, 31:3 (2012) 455–474.
3. **A. Kalogeratos** and A. Likas, Document clustering using synthetic cluster prototypes, Data and Knowledge Engineering, 70:3 (2011) 284–306.
4. V. Chasanis, **A. Kalogeratos** and A. Likas, Movie Segmentation into Scenes and Chapters Using Locally Weighted Bag of Visual Words, ACM International Conference on Image and Video Retrieval (CIVR09) (2009).
5. **A. Kalogeratos** and A. Likas, A Significance-Based Graph Model for Clustering Web Documents, G. Antoniou et al. (Eds.): SETN06, LNAI 3955, Springer Verlag (2006).

Short Curriculum Vitae

Argyris Kalogeratos was born in Patras, Greece, in 1982. He received the B.Sc in Computer Science in 2005 and the M.Sc. in Computer Science (Technologies and Applications) in 2007, from the Department of Computer Science, University of Ioannina, Greece. Since the end of 2007 he has been a Ph.D. candidate in the same Department. He has been involved in a research project and has published 2 papers in peer-review scientific journals and 3 papers in refereed conference proceedings. His research interests are in the areas of machine learning and data mining.