Clustering Using Similarity and Kernel Matrices

Grigorios Tzortzis

Ph.D. Dissertation

Ioannina, June 2014

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Τεχνικές Ομαδοποίησης Δεδομένων Βασισμένες σε Πίνακες Ομοιότητας

Η ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης

του Τμήματος Μηχανικών Η/Υ και Πληροφορικής Εξεταστική Επιτροπή

από τον

Γρηγόριο Τζώρτζη

ως μέρος των Υποχρεώσεων για τη λήψη του

ΔΙΔΑΚΤΟΡΙΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

Ιούνιος 2014

Τριμελής Συμβουλευτική Επιτροπή

- Αριστείδης Λύκας, Καθηγητής του Τμήματος Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Ιωαννίνων (Επιβλέπων)
- Κωνσταντίνος Μπλέκας, Επίκουρος Καθηγητής του Τμήματος Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Ιωαννίνων
- Ευάγγελος Καρκαλέτσης, Ερευνητής Α′ του Ινστιτούτου Πληροφορικής και Τηλεπικοινωνιών του ΕΚΕΦΕ "Δημόκριτος"

Επταμελής Εξεταστική Επιτροπή

- Αριστείδης Λύκας, Καθηγητής του Τμήματος Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Ιωαννίνων (Επιβλέπων)
- Κωνσταντίνος Μπλέκας, Επίκουρος Καθηγητής του Τμήματος Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Ιωαννίνων
- Ευάγγελος Καρκαλέτσης, Ερευνητής Α′ του Ινστιτούτου Πληροφορικής και Τηλεπικοινωνιών του ΕΚΕΦΕ "Δημόκριτος"
- Μιχαήλ Βαζιργιάννης, Καθηγητής του Τμήματος Πληροφορικής του Οικονομικού Πανεπιστημίου Αθηνών
- Χριστόφορος Νίκου, Αναπληρωτής Καθηγητής του Τμήματος Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Ιωαννίνων
- Αναστάσιος Τέφας, Επίκουρος Καθηγητής του Τμήματος Πληροφορικής του Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης
- Μιχαήλ Τίτσιας, Λέκτορας του Τμήματος Πληροφορικής του Οικονομικού Πανεπιστημίου Αθηνών

Dedication

To my family, Marina and Moha.

Acknowledgements

I would like to express my sincere thanks and gratitude to my advisor Prof. Aristidis Likas for the valuable guidance, advice and encouragement he has offered during the elaboration of this thesis. For the time and effort he spent on my work throughout all these years, dating back to 2006. For his excellent work ethic. Our collaboration has been a pleasant and memorable experience that has helped me develop strong research skills.

I am also grateful to Prof. Konstantinos Blekas and Research Director Vangelis Karkaletsis for their suggestions and insightful remarks and for acting as secondary referees for my thesis. Many thanks also to Prof. Michalis Vazirgiannis, Prof. Christophoros Nikou, Prof. Anastasios Tefas and Prof. Michalis Titsias for serving in my thesis committee.

I would also like to thank my colleagues Dr. Vasileios Chasanis, Dr. Argyris Kalogeratos, Antonis Ioannidis and Panagiotis Zagorisios for creating a pleasant and friendly environment at the office and for the useful conversations we had during the past years. Its has been a privilege to conduct my research among them.

A big thanks goes to my parents, Fotis and Athanasia, and my sister Katia for always believing and unconditionally supporting me.

Finally, a very special thanks goes to Marina who has been beside me since our undergraduate studies. I consider myself very lucky to have her in my life.

Ioannina, June 2014
Grigorios Tzortzis

CONTENTS

List of Figures

List of Tables

List Of Algorithms

Abstract

Grigorios F. Tzortzis.
PhD, Computer Science & Engineering Department, University of Ioannina, Greece. June, 2014.
Thesis Title: Clustering Using Similarity and Kernel Matrices.
Thesis Supervisor: Aristidis Likas.

This thesis studies the (unsupervised) clustering problem, which aims at partitioning a dataset into groups, called clusters, such that instances falling in the same cluster are similar to each other and dissimilar to those of other clusters according to some similarity/dissimilarity measure. Specifically, this thesis concerns the development, implementation and evaluation of clustering methodologies, mainly focusing on three different axes: i) proximity-based clustering, where only the pairwise proximity matrix (i.e. similarity or distance matrix) of the data is available during training and not the instances themselves, ii) multi-view learning, where for the same instances multiple representations (views) are available, coming from different sources and/or feature spaces and iii) multiple kernel learning, where a kernel that suits the data is learned together with the cluster assignments. Usually, the kernel is parametrized as a combination of some predefined kernels, called basis kernels, and we wish to infer appropriate values for the combination parameters.

We begin, by presenting an approach that tackles the initialization problem of the k-means algorithm by altering its sum of the intra-cluster variances objective. Weights are assigned to the clusters in proportion to their variance, which predispose the optimization towards primarily minimizing those clusters, that in the current iteration, exhibit large intra-cluster variance. In this way, the solution space is gradually restricted towards clusters with similar variances, allowing our method to systematically produce higher quality partitionings than k-means, while restarted from random initial centers. Moreover, we adapt our approach to perform clustering in kernel space, by altering the objective of the kernel k-means algorithm. The kernel space extension requires only the kernel matrix and not the instances as input, i.e. it is a proximity-based method.

Afterward, we consider the problem of unsupervised multi-view learning. Our main contribution in this field is the assignment of weights to the views, which are auto-

matically tuned to reflect the quality of the views and determine their contribution to the clustering solution accordingly. The majority of multi-view approaches treat all available views as being equally important, which may lead to a considerable drop in performance if degenerate views (e.g. noisy views) exist in the dataset. Analytically, we present two different methodologies for the above problem. In the first case, views are represented by convex mixture models and we develop an algorithm that associates a weight with each view and another that does not employ view weights. In the second case, each view is represented by a kernel matrix and a weighted combination of those kernel matrices is learned. This formulation includes a parameter that controls the sparsity of the weights, allowing its adaptation to the dataset.

Finally, we focus on multiple kernel learning, where most of the existing clustering approaches of this type exploit the large margin principle of SVM and perform learning by maximizing the margin. Instead, here, we propose an objective that utilizes the ratio between the margin and the intra-cluster variance. Since the objective explicitly takes into account both the separation (margin) and the compactness (intra-cluster variance) of the clusters, higher quality solutions can be possibly attained compared to approaches that rely solely on either of the two. Moreover, it has been shown that the margin alone is an unsuitable measure of the goodness of the learned kernel, as it can become arbitrary large by simply scaling the kernel. We prove that our ratio-based objective is invariant to kernel scaling and, also, that its global optimum solution is invariant to the type of norm constraint on the kernel parameters. Experiments verify the properties of our objective and reveal the superior clustering performance of the proposed multiple kernel formulation.

Εκτεταμενη Περιληψη στα Ελληνικα

Γρηγόριος Τζώρτζης του Φωτίου και της Αθανασίας.
PhD, Τμήμα Μηχανικών Η/Υ & Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ιούνιος, 2014.
Τίτλος Διατριβής: Τεχνικές Ομαδοποίησης Δεδομένων Βασισμένες σε Πίνακες Ομοιότητας.
Επιβλέπων: Αριστείδης Λύκας.

Η παρούσα διατριβή μελετά το πρόβλημα της ομαδοποίησης (clustering), που έχει ως στόχο τον διαχωρισμό ενός συνόλου δεδομένων σε ομάδες (clusters), χωρίς τη χρήση επίβλεψης, ώστε τα δεδομένα που ανήκουν στη ίδια ομάδα να είναι όμοια μεταξύ τους και ανόμοια με αυτά των άλλων ομάδων, βάσει ενός μέτρου ομοιότητας/ανομοιότητας. Συγκεκριμένα, η διατριβή επικεντρώνεται στην παρουσίαση μεθόδων ομαδοποίησης που αφορούν τρεις βασικούς θεματικούς άξονες: α) την ομαδοποίηση δεδομένων για τα οποία έχουμε διαθέσιμο μόνο τον πίνακα εγγύτητας και όχι τα ίδια τα δεδομένα (proximity-based clustering), β) την μάθηση με πολλαπλές όψεις (multi-view learning), όπου για τα ίδια δεδομένα έχουμε στη διάθεσή μας πολλαπλές αναπαραστάσεις (όψεις) που προέρχονται από διαφορετικές πηγές ή/και διαφορετικούς χώρους χαρακτηριστικών και γ) την μάθηση με πολλαπλούς πυρήνες (multiple kernel learning), όπου ταυτόχρονα με την ομαδοποίηση θέλουμε να μάθουμε και τον κατάλληλο πυρήνα (kernel) για τα δεδομένα. Συνήθως ο πυρήνας παραμετροποιείται ως ένας συνδυασμός από δοθέντες πυρήνες (basis kernels) και στοχεύουμε στην μάθηση κατάλληλων τιμών για τις παραμέτρους του συνδυασμού.

Αρχικά προτείνεται μια μέθοδος για την αντιμετώπιση του γνωστού προβλήματος της αρχικοποίησης (initialization problem), από το οποίο πάσχει ο αλγόριθμος k-means. Συγκεκριμένα, τροποποιούμε το κριτήριο (objective function) του k-means έτσι ώστε να δίδεται μεγαλύτερη έμφαση στην ελαχιστοποίηση των ομάδων που στην τρέχουσα επανάληψη εμφανίζουν μεγάλη διακύμανση (intra-cluster variance). Κατ' αυτόν τον τρόπο ο χώρος λύσεων σταδιακά περιορίζεται σε ομάδες που εμφανίζουν παρεμφερή διακύμανση, το οποίο επιτρέπει στη μέθοδό μας να εντοπίζει σε συστηματική βάση λύσεις καλύτερης ποιότητας σε σχέση με τον k-means, καθώς επανεκκινείται από τυχαία αρχικά κέντρα. Επιπλέον, παρουσιάζεται η προσαρμογή της μεθόδου ώστε να μπορεί να εφαρμοστεί για ομαδοποίηση με πίνακα ομοιότητας (kernel matrix), τροποποιώντας το κριτήριο του αλγορίθμου kernel k-means.

Στη συνέχεια, η διατριβή εστιάζεται στο πρόβλημα της ομαδοποίησης με πολλαπλές όψεις. Η βασική συνεισφορά στο αντικείμενο αυτό σχετίζεται με την ανάθεση βαρών στις

όψεις, τα οποία μαθαίνονται αυτόματα και τα οποία αντικατοπτρίζουν την ποιότητα των όψεων. Οι υπάρχουσες προσεγγίσεις θεωρούν όλες τις όψεις εξίσου σημαντικές, κάτι που μπορεί να οδηγήσει σε σημαντική μείωση της απόδοσης εάν υπάρχουν εκφυλισμένες όψεις (π.χ. όψεις με θόρυβο) στο σύνολο δεδομένων. Ειδικότερα, παρουσιάζονται για το ανωτέρω πρόβλημα δύο διαφορετικές μεθοδολογίες. Στην πρώτη περίπτωση αναπαριστούμε τις όψεις μέσω κυρτών μικτών μοντέλων (convex mixture models) λαμβάνοντας υπόψη τις διαφορετικές στατιστικές ιδιότητές τους και παρουσιάζουμε έναν αλγόριθμο με βάρη στις όψεις και έναν χωρίς βάρη. Στην δεύτερη περίπτωση αναπαριστούμε την κάθε όψη μέσω ενός πίνακα ομοιότητας (kernel matrix) και μαθαίνουμε ένα συνδυασμό με βάρη από τους πίνακες αυτούς. Το προτεινόμενο μοντέλο διαθέτει μία παράμετρο που ελέγχει την αραιότητα των βαρών, επιτρέποντας την καλύτερη προσαρμογή του συνδυασμού στα δεδομένα.

Η τελευταία ενότητα της διατριβής αφορά στην ομαδοποίηση με πολλαπλούς πυρήνες, όπου συνήθως το κριτήριο που βελτιστοποιείται είναι το εύρος (margin) της λύσης, όπως είναι γνωστό από τον ταξινομητή SVM (support vector machine). Στην προσέγγιση που προτείνεται, βελτιστοποιείται ο λόγος μεταξύ του εύρους και της διακύμανσης (intra-cluster variance) των ομάδων, λαμβάνοντας έτσι υπόψη τόσο τον διαχωρισμό (separability) τους όσο και το πόσο συμπαγείς (compactness) είναι οι ομάδες, το οποίο δύναται να οδηγήσει σε καλύτερες λύσεις. Έχει δειχθεί ότι το εύρος από μόνο του δεν επαρκεί ως κριτήριο για την μάθηση του κατάλληλου πυρήνα, καθότι μπορεί να γίνει αυθαίρετα μεγάλο μέσω μίας απλής κλιμάκωσης (scaling) του πυρήνα. Αντιθέτως, το κριτήριο που προτείνουμε είναι αμετάβλητο (invariant) σε κλιμακώσεις του πυρήνα και, επιπλέον, το ολικό του βέλτιστο είναι αμετάβλητο ως προς τον τύπο της νόρμας που εφαρμόζεται στους περιορισμούς (constraints) των παραμέτρων του πυρήνα. Τα πειραματικά αποτελέσματα επιβεβαιώνουν τις ιδιότητες του κριτηρίου μας, καθώς και τις αναμενόμενες βελτιωμένες επιδόσεις ομαδοποίησης.

CHAPTER 1

INTRODUCTION

As an ever increasing amount of information becomes available, users try to find ways of organizing, interpreting and handling this information. Locating valuable or important information can prove out to be an overwhelming task, due to the great volume of accessible data. One key step in dealing with this vast amount of data is to *classify* or *group* it into a set of categories or clusters. In this way, users are provided with a condensed representation of the data, where similarities and differences, as well as hidden structures and patterns in the data are exposed.

Machine learning concerns the development of methods that learn from examples and automatically detect patterns in the data [12, 99]. The two main areas of machine learning are *supervised* and *unsupervised learning*, whose most representative problems are classification and clustering, respectively. In the classification problem, the underlying task is to assign to an instance $\mathbf{x}$ one of a finite set of discrete class labels, i.e. assign $\mathbf{x}$ to a category. To accomplish this, a classifier is built that outputs the class label of an instance using a parametric function. To determine the values of the involved parameters, an inductive learning algorithm is employed, which optimizes an empirical risk objective function over a finite labeled dataset $\mathcal{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $\mathbf{x}_i$ is an instance and y_i is the corresponding class label. Overall, the aim of classification is to construct the classifier, by utilizing the labeled data in $\mathcal{X}$, and use it to *predict the labels of new unseen instances*.

For the clustering problem [123], in which this thesis focuses on, the dataset does not contain any notion of labels and is of the form $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, i.e. it consists only of the instances. The goal of clustering is to *partition $\mathcal{X}$ into a finite number of groups*, called clusters, which unveil the intrinsic structures present in the data. Exposing these hidden structures provides meaningful insight into the data that can be of particular interest. For this reason, clustering has found applications in a variety of fields, such as pattern recognition, image segmentation, spatial database analysis, life and medical sciences, economics and many more. Usually, a similarity or dissimilarity measure, such as Euclidean distance, is used to describe the relations among the instances, and clusters are obtained by employing a clustering algorithm which seeks to group the data in a way that *instances falling in the same cluster are similar to each other and dissimilar to those of other clusters*. The choice of proximity measure and clustering algorithm has a huge impact on the resulting partitioning and different choices can lead to different number of clusters, different cluster shapes etc. Note that clustering is a subjective process in nature, as there is no ground-truth to guide on how the instances should be grouped together, in contrast to classification where the class of each dataset point $\mathbf{x}_i$ is available. Even the number of clusters may not be known in advance. Thus, it is not straightforward how to evaluate a clustering result and the evaluation heavily depends on the application under consideration.

This thesis concerns the development, implementation and evaluation of (unsupervised) clustering methodologies for three important and very active machine learning problems, namely proximity-based clustering [92, 113], multi-view learning [98, 120] and multiple kernel learning [49]. In the following, we describe these problems in detail, along with a review of the related work. Afterward, we present the main contributions and the layout of the thesis.

1.1 Proximity-based Clustering

Given a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, a *proximity* matrix $P \in \Re^{N \times N}$, $P_{ij} = prox(\mathbf{x}_i, \mathbf{x}_j)$, is defined as the matrix that contains the pairwise proximities (similarities or distances) of all dataset points. If proximity corresponds to distance, then it is called a *distance* matrix and is denoted by $D \in \Re^{N \times N}$, $D_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$, while if it corresponds to similarity it is called a *similarity* matrix and is denoted by $S \in \Re^{N \times N}$, $S_{ij} = s(\mathbf{x}_i, \mathbf{x}_j)$. Several clustering algorithms, such as *k*-means, fuzzy *c*-means, Gaussian mixture models and many others [99], require as input the instances themselves which must be in the form of vectors. However, there exist problems where only the proximity matrix of the data is available, making the application of these methods impossible. This happens basically for two reasons. Either the pairwise proximities are extracted beforehand and only these are provided, or instances may not be vectors at all. Consider for example graph partitioning [37], where instances correspond to graph vertices and edge weights describe their proximity. In this case there are no vectors and only a proximity matrix

Table 1.1: Examples of kernel functions.

Linear	$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$
Polynomial	$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \left(\mathbf{x}_i^\top \mathbf{x}_j + \gamma\right)^\delta$
RBF (or Gaussian)	$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2\right)$
Sigmoid	$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(\gamma(\mathbf{x}_i^\top \mathbf{x}_j) + \theta\right)$

can be extracted. Hence, developing clustering algorithms that can directly work on the proximity matrix is of great importance. Of course, if we are given vectorial data, i.e. $\mathbf{x}_i \in \Re^d$, proximity-based methods can still be applied, by first calculating the pairwise proximities using an appropriate measure such as the Euclidean distance.

A special form of proximity (similarity) matrix utilized by several machine learning algorithms is the *kernel* matrix $K \in \Re^{N \times N}$ [92]. The dataset $\mathcal{X}$ is mapped from input space to a higher dimensional reproducing kernel Hilbert space $\mathcal{H}$, a.k.a. *feature space*, via a nonlinear transformation $\phi : \mathcal{X} \to \mathcal{H}$ and the kernel matrix contains the pairwise inner products in the feature space, i.e. $K_{ij} = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$, $\mathbf{x}_i \in \Re^d$. Note that any positive semidefinite matrix is a kernel matrix, since it can be interpreted as a Gram matrix. Usually a kernel function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \Re$ [41] is applied to directly provide the inner products in feature space without explicitly defining transformation ϕ, i.e. $K_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$. Some kernel function examples are given in Table 1.1. The use of a kernel function is not a mere technicality, as for certain kernel functions the corresponding transformation is intractable. By employing the kernel matrix, learning is carried out in feature space instead of input space, which allows for nonlinearities in the data to be uncovered and has been shown to boost performance in several cases. Some examples of kernel methods are kernel PCA [90], SVM [19], RVM [101], kernel k-means [37, 90] and kernel ridge regression [89].

In the rest of this section, we present some popular proximity-based clustering approaches from the literature.

1.1.1 *k*-medoids

The k-medoids algorithm is closely related to the well-known k-means algorithm [76, 123]. Their main difference is that k-medoids represents a cluster with a medoid instead of a centroid. While the centroid does not, in general, correspond to a dataset point, the medoid, by its definition, must necessarily be a dataset point. One can think of the medoid as the most "central" instance of the cluster with respect to some proximity measure. The utilization of medoids instead of centroids allows k-medoids to derive a partitioning using only the proximity matrix, without requiring data to be in vectorial form.

Given a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, k-medoids splits $\mathcal{X}$ into a predefined number M of

clusters, $\{\mathcal{C}_k\}_{k=1}^{M}$, by optimizing the following clustering criterion:

$$\mathcal{E} = \sum_{i=1}^{N}\sum_{k=1}^{M} \delta_{ik} prox(\mathbf{x}_i, \mathbf{m}_k), \tag{1.1}$$

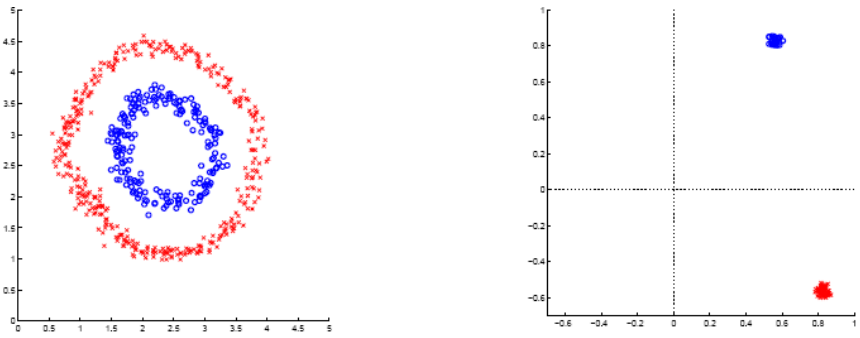
where $\delta_{ik} = 1$ if $\mathbf{x}_i \in \mathcal{C}_k$ and $\delta_{ik} = 0$ otherwise, $\mathbf{m}_k \in \mathcal{X}$ is the cluster medoid and $prox(\mathbf{x}_i, \mathbf{m}_k)$ is the proximity between data point $\mathbf{x}_i$ and medoid $\mathbf{m}_k$. If proximity represents similarity, the above criterion is maximized, while if it represents distance it is minimized. The optimization is done with an iterative procedure almost identical to that of *k*-means. The algorithm starts by (randomly) selecting M data points as initial medoids and proceeds by alternating between assigning each data point to the closest medoid (i.e. to the most similar, or the least distant medoid) and computing the medoid of each cluster, until the medoids do not change. The closest medoid can be found by a simple look up on the proximity matrix. The medoid of a cluster is an instance that belongs to that cluster and can be found through a discrete search over the cluster instances. We select as medoid of the k-th cluster the instance $\mathbf{x}_{i^*}$, where $\mathbf{x}_{i^*} = \underset{\mathbf{x}_i \in \mathcal{C}_k}{\operatorname{argmin}} \sum_{\mathbf{x}_j \in \mathcal{C}_k} prox(\mathbf{x}_i, \mathbf{x}_j)$ if proximity corresponds to distance and $\mathbf{x}_{i^*} = \underset{\mathbf{x}_i \in \mathcal{C}_k}{\operatorname{argmax}} \sum_{\mathbf{x}_j \in \mathcal{C}_k} prox(\mathbf{x}_i, \mathbf{x}_j)$ if proximity corresponds to similarity. This search can be performed using only the proximity matrix entries.

The advantages of *k*-medoids over *k*-means are that it requires only the proximity matrix, that cluster representatives are more robust to outliers and that any similarity (or distance) measure can be used in the objective function as long as it can be readily evaluated and guarantees convergence. There are also a number of drawbacks, such as the higher computational complexity, since *k*-medoids requires $O(\tau(N^2 + MN))$ operations (τ is the number of iterations). This complexity is a result of the discrete search required to identify the medoids. Finally, note that the solution depends on the initial medoids, hence local optima of the clustering criterion are attained.

1.1.2 Spectral Clustering

Spectral clustering [113] is a relatively new approach to clustering that produces a partitioning of the dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ using the eigenvectors of a matrix derived from the data. More specifically, a matrix containing the pairwise similarities, called *affinity* matrix in this case, is used and the eigenvectors of this matrix, or a matrix derived from the affinity matrix, are calculated. The eigenvectors are then processed to obtain a clustering of the dataset. We shall denote the affinity matrix by $A \in \Re^{N \times N}$, where $A_{ij} = s(\mathbf{x}_i, \mathbf{x}_j)$.

A number of variants of the spectral approach appear in the literature which basically differ on the matrix used to calculate the eigenvectors and the way the eigenvectors are subsequently processed to obtain the final partitioning. Some of them are summarized in [113, 118]. Here we present the typical algorithm proposed by Ng et al. [82], which is described in Algorithm 1.1. This algorithm does not require as input



(a) (b)

Figure 1.1: Spectral clustering on the two rings dataset from [82]: (a) The rings cannot be separated by *k*-means in the original space; (b) The two rings can be separated with *k*-means after being mapped in $\Re^2$ with the use of eigenvectors.

Algorithm 1.1 Spectral clustering with the Ng et al. [82] algorithm.

Input: Affinity matrix A, Number of clusters M

Output: Final clusters $\{\mathcal{C}_k\}_{k=1}^{M}$

1: Define the diagonal matrix D where $D_{ii} = \sum_{j=1}^{N} A_{ij}$

2: Construct the matrix $L = D^{-1/2}AD^{-1/2}$

3: Calculate $e_1, \ldots, e_M$, the M top eigenvectors of L, and form the matrix $E = [e_1, \ldots, e_M] \in \Re^{N \times M}$

4: Normalize each of E's rows to unit length and construct matrix Y, $Y_{ij} = E_{ij} \Big/ \left(\sum_{l=1}^{M} E_{il}^2\right)^{1/2}$

5: Treat each row of Y as a point in $\Re^M$ and cluster them into M clusters using *k*-means

6: Assign instance $\mathbf{x}_i$ to cluster $\mathcal{C}_k$ only if the i-th row of matrix Y was assigned to the k-th cluster

the dataset in vectorial form, however if the dataset is available, Ng et al. proposed computing the affinity matrix as $A_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$ for $i \neq j$ and $A_{ii} = 0$. The parameter σ controls how rapidly the affinity falls off with the distance. One may wonder, since in step 5 we apply *k*-means on the eigenvectors, why not apply *k*-means directly on the data. The answer is that mapping the points in $\Re^M$ (M is the number of clusters) using eigenvectors can lead to tight clusters that can be uncovered by *k*-means more easily. An example is shown in Figure 1.1, where the two rings cannot be identified directly by *k*-means, since they are not linearly separable, but when they are mapped to $\Re^2$, through the eigenvectors, ring identification is possible.

To understand how this algorithm works, consider the "ideal" case in which data points belonging to different clusters are infinitely far apart, hence their affinity is zero. This results in an affinity matrix that is block diagonal and thus matrix L (step 2) is also block diagonal. The eigenvectors and eigenvalues of a block diagonal matrix are the union of the eigenvalues and eigenvectors of its blocks (the latter padded appropriately with zeros). If the affinity matrix is also symmetric and $A_{ij} > 0$ for $i \neq j$ in each block, it

follows that each block of matrix L has an eigenvalue equal to 1 and the next eigenvalue is strictly less than 1. Thus, taking as many top eigenvectors[1] as the number of blocks in matrix L (step 3) results in taking the top eigenvector of each block of L, padded appropriately with zeros. Then matrix E is of the form:

$$E = \begin{bmatrix} e_1^{(1)} & \vec{0} & \cdots & \vec{0} & \vec{0} \\ \vec{0} & e_1^{(2)} & \cdots & \vec{0} & \vec{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vec{0} & \vec{0} & \cdots & e_1^{(M-1)} & \vec{0} \\ \vec{0} & \vec{0} & \cdots & \vec{0} & e_1^{(M)} \end{bmatrix} \in \Re^{N \times M}$$

where $e_1^{(k)}$ is the top eigenvector of the k-th block. Clustering the rows of this matrix (or the row-normalized matrix Y (step 4)) is straightforward as rows with nonzero value in the first dimension belong to the first cluster, rows with nonzero value in the second dimension belong to the second cluster etc. Note that this clustering corresponds to the true clustering of the instances. This "ideal" scenario provides the intuition behind all spectral clustering methods. Obviously, in real problems the affinity between points in different clusters will not be zero and the top eigenvectors will not define so clearly a partition, but with some processing we expect to approach the true clusters.

The main advantage of spectral methods is that they do not depend on initializations. Only the step that derives the clusters from the eigenvectors may require initialization (e.g. *k*-means in step 5 of Algorithm 1.1), but this is not expected to change the final solution considerably if the eigenvectors strongly indicate a partitioning of the dataset. Also there is no limitation on the form of the affinity matrix which can contain even non-metric similarities. Usually though, the affinity matrix is kept symmetric to avoid complex eigenvalues and its entries are positive numbers. Spectral methods have been employed to numerous problems with satisfactory results. They are particularly popular for graph partitioning as many graph cut criteria, such as normalized cut and ratio association, can be optimized by taking the eigenvectors of a matrix derived from the graph affinity matrix [37, 113, 118]. Spectral methods compute a globally optimum solution of a *relaxation* of the graph problem. The main disadvantage is the high computational complexity caused by the need to compute the eigenvalues and eigenvectors of an $N \times N$ matrix. Eigenvalue decomposition costs $O(N^3)$, which is very high for large datasets. Special numerical methods can be used to approximate the M top eigenvectors without computing all eigenvectors, thus reducing the above cost.

1.1.3 Kernel *k*-means

Kernel *k*-means [37, 90] is a generalization of the standard *k*-means algorithm where the dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x}_i \in \Re^d$ is mapped from input space to a higher dimensional reproducing kernel Hilbert space (or feature space) $\mathcal{H}$ via a nonlinear transformation

[1]The top eigenvectors are the ones corresponding to the largest eigenvalues.

Algorithm 1.2 Kernel *k*-means.

Input: Kernel matrix K, Number of clusters M, Initial clusters $\{\mathcal{C}_k\}_{k=1}^{M}$
Output: Final clusters $\{\mathcal{C}_k\}_{k=1}^{M}$, Intra-cluster variance $\mathcal{E}_{\mathcal{H}}$

1: **for all** points $\mathbf{x}_i$, $i = 1$ to N **do**
2: **for all** clusters $\mathcal{C}_k$, $k = 1$ to M **do**
3: Compute $\|\phi(\mathbf{x}_i) - \mathbf{m}_k\|^2$ using (1.3)
4: **end for**
5: Find $c^*(\mathbf{x}_i) = \underset{k}{\operatorname{argmin}} \left(\|\phi(\mathbf{x}_i) - \mathbf{m}_k\|^2\right)$
6: **end for**
7: **for all** clusters $\mathcal{C}_k$, $k = 1$ to M **do**
8: Update cluster $\mathcal{C}_k = \{\mathbf{x}_i | c^*(\mathbf{x}_i) = k\}$
9: **end for**
10: **if** converged **then**
11: **return** final clusters $\{\mathcal{C}_k\}_{k=1}^{M}$ and $\mathcal{E}_{\mathcal{H}}$ calculated using (1.2)
12: **else**
13: Go to step 1
14: **end if**

$\phi : \mathcal{X} \to \mathcal{H}$. Actually, kernel *k*-means is equivalent to performing *k*-means in feature space. This results in linear separators in feature space which correspond to nonlinear separators in input space. Thus kernel *k*-means avoids the limitation of linearly separable clusters that *k*-means suffers from.

To partition dataset $\mathcal{X}$ into M disjoint clusters, $\{\mathcal{C}_k\}_{k=1}^{M}$, the intra-cluster variance in feature space (1.2) is minimized over clusters $\{\mathcal{C}_k\}_{k=1}^{M}$, where $\mathbf{m}_k$ is the k-th cluster center and δ_{ik} is an indicator variable with $\delta_{ik} = 1$ if $\mathbf{x}_i \in \mathcal{C}_k$ and $\delta_{ik} = 0$ otherwise.

$$\mathcal{E}_{\mathcal{H}} = \sum_{i=1}^{N} \sum_{k=1}^{M} \delta_{ik} \|\phi(\mathbf{x}_i) - \mathbf{m}_k\|^2, \ \mathbf{m}_k = \frac{\sum_{i=1}^{N} \delta_{ik} \phi(\mathbf{x}_i)}{\sum_{i=1}^{N} \delta_{ik}} \tag{1.2}$$

A kernel function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \Re$ [41] (see Table 1.1 for examples) is applied to get the inner products in feature space without explicitly defining transformation ϕ, giving rise to the kernel matrix $K \in \Re^{N \times N}$, $K_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$. Note that for certain kernel functions the corresponding transformation is intractable. The squared Euclidean distances in (1.2) can now be computed using solely the kernel matrix entries (centers $\mathbf{m}_k$ cannot be analytically calculated):

$$\|\phi(\mathbf{x}_i) - \mathbf{m}_k\|^2 = K_{ii} - \frac{2 \sum_{j=1}^{N} \delta_{jk} K_{ij}}{\sum_{j=1}^{N} \delta_{jk}} + \frac{\sum_{j=1}^{N} \sum_{l=1}^{N} \delta_{jk} \delta_{lk} K_{jl}}{\sum_{j=1}^{N} \sum_{l=1}^{N} \delta_{jk} \delta_{lk}}. \tag{1.3}$$

By iteratively updating the partitioning through assignments of the instances to their closest center in feature space (Algorithm 1.2), kernel *k*-means monotonically converges to a local minimum of the objective if the kernel matrix is positive semidefinite, i.e. is a valid kernel matrix. If the kernel matrix is not positive semidefinite the algorithm may still converge, but this is not guaranteed. The returned solution heavily depends

on the initial cluster assignments, thus multiple restarts are often employed to avoid poor minima. As for the computational complexity, in [37] it is shown that kernel *k*-means requires $O(N^2\tau)$ scalar operations, where τ is the number of iterations until convergence is achieved. If we also have to calculate the kernel matrix, an extra $O(N^2 d)$ scalar operations are necessary.

It must be noted that by associating a weight with each instance, the weighted kernel *k*-means algorithm is obtained [37]. Its objective can become equivalent to that of many graph cut criteria, such as ratio association, normalized cut etc, if the weights and the kernel matrix are appropriately set [37]. Hence, it can substitute the commonly used spectral methods for graph partitioning.

1.1.4 Global Kernel *k*-means

In a previous work of ours, we proposed the global kernel *k*-means algorithm [103, 105] to circumvent the initialization problem of kernel *k*-means. Our method builds on the ideas of global *k*-means [74] and kernel *k*-means (Section 1.1.3). It maps the dataset points from input space to a higher dimensional feature space with the help of a kernel matrix $K \in \Re^{N\times N}$ and optimizes the intra-cluster variance in feature space (1.2), as kernel *k*-means does. In this way nonlinearly separable clusters are located in input space. Also, global kernel *k*-means finds *near optimal solutions* to the M-clustering problem, by incrementally and deterministically adding a new cluster at each stage (similarly to global *k*-means) and by applying kernel *k*-means as a local search procedure, instead of initializing all M clusters at the beginning of the execution. Thus, the problems of cluster initialization and convergence to poor local minima are avoided.

Suppose we are given a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x}_i \in \Re^d$ and we want to solve the M-clustering problem by splitting this dataset into M disjoint clusters $\{\mathcal{C}_k\}_{k=1}^M$, while optimizing the intra-cluster variance in feature space (1.2). Global kernel *k*-means proceeds as follows. We start by considering the 1-clustering problem. The optimal solution to this problem is trivial as all data points are assigned to the same cluster. We continue with the 2-clustering problem where kernel *k*-means is executed N times. During the i-th execution the initialization is done by considering two clusters one of which contains only $\mathbf{x}_i$ and the other is $\mathcal{X} - \{\mathbf{x}_i\}$. Among the N solutions, the one with the lowest intra-cluster variance is kept as the solution with two clusters. In general, for the k-clustering problem let $\left\{\mathcal{C}_1^*, \ldots, \mathcal{C}_{k-1}^*\right\}$ denote the solution with $k-1$ clusters and assume that $\mathbf{x}_i \in \mathcal{C}_r^*$. We perform N executions of the kernel *k*-means algorithm, with $\left\{\mathcal{C}_1^*, \ldots, \mathcal{C}_{r-1}^*, \mathcal{C}_r = \mathcal{C}_r^* - \{\mathbf{x}_i\}, \mathcal{C}_{r+1}^*, \ldots, \mathcal{C}_{k-1}^*, \mathcal{C}_k = \{\mathbf{x}_i\}\right\}$ as initial clusters for the i-th run, and keep the one resulting in the lowest intra-cluster variance. The above procedure is repeated until $k = M$. The global kernel *k*-means pseudocode is given in Algorithm 1.3.

The rationale behind this method is based on the assumption that a near optimal solution with k clusters can be obtained by starting from an initial state with $k-1$ near optimal clusters (solution of the $(k-1)$-clustering problem) and the k-th cluster

Algorithm 1.3 Global kernel *k*-means.

Input: Kernel matrix K, Number of clusters M
Output: Final clusters $\{\mathcal{C}_k\}_{k=1}^{M}$

// There is no need to solve for one cluster, as the optimal solution is trivial. Let $\mathcal{C}_1^* = \mathcal{X}$.
for all k-clustering problems, $k = 2$ to M **do**
  for all points $\mathbf{x}_i$, $i = 1$ to N **do** // Suppose $\mathbf{x}_i \in \mathcal{C}_r^*$.
    Run kernel *k*-means with:
    input $\left(K, k, \left\{\mathcal{C}_1^*, \ldots, \mathcal{C}_{r-1}^*, \mathcal{C}_r = \mathcal{C}_r^* - \{\mathbf{x}_i\}, \mathcal{C}_{r+1}^*, \ldots, \mathcal{C}_{k-1}^*, \mathcal{C}_k = \{\mathbf{x}_i\}\right\}\right)$
    output $\left(\left\{\mathcal{C}_1^i, \ldots, \mathcal{C}_k^i\right\}, \mathcal{E}_{\mathcal{H}}^{ik}\right)$
  end for
  // Solution with k clusters.
  Find $i^* = \underset{i}{\operatorname{argmin}}\, \mathcal{E}_{\mathcal{H}}^{ik}$ and set $\{\mathcal{C}_1^*, \ldots, \mathcal{C}_k^*\}$ to the partitioning corresponding to $\mathcal{E}_{\mathcal{H}}^{i^*k}$
end for
return $\{\mathcal{C}_1 = \mathcal{C}_1^*, \ldots, \mathcal{C}_M = \mathcal{C}_M^*\}$

appropriately initialized. As we consider only a single data point belonging to the k-th cluster when it is initialized, this is equivalent to initializing, during the i-th run, the k-th cluster center at point $\phi(\mathbf{x}_i)$ in feature space. Limiting the set of possible initial positions for the k-th center to those of the dataset points in feature space seems a reasonable choice. Since the optimal solution to the 1-clustering problem is known, the above idea can be iteratively applied to get M clusters. Note that during the execution of the algorithm, solutions for every k-clustering problem, $k < M$, are also obtained without additional cost. This is a rather desirable property in case we want to decide on the number of clusters present in the dataset.

Due to its close relation to global *k*-means and kernel *k*-means, global kernel *k*-means inherits their computational cost. Given the kernel matrix, the complexity of kernel *k*-means is $O(N^2\tau)$ scalar operations, where τ is the number of iterations until convergence is achieved. In the global kernel *k*-means algorithm, in order to solve the M-clustering problem we must run kernel *k*-means MN times. This leads to a complexity of $O(N^3M\tau)$. To reduce the high computational burden, two speeding up schemes were developed in [103, 105].

Finally, a weighted version of the global kernel *k*-means framework was introduced in [105] and applied to graph partitioning, analogously to the weighted variant of kernel *k*-means [37].

1.1.5 Convex Mixture Models

Exemplar-based mixture models [71], also called convex mixture models (CMMs), are simplified mixture models [12] which result in probabilistic (soft) assignments of data points to clusters and in the extraction of representative exemplars[2] from the dataset. When training these models, which is done by maximizing the log-likelihood, all in-

[2]An exemplar is a dataset point that acts as a cluster representative, similar to a medoid.

stances compete to become cluster representatives (i.e. exemplars), since *the number of the CMM components is equal to the number of data points* and *each component distribution is centered at a distinct dataset point*. In the end, the instances corresponding to the components that have received during training the highest priors are selected as exemplars.

Given a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x}_i \in \Re^d$, the CMM distribution is:

$$Q(\mathbf{x}) = \sum_{j=1}^{N} q_j f_j(\mathbf{x}),\ \mathbf{x} \in \Re^d, \tag{1.4}$$

where $q_j \geq 0$ denotes the prior probability of the j-th component, satisfying the constraint $\sum_{j=1}^N q_j = 1$, and $f_j(\mathbf{x})$ is an exponential family distribution, with its *expectation parameter* equal to the j-th data point $\mathbf{x}_j$. Note that the same exponential family is used for all components. Taking into account the bijection between regular exponential families and Bregman divergences [9], we can write:

$$f_j(\mathbf{x}) = C_\varphi(\mathbf{x}) \exp(-\beta d_\varphi(\mathbf{x}, \mathbf{x}_j)), \tag{1.5}$$

with d_φ denoting the Bregman divergence corresponding to the components' distributions, $C_\varphi(\mathbf{x})$ being independent of $\mathbf{x}_j$ and β being a constant controlling the sharpness of the components [71].

A clustering is produced by maximizing the dataset log-likelihood (1.6) over the prior probabilities q_j, s.t. $q_j \geq 0$, $\sum_{j=1}^N q_j = 1$. Note that *the priors of the components are the only adjustable parameters* of a CMM.

$$L\left(\mathcal{X}; \{q_j\}_{j=1}^N\right) = \frac{1}{N}\sum_{i=1}^{N} \log\left(\sum_{j=1}^{N} q_j f_j(\mathbf{x}_i)\right) = \frac{1}{N}\sum_{i=1}^{N} \log\left(\sum_{j=1}^{N} q_j e^{-\beta d_\varphi(\mathbf{x}_i, \mathbf{x}_j)}\right) + \text{const.} \tag{1.6}$$

This optimization problem is convex and can be solved with an iterative algorithm, whose updates for the components' prior probabilities are given by:

$$q_j^{(t+1)} = q_j^{(t)} \sum_{\mathbf{x} \in \mathcal{X}} \frac{\hat{P}(\mathbf{x}) f_j(\mathbf{x})}{\sum_{j'=1}^N q_{j'}^{(t)} f_{j'}(\mathbf{x})}\ , \tag{1.7}$$

where $\hat{P}(\mathbf{x}) = 1/N$, $\mathbf{x} \in \mathcal{X}$ is the empirical dataset distribution. The above iterative approach is *guaranteed to converge to the global optimum* as long as $q_j^{(0)} > 0, \forall j$ [31]. Importantly, the prior probability q_j associated with instance $\mathbf{x}_j$ is a measure of *how likely this instance is to become an exemplar*.

The ability of always being able to locate the global optimum makes this model attractive as it avoids the initialization and local optima problems of standard mixture models, which demand multiple executions of the EM algorithm [12, 35, 99]. Another important feature is that only the pairwise data distances $d_\varphi(\mathbf{x}_i, \mathbf{x}_j)$ take part in the calculation of the priors as $C_\varphi(\mathbf{x}_i)$ cancels out, thus the values of the instances are

not required if we are given the distances. Note that updating the priors costs $O(N^2\tau)$ scalar operations, where τ is the number of iterations until convergence.

Splitting the dataset into M disjoint clusters is done by requiring the instances with the M highest q_j values to serve as exemplars and then assigning the remaining instances to the exemplar with the highest posterior probability.

Finally, when clustering with a CMM we must select an appropriate value for the constant β ($0 < \beta < \infty$). It is possible to identify a reasonable range of β values by determining a reference value β_0. Lashkari and Golland [71] proposed the following empirical rule for β_0:

$$\beta_0 = N^2 \log N / \sum_{i,j=1}^{N} d_\varphi(\mathbf{x}_i, \mathbf{x}_j). \tag{1.8}$$

1.2 Multi-view Learning

In various scientific areas, such as bioinformatics, computer vision, text categorization, social computing, person identification etc., data is available from different sources and/or feature spaces[3]. For example, a person can be identified by face, fingerprint and iris characteristics, while an image can be described using both color and texture descriptors. Another example is web pages, which can be represented using the text of the web page and the hyperlink graph among the web pages. Also, to represent scientific articles, the abstract text and the title, as well as the co-author and citation graphs could be utilized.

The above examples outline situations where for the same instance in the dataset multiple representations, called *views*, are available. The frequent occurrence of multi-view data in practice has raised interest in the so-called multi-view learning [98, 120], which concerns the development of machine learning algorithms that simultaneously exploit all views. Conventional machine learning approaches (e.g. SVM, *k*-means, mixture models, spectral methods etc.) can only handle data with a single view. A straightforward extension to the multi-view setting is possible by concatenating the views (if they are given in vectorial form) into a single vector. However, this strategy has proven to be significantly less effective compared to methods developed explicitly for multi-view datasets, that consider the distinct context and statistical properties of each view [120].

Research on multi-view learning was greatly inspired by the seminal work of Blum and Mitchell [14] under the semi-supervised setting [91]. As such, on the following, we shall briefly review semi-supervised multi-view methods, before discussing the literature on multi-view clustering that this thesis focuses on. We shall begin, however, by first presenting the two basic principles that motivate the integration of multiple views in the learning process.

[3]The term feature space is used here to refer to a set of attributes (characteristics) of the data and should not be confused with the one referring to the kernel Hilbert space (Section 1.1). On the following, this distinction is clarified whenever it is not obvious from the context.

1.2.1 Multi-view Learning Principles

The success of multi-view algorithms is based on the *complementarity* and *consensus* principles [120]. The first can be seen as the intuitive justification for multi-view learning, while the second provides the theoretical justification.

Complementarity states that the different views may contain complementary information, i.e. information not found in the other views. By combining the information available in all views, the intrinsic structures in the data will be better revealed, leading to improved performance compared to using a single view.

Consensus states that the disagreement between two independent hypotheses on two distinct views is an upper bound on the error rate of either hypothesis. In detail, suppose the data is drawn from some distribution over triples $\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, y\right)$, where $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$ are the representations of an instance in the first and second view, respectively, and y is the label. Moreover, assume that views are conditionally independent given the label, i.e. $P(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}|y) = P(\mathbf{x}^{(1)}|y)P(\mathbf{x}^{(2)}|y)$. Let $h^{(v)}$ denote a hypothesis trained on the v-th view which predicts the label of an instance. Dasgupta et al. [33] proved that the probability of the two hypotheses disagreeing on the predicted labels on a set of unlabeled data bounds the probability of either hypothesis predicting the wrong label:

$$P(h^{(1)}(\mathbf{x}^{(1)}) \neq h^{(2)}(\mathbf{x}^{(2)})) \geq \max_{v} P(h^{(v)}(\mathbf{x}^{(v)}) \neq y). \tag{1.9}$$

The above inequality clearly illustrates that employing multiple views and maximizing the agreement of predictions across the views can enhance accuracy. Note that the conditional independence is often too strong to be satisfied in real applications. Abney [2] showed that a weak dependence can also lead to successful results.

1.2.2 Semi-supervised Learning with Multiple Views

Multi-view learning in the semi-supervised domain has been introduced by Yarowsky [125] and Blum and Mitchell [14]. In [125], a two-view word sense disambiguation algorithm was described, which uses two classifiers that bootstrap each other. Blum and Mitchell [14] proposed the co-training algorithm to train a classifier from two representations and showed that the Yarowsky method falls under their framework. Co-training, together with the co-EM approach of Nigam and Ghani [84] which extends EM [12, 35, 99] to two-view semi-supervised problems, have laid the foundations of multi-view learning and inspired subsequent research in this area. We next analyze them in more detail.

Co-training

In semi-supervised problems we have access to both labeled and unlabeled data [91]. The basic idea of co-training [14] is to train two learners (classifiers) on distinct views of the labeled data and iteratively allow each learner to label the unlabeled instances that predicts with the highest confidence. This way, the newly labeled instances by one

learner may help the other to improve its model. The success of co-training is based on three assumptions: i) *Sufficiency*: Each of the two views is sufficient for classification on its own, ii) *Compatibility*: the target functions over each view predict the same label for most examples, i.e. we assume that an instance has the same label in both views with high probability, and iii) *Conditional independence*: the views are conditionally independent given the label (Section 1.2.1).

Analytically, let $\mathcal{X}_l = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N_l}$ be the labeled and $\mathcal{X}_u = \{\boldsymbol{x}_i\}_{i=1}^{N_u}$ the unlabeled dataset. Furthermore, assume each instance is decomposed into two views, $\boldsymbol{x}_i = \left\{\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)}\right\}$, where $\mathbf{x}_i^{(v)}$ is the v-th view representation for instance $\boldsymbol{x}_i$. Co-training starts by training two classifiers, $h^{(1)}$ and $h^{(2)}$, on the first and second view, respectively, using $\mathcal{X}_l$. Subsequently, each classifier labels the data in $\mathcal{X}_u$ and moves a number of them from each class to the labeled set. The chosen instances are those whose labels are predicted with the highest confidence by the underlying classifier for each class and their number is set proportional to the empirical distribution of classes in $\mathcal{X}_l$. Then, each classifier is rebuilt from the augmented labeled set, using the corresponding view, and the process is repeated for a predefined number of iterations, or until $\mathcal{X}_u$ is left empty.

The rationale behind using the labels predicted by the classifier operating on the first view to train the classifier operating on the second view, and vice versa, is to exchange complementary view information between the learners and gradually drive $h^{(1)}$ and $h^{(2)}$ to agree on the labels. Some important observations about co-training could be made. First, it is limited to two views only. Second, it can be seen as a generic framework for two-view learning, where any classifier can be employed as long as it outputs some measure of confidence for its predictions. In the original co-training paper [14] the naive Bayes classifier was applied. Third, it is unclear how to label an unseen test example, since we end up with two distinct classifiers. Possibly we could base the decision on either of the two or, somehow, combine their predictions. Fourth, it does not define some specific multi-view objective, thus it is unclear which criterion it tries to optimize. This also makes impossible to guarantee the convergence of the algorithm. The co-training method was modified in [26, 93], so that an objective function that measures the degree of agreement between the two views is optimized, something known as co-regularization.

Overall, and despite the previous limitations, several variants of the co-training idea have been developed, not only for classification [14, 26, 36, 61, 93], but also for regression [16, 132] and clustering [10, 65, 66] problems.

Co-EM

Co-EM [84] is an EM/co-training hybrid, where EM in the semi-supervised domain [85] is adapted to handle data with two views through an iterative procedure closely resembling that of co-training. Once again assume that both a labeled dataset $\mathcal{X}_l$ and an unlabeled dataset $\mathcal{X}_u$ are available, whose instances consist of two views. Co-EM

considers two classifiers, $h^{(1)}$ and $h^{(2)}$, and associates them with the first and second view, respectively. Execution begins by training $h^{(1)}$ on $\mathcal{X}_l$. Then, $h^{(1)}$ probabilistically labels all the instances in $\mathcal{X}_u$ by estimating class posterior probabilities $P(y|\mathbf{x}_i^{(1)}), i = 1, \ldots, N_u$ (E-step in view 1). The labeled data $\mathcal{X}_l$ together with the probabilistically labeled (by $h^{(1)}$) data of $\mathcal{X}_u$ are fed to $h^{(2)}$ which is trained on the second view (M-step in view 2). After that, $h^{(2)}$ relabels $\mathcal{X}_u$ by estimating probabilities $P(y|\mathbf{x}_i^{(2)}), i = 1, \ldots, N_u$ (E-step in view 2), which in turn are used in conjunction with the labeled instances of $\mathcal{X}_l$ to retrain $h^{(1)}$ (M-step in view 1). These steps are repeated for a predefined number of iterations, hence information is exchanged between the views by using the probabilistic labels computed in one view to update the model of the classifier operating on the other view.

The four observations and rationale regarding co-training (mentioned in the previous subsection), also apply to the co-EM case. However, there exist some key differences between the two. Unlike co-training, the co-EM approach does not commit to the (most confident) labels generated by the classifiers, but re-estimates the class posterior probabilities of the entire unlabeled dataset after each iteration. Also, classifiers in co-EM must be able to process probabilistic labels and output class probabilities. This is a much stronger prerequisite than simply requiring classifiers to output the confidence of their predictions. Therefore, co-EM has been mainly studied with naive Bayes as the underlying classifier [47, 80, 84]. A co-EM version of SVM has been proposed in [17]. Note that co-EM has been found to perform better than co-training in many cases [80, 84] and has been extended to clustering problems as well [10, 11].

1.2.3 Clustering with Multiple Views

Most of the existing work in multi-view clustering extends well-known clustering algorithms to the multi-view setting by exploiting the "minimizing disagreement" idea (consensus principle) and the complementarity of the views (Section 1.2.1). The clustering process is guided by the assumption that the true clustering assigns corresponding points in each view to the same cluster, a form of compatibility assumption similar to that in co-training. We next take a closer look to some of the most representative multi-view clustering studies and denote by $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$ the dataset whose instances consist of V views, $\boldsymbol{x}_i = \left\{\mathbf{x}_i^{(v)}\right\}_{v=1}^{V}$, where $\mathbf{x}_i^{(v)}$ is the v-th view representation for instance $\boldsymbol{x}_i$.

Multi-view EM and Multi-view *k*-means

Bickel and Scheffer [10] developed a two-view EM and a two-view *k*-means algorithm under the assumption that the two views are conditionally independent. Their EM variant is actually a straightforward application of the co-EM algorithm to unsupervised learning, where mixture models take the place of classifiers and no labeled data is available. Thus, the posterior probabilities of the hidden variables (cluster labels) in one view are used to estimate the parameters of the mixture model in the other view.

As an instantiation of the method they consider mixtures of multinomial distributions for document clustering. In the end, an instance is assigned to the cluster with the highest average posterior over the two views.

The iterative procedure of co-EM is also followed for the two-view *k*-means approach. The views, however, exchange partitions instead of posterior probabilities. In more detail, in the E-step of each view an instance is assigned to its closest center in that view, while in the M-step the centers are updated according to the assignments produced by the other view. The sequence of the expectation and maximization steps is the same as in co-EM, starting from some initial (random) centers for the first view. As the returned partitions in the two views are not necessarily identical, the final cluster for each instance is decided to be the one whose centers exhibit the lowest average distance from the instance over the two views.

The previous two methods closely mimic co-EM, thus inheriting many of its drawbacks, such as the two-view restriction, the inexistence of an objective function and the inability to guarantee convergence. A generalized co-EM scheme for mixture model estimation with arbitrarily many views has been proposed [11], to alleviate these deficiencies. It maximizes the sum of the log-likelihoods of the views, regularized by a consensus term Δ measuring the disagreement of the views on the posterior probabilities. Given a dataset $\mathcal{X}$ with V views, the optimized objective is:

$$\sum_{v=1}^{V} L(\mathcal{X};\boldsymbol{\theta}^{(v)}) - \eta\Delta,\ \Delta = \frac{1}{V-1}\sum_{v\neq u}\sum_{i=1}^{N}\sum_{j=1}^{M} P(j|\mathbf{x}_i^{(v)},\boldsymbol{\theta}_t^{(v)})\log\frac{P(j|\mathbf{x}_i^{(v)},\boldsymbol{\theta}^{(v)})}{P(j|\mathbf{x}_i^{(u)},\boldsymbol{\theta}^{(u)})}, \quad (1.10)$$

where $L(\mathcal{X};\boldsymbol{\theta}^{(v)})$ is the log-likelihood of the v-th view, $\boldsymbol{\theta}^{(v)}$ are the parameters of the v-th mixture model, η is the regularization constant and M is the number of mixture components (clusters). The maximization is done by iterating over the views and executing an appropriate E-step and an appropriate M-step in each view. It is shown to converge, if η is annealed toward zero as iterations progress. Note that the two-view EM [10] can be derived as a special case of (1.10) for $V = 2$ and $\eta = 1$. Analogously to [11], a regularized objective that incorporates view disagreement for multi-view fuzzy *c*-means clustering is considered in [25].

Multi-view Spectral Clustering

Spectral clustering (Section 1.1.2), due to its empirical success and close connection to graph partitioning [37, 113], has also attracted considerable attention in the multi-view setting. Kumar and Daumé III [65] integrated the ideas of co-training to spectral clustering and presented an iterative method where, first, spectral clustering is executed on the individual views and, then, the top (largest) eigenvectors from each view are used to alter the affinity matrices of the remaining views. Hence, views "communicate" by exchanging eigenvectors. Given the affinity matrices $A^{(v)} \in \Re^{N\times N}$ of the views, $A_{ij}^{(v)} = s(\mathbf{x}_i^{(v)},\mathbf{x}_j^{(v)})$, the top M eigenvectors (M is the number of clusters) of the matrix $L^{(v)} = (D^{(v)})^{-1/2}A^{(v)}(D^{(v)})^{-1/2}$ are calculated and stored in matrix $E^{(v)} \in \Re^{N\times M}$. $D^{(v)}$

is a diagonal matrix with $D_{ii}^{(v)} = \sum_{j=1}^{N} A_{ij}^{(v)}$. Subsequently, the affinity matrices are modified according to $A^{(v)} = sym\left(\left(\sum_{u \neq v} E^{(u)} E^{(u)^\top}\right) A^{(v)}\right)$[4] and the whole process is repeated using the new affinities. The intuition behind this approach is to project the affinity matrix of each view onto the union of subspaces spanned by the top eigenvectors of the other views. Since these eigenvectors carry discriminative information about the clusters, the projection will help in revealing the underlying structures in the data by utilizing information from all views. Note that the employed spectral technique is that of Ng et al. [82] (see Section 1.1.2 for details). The final partitioning is obtained by discretizing the eigenvectors of the most informative view, following the same steps as in [82]. Similar to co-training, the above algorithm does not optimize a specific clustering criterion and its convergence cannot be ensured.

A multi-view spectral objective was developed in [66]. The problem is formulated as a regularized trace maximization (over $U^{(v)} \in \Re^{N \times M}$):

$$\max_{U^{(v)}} \sum_{v=1}^{V} tr\left(U^{(v)^\top} L^{(v)} U^{(v)}\right) + \lambda \sum_{\substack{v,u \\ v \neq u}} tr\left(U^{(v)} U^{(v)^\top} U^{(u)} U^{(u)^\top}\right), \; s.t.\; U^{(v)^\top} U^{(v)} = I, \quad (1.11)$$

where $L^{(v)}$ is defined as above and λ is a regularization constant. The $tr\left(U^{(v)^\top} L^{(v)} U^{(v)}\right)$ term appearing in (1.11) is actually the objective optimized by the Ng et al. [82] algorithm when executed on the affinity matrix of the v-th view. This objective is also closely related to the normalized cut graph criterion [37]. Hence, the first term in (1.11) sums the spectral objectives of the individual views, while the second can be regarded as a consensus term which enforces agreement among the views. The optimization is performed by alternating over the views and updating $U^{(v)}$ for given $U^{(u)}$, $u \neq v$, until convergence is achieved. $U^{(v)}$ consists of the top eigenvectors of matrix $L^{(v)} + \lambda \sum_{\substack{u \\ u \neq v}} U^{(u)} U^{(u)^\top}$.

An approach that generalizes the single view normalized cut objective to the multi-view case and can handle both directed and undirected graphs was introduced by Zhou and Burges [131]. Their idea can be explained as a vertex-wise mixture of Markov chains associated with different graphs and is applicable to problems with arbitrary number of views. Finally, de Sa [34] proposed a two-view spectral clustering algorithm that creates a bipartite graph of the views, i.e. a graph where only connections between points (vertices) on different views exist, and the affinity matrix of this graph is processed with the method of Ng et al. [82] to recover the clusters.

CCA-based Multi-view Clustering

Canonical correlation analysis (CCA) [51, 53] can be seen as the problem of finding projection directions for two (or more) sets of variables, such that the correlation between the projections of the variables is maximized. Each set of variables can be interpreted

[4] $sym(A) = (A + A^\top)/2$ is applied in order to get a symmetric matrix.

as a different view of the data, hence CCA naturally lends itself to multi-view learning. Let $X^{(v)} \in \Re^{N \times d^{(v)}}$ be the data matrix of the v-th view, whose i-th row corresponds to $\mathbf{x}_i^{(v)} \in \Re^{d^{(v)}}$, and $\mathbf{w}^{(v)} \in \Re^{d^{(v)}}$ be the corresponding projection vector. We shall here describe CCA in the case of two views, however note that it is also applicable to problems with more views. CCA simultaneously searches for those projection directions $\mathbf{w}^{(1)}$ and $\mathbf{w}^{(2)}$ that maximize the correlation ρ between the projected views:

$$\rho = \frac{\mathbf{w}^{(1)^\top} X^{(1)^\top} X^{(2)} \mathbf{w}^{(2)}}{\sqrt{\left(\mathbf{w}^{(1)^\top} X^{(1)^\top} X^{(1)} \mathbf{w}^{(1)}\right)\left(\mathbf{w}^{(2)^\top} X^{(2)^\top} X^{(2)} \mathbf{w}^{(2)}\right)}}. \tag{1.12}$$

The solution to this optimization is given by the eigenvectors of a generalized eigenvalue problem. Note that different eigenvectors define different pairs of projection vectors. Depending on the desired number of dimensions for the CCA projection, the top pairs, i.e. directions, yielding the largest correlations are retained. In essence, CCA aims to recover a latent subspace that is shared by all views. Afterward, learning can be executed in this subspace. Notice that CCA considers linear projections of the instances, thus it is impossible to accurately capture the properties of data exhibiting nonlinearities. To circumvent this issue, the kernelized version of CCA can be employed [51].

Two interesting CCA-based methods for multi-view clustering are those of Blaschko and Lampert [13] and Chaudhuri et al. [24]. The first [13], projects the data onto the top directions obtained by kernel CCA across the views and then applies *k*-means to cluster those projections. In the second [24], each of the views is assumed to be generated by a mixture of distributions and CCA is employed to project the data to the subspace spanned by the distributions' means. Then, a standard clustering algorithm is used in this subspace to partition the instances. The subspace is endowed with an important property for clustering: when projected onto it, the means of the distributions are well-separated, yet the distances between points from the same distributions is smaller than in the original space. Additionally, the authors provided theoretical results which ensure the method can recover the correct clusters with high probability.

Other Approaches

Several multi-view clustering methods employ matrix factorization techniques, in particular non-negative matrix factorization (NMF) [72], to derive a partitioning of the data. Given a matrix $A \in \Re^{n \times d}$ with non-negative entries ($A \geq 0$), NMF aims to find two matrices $B \in \Re^{n \times k}$ and $C \in \Re^{k \times d}$, also with non-negative entries, whose product yields a good approximation of A, hence $A \approx BC$. Different factorizations emerge by considering different cost functions for measuring the reconstruction error of the approximation, such as the minimization of the squared error:

$$\min_{B,C} \|A - BC\|_F^2, \; s.t. \; B \geq 0, \; C \geq 0. \tag{1.13}$$

In general, the NMF solution is not unique, since $BC = (BQ^{-1})(QC)$ for an arbitrary invertible matrix $Q \in \Re^{k \times k}$. Thus, additional constraints are usually imposed to ensure

both uniqueness and that a clustering solution naturally emerges from the resulting factorization. Notice that for most cost functions the underlying optimization problem is not convex, therefore NMF algorithms can only locate local optimal solutions for B and C.

Gao et al. [46] apply NMF to factorize the data matrix of each view. Specifically, they optimize the sum of the NMF reconstruction errors of the individual views, regularized by a term enforcing consensus among the views' factorizations, in order to find a clustering that is consistent with all views. In [70], each view is represented by a similarity matrix and these matrices are linearly combined to get a composite matrix which contains similarity information from all views. A partitioning of the instances is found by performing NMF, with cross-entropy as the cost function, on the composite matrix. Importantly, the method learns appropriate weights for the linear combination, along with the factorization. Weights reflect the quality of the views and determine their contribution to the clustering task accordingly.

When considering problems with multiple views, it is possible to, first, cluster each view independently from the others, with an appropriate single view algorithm, and, then, combine the individual clusterings to produce a final partitioning which is based on all views [50, 77]. Here, we assume that each view's partitioning is described by a cluster indicator matrix, whose entries reflect the assignment of instances to clusters. In [50], an NMF-based approach is adopted to reconcile the groups arising from the individual views. Specifically, a matrix that contains the partitionings of all views is created by concatenating the cluster indicator matrices of the views and is then decomposed to two matrices (using NMF): the one showing the contribution of the individual partitionings to the final clusters, called meta-clusters, and the other the assignment of instances to the meta-clusters. In [77], a general model for multi-view unsupervised learning is proposed. According to this model, the final partitioning of the data, which is based on all views, is derived by minimizing an objective function that measures how close this final partitioning is to the clustering of each view, with the help of a mapping function. The whole process can be seen as performing NMF on the cluster indicator matrices of the views, where the factorizations across the views share a common factor. The generalized I-divergence is used as the NMF cost function.

Some other interesting approaches for clustering multi-view data can be found in [18, 20, 100, 102].

1.3 Multiple Kernel Learning

Kernel-based methods [92], e.g. SVM [19] and kernel *k*-means [37, 90], have become increasingly popular in recent years for both supervised and unsupervised machine learning tasks. Remember that (Section 1.1), a kernel implicitly induces a nonlinear transformation $\phi : \mathcal{X} \to \mathcal{H}$ that maps the instances, $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, from input space to feature space $\mathcal{H}$ and is defined through a kernel function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \Re$ [41], where

$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$ (see Table 1.1 for kernel function examples). Hence, learning is executed in feature space instead of input space, which allows for nonlinearities in the data to be uncovered and has been demonstrated to substantially enhance performance. The effectiveness of kernel methods, however, strongly depends on the choice of an appropriate kernel for the underlying problem. If an unsuitable kernel is selected, the quality of the solution can significantly degrade, which makes kernel selection a rather crucial step in the application of such methods. Unfortunately, the best kernel for a specific dataset is rarely known in advance.

Multiple kernel learning (MKL) [49] assumes a parametric form for the kernel and aims at estimating the values of these parameters during training, in order to automatically infer a kernel that suits the data. The most common MKL strategy is to parametrize the kernel as a linear (e.g. [62, 88]), or nonlinear (e.g. [28, 111]) combination of some predefined kernels, called *basis kernels*[5]. Hence, learning the kernel becomes equivalent to learning appropriate values for the combination coefficients. Basis kernels are obtained by applying a single type, or different types of kernel functions on the same instances (e.g. RBF kernels with different σ values and/or polynomial kernels of different degrees). Moreover, for multi-view data (Section 1.2), they can be derived by using the different representations of the instances describing the different sources (or modalities) of the data. Under this perspective, MKL can be seen as a special case of multi-view learning, where basis kernels correspond to views. Note that for supervised problems cross-validation can be employed to select the best kernel. However, cross-validation requires an additional validation set and retains a single kernel in the solution, while MKL approaches find a combination of the available kernels, thus utilizing information from all kernels and leading to improved results (e.g. [69, 88]).

MKL has been predominantly studied in the supervised domain under the SVM paradigm (e.g. [49, 69, 88, 96]), while related literature focusing on clustering problems, such as those studied in this thesis, is considerably more limited (e.g. [110, 130]). In Section 1.3.1 and Section 1.3.2 we review a number of approaches for both cases. Note that MKL has been also applied to the problems of dimensionality reduction [75] and metric learning [116].

1.3.1 Supervised MKL

Suppose we are given a labeled dataset $\mathcal{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, $\mathbf{x}_i \in \Re^d$ and $y_i \in \{\pm 1\}$, and a kernel $\widetilde{\mathcal{K}} : \mathcal{X} \times \mathcal{X} \rightarrow \Re$ that is parametrized by a vector $\boldsymbol{\theta}$ of parameters, inducing a transformation $\widetilde{\phi} : \mathcal{X} \rightarrow \widetilde{\mathcal{H}}$ that maps the instances to feature space $\widetilde{\mathcal{H}}$, hence $\widetilde{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_j) = \widetilde{\phi}(\mathbf{x}_i)^\top \widetilde{\phi}(\mathbf{x}_j)$. Most MKL methods based on the SVM classifier (the reader is referred to [12, 19] for details on SVM), in principle, derive from the following

[5] Basis kernels are kernels for which the parameters of their corresponding kernel functions (e.g. the RBF σ parameter) are fixed to specific values before training.

optimization problem:

$$\min_{\boldsymbol{\theta},\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i, \tag{1.14}$$

$$s.t.\ \ y_i\left(\mathbf{w}^\top\widetilde{\phi}(\mathbf{x}_i)+b\right) \geq 1-\xi_i,\ \xi_i \geq 0,\ \Omega(\boldsymbol{\theta}),$$

where $\mathbf{w}$, b are the coefficients of the SVM hyperplane ($\|\mathbf{w}\|$ is the reciprocal of the margin), $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_N]^\top$ is the vector of slack variables capturing the misclassification error, $C > 0$ is the regularization constant and $\Omega(\boldsymbol{\theta})$ is a set of constraints on the kernel parameters. This optimization closely resembles that of the standard SVM [12, 19], with the only difference being that we also minimize w.r.t. $\boldsymbol{\theta}$ (s.t. $\Omega(\boldsymbol{\theta})$) in order to simultaneously find the hyperplane with the largest margin and also learn the kernel. Unlike the standard (convex) SVM for which the global optimum can be obtained, the above problem is not convex in general, due to $\boldsymbol{\theta}$ and its associated constraints. Therefore, various optimization techniques, such as semidefinite programming (SDP) [68, 69], semi-infinite linear programming (SILP) [96, 133], gradient-based methods [88, 111] etc., have been employed to locate local optimal solutions.

Lanckriet et al. [68, 69] were of the first to study MKL and they considered learning linear combinations of basis kernels. A linear mixture of kernels gives rise to a composite kernel $\widetilde{\mathcal{K}}$:

$$\widetilde{\mathcal{K}}(\mathbf{x}_i,\mathbf{x}_j) = \sum_{v=1}^{V}\theta_v\mathcal{K}^{(v)}(\mathbf{x}_i,\mathbf{x}_j), \tag{1.15}$$

that is parametrized by $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_V]^\top$. Each of the V basis kernels $\mathcal{K}^{(v)} : \mathcal{X}\times\mathcal{X} \to \Re$ implicitly induces a transformation $\phi^{(v)} : \mathcal{X} \to \mathcal{H}^{(v)}$ on the instances to a feature space $\mathcal{H}^{(v)}$, hence $\mathcal{K}^{(v)}(\mathbf{x}_i,\mathbf{x}_j) = \phi^{(v)}(\mathbf{x}_i)^\top\phi^{(v)}(\mathbf{x}_j)$. Let us denote by $K^{(v)} \in \Re^{N\times N}$, $\widetilde{K} \in \Re^{N\times N}$ the kernel matrices corresponding to the basis kernels and the composite kernel respectively, i.e. $K^{(v)}_{ij} = \mathcal{K}^{(v)}(\mathbf{x}_i,\mathbf{x}_j)$ and $\widetilde{K}_{ij} = \widetilde{\mathcal{K}}(\mathbf{x}_i,\mathbf{x}_j)$. For the linear combination case it is easy to see that:

$$\widetilde{K} = \sum_{v=1}^{V}\theta_v K^{(v)}. \tag{1.16}$$

Note that a kernel matrix is always a positive semidefinite matrix and vice versa, since it can be interpreted as a Gram matrix. Hence, *when learning a parametric kernel we must ensure that positive semidefiniteness is satisfied in order to get a valid kernel.*

According to Lanckriet et al. [68, 69], MKL with the linear mixture (1.15)-(1.16) is formulated as:

$$\min_{\boldsymbol{\theta}}\ \omega(\boldsymbol{\theta}),\ s.t.\ \ \theta_v \geq 0,\ \widetilde{K} \succeq 0,\ tr(\widetilde{K}) = c, \tag{1.17}$$

$$\omega(\boldsymbol{\theta}) = \max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \widetilde{K}_{ij},\ s.t.\ 0 \leq \alpha_i \leq C,\ \sum_{i=1}^{N} \alpha_i y_i = 0, \tag{1.18}$$

where $\omega(\boldsymbol{\theta})$ at a given $\boldsymbol{\theta}$ is defined as the optimal objective value of the standard SVM dual in feature space $\widetilde{\mathcal{H}}$ and $tr(\widetilde{K})$ denotes the trace of matrix $\widetilde{K}$. To solve this problem they cast it into a quadratically constrained quadratic program (QCQP), where the constraint enforcing positive semidefiniteness ($\widetilde{K} \succeq 0$) can be dropped, since it is implied by the nonnegativity of θ_v. Moreover, they consider the more general case where θ_v is allowed to also take negative values and cast the problem into an SDP ($\widetilde{K} \succeq 0$ is now necessary to get a valid kernel).

When considering a linear combination of basis kernels whose coefficients are restricted to nonnegative values, a sufficient condition ensuring that $\widetilde{\mathcal{K}}$ is a valid kernel as mentioned above, it can be verified that $\widetilde{\phi}(\mathbf{x}_i) = \left[\sqrt{\theta_1}\phi^{(1)}(\mathbf{x}_i)^\top, \ldots, \sqrt{\theta_V}\phi^{(V)}(\mathbf{x}_i)^\top\right]^\top$, therefore (1.14) can be rewritten as ($\mathbf{w} = [\mathbf{w}_1^\top, \ldots, \mathbf{w}_V^\top]^\top$):

$$\min_{\boldsymbol{\theta}, \mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2} \sum_{v=1}^{V} \|\mathbf{w}_v\|^2 + C \sum_{i=1}^{N} \xi_i, \tag{1.19}$$

$$s.t.\ y_i \left(\sum_{v=1}^{V} \mathbf{w}_v^\top \left(\sqrt{\theta_v} \phi^{(v)}(\mathbf{x}_i) \right) + b \right) \geq 1 - \xi_i,\ \xi_i \geq 0,\ \theta_v \geq 0,\ \Omega(\boldsymbol{\theta}).$$

Rakotomamonjy et al. [87,88] address MKL by solving the following problem to infer a linear mixture of basis kernels:

$$\min_{\boldsymbol{\theta}, \mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2} \sum_{v=1}^{V} \frac{\|\mathbf{w}_v\|^2}{\theta_v} + C \sum_{i=1}^{N} \xi_i, \tag{1.20}$$

$$s.t.\ y_i \left(\sum_{v=1}^{V} \mathbf{w}_v^\top \phi^{(v)}(\mathbf{x}_i) + b \right) \geq 1 - \xi_i,\ \xi_i \geq 0,\ \theta_v \geq 0,\ \sum_{v=1}^{V} \theta_v = 1,$$

which can be directly derived from (1.19) by a simple change of variable ($\mathbf{w}_v = \sqrt{\theta_v}\mathbf{w}_v$). Their framework, called simpleMKL, imposes a 1-norm constraint on $\boldsymbol{\theta}$, $\sum_{v=1}^{V} \theta_v = 1$, to avoid overfitting, which is known to produce sparse solutions. To optimize (1.20) they reformulate it as:

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}),\ s.t.\ \theta_v \geq 0,\ \sum_{v=1}^{V} \theta_v = 1, \tag{1.21}$$

$$J(\boldsymbol{\theta}) = \min_{\mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2} \sum_{v=1}^{V} \frac{\|\mathbf{w}_v\|^2}{\theta_v} + C \sum_{i=1}^{N} \xi_i,\ s.t.\ y_i \left(\sum_{v=1}^{V} \mathbf{w}_v^\top \phi^{(v)}(\mathbf{x}_i) + b \right) \geq 1 - \xi_i,\ \xi_i \geq 0, \tag{1.22}$$

where $J(\boldsymbol{\theta})$ at a given $\boldsymbol{\theta}$ is actually the optimal objective value of the standard SVM in feature space $\widetilde{\mathcal{H}}$, and propose an iterative procedure that consists of two steps: i) solve a standard SVM for given $\boldsymbol{\theta}$ to get $J(\boldsymbol{\theta})$ and ii) update $\boldsymbol{\theta}$ using the gradient of $J(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$. Specifically, a projected gradient update is executed so that the constraints on the composite kernel parameters are not violated. SimpleMKL can be readily extended to regression and one-against-all, or one-against-one multiclass classification [88]. In [23], a second order method is presented, extending simpleMKL, and is shown to converge faster than simpleMKL.

Kloft et al. [62] generalize MKL to arbitrary p-norms ($p \geq 1$) on the composite kernel coefficients:

$$\min_{\boldsymbol{\theta},\mathbf{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\sum_{v=1}^{V}\frac{\|\mathbf{w}_v\|^2}{\theta_v} + C\sum_{i=1}^{N}\xi_i, \tag{1.23}$$

$$s.t. \ \ y_i\left(\sum_{v=1}^{V}\mathbf{w}_v^\top\phi^{(v)}(\mathbf{x}_i)+b\right) \geq 1-\xi_i, \ \xi_i \geq 0, \ \theta_v \geq 0, \ \|\boldsymbol{\theta}\|_p^p \leq 1.$$

To solve this problem they devise two alternating approaches, based on Newton descent and cutting planes. Note that simpleMKL (1.20) is a special case of (1.23) for $p = 1$. Moreover, they prove that an alternative formulation for p-norm MKL, where an additional regularizer $\mu\|\boldsymbol{\theta}\|_p^p$ is inserted into the objective function in place of the norm constraint $\|\boldsymbol{\theta}\|_p^p \leq 1$ (such a formulation is considered by Varma and Ray [112] for $p = 1$), is equivalent to (1.23). MKL for arbitrary p-norms has been also studied in [63, 124], where closed-form solutions are derived for updating $\boldsymbol{\theta}$. Both [63] and [124] arrive at the same solution, although they approach the problem from different perspectives. These closed-form solutions are utilized in an iterative optimization procedure that solves a standard SVM for fixed $\boldsymbol{\theta}$ and, subsequently, reestimates $\boldsymbol{\theta}$.

Bach et al. [4][6] and Sonnenburg et al. [95, 96] adopt a slightly different approach to MKL and use a $(2/1)$-block norm on the blocks of $\mathbf{w} = [\mathbf{w}_1^\top, \ldots, \mathbf{w}_V^\top]^\top$[7]:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\left(\sum_{v=1}^{V}\|\mathbf{w}_v\|\right)^2 + C\sum_{i=1}^{N}\xi_i, \ s.t. \ \ y_i\left(\sum_{v=1}^{V}\mathbf{w}_v^\top\phi^{(v)}(\mathbf{x}_i)+b\right) \geq 1-\xi_i, \ \xi_i \geq 0. \tag{1.24}$$

The solution of this problem yields a composite kernel of the form described in (1.15), where $\theta_v \geq 0$ and $\sum_{v=1}^{V}\theta_v = 1$, although the kernel parameters are omitted from the objective. Note that the block norm on w results on a 1-norm constraint on $\boldsymbol{\theta}$, thus promoting a sparse outcome. In [4], eq. (1.24) is treated as a second-order cone program (SOCP) which allows for the development of an SMO-based (sequential minimal optimization) algorithm, while in [95, 96] it is converted to a SILP formulation that is capable of handling large scale problems with hundreds of basis kernels.

[6] For simplicity we set d_j in Bach et al. [4] to one.

[7] A 2-norm is applied within each block ($\|\mathbf{w}_v\|$), while the 1-norm is applied over the blocks, giving $\sum_{v=1}^{V}\|\mathbf{w}_v\|$.

Performing MKL with SVM as the underlying classifier suffers from a serious limitation: due to the SVM we are restricted to problems with two classes only. One possible way of dealing with multiclass problems is to decompose them into binary problems, following a one-against-all, or a one-against-one strategy (e.g. [88]). Zien and Ong [133] propose an SVM-based MKL framework that can directly handle multiclass data, by extending the formulation in (1.20). For the optimization they consider both QCQP and SILP techniques. Additionally, they make an important observation. Their formulation (in the case of two classes) is equivalent to that of Bach et al. [4] and Sonnenburg et al. [95, 96]. Kloft et al. [62] further extend this result and show that the formulations of Rakotomamonjy et al. [87, 88] and Varma and Ray [112], as well as their own p-norm MKL (1.23) for $p = 1$, fall into the same equivalence class with [4, 95, 96, 133]. Moreover, in [124] it is proved that the p-norm MKL (1.23) for $p \geq 1$ is equivalent to the following $(2/q)$-block norm problem:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\left(\sum_{v=1}^{V}\|\mathbf{w}_v\|^q\right)^{\frac{2}{q}} + C\sum_{i=1}^{N}\xi_i,\ s.t.\ y_i\left(\sum_{v=1}^{V}\mathbf{w}_v^\top\phi^{(v)}(\mathbf{x}_i)+b\right)\geq 1-\xi_i,\ \xi_i\geq 0, \quad (1.25)$$

with $q = \frac{2p}{p+1}$. Note that when varying p in the interval $[1,\infty)$, q is limited to $[1,2]$. Hence, the block norm formulation is strictly more general and was pursued in [64], together with an elastic net regularizer, for $1 \leq q < \infty$.

Despite their variety, all the aforementioned MKL methods focus on a linear combination of the basis kernels (1.15). On the remainder of this section, we will review approaches that learn a composite kernel as a nonlinear mixture of basis kernels, or even accommodate general types of parametric kernels. Cortes et al. [28] develop a polynomial combination of basis kernels:

$$\widetilde{\mathcal{K}}(\mathbf{x}_i,\mathbf{x}_j) = \sum_{\substack{k_1+\ldots+k_V=d\\ k_v\in\mathbb{Z}_+\cup\{0\}}} \theta_1^{k_1}\theta_2^{k_2}\ldots\theta_V^{k_V}\left(\mathcal{K}^{(1)}(\mathbf{x}_i,\mathbf{x}_j)\right)^{k_1}\left(\mathcal{K}^{(2)}(\mathbf{x}_i,\mathbf{x}_j)\right)^{k_2}\ldots\left(\mathcal{K}^{(V)}(\mathbf{x}_i,\mathbf{x}_j)\right)^{k_V}, \quad (1.26)$$

that is parametrized by $\boldsymbol{\theta} = [\theta_1,\ldots,\theta_V]^\top$ and its degree is determined by the choice of the constant d. Note that all k_v exponents are nonnegative integers and the summation in (1.26) is over all possible combinations of $k_1, k_2, \ldots, k_V$ values whose sum equals d. This composite kernel is applied to regression problems using kernel ridge regression (KRR) as the underlying learning algorithm. The coefficients $\boldsymbol{\theta}$ are restricted to nonnegative values ($\theta_v \geq 0$), in order to get a valid kernel, and, furthermore, either a 1-norm constraint, $\|\boldsymbol{\theta}-\boldsymbol{\theta}_0\|_1 \leq \Lambda$, or a 2-norm constraint, $\|\boldsymbol{\theta}-\boldsymbol{\theta}_0\| \leq \Lambda$, is imposed ($\boldsymbol{\theta}_0$ and Λ are predefined constants). An iterative gradient-based procedure is employed to optimize the resulting objective. Gönen and Alpaydin [48] introduce a composite kernel that assigns different coefficients to the basis kernels in different regions of the input space, thus accounting for localities in the data:

$$\widetilde{\mathcal{K}}(\mathbf{x}_i,\mathbf{x}_j) = \sum_{v=1}^{V}\theta_v(\mathbf{x}_i)\mathcal{K}^{(v)}(\mathbf{x}_i,\mathbf{x}_j)\theta_v(\mathbf{x}_j). \quad (1.27)$$

As it can be seen, the parameters of $\widetilde{\mathcal{K}}$ are actually functions of the instances and their values are calculated through a gating model. In [48], the softmax function is considered as a possible instantiation of the gating model:

$$\theta_v(\mathbf{x}) = \frac{\exp(\mathbf{f}_v^\top \mathbf{x} + f_{v0})}{\sum_{v=1}^{V} \exp(\mathbf{f}_v^\top \mathbf{x} + f_{v0})}. \tag{1.28}$$

Hence, learning $\widetilde{\mathcal{K}}$ equals learning the gating model parameters $\mathbf{f}_v$, f_{v0}, which is done by incorporating the composite kernel into the SVM classifier and obtaining the following MKL problem:

$$\min_{\mathbf{f}_v, f_{v0}, \mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2} \sum_{v=1}^{V} \|\mathbf{w}_v\|^2 + C \sum_{i=1}^{N} \xi_i, \ s.t. \ y_i \left(\sum_{v=1}^{V} \mathbf{w}_v^\top \left(\theta_v(\mathbf{x}_i) \phi^{(v)}(\mathbf{x}_i) \right) + b \right) \geq 1 - \xi_i, \ \xi_i \geq 0. \tag{1.29}$$

To solve (1.29), a two step iterative procedure is proposed. This procedure is inspired by simpleMKL [88] and alternates between solving a standard SVM for fixed $\mathbf{f}_v, f_{v0}$, i.e. fixed $\theta_v(\mathbf{x})$, and updating the parameters of the gating model by means of gradient descent. Another nonlinear MKL approach can be found in [73].

Varma and Babu [111] propose an MKL framework that can handle generic types of parametric kernels $\widetilde{\mathcal{K}}$ and, therefore, unlike the methods we have so far presented in this section, is not limited to a specific form of $\widetilde{\mathcal{K}}$. This generalized MKL framework optimizes the following objective:

$$\min_{\boldsymbol{\theta}, \mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \xi_i + r(\boldsymbol{\theta}), \ s.t. \ y_i \left(\mathbf{w}^\top \widetilde{\phi}(\mathbf{x}_i) + b \right) \geq 1 - \xi_i, \ \xi_i \geq 0, \ \theta_v \geq 0, \tag{1.30}$$

where $r(\boldsymbol{\theta})$ is a regularizer on the kernel parameters $\boldsymbol{\theta}$. To efficiently solve (1.30), it is reformulated as:

$$\min_{\boldsymbol{\theta}} \ T(\boldsymbol{\theta}), \ s.t. \ \theta_v \geq 0, \tag{1.31}$$

$$T(\boldsymbol{\theta}) = \min_{\mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \xi_i + r(\boldsymbol{\theta}), \ s.t. \ y_i \left(\mathbf{w}^\top \widetilde{\phi}(\mathbf{x}_i) + b \right) \geq 1 - \xi_i, \ \xi_i \geq 0, \tag{1.32}$$

and an iterative process, closely resembling that of simpleMKL [88], is applied, where a standard SVM is executed for given $\boldsymbol{\theta}$ to get $T(\boldsymbol{\theta})$ and, then, $\boldsymbol{\theta}$ is updated using the gradient of $T(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$ while preserving $\theta_v \geq 0$. Note that the only restriction imposed on the learned kernel $\widetilde{\mathcal{K}}$ and the regularizer $r(\boldsymbol{\theta})$ is that they must be continuously differentiable functions of $\boldsymbol{\theta}$, to ensure the gradient exists. In [111], gender identification experiments on a collection of face images are conducted, with $\widetilde{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_j) = \prod_{v=1}^{d} \exp(-\theta_v (x_{iv} - x_{jv})^2)$ (x_{iv} is the v-th attribute of instance $\mathbf{x}_i$) and a 1-norm regularizer. Gai et al. [45] also present an MKL scheme that is able to handle various forms of parametric kernels $\widetilde{\mathcal{K}}$. However, they utilize the ratio between the

margin of the SVM and the radius of the minimum enclosing ball of the data in feature space $\widetilde{\mathcal{H}}$, denoted by $R(\boldsymbol{\theta})$, to perform MKL. The resulting objective is given by:

$$\min_{\boldsymbol{\theta},\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}R^2(\boldsymbol{\theta})\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i,\ s.t.\ y_i\left(\mathbf{w}^\top\widetilde{\phi}(\mathbf{x}_i)+b\right) \geq 1-\xi_i,\ \xi_i \geq 0,\ \Omega(\boldsymbol{\theta}), \quad (1.33)$$

where the square of the radius of the minimum enclosing ball for a specific $\boldsymbol{\theta}$ can be obtained by:

$$R^2(\boldsymbol{\theta}) = \min_{t,\mathbf{c}}\ t,\ s.t.\ t \geq \|\widetilde{\phi}(\mathbf{x}_i)-\mathbf{c}\|^2, \quad (1.34)$$

which is a convex problem that is optimally solved through its dual. Integrating $R(\boldsymbol{\theta})$ into (1.33) is motivated by the observation that the margin alone is not a suitable measure of the goodness of the learned kernel, as it can become arbitrarily large by simply scaling $\widetilde{\mathcal{K}}$, something known as the scaling problem. Hence, even a poor performing kernel can give an arbitrary large margin. Moreover, the scaling problem causes the initialization problem, i.e. the solution is affected by the initial scalings of the basis kernels. Gai et al. [45] prove that their formulation is scale invariant and, in the case of a linear combination of basis kernels, it also avoids the initialization problem and is invariant to the choice of p-norm constraint on $\boldsymbol{\theta}$. The whole MKL process is transformed into a tri-level optimization problem that is solved by an iterative gradient-based algorithm. This algorithm alternates among finding $R(\boldsymbol{\theta})$ for fixed $\boldsymbol{\theta}$, solving a standard SVM for the current $\{R(\boldsymbol{\theta}),\ \boldsymbol{\theta}\}$ values and taking a step along the projected gradient, to update $\boldsymbol{\theta}$ without violating the constraints $\Omega(\boldsymbol{\theta})$. For the gradient to exist, $\widetilde{\mathcal{K}}$ must be a continuously differentiable function of $\boldsymbol{\theta}$.

Although the vast majority of MKL methods is built upon the SVM classifier, it is worth mentioning that there exist approaches that employ other types of base learners. For example, kernel ridge regression is utilized in [27] and [28] to learn a linear and a nonlinear combination of basis kernels, respectively, while in [44, 60] linear mixtures of basis kernels are found using kernel Fisher discriminant analysis. Another popular direction to perform MKL is to consider the kernel alignment criterion. Kernel alignment [30] defines a notion of cosine similarity between two arbitrary kernel matrices $K^{(1)} \in \Re^{N\times N}$ and $K^{(2)} \in \Re^{N\times N}$ as follows:

$$A(K_1,K_2) = \frac{\left\langle K^{(1)},K^{(2)}\right\rangle_F}{\sqrt{\left\langle K^{(1)},K^{(1)}\right\rangle_F\left\langle K^{(2)},K^{(2)}\right\rangle_F}}, \quad (1.35)$$

where $\left\langle K^{(1)},K^{(2)}\right\rangle_F = \sum_{i=1}^{N}\sum_{j=1}^{N}K_{ij}^{(1)}K_{ij}^{(2)}$ is the Frobenius product. For a binary classification task, we can view the matrix $\mathbf{y}\mathbf{y}^\top$ as the "ideal" kernel matrix, where $\mathbf{y} = [y_1,\ldots,y_N]^\top$, $y_i \in \{\pm 1\}$, is the vector of the class labels. Hence, to learn a parametric kernel $\widetilde{\mathcal{K}}$, we try to maximize the alignment between its kernel matrix $\widetilde{K}$ and $\mathbf{y}\mathbf{y}^\top$, w.r.t. the parameters $\boldsymbol{\theta}$. Afterward, a classifier can be built using the learned kernel. Kernel alignment is applied to infer linear combinations of basis kernels in [29, 55, 57, 69], which are then used to train a standard SVM.

1.3.2 MKL Clustering

Enhancing clustering algorithms by learning an appropriate kernel for the dataset at hand along with the cluster assignments, using MKL techniques, constitutes an interesting research direction. However, few approaches have been presented in the literature to tackle this problem. Here, we will mainly focus on those approaches that extend a relatively new clustering paradigm, called maximum margin clustering [121], to the MKL setting and briefly mention some other MKL methods for clustering as well. First of all, however, we shall provide a quick overview of maximum margin clustering.

Maximum Margin Clustering

Maximum margin clustering (MMC) has been proposed by Xu et al. [121] and expands the large margin principle of SVM to the clustering domain. Given a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x}_i \in \Re^d$, MMC attempts to find a labeling (clustering) $\mathbf{y} = [y_1, \ldots, y_N]^\top$, $y_i \in \{\pm 1\}$, of the instances, such that a subsequent training of a standard SVM [12, 19] would result in a margin that is maximal over all possible labellings. MMC is formulated as:

$$\min_{\mathbf{y}} \min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^N \xi_i, \tag{1.36}$$

$$s.t. \;\; -\ell \leq \sum_{i=1}^N y_i \leq \ell, \; \mathbf{y} \in \{\pm 1\}^N, \; y_i\left(\mathbf{w}^\top \phi(\mathbf{x}_i) + b\right) \geq 1 - \xi_i, \; \xi_i \geq 0,$$

where $\mathbf{w}$, b, $\boldsymbol{\xi}$ and C are the usual SVM parameters (see (1.14) for details) and a kernel $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \Re$ is used to implicitly define a transformation $\phi : \mathcal{X} \to \mathcal{H}$ of the instances to a feature space $\mathcal{H}$, i.e. $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$. To prevent the trivially "optimal" solution of assigning all instances to the same cluster and thus obtaining an infinite margin ($\|\mathbf{w}\| = 0$), a cluster balance constraint ($-\ell \leq \sum_{i=1}^N y_i \leq \ell$) is necessary, where $\ell \geq 0$ is a constant controlling the imbalance of the clusters. Note that MMC can only handle datasets exhibiting a two-cluster structure ($y_i \in \{\pm 1\}$).

The MMC problem is non-convex with integer parameters $\mathbf{y}$, making the optimization much trickier than that of the standard (convex) SVM where the labels $\mathbf{y}$ are known in advance. To ease the optimization, Xu et al. [121] make a number of relaxations with regard to $\mathbf{y}$ and also remove the bias b ($b = 0$), hence the recovered hyperplane is required to pass through the origin. Subsequently, they cast MMC into an SDP problem, whose number of parameters is quadratic in the number of instances, and the cluster assignments are recovered by eigendecomposition of an appropriate matrix. Valizadegan and Jin [110] reduce the number of parameters in the SDP from N^2 to N and reinstate the bias term in the hyperplane, leading to an improved MMC framework. Solving an SDP is computationally expensive and the algorithms in [110, 121] can only cluster datasets containing a few hundreds of samples. Zhang et al. [129]

present a more efficient optimization strategy that alternates between the inner and the outer minimization in (1.36). However, instead of relying on a straightforward implementation, where a standard SVM is executed for given $\mathbf{y}$ and, then, $\mathbf{y}$ is updated according to $y_i = \text{sign}(\mathbf{w}^\top \phi(\mathbf{x}_i) + b)$, they perform a key modification. The SVM arising in the inner subproblem is replaced by support vector regression (SVR), to avoid the premature convergence issue observed when SVM is employed. Moreover, the bias b is not directly determined through the SVR, but is set (after obtaining $\mathbf{w}$ for the current $\mathbf{y}$ using the SVR) to an appropriate value that guarantees the cluster update rule $y_i = \text{sign}(\mathbf{w}^\top \phi(\mathbf{x}_i) + b)$ will respect the cluster balance constraint. Another efficient approach for solving MMC, based on the cutting plane method, is reported in [115]. Finally, [115, 122] extend MMC to the multiclass case (i.e. to problems with more than two clusters).

Combining MMC with MKL

As in Section 1.3.1, we assume a kernel $\widetilde{\mathcal{K}} : \mathcal{X} \times \mathcal{X} \to \Re$ (parametrized by a vector $\boldsymbol{\theta}$ of parameters) to which a transformation $\widetilde{\phi} : \mathcal{X} \to \widetilde{\mathcal{H}}$ corresponds. There exist two methods that couple MMC with MKL techniques [110, 130]. Both [110] and [130] infer a linear mixture of basis kernels, of the form described in (1.15)-(1.16), and impose a 1-norm and a 2-norm constraint on $\boldsymbol{\theta}$, respectively.

In more detail, Zhao et al. [130] express kernel learning under the MMC paradigm analogously to the p-norm MKL (1.23):

$$\min_{\boldsymbol{\theta},\mathbf{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\sum_{v=1}^{V} \frac{\|\mathbf{w}_v\|^2}{\theta_v} + C\sum_{i=1}^{N} \xi_i, \tag{1.37}$$

$$s.t. \quad -\ell \leq \sum_{i=1}^{N}\left(\sum_{v=1}^{V} \mathbf{w}_v^\top \phi^{(v)}(\mathbf{x}_i) + b\right) \leq \ell, \ \left|\sum_{v=1}^{V} \mathbf{w}_v^\top \phi^{(v)}(\mathbf{x}_i) + b\right| \geq 1 - \xi_i,$$

$$\xi_i \geq 0, \ \theta_v \geq 0, \ \|\boldsymbol{\theta}\|^2 \leq 1.$$

Note that a slightly relaxed cluster balance constraint is used compared to (1.36) and the cluster assignments are calculated as $y_i = \text{sign}\left(\sum_{v=1}^{V} \mathbf{w}_v^\top \phi^{(v)}(\mathbf{x}_i) + b\right)$, allowing for the removal of $\mathbf{y}$ from the objective. Zhao et al. [130] devise a cutting plane method to optimize (1.37) and also expand their formulation to problems with several clusters.

As already mentioned, Valizadegan and Jin [110] convert MMC (1.36) into an SDP problem (see the MMC subsection above). This SDP formulation is further extended to perform MKL by incorporating a linear mixture of basis kernels into it, leading to the following optimization problem:

$$\max_{\boldsymbol{\theta},\boldsymbol{\gamma}} \ \sum_{i=1}^{N} \gamma_i, \tag{1.38}$$

$$s.t.\ \ P^\top \widetilde{K}^{-1} P + C_e \mathbf{e}_0 \mathbf{e}_0^\top - \sum_{i=1}^{N} \gamma_i I_{N+1}^i \succeq 0,\ 0 \leq \gamma_i \leq C_\delta,\ \theta_v \geq 0,\ \sum_{v=1}^{V} \theta_v = 1,$$

where the kernel matrix $\widetilde{K}$ is defined as in (1.16), $\boldsymbol{\gamma}$ is a vector of dual parameters, C_e and C_δ are regularization constants, $\mathbf{e}_0$ is an $(N+1)$-dimensional vector whose last element is zero and all other elements are equal to 1, I_{N+1}^i is an $(N+1) \times (N+1)$ matrix with all the elements being zero except the i-th diagonal element which is equal to 1 and $P = [I, \mathbf{e}]$ is an $N \times (N+1)$ matrix, where I is the identity matrix and $\mathbf{e}$ is a vector with all its elements equal to 1. By solving (1.38), we are able to simultaneously resolve the cluster memberships (through $\boldsymbol{\gamma}$) and learn the composite kernel $\widetilde{\mathcal{K}}$ (1.15) (through $\boldsymbol{\theta}$).

Other Approaches

Zeng and Cheung [126, 127] consider a linear combination of basis kernels (1.15) within the framework of local learning based clustering [119]. An iterative gradient-based procedure is deployed to locate the clusters and learn the composite kernel. Baili and Frigui [7] utilize a kernelized version of the popular fuzzy *c*-means algorithm [99] and learn a cluster-dependent linear mixture of basis kernels, i.e. the coefficients of the basis kernels have different values across the different clusters. The optimization process resembles that of the original fuzzy *c*-means algorithm, where an additional step is included to update the cluster-dependent coefficients using closed-form expressions.

Some other interesting unsupervised methods, which infer a linear combination of basis kernels, can be found in [1, 117].

1.4 Thesis Contribution

In this thesis, we study the clustering problem, mainly focusing on three different axes: i) proximity-based clustering, ii) clustering of data available in multiple views and iii) learning the kernel along with the cluster assignments using multiple kernel learning (MKL) techniques. Note that these problems are not completely independent and there is certain overlap among them. For example, MKL is related to proximity-based approaches, since instances are represented by kernel matrices, or to multi-view learning if the basis kernels are derived from different views. Next, we summarize the contribution of this thesis.

In Chapter 2, we present an approach that improves the *k*-means algorithm [76]. Specifically, we alter the *k*-means objective and instead of minimizing the sum of the intra-cluster variances, we optimize a weighted sum of the intra-cluster variances [108]. The optimization is performed using an efficient iterative algorithm, which closely resembles *k*-means, and the values of the weights are updated through closed-form expressions. The main novelty of the proposed objective is that the weights on clusters predispose our model towards primarily minimizing those clusters which in the current

iteration exhibit large intra-cluster variances. In this way, the solution space is gradually restricted towards clusters with similar variances. The unweighted summation in the original k-means objective does not account for the relative differences of the cluster variances, thus is it very easy, after a bad initialization, to come up with a solution where some of the natural groups in the data have been merged (large variance clusters) and others have been split (small variance clusters). Hence, our approach can *produce high quality partitionings more regularly* than k-means, while restarted from random initial centers. Moreover, our clustering scheme utilizes a parameter, whose value must be set prior to execution, that controls how strongly the weighted objective penalizes larger variance clusters relative to smaller variance clusters. A practical framework is developed to automatically tune this parameter to the underlying dataset, so that the intrinsic structures in the data are successfully uncovered. Finally, we also present a kernelized version of the weighted objective to perform clustering in kernel space (i.e. feature space) [41] and thus locate nonlinearly separable clusters. The feature space extension requires only the kernel matrix and not the instances as input, i.e. it is a proximity-based method.

In a nutshell, we make the following contributions:

- we introduce a novel weighted objective to circumvent the initialization problem of k-means,
- we incorporate in our model a parameter that controls its behavior, allowing its adaptation to the dataset, and devise a practical framework to automatically adjust the value of this parameter,
- we extend our approach to kernel space,
- we experimentally evaluate the efficacy of our approach in overcoming bad initializations and systematically obtaining high quality solutions.

Chapter 3 and Chapter 4, focus on unsupervised multi-view learning (see Section 1.2). Most of the existing multi-view methods treat all available *views as being equally important* during training. However, in practice, degeneracies may occur, as views can contain noise or be irrelevant to the problem at hand. Including such views in the learning process can result in a significant performance degradation. Our key contribution in multi-view learning is that we present approaches that assign weights to the views and automatically tune these weights to *reflect the quality of the views* and determine their contribution to the clustering solution accordingly, thus providing robustness over degenerate views.

In more detail, in Chapter 3, we develop two multi-view algorithms that are based on convex mixture models (CMMs) [71]. CMMs are simplified mixture models that locate exemplars in the dataset and are characterized by the ability to converge to the global optimum, thus avoiding the initialization problem of standard mixture models (see Section 1.1.5). Our first approach is a simple extension of CMMs to data with

multiple views that considers all views as equally important [104]. Its most interesting features are the global optimality of the returned solution, the use of the pairwise distance matrix of the instances and that the different statistical properties of the views are taken into account. Our second approach represents each view with a CMM and learns a weighted combination of those CMMs [106]. Hence, degenerate views can be identified and appropriately handled. Moreover, this weighted combination has a probabilistic interpretation and takes into account the statistical properties of each view. Experiments reveal the superiority of assigning weights to the views.

In a nutshell, we make the following contributions:

- we present two CMM-based multi-view methods, where the first relies equally on every view and can find the global optimum solution, while the second associates a weight with each view and, therefore, can spot degenerate views,
- we consider the different statistical properties of the views,
- we experimentally evaluate the effectiveness of our approaches on both noisy and noise-free artificial datasets, as well as on real data.

Also, in Chapter 4, we draw inspiration from the MKL literature and perform multi-view clustering by representing each view with a kernel matrix (derived by employing a kernel function [41] on the view) and learning a weighted combination of the kernel matrices [107]. The influence of the views can be adjusted through the weights, thus the effects of degenerate views can be confined. The formulation includes a parameter, whose value must be set prior to execution, that controls the sparsity of the weights. A low value results in retaining only the best view (sparse solution), which is useful if most views are of poor quality, while a large value leads towards a uniform solution, which is preferable when all views are of similar quality. Intermediate values provide a tradeoff between these ends. To infer the weights and the cluster assignments, we develop an iterative algorithm that is based on kernel k-means [37, 90] (see Section 1.1.3) and estimates the weights using closed-form updates. Moreover, we exploit the connection between kernel k-means and spectral clustering [37] and present an alternative optimization algorithm under the spectral perspective.

In a nutshell, we make the following contributions:

- we present a multi-view method that assigns weights to the views,
- we use kernel matrices to represent the views, which connects multi-view learning to MKL,
- we utilize a parameter in our formulation that controls the sparsity of the weights,
- we develop two optimization strategies, one based on kernel k-means and the other on spectral techniques,
- we perform experiments on both synthetic and real data.

Finally, in Chapter 5, we focus on the MKL problem, to simultaneously infer an appropriate kernel (i.e. infer its parameters) and cluster the instances. As already mentioned (see Section 1.3), the vast majority of existing (supervised and unsupervised) MKL frameworks exploit the large margin principle of SVM [12, 19] and perform learning by maximizing the margin. Instead, here, we propose an objective that utilizes the *ratio* between the margin and the intra-cluster variance criterion of kernel k-means [37, 90]. The main advantages of this objective are that it explicitly considers both the separation (margin) and the compactness (intra-cluster variance) of the clusters, hence higher quality solutions can be possibly attained compared to approaches that rely on either of the two. Moreover, it has been shown that the margin alone is an unsuitable measure of the goodness of the learned kernel [45], as it can become arbitrary large by simply scaling the kernel. We prove that our ratio-based objective is invariant to kernel scaling and, also, that its global optimum solution is invariant to the type of norm constraint on the kernel parameters when a linear combination of basis kernels is considered. Additionally, our formulation can learn different types of parametric kernels (if some mild conditions are satisfied), while most of the available MKL methods can only handle parametric kernels of a specific form (usually a linear mixture of basis kernels). For the optimization, a gradient-based iterative algorithm has been developed that alternates between updating the kernel parameters and the cluster assignments.

In a nutshell, we make the following contributions:

- we introduce a novel ratio-based objective for MKL clustering that considers both the separation and the compactness of the clusters,
- we prove that our formulation is invariant to scalings of the learned kernel, as well as invariant (on its global optimum) to the type of norm constraint on the kernel parameters when basis kernels are linearly mixed,
- we show that our approach can handle different types of parametric kernels,
- we perform experiments on several diverse real datasets to demonstrate the potential of our framework.

1.5 Thesis Layout

The rest of this thesis is structured as follows. In Chapter 2, we introduce a novel approach to circumvent the initialization problem of k-means and, also, provide a kernel space extension of our approach. In Chapter 3 and Chapter 4, we consider the multi-view clustering problem from different perspectives and develop methods that assign weights to the views, so that degenerate views are automatically identified and appropriately handled. Chapter 5 presents a multiple kernel learning method, where the kernel is learned along with the cluster assignments using a ratio-based objective. Finally, Chapter 6 summarizes this thesis and overviews directions for future work.

Chapter 2

The MinMax k-means Clustering Algorithm

Clustering, is a fundamental problem in data analysis that arises in a variety of fields, such as pattern recognition, machine learning, bioinformatics and image processing [41, 123]. It aims at partitioning a set of instances into homogeneous groups, i.e. the intra-cluster similarities are high while the inter-cluster similarities are low, according to some clustering objective. However, exhaustively searching for the optimal assignment of instances to clusters is computationally infeasible and usually a good local optimum of the clustering objective is sought.

One of the most well-studied clustering algorithms is k-means [76], which minimizes the sum of the intra-cluster variances. Its simplicity and efficiency have established it as a popular means for performing clustering across different disciplines. Even an extension to kernel space has been developed [37, 90], namely kernel k-means, to enable the identification of nonlinearly separable groups. Despite its wide acceptance, k-means suffers from a serious limitation. Its solution heavily depends on the initial positions of the cluster centers, thus after a bad initialization it easily gets trapped in poor local minima [21, 86]. To alleviate this shortcoming, k-means with multiple random restarts is often employed in practice.

Several methods attempt to overcome the sensitivity to the initialization in a more principled way. A first group of methods applies special techniques aiming at systematically avoiding partitionings of poor quality during the restarts. In [3], the initial centers are selected through a stochastic procedure such that the entire data space is covered. Theoretical guarantees are provided about the capability of the method to approximate the optimal clustering. Two approaches that start from random centers and penalize clusters relative to the winning frequency of their representatives are presented in [8, 114]. Discouraging clusters to which several points have already been assigned from attracting new points in the subsequent steps has a regularizing effect. Centers that were initially ill-placed and are currently underutilized can actively participate in the solution on the following steps, which obstructs outlier clusters from forming and in effect balances the sizes of the clusters. Some other, analogous, strategies can be found in [15, 97].

A second group of methods attempts to eliminate the dependence on random initial conditions, hence restarts are not anymore necessary. Global k-means [74] and its modifications [5, 6] are incremental approaches that start from a single cluster and at each step a new cluster is deterministically added to the solution according to an appropriate criterion. A kernel-based version of global k-means is also available [103, 105]. In [128] and its extension [38], spectral clustering is applied to locate the global optimum of a relaxed version of the k-means objective, by formulating the problem as a trace maximization. Although these algorithms are not susceptible to bad initializations, they are computationally more expensive.

In this chapter we present MinMax k-means, a novel approach that tackles the k-means initialization problem by altering its objective [108]. Our method starts from a randomly picked set of centers and tries to minimize the maximum intra-cluster variance instead of the sum of the intra-cluster variances. Specifically, *a weight is associated with each cluster*, such that clusters with larger variance[1] are allocated higher weights, and a weighted version of the sum of the intra-cluster variances criterion is derived. Different notions of weights have been exploited in the literature across several k-means variants. In fuzzy c-means and Gaussian mixture models [99] weights are used to compute the degree of cluster membership of the instances, while in other variants weights are assigned to features, or groups of features, such that the tasks of clustering and feature selection are simultaneously performed [54, 79]. Also, in [58], a weighting factor is added to each instance in order to detect outliers.

The per cluster weights predispose our algorithm towards primarily minimizing those clusters that currently exhibit a large variance, in essence confining the occurrence of large variance clusters in the outcome, and are *learned automatically*, along with the cluster assignments. The proposed method alternates between a minimization step, resembling the k-means procedure, and an additional maximization step,

[1] To avoid cluttering the text, we shall also refer to the intra-cluster variances, simply, as the variances of the clusters.

in which the weights are calculated using closed-form expressions. By applying this weighting mechanism, results become less affected by the initialization and *solutions of high quality can be more consistently discovered*, even after starting from a bad initial set of centers. In addition, the obtained clusters are balanced with respect to their variance.

The presented algorithm also *incorporates a parameter* p, whose value must be specified prior to execution, that adjusts the degree of its bias towards penalizing large variance clusters. When $p = 0$, *k*-means, which has a zero bias, can be deduced as a special case of our method. A practical framework extending MinMax *k*-means to automatically adapt this parameter to the dataset is also developed here, so that the hidden group structures in the data can be successfully uncovered.

Experiments are conducted on several diverse datasets, including images, handwritten digits, proteins and patient records. MinMax *k*-means is compared to *k*-means, as well as to *k*-means++ [3] and pifs *k*-means [8] that evade degenerate optima, the first by methodically picking the initial centers and the second by balancing the cluster sizes. Our empirical evaluation reveals the effectiveness of the proposed clustering scheme in restricting the emergence of large variance clusters and producing superior solutions compared to the other three approaches, while restarted from random initializations. Furthermore, we observe that our algorithm constitutes a very promising technique for initializing *k*-means.

Finally, we provide a kernel space extension of MinMax *k*-means, which enables the identification of nonlinearly separable clusters in the data. This kernelized version of MinMax *k*-means is compared to kernel *k*-means [37, 90] and the obtained results confirm the superiority of our approach in kernel space as well.

The rest of this chapter is organized as follows. We next briefly describe *k*-means, while in Section 2.2 the proposed MinMax *k*-means algorithm is presented and its properties are analyzed. Section 2.3 introduces our practical framework for setting the p parameter. The experiments with regard to MinMax *k*-means follow in Section 2.4, before its extension to kernel space is presented in Section 2.5. Section 2.6 concludes the chapter.

2.1 *k*-means

To partition a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x}_i \in \Re^d$ into M disjoint clusters, $\{\mathcal{C}_k\}_{k=1}^M$, *k*-means [76] minimizes the sum of the intra-cluster variances (2.1), where $\mathcal{V}_k = \sum_{i=1}^N \delta_{ik}\|\mathbf{x}_i - \mathbf{m}_k\|^2$ and $\mathbf{m}_k = \sum_{i=1}^N \delta_{ik}\mathbf{x}_i / \sum_{i=1}^N \delta_{ik}$ are the variance and the center of the k-th cluster, respectively, and δ_{ik} is a cluster indicator variable with $\delta_{ik} = 1$ if $\mathbf{x}_i \in \mathcal{C}_k$ and $\delta_{ik} = 0$ otherwise.

$$\mathcal{E}_{sum} = \sum_{k=1}^{M} \mathcal{V}_k = \sum_{k=1}^{M}\sum_{i=1}^{N} \delta_{ik}\|\mathbf{x}_i - \mathbf{m}_k\|^2 \tag{2.1}$$

Clustering proceeds by alternating between assigning instances to their closest center and recomputing the centers, until a local minimum is (monotonically) reached.

Despite its simplicity and speed, *k*-means has some drawbacks, with the most prominent being the dependence of the solution on the choice of initial centers [21, 86]. Bad initializations can lead to poor local minima, thus multiple random restarts are usually executed to circumvent the initialization problem. Often, the solutions returned by the restarts significantly vary in terms of the achieved objective value, ranging from good to very bad ones, particularly for problems with a large search space (e.g. many clusters and dimensions). Therefore, numerous runs of the algorithm are required to increase the possibility of locating a good local minimum.

2.2 MinMax *k*-means

As discussed in Section 2.1, the sensitivity of *k*-means to initialization and the diverse solutions uncovered during the restarts make it difficult to find a good partitioning of the data. Motivated by this, we propose the optimization of a different objective and a new methodology that allows *k*-means to *produce high quality partitionings more systematically*, while restarted from random initial centers.

2.2.1 The Maximum Variance Objective

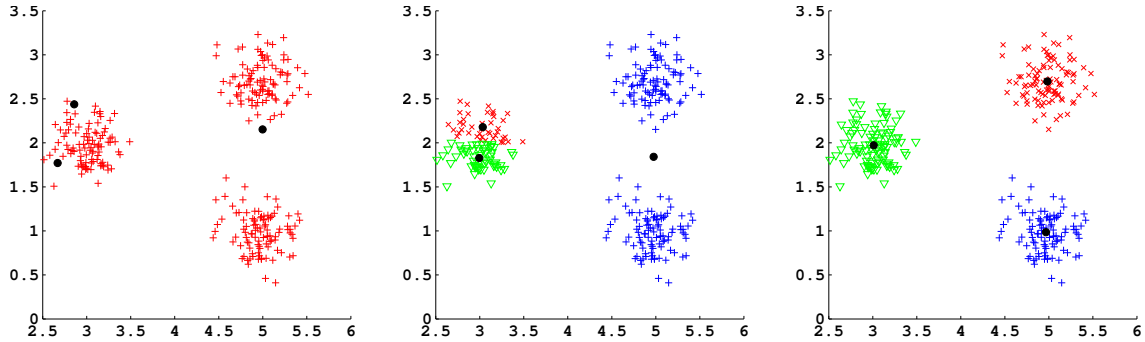
Consider a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, $\mathbf{x}_i \in \Re^d$ to be split into M disjoint clusters, $\{\mathcal{C}_k\}_{k=1}^{M}$. Instead of minimizing the *sum* of the intra-cluster variances (2.1), we propose to minimize the *maximum* intra-cluster variance:

$$\mathcal{E}_{max} = \max_{1 \le k \le M} \mathcal{V}_k = \max_{1 \le k \le M} \left\{ \sum_{i=1}^{N} \delta_{ik} \|\mathbf{x}_i - \mathbf{m}_k\|^2 \right\}, \tag{2.2}$$

where $\mathcal{V}_k$, $\mathbf{m}_k$ and δ_{ik} are defined as in (2.1).

The rationale for this approach is the following: the summation over all clusters in the *k*-means objective (2.1) allows for similar $\mathcal{E}_{sum}$ values to be achieved either by having a few clusters with large variance that are counterbalanced by others with small variance, or by having a moderate variance for all clusters. This means that the relative differences among the cluster variances are not taken into account. Note that the variance of a cluster is a measure of its quality. The above remark does not hold when minimizing $\mathcal{E}_{max}$ though, as the first case above would lead to a higher objective value. Hence, when minimizing $\mathcal{E}_{max}$, large variance clusters are avoided and *the solution space is now restricted towards clusters that exhibit more similar variances*.

The previous observation has two important implications. Since *k*-means minimizes $\mathcal{E}_{sum}$, it cannot distinguish between the two cases, thus a bad initialization yields a poor solution that is characterized by substantially different variances among the returned clusters; a result of natural groups getting merged (large variance clusters) and others



(a) (b) (c)

Figure 2.1: Example (a) of a bad initialization that (b) leads to a poor k-means solution, consisting of clusters with significantly different variances. On the contrary, our method, which is based on the notion of the maximum intra-cluster variance, manages to correctly locate the clusters (c), starting from the same initial centers. Different symbols and colors represent the cluster assignments and centers.

getting split (small variance clusters), or of outlier clusters being formed[2]. As explained, the maximum intra-cluster variance objective $\mathcal{E}_{max}$ is less likely to converge to such solutions, hence it is *easier to overcome a bad initialization*. Thus, we expect a k-means type algorithm coupled with this objective to be able to uncover better group structures more consistently during the restarts. An example is illustrated in Figure 2.1.

Additionally, a balancing effect on the clusters occurs. Balanced outcomes have been pursued in different ways in the literature. For example, in [8] k-means and spherical k-means are modified to penalize clusters in proportion to the number of instances assigned to them, while in [39, 83] a graph cut criterion is optimized which favors the creation of subgraphs where the sums of the edge weights within the subgraphs (subgraph associations) are similar. In our case, balancing is done with regard to the variance of the clusters and not the number of cluster instances. As many real life applications demand partitionings of comparable size for subsequent data analysis [8], this is a nice and desired property of the presented objective (2.2). Note that a known shortcoming of k-means is its tendency to produce skewed solutions, i.e. clusters with widely varying number of instances and/or near-empty clusters, especially for data with many dimensions and clusters, since the solution space vastly expands in this case [8, 99].

2.2.2 A Relaxed Maximum Variance Objective

Despite the aforementioned advantages, directly minimizing the maximum intra-cluster variance $\mathcal{E}_{max}$ poses a non-trivial optimization problem. To tackle this problem, the ob-

[2]Let us clarify that a solution with quite different variances on the clusters is not necessarily a bad one. There are datasets where the natural groups exhibit such structure. We simply claim that such behavior also arises after a bad initialization, where some groups are merged and others are split.

jective is relaxed so it can be readily optimized in a *k*-means iterative fashion. Specifically, we construct *a weighted formulation* $\mathcal{E}_w$ of the sum of the intra-cluster variances (2.3), where greater emphasis, i.e. a higher weight w_k, is placed on clusters with large variance, to mimic the behavior of the maximum variance criterion (2.2).

$$\mathcal{E}_w = \sum_{k=1}^{M} w_k^p \mathcal{V}_k = \sum_{k=1}^{M} w_k^p \sum_{i=1}^{N} \delta_{ik} \|\mathbf{x}_i - \mathbf{m}_k\|^2, \; w_k \geq 0, \; \sum_{k=1}^{M} w_k = 1, \; 0 \leq p < 1 \tag{2.3}$$

In contrast to $\mathcal{E}_{max}$, now, all clusters contribute to the objective, albeit to different degrees regulated by the w_k values (the w_k^p values, to be precise). Obviously, the more a cluster contributes (higher weight), the more intensely its variance will be minimized, as in this way a bigger reduction of the objective is possible. Note that the weights are not constants, but parameters that must be optimized together with the cluster labels. We treat weights as parameters to allow their values to accurately reflect the variance of their respective clusters at each iteration during training and constrain them to sum to unity to avoid overfitting and get a meaningful optimization problem. The p exponent is a user specified constant that takes values in the range $[0, 1)$ and controls the sensitivity of the weight updates to the relative differences of the cluster variances, i.e. how strongly these differences are echoed by the weights. We shall shortly provide more insight into the $\mathcal{E}_w$ objective and thoroughly explain the role of p.

The general goal of clustering is to produce a partitioning with low intra-cluster variances (compact clusters) and at the same time we wish to rigorously guard against solutions in which large variance clusters occur, analogously to $\mathcal{E}_{max}$. In order for the relaxed objective to penalize large clusters, a higher variance should lead to a higher weight, which can be realized by maximizing $\mathcal{E}_w$ with respect to the weights. The resulting optimization problem is a min-max problem of the form:

$$\min_{\{\mathcal{C}_k\}_{k=1}^{M}} \max_{\{w_k\}_{k=1}^{M}} \mathcal{E}_w, \; s.t. \; w_k \geq 0, \; \sum_{k=1}^{M} w_k = 1, \; 0 \leq p < 1. \tag{2.4}$$

We propose an iterative algorithm, called MinMax *k*-means, that alternates between the $\mathcal{C}_k$ and w_k optimization steps to get a local optimum of $\mathcal{E}_w$ and the corresponding derivations are presented next. Note that p is not part of the optimization and must be fixed a priori.

Minimization Step

By keeping the weights fixed, new cluster assignments and representatives $\mathbf{m}_k$ are calculated. For the assignments, because the terms of $\mathcal{E}_w$ involving the cluster indicator variables δ_{ik} for the i-th instance are independent of the other instances, the optimization is straightforward, giving:

$$\delta_{ik} = \begin{cases} 1, & k = \operatorname{argmin}_{1 \leq k' \leq M} w_{k'}^p \|\mathbf{x}_i - \mathbf{m}_{k'}\|^2 \\ 0, & \text{otherwise} \end{cases} . \tag{2.5}$$

Hence, each instance is assigned to the cluster whose weighted distance from the representative to the instance is the smallest. Moreover, it is evident that as the weight w_k increases, only instances that are very close to the representative $\mathbf{m}_k$ are assigned to the k-th cluster.

To estimate the representatives, the derivatives of the objective function with respect to $\mathbf{m}_k$ are set to zero, which yields:

$$\mathbf{m}_k = \frac{\sum_{i=1}^{N} \delta_{ik} \mathbf{x}_i}{\sum_{i=1}^{N} \delta_{ik}} \,. \tag{2.6}$$

As for *k*-means, the representatives coincide with the centroids of the clusters and are independent of the weights.

Maximization Step

To update the weights for given cluster assignments and centers, the weight constraints (2.4) are incorporated into the objective via a Lagrange multiplier and the derivatives with respect to w_k are set to zero. It is easy to verify that the constrained objective is concave with respect to the weights when $0 \leq p < 1$, hence their optimal values that maximize $\mathcal{E}_w$ given the current partitioning can be determined. After some manipulation the following closed-form solution emerges[3]:

$$w_k = \mathcal{V}_k^{\frac{1}{1-p}} \Big/ \sum_{k'=1}^{M} \mathcal{V}_{k'}^{\frac{1}{1-p}}, \text{ where } \mathcal{V}_k = \sum_{i=1}^{N} \delta_{ik} \|\mathbf{x}_i - \mathbf{m}_k\|^2. \tag{2.7}$$

As $1/(1-p) > 0$, since $0 \leq p < 1$, it can be observed that the larger the cluster variance $\mathcal{V}_k$ the higher the weight w_k.

2.2.3 Discussion

In this section some aspects of the MinMax *k*-means algorithm and its relaxed objective (2.3) are analyzed in more detail. According to (2.7), for a given partitioning of the data the weights are set proportionally to the cluster variances. In the subsequent minimization step, the assignment of instances to clusters is made using the weighted distance from the cluster centers (2.5). Apparently, for highly weighted clusters, the weighted distance of their representatives from the instances increases. Consequently, a cluster with large variance may lose some of its current instances that are away from its center (instances on the periphery of the cluster) and its variance is expected to decrease. At the same time, low variance clusters, due to the small weights, may also acquire instances that are not close to their centers and their variance will increase. Therefore, the iterative procedure of MinMax *k*-means *punishes large variance clusters*

[3]The proof of (2.7) is very similar to that provided in Appendix B (Section B.1) for the MVKKM method, presented in Chapter 4 of the thesis.

and operates towards clusters with similar variances, resembling the maximum variance objective (2.2) whose advantages are carried over.

MinMax *k*-means requires initial values for the cluster representatives and the weights. At the start no information about the variance of the clusters is available and the weights should be uniformly initialized, i.e. $w_k = 1/M$. Similar to *k*-means, the solution depends on the initialization of the centers and multiple restarts are necessary. However, as $\mathcal{E}_w$ shares the same properties with $\mathcal{E}_{max}$, high quality solutions are anticipated on a regular basis compared to *k*-means.

Regarding the p values, the most natural choice would be to propose a method where $p = 1$. For $p = 1$ the estimation of the weights simplifies to:

$$w_k = \begin{cases} 1, & k = \text{argmax}_{1 \leq k' \leq M} \mathcal{V}_{k'} \\ 0, & \text{otherwise} \end{cases} . \tag{2.8}$$

Obviously, in each iteration only the highest variance cluster receives a nonzero weight and thus in the following minimization step all its instances will be randomly assigned (2.5) to one of the other, zero-weight, clusters, which clearly signifies a degenerate case. If $p > 1$ is selected, the relaxed objective becomes convex with respect to the weights, thus the weight updates, which take the same form as in (2.7), will minimize $\mathcal{E}_w$ instead of maximizing it as required by (2.4). Therefore, only $0 \leq p < 1$ can be permitted.

As for the effect of the p exponent ($0 \leq p < 1$), based on (2.7) it can be shown that the greater (smaller) the p value the less (more) similar the weight values become, as the relative differences of the variances among the clusters are enhanced (suppressed). This remark also holds for the w_k^p values, which are the actual coefficients used in the relaxed objective (2.3). To demonstrate the above in detail, the ratio between any two weights, $w_k/w_{k'}$, can be considered as an indicator of their similarity. The more this ratio tends to 1 the more similar the weights. Assume a fixed clustering, i.e. fixed cluster variances $\mathcal{V}_k$ and $\mathcal{V}_{k'}$. From (2.7), $\frac{w_k}{w_{k'}} = \left(\frac{\mathcal{V}_k}{\mathcal{V}_{k'}}\right)^{\frac{1}{1-p}}$ and $\frac{w_k^p}{w_{k'}^p} = \left(\frac{\mathcal{V}_k}{\mathcal{V}_{k'}}\right)^{\frac{p}{1-p}}$, $0 \leq p < 1$. As p increases, the value of the $1/(1-p)$ and $p/(1-p)$ exponents grows, thus the relative differences of the cluster variances are enhanced and both ratios deviate more from 1, i.e. the weights and coefficients w_k^p attain less similar values (the exact opposite holds when p is decreased). In other words, p adjusts how intensely the differences of the cluster variances are reflected on the weights.

Therefore, for a high p value, large variance clusters accumulate considerably higher w_k and w_k^p values compared to low variance clusters, resulting in an objective that severely penalizes clusters with high variance. Note that an extremely high p may force clusters with large variance to lose most, or even all their instances, as their enormous weights will excessively distance the instances from their centers (2.5), something not desired of course. On the other hand, for $p = 0$, all w_k^p coefficients equal 1, hence the differences of the cluster variances are ignored and actually the *k*-means criterion is recovered, which permits high variance clusters. As shown in Section 2.2.1, preventing the appearance of large variance clusters is helpful in evading poor solutions after a bad

initialization and also balances the clusters. However, this tactic may prove problematic when natural groups with different amounts of variance exist in the dataset, a common scenario in practice, as it will hinder the clustering process from unveiling the true structure of the data. We believe that intermediate p values provide a good compromise, since high variance clusters will be admitted up to a certain extent. In a nutshell, *the p exponent controls how strongly the relaxed objective of MinMax k-means restricts the occurrence of large variance clusters*, allowing its adaptation to the dataset. This is an important advantage over the maximum variance objective $\mathcal{E}_{max}$, whose strictness over large variance clusters cannot be adjusted.

2.3 Improving MinMax *k*-means

A crucial limitation of the MinMax *k*-means algorithm is the treatment of the p exponent as a predefined constant. While from the above discussion it is clear that a moderate p is preferable, this is a rough assessment that hardly provides an indication as to which exact p values suit a specific dataset. Therefore, manually selecting an appropriate p is not trivial and requires repeating the clustering for several p values. This task becomes even harder given the dependence of the solution on the initial centers for a particular p.

To circumvent this limitation, we devise a practical framework that extends MinMax *k*-means to automatically adapt the exponent to the dataset, while alternating between the minimization and maximization steps as before. Specifically, we begin with a small p (p_{init}) that after each iteration is increased by step p_{step}, until a maximum value is attained (p_{max}). After p_{max} is reached, clustering continues without changing p. The idea behind this strategy is that the clusters formed during the first steps are heavily influenced by the initialization and should be allowed to freely evolve without considering their differences in variance, thus a small p is desirable (we set $p_{init} = 0$). As clustering progresses, p is gradually increased to restrain large variance clusters that persist in the solution and result in poor outcomes, especially after a bad initialization. Note that such a progressive punishment of large variance clusters is not possible when p is fixed a priori. Moreover, since clusters with high variance must not be completely eliminated in order to correctly uncover the intrinsic structures in the data (Section 2.2.3), extremely high values for p_{max} should be avoided.

As p grows, large variance clusters are susceptible to relinquishing most of their current instances (see Section 2.2.3). If an empty or singleton cluster appears, it will receive zero weight in the maximization step as $\mathcal{V}_k = 0$. This will cause all the dataset instances to be assigned to it in the subsequent minimization step (2.5) and the clustering process will collapse. This situation indicates that p has attained a very high value for the particular dataset. Whenever an empty or singleton cluster emerges, irrespective of whether p_{max} has been reached or not, we decrease p by p_{step}, revert back to the cluster assignments corresponding to the previous p value and resume

clustering from there. Note that p is never increased again in the following iterations. This manipulation of the p exponent has the same effect as setting p_{max} to be equal to the reduced p value from the beginning (here the adjustment is done automatically though) and actually shows that the presented framework is not very sensitive to the choice of p_{max}, as p will stop increasing when necessary.

To enhance the stability of the MinMax *k*-means algorithm, a memory effect could be added to the weights:

$$w_k^{(t)} = \beta w_k^{(t-1)} + (1-\beta)\left(\mathcal{V}_k^{\frac{1}{1-p}} / \sum_{k'=1}^{M} \mathcal{V}_{k'}^{\frac{1}{1-p}}\right), \quad 0 \leq \beta \leq 1, \tag{2.9}$$

where β controls the influence of the previous iteration weights to the current update, allowing for smoother transitions of the weight values between consecutive iterations. It should be stressed that when memory is applied ($\beta > 0$), the newly derived weights no more correspond to their optimal values for the current partitioning, in contrast to the case where memory is not employed (see the Maximization Step subsection in Section 2.2.2). However, this does not negatively affect our method, as convergence to a local optimum cannot be guaranteed, irrespective of the use of memory, since at every iteration both a minimization and a maximization of the objective is performed. On the contrary, our empirical evaluation has shown that memory is beneficial in several cases and that fewer empty clusters are created.

The change of the relaxed objective's value (2.3) between two consecutive iterations serves as the termination criterion. When this change is less than a tiny value ϵ, we stop. However, as mentioned above, convergence cannot be theoretically ensured, therefore we also stop if a predefined number of iterations t_{max} is exceeded. In practice we observed that convergence was achieved in many of our experiments. The pseudocode for the complete MinMax *k*-means algorithm is shown in Algorithm 2.1.

2.4 Empirical Evaluation

The performance of MinMax *k*-means[4] is studied on several datasets and we wish to investigate if indeed its relaxed objective (2.3) limits the occurrence of large variance clusters and how effective the proposed method is in overcoming bad initializations and attaining good solutions more regularly than *k*-means.

To demonstrate the above, first, a comparison to the basic *k*-means algorithm is made. As already discussed, *k*-means does not consider the relative differences of the clusters, allowing high variance clusters to emerge. Also, its solution is greatly affected by the initial centers. Hence, this comparison will provide strong evidence on the effectiveness of MinMax *k*-means. Moreover, we also experiment with two *k*-means variants, called *k*-means++ and partially incremental frequency sensitive (pifs) *k*-means.

[4]Matlab code is available at: `http://www.cs.uoi.gr/~gtzortzi`.

Algorithm 2.1 MinMax k-means.

Input: Dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, Initial centers $\left\{\mathbf{m}_k^{(0)}\right\}_{k=1}^{M}$, Number of clusters M, Secondary parameters (see text) p_{max}, p_{step}, β, ϵ, t_{max}

Output: Cluster assignments $\{\delta_{ik}\}_{i=1,\ldots,N,k=1,\ldots,M}$, Final centers $\{\mathbf{m}_k\}_{k=1}^{M}$

Set $t = 0$
Set $p_{init} = 0$
Set $w_k^{(0)} = 1/M$, $\forall k = 1, \ldots, M$
Set $empty =$ **false** // No empty or singleton clusters yet detected.
$p = p_{init}$
repeat
  $t = t + 1$
  for $i = 1$ to N **do** // Update the cluster assignments.
    for $k = 1$ to M **do**
      $\delta_{ik}^{(t)} = \begin{cases} 1, & k = \text{argmin}_{1 \le k' \le M} \left(w_{k'}^{(t-1)}\right)^p \|\mathbf{x}_i - \mathbf{m}_{k'}^{(t-1)}\|^2 \\ 0, & \text{otherwise} \end{cases}$
    end for
  end for
  if empty or singleton clusters have occurred at time t **then** // Reduce p.
    $empty =$ **true**
    $p = p - p_{step}$
    if $p < p_{init}$ **then**
      return NULL
    end if
    // Revert to the assignments and weights corresponding to the reduced p.
    $\delta_{ik}^{(t)} = [\Delta^{(p)}]_{ik}$, $\forall k = 1, \ldots, M$, $\forall i = 1, \ldots, N$
    $w_k^{(t-1)} = [\mathbf{w}^{(p)}]_k$, $\forall k = 1, \ldots, M$
  end if
  for all $\mathbf{m}_k$, $k = 1$ to M **do** // Update the centers.
    $\mathbf{m}_k^{(t)} = \sum_{i=1}^{N} \delta_{ik}^{(t)} \mathbf{x}_i / \sum_{i=1}^{N} \delta_{ik}^{(t)}$
  end for
  if $p < p_{max}$ **and** $empty =$ **false then** // Increase p.
    $\Delta^{(p)} = [\delta_{ik}^{(t)}]$ // Store the current assignments in matrix $\Delta^{(p)}$.
    $\mathbf{w}^{(p)} = [w_k^{(t-1)}]$ // Store the previous weights in vector $\mathbf{w}^{(p)}$.
    $p = p + p_{step}$
  end if
  for all w_k, $k = 1$ to M **do** // Update the weights.
    $w_k^{(t)} = \beta w_k^{(t-1)} + (1 - \beta) \left(\left(\mathcal{V}_k^{(t)}\right)^{\frac{1}{1-p}} \Big/ \sum_{k'=1}^{M} \left(\mathcal{V}_{k'}^{(t)}\right)^{\frac{1}{1-p}} \right)$,
    where $\mathcal{V}_k^{(t)} = \sum_{i=1}^{N} \delta_{ik}^{(t)} \|\mathbf{x}_i - \mathbf{m}_k^{(t)}\|^2$
  end for
until $\left|\mathcal{E}_w^{(t)} - \mathcal{E}_w^{(t-1)}\right| < \epsilon$ **or** $t \ge t_{max}$
return $\left\{\delta_{ik}^{(t)}\right\}_{i=1,\ldots,N,k=1,\ldots,M}$, $\left\{\mathbf{m}_k^{(t)}\right\}_{k=1}^{M}$

Figure 2.2: The COIL-20 objects considered in the experiments.

Figure 2.3: Indicative images belonging to three individuals from the Olivetti collection.

In *k*-means++ [3] a stochastic procedure is employed to pick the initial cluster centers and then *k*-means is executed from these centers. Specifically, given that $k-1$ centers have already been selected, instance $\mathbf{x}_i$ may be selected as the k-th initial center with a probability that is proportional to its minimum distance from the $k-1$ centers. The above procedure aims at selecting initial centers that cover the entire data space, thus providing better initializations to *k*-means (compared to random starts), and, therefore, constituting a worthy competitor against which to measure our method.

Pifs *k*-means [8] explicitly penalizes clusters in proportion to the number of instances already assigned to them, according to the following cluster update rule[5]:

$$\delta_{ik} = \begin{cases} 1, & k = \text{argmin}_{1 \leq k' \leq M} |\mathcal{C}_{k'}| \|\mathbf{x}_i - \mathbf{m}_{k'}\|^2 \\ 0, & \text{otherwise} \end{cases}, \tag{2.10}$$

where $|\mathcal{C}_k|$ is the current size of the k-th cluster. Based on (2.10), the larger the cluster the lower the chance of an instance being acquired by that cluster. Thus, clusters are balanced in terms of their size, which has been shown to decrease the sensitivity to bad initializations [8, 114]. Remember (Section 2.2.3), that MinMax *k*-means, implicitly, through its weighting strategy, operates towards clusters with similar variances. Therefore, it is interesting to examine how these two different balancing approaches compare against each other.

2.4.1 Datasets

Six popular datasets are utilized in our empirical study for which the ground-truth is available. Their features are normalized to zero mean and unit variance, unless stated otherwise.

[5]Note that the exact cluster update rule proposed in [8] contains an additional $d \ln |\mathcal{C}_{k'}|$ term, where d is the dataset dimensionality. However, better results were obtained without using this term in our experiments.

Table 2.1: Main characteristics of the tested datasets.

Dataset	Instances	Features	Classes	Balanced
Coil1 & Coil2	216	1000	3	Yes
Coil3	360	1000	5	Yes
Multiple features - pixel averages	2000	240	10	Yes
Multiple features - profile correlations	2000	216	10	Yes
Pendigits	10992	16	10	Almost
Olivetti	900	2500	10	Yes
Ecoli	307	7	4	No
Dermatology	366	34	6	No

Coil-20 [81] contains 72 images taken from different angles for each of the 20 included objects. As in [56], SIFT descriptors [78] are first extracted from the images which are then represented by the bag of visual words model using 1000 visual words and the data vectors are normalized to unit length. For our purposes, three subsets of Coil-20 were created, Coil1 (objects 3, 9 and 10), Coil2 (objects 15, 18 and 19) and Coil3 (objects 2, 4, 7, 10 and 11). The tested objects are shown in Figure 2.2.

Multiple features & Pendigits are two collections of handwritten digits (0-9) from the UCI repository [42]. Multiple features digits (200 per class) are described in terms of six different feature sets and we select two of them, namely pixel averages and profile correlations. Pendigits consists of 10992 instances (roughly 1100 samples per numeral) in 16-dimensional space.

Olivetti is a face database of 40 individuals with ten 64×64 grayscale images per individual. Based on [43], we only retain the first 100 images, belonging to ten persons, and apply the same preprocessing. Specifically, each image is smoothed using a Gaussian kernel and then rotated by -10, 0 and 10 degrees and scaled by a factor of 0.9, 1.0 and 1.1, resulting in 900 images. Finally, a central 50×50 window of the images is kept and its pixels are normalized to zero mean and 0.1 variance. A subset of the considered images is shown in Figure 2.3.

Ecoli (UCI) [42] includes 336 proteins from the Escherichia coli bacterium and seven attributes, calculated from the amino acid sequences, are provided. Proteins belong to eight categories according to their cellular localization sites. Four of the classes are extremely underrepresented and are not considered in our evaluation. Note that classes differ in size, i.e. it is an unbalanced dataset.

Dermatology (UCI) [42] is comprised of 366 patient records that suffer from six different types of the Eryhemato-Squamous disease. Each patient is described by both clinical and histopathological features (34 in total). This dataset is also unbalanced.

A summary of the datasets is provided in Table 2.1.

2.4.2 Experimental Protocol

All tested algorithms, apart from *k*-means++, are restarted 500 times from the same randomly chosen initial centers. For *k*-means++, the stochastic initialization procedure is executed 500 times. The number of clusters is set equal to the number of classes in each dataset, throughout the experiments. To evaluate the quality of the returned solutions, the maximum cluster variance $\mathcal{E}_{max}$, defined in (2.2), and the sum of the cluster variances $\mathcal{E}_{sum}$, defined in (2.1), serve as the main performance measures and their average and standard deviation over the 500 runs is reported. Note that $\mathcal{E}_{sum}$ favors *k*-means and *k*-means++ in the comparisons, since this is the objective optimized by these two methods. Likewise, $\mathcal{E}_{max}$ favors our framework which optimizes a relaxed version of (2.2). Since the ground-truth is available, the achieved NMI score (2.11)[6] is also reported. Higher NMI values indicate a better match between the cluster labels and the class labels.

$$NMI = \frac{2\sum_{k=1}^{M}\sum_{h=1}^{C}\frac{n_k^h}{N}\log\frac{n_k^h N}{\sum_{i=1}^{M} n_i^h \sum_{i=1}^{C} n_k^i}}{H_M + H_C} \tag{2.11}$$

Moreover, to assess the computational complexity of the algorithms, their average execution time (in seconds) is reported.

In a second series of experiments, the cluster centers derived by each execution of MinMax *k*-means and pifs *k*-means are used to initialize a subsequent *k*-means run. This allows us to determine if *k*-means performance can be improved when initialized by these two approaches and also facilitates the comparison of the tested methods under a common objective ($\mathcal{E}_{sum}$).

For MinMax *k*-means, some additional parameters must be fixed prior to execution (p_{max}, p_{step}, β, ϵ and t_{max}). Our method is not particularly sensitive to either p_{max} or p_{step}. Regarding p_{max}, p stops increasing when empty or singleton clusters are detected. For p_{step}, one should simply avoid a large step which will cause abrupt changes to the p value between consecutive iterations. Thus, we do not fine-tune these two parameters for each dataset and for all the experiments we set $p_{max} = 0.5$ and $p_{step} = 0.01$. Note that empty clusters appear quite often for the selected p_{max} value, indicating that it is already set to a high value. For β, we tried three different levels of memory, $\beta \in \{0, 0.1, 0.3\}$, and present the corresponding results. Finally, $\epsilon = 10^{-6}$ and $t_{max} = 500$ for all experiments.

2.4.3 Performance Analysis

The comparison of the algorithms across the various datasets is shown in Tables 2.2-2.10, where MinMax *k*-means and pifs *k*-means are abbreviated as MinMax and pifs, respectively. Tables 2.2(b)-2.10(b), labeled as "method + *k*-means", refer to the experiments where *k*-means is initialized from the solution of the method designated by

[6] N is the dataset size, M the number of clusters, C the number of classes, n_k^h the number of points in cluster k belonging to class h, and H_M, H_C the entropy of the clusters and the classes, respectively.

the corresponding row. For example, we denote as MinMax+k-means (pifs+k-means), the method where MinMax k-means (pifs k-means) is executed first and its solution is used to initialize a subsequent run of k-means. Of course, reinitializing k-means with its own, or the k-means++ solution has no effect and the results are just replicated from Tables 2.2(a)-2.10(a) for readers' convenience. Asterisk (*), dagger (†) and double dagger (‡) superscripts denote that MinMax k-means has a statistically significant difference to k-means, k-means++ and pifs k-means, respectively, according to the t-test (the significance level is taken as 0.05). A line above (below) these symbols stands for a higher (lower) average.

From the tables two main observations can be made. First, all memory levels of MinMax k-means exhibit better (smaller) $\mathcal{E}_{max}$ than k-means, k-means++ and pifs k-means for every dataset (Tables 2.2(a)-2.10(a)), but Pendigits. This clearly displays that the relaxed objective (2.3) minimizes large variance clusters and mimics the maximum variance criterion (2.2). Note also that k-means, when initialized by our algorithm, leads to clusters with lower $\mathcal{E}_{max}$ for most datasets (Tables 2.2(b)-2.10(b)). However, k-means optimizes the sum of the variances and does not consider the maximum variance. Hence, it is reasonable in this case that $\mathcal{E}_{max}$ increases compared to that before employing k-means and that pifs+k-means produces equal or better $\mathcal{E}_{max}$ scores than MinMax+k-means for half of the datasets.

Second, our method outperforms k-means for all the metrics (apart from execution time) reported in Tables 2.2-2.10, demonstrating its ability to attain good partitionings on a more frequent basis. To add to the above, MinMax+k-means obtains lower $\mathcal{E}_{sum}$ and higher NMI values than k-means, i.e. k-means converges to better local minima when initialized by MinMax k-means. Actually, the main difference between k-means and MinMax+k-means is that some restarts of the former return solutions with excessively high $\mathcal{E}_{sum}$ (its higher standard deviation is indicative of that), while for the latter such poor outcomes do not emerge, illustrating the robustness of MinMax k-means over bad initializations.

Considering k-means++, its stochastic initialization process improves performance, as lower $\mathcal{E}_{max}$ and $\mathcal{E}_{sum}$ (and equal NMI) values are acquired on most cases compared to the randomly restarted k-means. When put up against MinMax k-means though, similar conclusions to those mentioned above for k-means can be drawn, further establishing the potential of the presented framework. It is of particular interest that MinMax+k-means yields better $\mathcal{E}_{sum}$ and NMI scores on every dataset, despite k-means++ carefully picking the initial centers. This definitely reveals that the centers outputted by MinMax k-means consist good initializations for k-means.

The proposed algorithm is also superior to pifs k-means. Specifically, it always reaches a lower $\mathcal{E}_{max}$ (the exception being Multiple features-profile correlations for $\beta = 0.1$ and Pendigits for $\beta = 0.1$ and $\beta = 0.3$), while for $\mathcal{E}_{sum}$ it gets ahead on four of the nine datasets and it is only beaten two times. For the remaining three (Coil3, Multiple features-profile correlations and Pendigits), there is at least one memory level for which

Table 2.2(a): Comparative results on the Coil1 dataset.

Method	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	NMI	Time
MinMax ($\beta = 0$)	$46.61 \pm 0.00^{*\dagger\ddagger}$	$119.02 \pm 0.00^{*\dagger\ddagger}$	$0.88 \pm 0.00^{\dagger\ddagger}$	0.54 (0.54)
MinMax ($\beta = 0.1$)	$45.75 \pm 0.00^{*\dagger\ddagger}$	$119.24 \pm 0.00^{*\dagger\ddagger}$	$0.87 \pm 0.00^{*\dagger\ddagger}$	0.42 (0.42)
MinMax ($\beta = 0.3$)	$\mathbf{45.04 \pm 0.00}^{*\dagger\ddagger}$	$119.40 \pm 0.00^{*\dagger\ddagger}$	$0.87 \pm 0.00^{*\dagger\ddagger}$	0.42 (0.42)
***k*-means**	66.33 ± 19.46	121.24 ± 7.12	0.89 ± 0.16	**0.07**
***k*-means++**	64.92 ± 18.83	121.01 ± 7.18	0.90 ± 0.16	0.09
Pifs	53.43 ± 0.00	$\mathbf{117.82 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	**0.07**

Table 2.2(b): Comparative results on the Coil1 dataset when *k*-means is initialized by the solution returned by MinMax *k*-means and pifs *k*-means.

Method + *k*-means	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	NMI	Time
MinMax ($\beta = 0$)	$\mathbf{53.43 \pm 0.00}^{*\dagger}$	$\mathbf{117.82 \pm 0.00}^{*\dagger}$	$\mathbf{1.00 \pm 0.00}^{*\dagger}$	0.58 (0.58)
MinMax ($\beta = 0.1$)	$\mathbf{53.43 \pm 0.00}^{*\dagger}$	$\mathbf{117.82 \pm 0.00}^{*\dagger}$	$\mathbf{1.00 \pm 0.00}^{*\dagger}$	0.45 (0.45)
MinMax ($\beta = 0.3$)	$\mathbf{53.43 \pm 0.00}^{*\dagger}$	$\mathbf{117.82 \pm 0.00}^{*\dagger}$	$\mathbf{1.00 \pm 0.00}^{*\dagger}$	0.46 (0.46)
***k*-means**	66.33 ± 19.46	121.24 ± 7.12	0.89 ± 0.16	**0.07**
***k*-means++**	64.92 ± 18.83	121.01 ± 7.18	0.90 ± 0.16	0.09
Pifs	$\mathbf{53.43 \pm 0.00}$	$\mathbf{117.82 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	0.09

Table 2.3(a): Comparative results on the Coil2 dataset.

Method	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	NMI	Time
MinMax ($\beta = 0$)	$58.74 \pm 0.36^{*\dagger\ddagger}$	$\mathbf{154.49 \pm 1.04}^{*\dagger\ddagger}$	$\mathbf{0.95 \pm 0.14}^{*\dagger\ddagger}$	1.94 (0.47)
MinMax ($\beta = 0.1$)	$\mathbf{57.14 \pm 0.35}^{*\dagger\ddagger}$	$155.09 \pm 0.85^{*\ddagger}$	$0.91 \pm 0.13^{*\dagger\ddagger}$	0.45 (0.44)
MinMax ($\beta = 0.3$)	$58.73 \pm 0.42^{*\dagger\ddagger}$	$154.56 \pm 1.09^{*\dagger\ddagger}$	$0.94 \pm 0.14^{*\dagger\ddagger}$	0.52 (0.52)
***k*-means**	77.46 ± 18.74	155.49 ± 2.26	0.80 ± 0.16	**0.09**
***k*-means++**	74.33 ± 16.87	155.18 ± 1.85	0.82 ± 0.16	0.10
Pifs	59.48 ± 1.11	155.96 ± 1.61	0.75 ± 0.19	0.11

Table 2.3(b): Comparative results on the Coil2 dataset when *k*-means is initialized by the solution returned by MinMax *k*-means and pifs *k*-means.

Method + *k*-means	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	NMI	Time
MinMax ($\beta = 0$)	$\mathbf{58.95 \pm 0.97}^{*\dagger\ddagger}$	$\mathbf{154.38 \pm 0.93}^{*\dagger\ddagger}$	$\mathbf{0.95 \pm 0.13}^{*\dagger\ddagger}$	1.97 (0.50)
MinMax ($\beta = 0.1$)	$59.03 \pm 1.70^{*\dagger\ddagger}$	$154.39 \pm 0.94^{*\dagger\ddagger}$	$\mathbf{0.95 \pm 0.13}^{*\dagger\ddagger}$	0.47 (0.46)
MinMax ($\beta = 0.3$)	$59.11 \pm 1.98^{*\dagger\ddagger}$	$154.42 \pm 0.96^{*\dagger\ddagger}$	$0.94 \pm 0.14^{*\dagger\ddagger}$	0.55 (0.55)
***k*-means**	77.46 ± 18.74	155.49 ± 2.26	0.80 ± 0.16	**0.09**
***k*-means++**	74.33 ± 16.87	155.18 ± 1.85	0.82 ± 0.16	0.10
Pifs	62.84 ± 7.06	155.58 ± 1.50	0.77 ± 0.19	0.14

Table 2.4(a): Comparative results on the Coil3 dataset.

Method	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	NMI	Time
MinMax ($\beta = 0$)	$58.00 \pm 0.27^{*\dagger\ddagger}$	$245.95 \pm 0.71^{*\dagger\ddagger}$	$\mathbf{0.99 \pm 0.03}^{*\dagger\ddagger}$	3.29 (0.74)
MinMax ($\beta = 0.1$)	$57.90 \pm 0.25^{*\dagger\ddagger}$	$\mathbf{245.64 \pm 0.75}^{*\dagger\ddagger}$	$\mathbf{0.99 \pm 0.03}^{*\dagger\ddagger}$	5.46 (0.81)
MinMax ($\beta = 0.3$)	$\mathbf{53.24 \pm 0.40}^{*\dagger\ddagger}$	$249.82 \pm 0.24^{\dagger\ddagger}$	$0.94 \pm 0.01^{*\dagger}$	3.36 (0.82)
***k*-means**	101.95 ± 29.81	249.64 ± 5.64	0.88 ± 0.08	**0.15**
***k*-means++**	96.35 ± 28.37	249.13 ± 5.45	0.89 ± 0.07	0.18
Pifs	58.39 ± 1.07	246.47 ± 2.52	0.95 ± 0.08	0.20

Table 2.4(b): Comparative results on the Coil3 dataset when *k*-means is initialized by the solution returned by MinMax *k*-means and pifs *k*-means.

Method + *k*-means	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	NMI	Time
MinMax ($\beta = 0$)	$58.30 \pm 2.85^{*\dagger\ddagger}$	$245.41 \pm 0.41^{*\dagger\ddagger}$	$\mathbf{0.99 \pm 0.02}^{*\dagger\ddagger}$	3.33 (0.82)
MinMax ($\beta = 0.1$)	$58.26 \pm 2.72^{*\dagger\ddagger}$	$245.41 \pm 0.41^{*\dagger\ddagger}$	$\mathbf{0.99 \pm 0.02}^{*\dagger\ddagger}$	5.51 (0.90)
MinMax ($\beta = 0.3$)	$\mathbf{58.03 \pm 1.77}^{*\dagger\ddagger}$	$\mathbf{245.40 \pm 0.23}^{*\dagger\ddagger}$	$\mathbf{0.99 \pm 0.01}^{*\dagger\ddagger}$	3.40 (0.89)
***k*-means**	101.95 ± 29.81	249.64 ± 5.64	0.88 ± 0.08	**0.15**
***k*-means++**	96.35 ± 28.37	249.13 ± 5.45	0.89 ± 0.07	0.18
Pifs	64.12 ± 9.50	245.68 ± 2.01	0.96 ± 0.06	0.25

Table 2.5(a): Comparative results on the Multiple features (pixel averages) dataset.

Method	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	NMI	Time
MinMax ($\beta = 0$)	$149.60 \pm 9.56^{*\dagger\ddagger}$	$1239.33 \pm 6.19^{\dagger\ddagger}$	$0.68 \pm 0.03^{*\dagger\ddagger}$	2.59 (2.00)
MinMax ($\beta = 0.1$)	$146.73 \pm 14.70^{*\dagger\ddagger}$	$1240.49 \pm 8.61^{*\dagger\ddagger}$	$0.68 \pm 0.03^{*\dagger\ddagger}$	2.36 (1.98)
MinMax ($\beta = 0.3$)	$\mathbf{145.00 \pm 17.17}^{*\dagger\ddagger}$	$1243.09 \pm 13.05^{*\dagger\ddagger}$	$0.68 \pm 0.04^{*\dagger\ddagger}$	2.22 (1.50)
***k*-means**	222.50 ± 33.95	1238.36 ± 12.51	0.71 ± 0.04	**0.66**
***k*-means++**	219.63 ± 36.34	$\mathbf{1237.24 \pm 11.18}$	0.71 ± 0.04	0.80
Pifs	150.75 ± 4.47	1237.84 ± 4.31	$\mathbf{0.72 \pm 0.05}$	1.03

Table 2.5(b): Comparative results on the Multiple features (pixel averages) dataset when *k*-means is initialized by the solution returned by MinMax *k*-means and pifs *k*-means.

Method + *k*-means	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	NMI	Time
MinMax ($\beta = 0$)	$202.03 \pm 23.73^{*\dagger\ddagger}$	$1230.64 \pm 5.56^{*\dagger\ddagger}$	$0.72 \pm 0.03^{*\dagger\ddagger}$	2.87 (2.28)
MinMax ($\beta = 0.1$)	$200.20 \pm 23.89^{*\dagger\ddagger}$	$1230.52 \pm 5.38^{*\dagger\ddagger}$	$0.72 \pm 0.03^{*\dagger\ddagger}$	2.66 (2.28)
MinMax ($\beta = 0.3$)	$198.91 \pm 24.51^{*\dagger\ddagger}$	$\mathbf{1229.77 \pm 4.27}^{*\dagger\ddagger}$	$0.72 \pm 0.03^{*\dagger\ddagger}$	2.55 (1.83)
***k*-means**	222.50 ± 33.95	1238.36 ± 12.51	0.71 ± 0.04	**0.66**
***k*-means++**	219.63 ± 36.34	1237.24 ± 11.18	0.71 ± 0.04	0.80
Pifs	$\mathbf{177.06 \pm 21.25}$	1232.07 ± 3.53	$\mathbf{0.74 \pm 0.04}$	1.30

Table 2.6(a): Comparative results on the Multiple features (profile correlations) dataset.

Method	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	NMI	Time
MinMax ($\beta = 0$)	$\mathbf{118.60 \pm 7.63}^{*\dagger\ddagger}$	$\mathbf{966.96 \pm 8.43}^{*\dagger\ddagger}$	$0.69 \pm 0.04^{\ddagger}$	2.84 (1.98)
MinMax ($\beta = 0.1$)	$150.97 \pm 52.71^{*\dagger\ddagger}$	$1004.81 \pm 52.86^{*\dagger\ddagger}$	$0.67 \pm 0.04^{*\dagger\ddagger}$	3.93 (1.82)
MinMax ($\beta = 0.3$)	$120.21 \pm 15.16^{*\dagger\ddagger}$	$972.86 \pm 13.50^{*\dagger\ddagger}$	$0.69 \pm 0.04^{\ddagger}$	2.13 (1.03)
***k*-means**	179.22 ± 41.17	970.18 ± 15.90	0.69 ± 0.04	**0.49**
***k*-means++**	175.74 ± 37.88	968.81 ± 15.43	0.69 ± 0.03	0.63
Pifs	133.29 ± 10.57	974.54 ± 5.63	$\mathbf{0.71 \pm 0.04}$	1.00

Table 2.6(b): Comparative results on the Multiple features (profile correllations) dataset when *k*-means is initialized by the solution returned by MinMax *k*-means and pifs *k*-means.

Method + *k*-means	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	NMI	Time
MinMax ($\beta = 0$)	$155.36 \pm 16.13^{*\dagger\ddagger}$	$958.28 \pm 6.97^{*\dagger\ddagger}$	$0.70 \pm 0.03^{*\dagger\ddagger}$	3.05 (2.19)
MinMax ($\beta = 0.1$)	$154.59 \pm 13.08^{*\dagger\ddagger}$	$957.78 \pm 6.54^{*\dagger\ddagger}$	$0.70 \pm 0.03^{*\dagger\ddagger}$	4.17 (2.04)
MinMax ($\beta = 0.3$)	$\mathbf{153.97 \pm 12.22}^{*\dagger\ddagger}$	$\mathbf{957.63 \pm 6.28}^{*\dagger\ddagger}$	$0.70 \pm 0.03^{*\dagger\ddagger}$	2.37 (1.26)
***k*-means**	179.22 ± 41.17	970.18 ± 15.90	0.69 ± 0.04	**0.49**
***k*-means++**	175.74 ± 37.88	968.81 ± 15.43	0.69 ± 0.03	0.63
Pifs	160.16 ± 11.63	962.93 ± 3.56	$\mathbf{0.72 \pm 0.04}$	1.26

Table 2.7(a): Comparative results on the Pendigits dataset.

Method	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	NMI	Time
MinMax ($\beta = 0$)	$\mathbf{7769.50 \pm 1249.80}^{*\dagger\ddagger}$	$61140.86 \pm 659.81^{\dagger\ddagger}$	$0.68 \pm 0.01^{*\dagger\ddagger}$	2.72 (2.23)
MinMax ($\beta = 0.1$)	$17497.21 \pm 5431.65^{*\dagger\ddagger}$	$71599.61 \pm 5066.73^{*\dagger\ddagger}$	$0.64 \pm 0.03^{*\dagger\ddagger}$	4.79 (1.47)
MinMax ($\beta = 0.3$)	$8849.21 \pm 1706.73^{*\dagger\ddagger}$	$62345.44 \pm 1266.36^{*\dagger\ddagger}$	$0.69 \pm 0.01^{\ddagger}$	2.27 (0.91)
***k*-means**	11576.43 ± 3125.47	61024.17 ± 1333.92	0.69 ± 0.02	**0.55**
***k*-means++**	11857.89 ± 3039.04	$\mathbf{60940.96 \pm 1294.01}$	0.69 ± 0.02	0.56
Pifs	8623.37 ± 329.35	61895.12 ± 643.98	$\mathbf{0.70 \pm 0.01}$	3.06

Table 2.7(b): Comparative results on the Pendigits dataset when *k*-means is initialized by the solution returned by MinMax *k*-means and pifs *k*-means.

Method + *k*-means	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	NMI	Time
MinMax ($\beta = 0$)	$9403.21 \pm 2760.33^{*\dagger}$	$60681.71 \pm 710.50^{*\dagger}$	$0.69 \pm 0.01^{\ddagger}$	2.88 (2.39)
MinMax ($\beta = 0.1$)	$9835.21 \pm 2444.54^{*\dagger\ddagger}$	$60447.71 \pm 751.37^{*\dagger\ddagger}$	$0.70 \pm 0.01^{*\dagger\ddagger}$	4.99 (1.60)
MinMax ($\beta = 0.3$)	$\mathbf{9258.11 \pm 2590.49}^{*\dagger}$	$\mathbf{60366.92 \pm 731.99}^{*\dagger\ddagger}$	$0.69 \pm 0.01^{\ddagger}$	2.50 (1.07)
***k*-means**	11576.43 ± 3125.47	61024.17 ± 1333.92	0.69 ± 0.02	**0.55**
***k*-means++**	11857.89 ± 3039.04	60940.96 ± 1294.01	0.69 ± 0.02	0.56
Pifs	9289.79 ± 672.91	60722.65 ± 684.59	$\mathbf{0.71 \pm 0.00}$	3.25

Table 2.8(a): Comparative results on the Olivetti dataset.

Method	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	**NMI**	**Time**
MinMax ($\beta = 0$)	$1217.72 \pm 55.18^{*\dagger\ddagger}$	$\mathbf{11016.58 \pm 44.35}^{*\dagger\ddagger}$	$\mathbf{0.34 \pm 0.04}$	7.80 (7.43)
MinMax ($\beta = 0.1$)	$1207.91 \pm 86.61^{*\dagger\ddagger}$	$11019.11 \pm 83.40^{*\dagger}$	$\mathbf{0.34 \pm 0.04}$	7.26 (7.10)
MinMax ($\beta = 0.3$)	$\mathbf{1198.19 \pm 92.13}^{*\dagger\ddagger}$	$11019.25 \pm 69.03^{*\dagger}$	$\mathbf{0.34 \pm 0.04}$	6.50 (6.22)
***k*-means**	1610.49 ± 152.77	11034.37 ± 61.38	$\mathbf{0.34 \pm 0.03}$	**2.40**
***k*-means++**	1624.46 ± 158.38	11031.70 ± 64.07	$\mathbf{0.34 \pm 0.03}$	2.82
Pifs	1305.87 ± 36.61	11024.36 ± 45.72	$\mathbf{0.34 \pm 0.03}$	2.97

Table 2.8(b): Comparative results on the Olivetti dataset when *k*-means is initialized by the solution returned by MinMax *k*-means and pifs *k*-means.

Method + *k*-means	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	**NMI**	**Time**
MinMax ($\beta = 0$)	$1383.35 \pm 120.45^{*\dagger\ddagger}$	$10985.52 \pm 41.70^{*\dagger\ddagger}$	$\mathbf{0.34 \pm 0.04}$	8.61 (8.24)
MinMax ($\beta = 0.1$)	$1374.73 \pm 117.89^{*\dagger}$	$10984.49 \pm 41.86^{*\dagger\ddagger}$	$\mathbf{0.34 \pm 0.04}$	8.04 (7.88)
MinMax ($\beta = 0.3$)	$1367.46 \pm 116.57^{*\dagger}$	$\mathbf{10980.86 \pm 42.48}^{*\dagger\ddagger}$	$\mathbf{0.34 \pm 0.04}$	7.33 (7.05)
***k*-means**	1610.49 ± 152.77	11034.37 ± 61.38	$\mathbf{0.34 \pm 0.03}$	**2.40**
***k*-means++**	1624.46 ± 158.38	11031.70 ± 64.07	$\mathbf{0.34 \pm 0.03}$	2.82
Pifs	$\mathbf{1362.69 \pm 101.90}$	10993.37 ± 40.90	$\mathbf{0.34 \pm 0.03}$	3.91

Table 2.9(a): Comparative results on the Ecoli dataset.

Method	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	**NMI**	**Time**
MinMax ($\beta = 0$)	$5.29 \pm 0.15^{*\dagger}$	$15.94 \pm 0.24^{*\dagger\ddagger}$	$0.58 \pm 0.01^{*\dagger\ddagger}$	0.18 (0.06)
MinMax ($\beta = 0.1$)	$5.02 \pm 0.25^{*\dagger\ddagger}$	$15.72 \pm 0.04^{\ddagger}$	$0.57 \pm 0.01^{*\dagger\ddagger}$	0.11 (0.05)
MinMax ($\beta = 0.3$)	$\mathbf{4.80 \pm 0.00}^{*\dagger\ddagger}$	$15.73 \pm 0.00^{\ddagger}$	$0.58 \pm 0.00^{*\dagger\ddagger}$	0.05 (0.05)
***k*-means**	6.38 ± 0.88	$\mathbf{15.68 \pm 0.54}$	$\mathbf{0.61 \pm 0.02}$	0.02
***k*-means++**	6.60 ± 1.58	15.79 ± 1.02	$\mathbf{0.61 \pm 0.03}$	**0.01**
Pifs	5.30 ± 0.28	16.19 ± 0.15	0.55 ± 0.01	0.04

Table 2.9(b): Comparative results on the Ecoli dataset when *k*-means is initialized by the solution returned by MinMax *k*-means and pifs *k*-means.

Method + *k*-means	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	**NMI**	**Time**
MinMax ($\beta = 0$)	$6.29 \pm 0.11^{*\dagger\ddagger}$	$15.40 \pm 0.03^{*\dagger\ddagger}$	$\mathbf{0.63 \pm 0.00}^{*\dagger\ddagger}$	0.19 (0.06)
MinMax ($\beta = 0.1$)	$6.29 \pm 0.00^{*\dagger\ddagger}$	$\mathbf{15.39 \pm 0.00}^{*\dagger\ddagger}$	$\mathbf{0.63 \pm 0.00}^{*\dagger\ddagger}$	0.12 (0.06)
MinMax ($\beta = 0.3$)	$6.29 \pm 0.00^{*\dagger\ddagger}$	$\mathbf{15.39 \pm 0.00}^{*\dagger\ddagger}$	$\mathbf{0.63 \pm 0.00}^{*\dagger\ddagger}$	0.05 (0.05)
***k*-means**	6.38 ± 0.88	15.68 ± 0.54	0.61 ± 0.02	0.02
***k*-means++**	6.60 ± 1.58	15.79 ± 1.02	0.61 ± 0.03	**0.01**
Pifs	$\mathbf{6.04 \pm 0.35}$	15.65 ± 0.19	0.61 ± 0.02	0.05

Table 2.10(a): Comparative results on the Dermatology dataset.

Method	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	NMI	Time
MinMax ($\beta = 0$)	$1513.85 \pm 316.42^{*\dagger\ddagger}$	$\mathbf{5672.82 \pm 272.21}^{*\dagger\ddagger}$	$\mathbf{0.82 \pm 0.03}^{\ddagger}$	0.37 (0.18)
MinMax ($\beta = 0.1$)	$1439.76 \pm 296.76^{*\dagger\ddagger}$	$5685.16 \pm 237.35^{*\dagger\ddagger}$	$\mathbf{0.82 \pm 0.03}^{\ddagger}$	0.37 (0.16)
MinMax ($\beta = 0.3$)	$\mathbf{1368.05 \pm 347.04}^{*\dagger\ddagger}$	$5703.26 \pm 195.87^{*\dagger\ddagger}$	$\mathbf{0.82 \pm 0.01}^{\ddagger}$	0.49 (0.16)
***k*-means**	2247.59 ± 804.75	5885.92 ± 542.49	$\mathbf{0.82 \pm 0.07}$	0.10
***k*-means++**	2134.54 ± 681.34	5800.23 ± 448.38	$\mathbf{0.82 \pm 0.07}$	0.11
Pifs	1650.13 ± 91.99	6057.18 ± 50.62	0.80 ± 0.01	**0.08**

Table 2.10(b): Comparative results on the Dermatology dataset when *k*-means is initialized by the solution returned by MinMax *k*-means and pifs *k*-means.

Method + *k*-means	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	NMI	Time
MinMax ($\beta = 0$)	$1683.33 \pm 402.51^{*\dagger\ddagger}$	$5578.95 \pm 295.56^{*\dagger\ddagger}$	$0.86 \pm 0.04^{*\dagger\ddagger}$	0.42 (0.23)
MinMax ($\beta = 0.1$)	$1609.88 \pm 379.81^{*\dagger\ddagger}$	$5548.56 \pm 263.49^{*\dagger\ddagger}$	$0.86 \pm 0.03^{*\dagger\ddagger}$	0.42 (0.21)
MinMax ($\beta = 0.3$)	$\mathbf{1395.32 \pm 109.48}^{*\dagger\ddagger}$	$\mathbf{5441.13 \pm 107.40}^{*\dagger\ddagger}$	$\mathbf{0.87 \pm 0.01}^{*\dagger}$	0.54 (0.21)
***k*-means**	2247.59 ± 804.75	5885.92 ± 542.49	0.82 ± 0.07	**0.10**
***k*-means++**	2134.54 ± 681.34	5800.23 ± 448.38	0.82 ± 0.07	0.11
Pifs	1761.13 ± 358.36	5496.97 ± 207.25	$\mathbf{0.87 \pm 0.02}$	**0.10**

pifs *k*-means is outperformed. As $\mathcal{E}_{max}$ is biased towards MinMax *k*-means and $\mathcal{E}_{sum}$ is optimized by neither algorithm, to get a more meaningful and fair comparison we should focus on MinMax+*k*-means and pifs+*k*-means. In this case, $\mathcal{E}_{sum}$ is the most informative measure, since it coincides with the *k*-means objective, and consistently MinMax+*k*-means edges ahead (apart from Dermatology when $\beta = 0$ or $\beta = 0.1$), signifying that the MinMax *k*-means solutions are of higher quality and thus when fed to *k*-means improved local optima are attained. In terms of NMI, they are closely matched, each achieving a better score than the other on half of the datasets (Tables 2.2-2.10). Note that apart from Ecoli and Dermatology, all other datasets consist of classes of equal size, thus we would expect pifs *k*-means, which explicitly balances the cluster sizes, to have the upper hand for this metric. Therefore, we can conclude that balancing the variance of the clusters is a more effective strategy.

By examining how memory affects the results, the following pattern arises. As the amount of memory grows, a greater reduction of $\mathcal{E}_{max}$ is possible, which is usually accompanied by an increase over $\mathcal{E}_{sum}$ (Tables 2.2(a)-2.10(a)). This can be explained from Table 2.11, which depicts a remarkable rise on the number of restarts that are free of empty or singleton clusters as memory increases. When no empty or singleton clusters are detected, p reaches p_{max} in our framework and, remember, that for higher p values large variance clusters are heavily punished, while less effort is put into minimizing the sum of the cluster variances. Two datasets severely deviate from the previous pattern, Multiple features-profile correlations and Pendigits, for which the use of memory (especially $\beta = 0.1$) yields partitionings of very poor quality. NMI-wise, $\beta = 0.1$ seems to be slightly worse than $\beta = 0$ and $\beta = 0.3$. For MinMax+*k*-means, the setting where

Table 2.11: Percentage (%) of MinMax *k*-means restarts over all nine datasets for which empty or singleton clusters never occur, in relation to the memory level.

Memory Level	**Percentage**
$\beta = 0$	14.96
$\beta = 0.1$	54.37
$\beta = 0.3$	91.19

$\beta = 0.3$ always displays (apart from Coil2) a better or, at least, equal score for $\mathcal{E}_{max}$, $\mathcal{E}_{sum}$ and NMI than the other two β settings. However, the performance differences between the memory levels for MinMax+*k*-means are small and, in general, not statistically significant on most datasets. Hence, larger memory seems to only slightly boost efficacy when initializing *k*-means.

The average execution time per run (in seconds) unveils, as anticipated, that *k*-means is the fastest method, followed by *k*-means++, pifs *k*-means and MinMax *k*-means. MinMax *k*-means is slower than *k*-means by a factor ranging between 3-6, depending on the dataset. This higher execution time is a direct consequence of our method requiring more iterations to converge, due to the process employed for adapting p to the data, and also the fact that for some restarts convergence is not achieved, hence t_{max} iterations are performed. Note that t_{max} is set to a high value in the experiments ($t_{max} = 500$). For this reason, the execution time for only those restarts that do converge is also shown (in parentheses) and for Coil3, Multiple features-profile correlations, Pendigits, Ecoli and Dermatology a significant reduction is observed. However, MinMax *k*-means is still more time consuming.

Overall, the experimental evaluation has revealed that MinMax *k*-means is superior to *k*-means, *k*-means++ and pifs *k*-means, although it incurs a higher computational cost. Importantly, our method guards against large variance clusters and evades poor solutions after bad initializations. Furthermore, it constitutes a sound approach for initializing *k*-means. This superior performance has been attained for general p_{max} and p_{step} values that were not tailored to each particular dataset, which greatly enhances the applicability of the presented algorithm. Regarding the use of memory, a higher memory further limits large variance clusters as well as the occurrence of empty or singleton clusters, but increases $\mathcal{E}_{sum}$ and its gains when used to initialize *k*-means are small. We could argue that memory is helpful, but not considerably, and even without memory ($\beta = 0$) solutions of very good quality can be obtained. As already discussed, the convergence of MinMax *k*-means cannot be theoretically guaranteed. However, for the conducted experiments about 60% of the restarts across all datasets do converge, empirically validating that runs which stop at a local optimum of the relaxed objective (2.3) are frequently encountered. Finally, a note on the Olivetti dataset, where the compared methods attain identical NMI scores (Tables 2.8(a)-2.8(b)): despite the NMI being equal on average, many of the individual restarts exhibit significant differences across the different methods.

2.5 Extension to Kernel Space

The MinMax *k*-means algorithm can be readily extended to perform kernel-based clustering [41], so that nonlinearly separable clusters are detected in the data. Analogously to kernel *k*-means[7] [37, 90], which generalizes *k*-means to kernel space, the dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x}_i \in \Re^d$ is mapped from input space to a higher dimensional reproducing kernel Hilbert space $\mathcal{H}$, a.k.a. feature space, via a nonlinear transformation $\phi : \mathcal{X} \to \mathcal{H}$. The MinMax *k*-means relaxed objective becomes:

$$\mathcal{E}_w = \sum_{k=1}^{M} w_k^p \sum_{i=1}^{N} \delta_{ik} \|\phi(\mathbf{x}_i) - \mathbf{m}_k\|^2, \; w_k \geq 0, \; \sum_{k=1}^{M} w_k = 1, \; 0 \leq p < 1, \tag{2.12}$$

where $\mathbf{m}_k = \sum_{i=1}^N \delta_{ik}\phi(\mathbf{x}_i) / \sum_{i=1}^N \delta_{ik}$ is the center of the k-th cluster in feature space. The cluster indicator variables δ_{ik}, the weights w_k and the p exponent are defined as in (2.3). To obtain a partitioning of the data, we must solve the min-max optimization problem of (2.4), but with $\mathcal{E}_w$ defined according to (2.12).

As for kernel *k*-means, a kernel function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \Re$ [41] is applied to directly provide the inner products in feature space without explicitly determining transformation ϕ, giving rise to the kernel matrix $K \in \Re^{N \times N}$, $K_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$. The squared Euclidean distances in (2.12) can now be computed using solely the kernel matrix entries:

$$\|\phi(\mathbf{x}_i) - \mathbf{m}_k\|^2 = K_{ii} - \frac{2\sum_{j=1}^N \delta_{jk} K_{ij}}{\sum_{j=1}^N \delta_{jk}} + \frac{\sum_{j=1}^N \sum_{l=1}^N \delta_{jk}\delta_{lk} K_{jl}}{\sum_{j=1}^N \sum_{l=1}^N \delta_{jk}\delta_{lk}} \; . \tag{2.13}$$

Note that although the centers $\mathbf{m}_k$ cannot be analytically calculated, since ϕ is unknown, Algorithm 2.1 requires only a few, tiny modifications in order to conduct clustering in feature space, giving rise to the MinMax kernel *k*-means algorithm.

2.5.1 Empirical Evaluation

To investigate whether the potential observed in input space for MinMax *k*-means carries over to feature space, MinMax kernel *k*-means is compared to kernel *k*-means [37, 90] on the Multiple features and Pendigits handwritten numerals, the Olivetti face database, and the artificial, Ten rings, dataset shown in Figure 2.4. The description of the first three datasets and their preprocessing details can be found in Section 2.4.1. Ten rings consists of five copies of two rings, where the inner ring is dense and has 700 points, while the outer ring has 300 points, yielding a total of 5000 points and ten clusters.

For all experiments we employ the rbf kernel function. The same experimental protocol as for the input space case is adopted (Section 2.4.2), however, $\mathcal{E}_{max}$ and $\mathcal{E}_{sum}$ are appropriately redefined to reflect the maximum cluster variance and the sum of

[7] For details on kernel *k*-means see Chapter 1, Section 1.1.3.

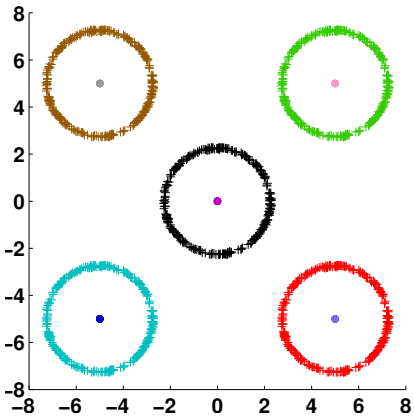


Figure 2.4: The Ten rings dataset.

the cluster variances in feature space, respectively. Similarly to MinMax k-means, we execute an additional series of experiments, where the solution returned by MinMax kernel k-means is used to initialize a subsequent run of kernel k-means.

Results are reported in Tables 2.12-2.16, where MinMax kernel k-means is abbreviated as Kernel MinMax. It can be observed that our method attains lower $\mathcal{E}_{max}$ values for all datasets, demonstrating, once again, its ability to penalize large variance clusters. It also outperforms kernel k-means in terms of $\mathcal{E}_{sum}$ and NMI, hence higher quality solutions are located on a more regular basis. Importantly, kernel k-means converges to better local minima when initialized by MinMax kernel k-means (Tables 2.12(b)-2.16(b)). Let us stress that for the Ten rings dataset, kernel k-means uncovers the correct partitioning three times over the 500 restarts, while, when initialized by our approach it does so for 11 ($\beta = 0$), 26 ($\beta = 0.1$) and 29 ($\beta = 0.3$) times. Of course, kernel k-means is faster on all tested data for the same reasons k-means is faster than MinMax k-means (Section 2.4.3).

Overall, the superior performance of MinMax k-means in input space is carried over to feature space and similar conclusions can be drawn for our method.

2.6 Summary

In this chapter, we have presented the MinMax k-means algorithm, a principled approach to circumvent the initialization problem associated with k-means [108]. Weights are assigned to the clusters in proportion to their variance and a weighted version of the k-means objective is optimized to restrain large variance clusters from appearing in the solution. A user specified p exponent is utilized to control the strictness of our method over large variance clusters. By punishing large variance clusters, bad initializations can be readily overcome to consistently uncover partitionings of high quality, irrespective of the initial choice of the cluster centers. Additionally, clusters are balanced in terms of their variance, which may prove useful as many data analysis scenarios re-

Table 2.12(a): Comparative results on the Ten rings dataset.

Method $\sigma = 1.8$	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	**NMI**	**Time**
Kernel MinMax ($\beta = 0$)	$671.82 \pm 390.04^{\overline{*}}$	$1290.80 \pm 175.90^{\overline{*}}$	$0.85 \pm 0.03^{\underline{*}}$	119.63 (13.02)
Kernel MinMax ($\beta = 0.1$)	$297.43 \pm 272.89^{\underline{*}}$	1103.29 ± 113.92	$0.88 \pm 0.01^{\underline{*}}$	80.28 (14.77)
Kernel MinMax ($\beta = 0.3$)	$\mathbf{150.81 \pm 95.40}^{\underline{*}}$	$\mathbf{1062.91 \pm 47.54}^{\underline{*}}$	$0.88 \pm 0.01^{\underline{*}}$	32.71 (15.30)
Kernel *k*-means	337.63 ± 225.56	1092.89 ± 165.66	$\mathbf{0.91 \pm 0.04}$	**5.05**

Table 2.12(b): Comparative results on the Ten rings dataset when kernel *k*-means is initialized by the solution returned by MinMax kernel *k*-means.

Method + kernel *k*-means $\sigma = 1.8$	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	**NMI**	**Time**
Kernel MinMax ($\beta = 0$)	$280.59 \pm 85.07^{\underline{*}}$	$1052.48 \pm 67.93^{\underline{*}}$	$0.92 \pm 0.03^{\overline{*}}$	124.00 (15.99)
Kernel MinMax ($\beta = 0.1$)	$249.04 \pm 66.71^{\underline{*}}$	$1028.55 \pm 51.96^{\underline{*}}$	$0.93 \pm 0.03^{\overline{*}}$	83.45 (18.18)
Kernel MinMax ($\beta = 0.3$)	$\mathbf{208.90 \pm 42.50}^{\underline{*}}$	$\mathbf{1011.98 \pm 47.74}^{\underline{*}}$	$\mathbf{0.95 \pm 0.03}^{\overline{*}}$	36.27 (18.92)
Kernel *k*-means	337.63 ± 225.56	1092.89 ± 165.66	0.91 ± 0.04	**5.05**

Table 2.13(a): Comparative results on the Multiple features (pixel averages) dataset.

Method $\sigma = 15$	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	**NMI**	**Time**
Kernel MinMax ($\beta = 0$)	$109.18 \pm 6.22^{\underline{*}}$	937.11 ± 4.10	$0.70 \pm 0.04^{\underline{*}}$	4.30 (3.80)
Kernel MinMax ($\beta = 0.1$)	$106.48 \pm 4.73^{\underline{*}}$	937.26 ± 3.98	$0.70 \pm 0.05^{\underline{*}}$	4.36 (3.86)
Kernel MinMax ($\beta = 0.3$)	$\mathbf{103.70 \pm 6.53}^{\underline{*}}$	937.41 ± 3.55	$0.70 \pm 0.05^{\underline{*}}$	4.06 (3.33)
Kernel *k*-means	172.10 ± 27.68	$\mathbf{937.04 \pm 7.13}$	$\mathbf{0.72 \pm 0.04}$	**1.18**

Table 2.13(b): Comparative results on the Multiple features (pixel averages) dataset when kernel *k*-means is initialized by the solution returned by MinMax kernel *k*-means.

Method + kernel *k*-means $\sigma = 15$	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	**NMI**	**Time**
Kernel MinMax ($\beta = 0$)	$150.30 \pm 17.19^{\underline{*}}$	$931.87 \pm 3.47^{\underline{*}}$	$\mathbf{0.74 \pm 0.04}^{\overline{*}}$	4.99 (4.49)
Kernel MinMax ($\beta = 0.1$)	$148.84 \pm 17.40^{\underline{*}}$	$931.76 \pm 3.22^{\underline{*}}$	$\mathbf{0.74 \pm 0.04}^{\overline{*}}$	5.07 (4.58)
Kernel MinMax ($\beta = 0.3$)	$\mathbf{145.24 \pm 17.27}^{\underline{*}}$	$\mathbf{931.66 \pm 3.15}^{\underline{*}}$	$\mathbf{0.74 \pm 0.04}^{\overline{*}}$	4.76 (4.02)
Kernel *k*-means	172.10 ± 27.68	937.04 ± 7.13	0.72 ± 0.04	**1.18**

Table 2.14(a): Comparative results on the Multiple features (profile correlations) dataset.

Method $\sigma = 9$	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	**NMI**	**Time**
Kernel MinMax ($\beta = 0$)	$154.81 \pm 7.88^{\underline{*}}$	$1328.24 \pm 7.83^{\underline{*}}$	$0.74 \pm 0.05^{\overline{*}}$	4.65 (3.94)
Kernel MinMax ($\beta = 0.1$)	$152.03 \pm 9.64^{\underline{*}}$	$\mathbf{1328.23 \pm 7.63}^{\underline{*}}$	$0.74 \pm 0.05^{\overline{*}}$	4.19 (3.96)
Kernel MinMax ($\beta = 0.3$)	$\mathbf{146.88 \pm 6.63}^{\underline{*}}$	$1328.34 \pm 6.83^{\underline{*}}$	$\mathbf{0.75 \pm 0.05}^{\overline{*}}$	4.01 (3.30)
Kernel *k*-means	263.28 ± 57.90	1332.17 ± 12.37	0.72 ± 0.03	**1.02**

Table 2.14(b): Comparative results on the Multiple features (profile correlations) dataset when kernel *k*-means is initialized by the solution returned by MinMax kernel *k*-means.

Method + kernel *k*-means $\sigma = 9$	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	**NMI**	**Time**
Kernel MinMax ($\beta = 0$)	$213.95 \pm 14.14^{\underline{*}}$	$1320.24 \pm 7.16^{\underline{*}}$	$0.75 \pm 0.03^{\overline{*}}$	5.26 (4.56)
Kernel MinMax ($\beta = 0.1$)	$213.23 \pm 12.83^{\underline{*}}$	$1319.68 \pm 6.95^{\underline{*}}$	$0.75 \pm 0.03^{\overline{*}}$	4.81 (4.58)
Kernel MinMax ($\beta = 0.3$)	$\mathbf{212.91 \pm 11.86}^{\underline{*}}$	$\mathbf{1318.93 \pm 6.23}^{\underline{*}}$	$\mathbf{0.76 \pm 0.03}^{\overline{*}}$	4.64 (3.93)
Kernel *k*-means	263.28 ± 57.90	1332.17 ± 12.37	0.72 ± 0.03	**1.02**

Table 2.15(a): Comparative results on the Pendigits dataset.

Method $\sigma = 2.1$	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	**NMI**	**Time**
Kernel MinMax ($\beta = 0$)	$770.06 \pm 26.33^{\underline{*}}$	$\mathbf{6622.69 \pm 12.46}^{\underline{*}}$	$0.74 \pm 0.01^{\overline{*}}$	90.71 (86.48)
Kernel MinMax ($\beta = 0.1$)	$753.16 \pm 49.07^{\underline{*}}$	$6627.69 \pm 12.52^{\underline{*}}$	$0.74 \pm 0.01^{\overline{*}}$	94.08 (89.68)
Kernel MinMax ($\beta = 0.3$)	$\mathbf{728.18 \pm 46.54}^{\underline{*}}$	$6632.42 \pm 8.12^{\underline{*}}$	$\mathbf{0.75 \pm 0.01}^{\overline{*}}$	79.13 (64.87)
Kernel *k*-means	1712.13 ± 291.25	6659.10 ± 108.98	0.74 ± 0.03	**27.75**

Table 2.15(b): Comparative results on the Pendigits dataset when kernel *k*-means is initialized by the solution returned by MinMax kernel *k*-means.

Method + kernel *k*-means $\sigma = 2.1$	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	**NMI**	**Time**
Kernel MinMax ($\beta = 0$)	$\mathbf{1568.09 \pm 34.37}^{\underline{*}}$	$\mathbf{6515.30 \pm 4.46}^{\underline{*}}$	$\mathbf{0.78 \pm 0.00}^{\overline{*}}$	100.51 (96.31)
Kernel MinMax ($\beta = 0.1$)	$1570.27 \pm 43.60^{\underline{*}}$	$6516.96 \pm 11.03^{\underline{*}}$	$\mathbf{0.78 \pm 0.00}^{\overline{*}}$	103.72 (99.30)
Kernel MinMax ($\beta = 0.3$)	$1568.22 \pm 17.76^{\underline{*}}$	$6515.51 \pm 5.46^{\underline{*}}$	$\mathbf{0.78 \pm 0.00}^{\overline{*}}$	88.25 (73.83)
Kernel *k*-means	1712.13 ± 291.25	6659.10 ± 108.98	0.74 ± 0.03	**27.75**

Table 2.16(a): Comparative results on the Olivetti dataset.

Method $\sigma = 6.5$	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	**NMI**	**Time**
Kernel MinMax ($\beta = 0$)	$24.42 \pm 1.04^{\underline{*}}$	$223.48 \pm 0.80^{\underline{*}}$	$\mathbf{0.35 \pm 0.04}$	0.52 (0.49)
Kernel MinMax ($\beta = 0.1$)	$24.18 \pm 0.86^{\underline{*}}$	$223.47 \pm 0.81^{\underline{*}}$	$\mathbf{0.35 \pm 0.04}$	0.49 (0.47)
Kernel MinMax ($\beta = 0.3$)	$\mathbf{23.96 \pm 0.95}^{\underline{*}}$	$\mathbf{223.44 \pm 0.80}^{\underline{*}}$	$\mathbf{0.35 \pm 0.04}$	0.44 (0.43)
Kernel *k*-means	32.69 ± 3.40	223.88 ± 1.19	$\mathbf{0.35 \pm 0.03}$	**0.18**

Table 2.16(b): Comparative results on the Olivetti dataset when kernel *k*-means is initialized by the solution returned by MinMax kernel *k*-means.

Method + kernel *k*-means $\sigma = 6.5$	$\mathcal{E}_{max}$	$\mathcal{E}_{sum}$	**NMI**	**Time**
Kernel MinMax ($\beta = 0$)	$27.66 \pm 2.33^{\underline{*}}$	$222.91 \pm 0.76^{\underline{*}}$	$\mathbf{0.35 \pm 0.04}$	0.60 (0.56)
Kernel MinMax ($\beta = 0.1$)	$27.52 \pm 2.40^{\underline{*}}$	$222.90 \pm 0.77^{\underline{*}}$	$\mathbf{0.35 \pm 0.04}$	0.57 (0.55)
Kernel MinMax ($\beta = 0.3$)	$\mathbf{27.37 \pm 2.38}^{\underline{*}}$	$\mathbf{222.87 \pm 0.75}^{\underline{*}}$	$\mathbf{0.35 \pm 0.04}$	0.52 (0.50)
Kernel *k*-means	32.69 ± 3.40	223.88 ± 1.19	$\mathbf{0.35 \pm 0.03}$	**0.18**

quire groups of roughly the same size. Training involves a min-max problem that is iteratively solved, where the weights are updated in the maximization step to accurately reflect the variances of the clusters at each iteration. Moreover, we have presented a methodology for adjusting the p exponent to the underlying dataset properties, so that the intrinsic group structures can be identified, which greatly facilitates the application of our algorithm. Furthermore, we have developed the MinMax kernel *k*-means algorithm, which locates nonlinearly separable clusters by extending MinMax *k*-means to perform clustering in kernel space.

To draw reliable conclusions, MinMax *k*-means was extensively tested on various datasets. Results demonstrate its robustness over bad initializations and its efficacy, as for most cases it outperforms (in terms of clustering quality) all three compared methods, namely *k*-means, *k*-means++ [3] and pifs *k*-means [8]. Furthermore, we noticed that *k*-means solutions can be significantly improved when initialized by MinMax *k*-means, suggesting an important additional usage of our approach. Similar observations to the above can be also made for MinMax kernel *k*-means, which was compared to kernel *k*-means. Overall, MinMax *k*-means appears to be a very competitive and easy to employ method for dealing with the sensitivity to initialization of *k*-means.

Chapter 3

Multiple View Clustering Using Exemplar-based Mixture Models

The most common approach for the machine learning and data mining settings is to assume that instances are represented in a single, vector or graph, space. However, in many real-life problems data with multiple views naturally arise. Multi-view data consist of instances that have multiple representations (views) from different feature spaces. Usually these multiple views are from different vector spaces, or different graph spaces, or a combination of vector and graph spaces. The most typical example are web pages. Web pages can be represented using the text of the web page and the hyperlink graph among the web pages. Another example is scientific articles, which can be represented by utilizing the text appearing in the abstract and the title, as well as the co-author and citation graphs.

The natural and frequent occurrence of multi-view data has raised interest in the so called *multi-view learning*[1]. The main challenge of multi-view learning is to develop algorithms that use multiple views simultaneously, given the diversity of the views. Most studies on this topic address the semi-supervised classification problem and multi-view classification algorithms have often proven to utilize unlabeled data effectively and substantially improve classification accuracy (e.g. [14, 17, 80, 84]).

[1]For details on multi-view learning see Chapter 1, Section 1.2.

This chapter focuses on multi-view unsupervised learning and particularly in *multi-view clustering*. Multi-view clustering explores and exploits multiple representations simultaneously, in order to produce a more accurate and robust partitioning of the data. The intuition behind this approach is that the different representations are potentially more informative regarding the correct partitioning of the dataset, than a single view. Therefore, by taking advantage of all the available views, we expect to locate a better split of the data. The available literature for this topic is growing fast (e.g. [10, 13, 24, 34, 50, 77, 131]), with encouraging results.

On the following, we present a multi-view clustering algorithm based on the *convex mixture models* (CMMs) of Lashkari and Golland [71]. CMMs (a.k.a. exemplar-based mixture models) are a special case of mixture models that identify exemplars in the dataset (i.e. instances that serve as the representatives of the clusters), by optimizing a convex criterion whose global optimum solution can be found. Hence, they avoid the initialization and local optima problems of standard mixture models, which require multiple executions of the EM algorithm [12, 35]. Moreover, they can be applied to problems where only the pairwise distance matrix of the data is available and not the instances. The proposed *multi-view convex mixture model* (multi-view CMM) generalizes CMMs[2] to data with multiple representations and locates exemplars, through a convex optimization, by equally considering all available views [104]. The aforementioned advantages of the single view CMM are retained and, additionally, the different statistical properties of the views are taken into account.

An important observation is that most multi-view algorithms rely equally on every view in order to compute a clustering. However, the useful information conveyed by the available views can vary significantly. For example, some views may contain noise, or outliers, or be irrelevant to the underlying task. Including such views in the partitioning process may result in performance degradation. Identifying and removing such views beforehand is not easy though. For this reason, we also present a multi-view clustering method that *assigns different weights to the views and learns those weights automatically* [106]. The weights reflect the quality of each view and, therefore, affect its contribution to the final clustering solution accordingly. Specifically, we extend multi-view CMMs to accommodate weights for the views. This *weighted multi-view convex mixture model* (weighted multi-view CMM) is actually a weighted combination of CMMs (one for each view) that incorporates most of the advantages of multi-view CMMs, plus the ability to spot irrelevant views through the weights. As we shall see, this model has also a probabilistic interpretation.

Experiments with our two algorithms on noisy artificial datasets and two real datasets, demonstrate in most cases a considerable improvement on the clustering performance, especially for the weighted multi-view CMM, when compared to: i) the single view CMM [71] applied on the individual views, ii) the single view CMM that uses

[2]We shall refer to CMMs as single view CMMs, whenever it is necessary to make the distinction to their multi-view counterparts explicit.

the concatenation of the views and iii) the multi-view clustering method of [13] that first combines the views through kernel canonical correlation analysis projection [51] and then clusters the derived projections. Moreover, results confirm the ability of the weighted multi-view CMM to correctly measure the quality of the views and adjust the weights so as to get good clustering solutions, not affected by the presence of noisy or non-informative views.

The rest of this chapter is organized as follows. Section 3.1 reviews single view CMMs. The proposed algorithms follow in Section 3.2 and Section 3.3. The experimental evaluation on artificial and real data is discussed in Section 3.4. Finally, Section 3.5 summarizes the chapter.

3.1 Convex Mixture Models

This section briefly describes *convex mixture models* (CMMs) [71], also called exemplar-based mixture models, since they consist a key part of our new algorithms. CMMs are simplified mixture models which result in probabilistic (soft) assignments of data points to clusters and in the extraction of representative exemplars from the dataset. When training these models, which is done by maximizing the log-likelihood, all instances compete to become cluster representatives (i.e. exemplars), since *the number of the CMM components is equal to the number of data points* and *each component distribution is centered at a distinct dataset point*. In the end, the instances corresponding to the components that have received during training the highest priors are selected as exemplars.

Given a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x}_i \in \Re^d$, the CMM distribution is:

$$Q(\mathbf{x}) = \sum_{j=1}^{N} q_j f_j(\mathbf{x}),\ \mathbf{x} \in \Re^d, \tag{3.1}$$

where $q_j \geq 0$ denotes the prior probability of the j-th component, satisfying the constraint $\sum_{j=1}^N q_j = 1$, and $f_j(\mathbf{x})$ is an exponential family distribution, with its *expectation parameter* equal to the j-th data point $\mathbf{x}_j$. Note that the same exponential family is used for all components. Taking into account the bijection between regular exponential families and Bregman divergences [9], we can write:

$$f_j(\mathbf{x}) = C_\varphi(\mathbf{x}) \exp(-\beta d_\varphi(\mathbf{x}, \mathbf{x}_j)), \tag{3.2}$$

with d_φ denoting the Bregman divergence corresponding to the components' distributions, $C_\varphi(\mathbf{x})$ being independent of $\mathbf{x}_j$ and β being a constant controlling the sharpness of the components [71].

A clustering is produced by maximizing the dataset log-likelihood (3.3) over the prior probabilities $\{q_j\}_{j=1}^N$, s.t. $q_j \geq 0$, $\sum_{j=1}^N q_j = 1$. Note that *the priors of the components are*

the only adjustable parameters of a CMM.

$$L\left(\mathcal{X};\{q_j\}_{j=1}^N\right)=\frac{1}{N}\sum_{i=1}^{N}\log\left(\sum_{j=1}^{N}q_j f_j(\mathbf{x}_i)\right)=\frac{1}{N}\sum_{i=1}^{N}\log\left(\sum_{j=1}^{N}q_j e^{-\beta d_\varphi(\mathbf{x}_i,\mathbf{x}_j)}\right)+\text{const.} \tag{3.3}$$

If we define $\hat{P}(\mathbf{x})=1/N$, $\mathbf{x}\in\mathcal{X}$ to be the empirical dataset distribution, we can equivalently formulate the above likelihood maximization problem in terms of the Kullback-Leibler (KL) divergence between $\hat{P}(\mathbf{x})$ and $Q(\mathbf{x})$, since their KL distance is:

$$D(\hat{P}\|Q)=-\sum_{i=1}^{N}\hat{P}(\mathbf{x}_i)\log Q(\mathbf{x}_i)-\mathbb{H}(\hat{P})=-L\left(\mathcal{X};\{q_j\}_{j=1}^N\right)+\text{const.}\,, \tag{3.4}$$

where $\mathbb{H}(\hat{P})$ is the entropy of the empirical distribution that does not depend on the parameters q_j of the CMM. Now the maximization of (3.3) is equivalent to the minimization of (3.4). This minimization problem is *convex* and can be solved with an iterative algorithm, whose updates for the components' prior probabilities are given by:

$$q_j^{(t+1)}=q_j^{(t)}\sum_{i=1}^{N}\frac{\hat{P}(\mathbf{x}_i)f_j(\mathbf{x}_i)}{\sum_{j'=1}^{N}q_{j'}^{(t)}f_{j'}(\mathbf{x}_i)}\,, \tag{3.5}$$

and the algorithm is *guaranteed to converge to the global minimum* as long as $q_j^{(0)}>0,\forall j$ [31]. Updating the priors costs $O(N^2\tau)$ scalar operations, where τ is the number of iterations until convergence. Importantly, the prior probability q_j associated with instance $\mathbf{x}_j$ is a measure of *how likely this instance is to become an exemplar*.

The ability of always being able to locate the global optimum makes this model attractive as it avoids the initialization and local optima problems of standard mixture models, which demand multiple executions of the EM algorithm [12, 35]. Another important feature is that only the pairwise data distances $d_\varphi(\mathbf{x}_i,\mathbf{x}_j)$ take part in the calculation of the priors (3.5) as $C_\varphi(\mathbf{x}_i)$ cancels out, thus the values of the instances are not required if we are given the distances.

Splitting the dataset into M disjoint clusters is done by requiring the instances with the M highest q_j values to serve as exemplars and then assigning the remaining instances to the exemplar with the highest posterior probability.

Finally, when clustering with a CMM we must select an appropriate value for the constant β ($0<\beta<\infty$). It is possible to identify a reasonable range of β values by determining a reference value β_0. Lashkari and Golland [71] proposed the following empirical rule for β_0:

$$\beta_0=N^2\log N/\sum_{i,j=1}^{N}d_\varphi(\mathbf{x}_i,\mathbf{x}_j). \tag{3.6}$$

3.2 Multi-view Convex Mixture Models

Motivated by the potential of CMMs, here, we extend them to data with multiple representations [104]. Suppose we are given a dataset with N instances and V views, $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^N$, where $\boldsymbol{x}_i$ contains the representations of the i-th instance across the views, i.e. $\boldsymbol{x}_i = \left\{\mathbf{x}_i^{(v)}\right\}_{v=1}^V$, $\mathbf{x}_i^{(v)} \in \Re^{d^{(v)}}$. Define for each view a CMM distribution:

$$Q^v(\mathbf{x}^{(v)}) = \sum_{j=1}^N q_j f_j^v(\mathbf{x}^{(v)}) = C_{\varphi_v}(\mathbf{x}^{(v)}) \sum_{j=1}^N q_j e^{-\beta^v d_{\varphi_v}(\mathbf{x}^{(v)}, \mathbf{x}_j^{(v)})},\ \mathbf{x}^{(v)} \in \Re^{d^{(v)}}, \tag{3.7}$$

and a uniform empirical dataset distribution $\hat{P}^v(\mathbf{x}^{(v)}) = 1/N, \mathbf{x}^{(v)} \in \{\mathbf{x}_1^{(v)}, \mathbf{x}_2^{(v)}, \ldots, \mathbf{x}_N^{(v)}\}$, analogously to Section 3.1. Note that all $Q^v(\mathbf{x}^{(v)})$ distributions *share the same prior probabilities* q_j to allow the views to interact, but have different component distributions $f_j^v(\mathbf{x}^{(v)})$.

Our aim is to locate high quality exemplars (cluster centroids) in the dataset, by considering all views simultaneously, around which the remaining instances will cluster. To achieve this, the proposed *multi-view convex mixture model* (multi-view CMM) generalizes (3.4) by minimizing the sum of KL divergences between $\hat{P}^v(\mathbf{x}^{(v)})$ and $Q^v(\mathbf{x}^{(v)})$ across all views:

$$\begin{aligned} &\min_{\{q_j\}_{j=1}^N} \sum_{v=1}^V D(\hat{P}^v \| Q^v),\ s.t.\ \ q_j \geq 0,\ \sum_{j=1}^N q_j = 1 \\ \Rightarrow &\min_{\{q_j\}_{j=1}^N} -\sum_{v=1}^V \sum_{i=1}^N \hat{P}^v(\mathbf{x}_i^{(v)}) \log Q^v(\mathbf{x}_i^{(v)}) - \sum_{v=1}^V \mathbb{H}(\hat{P}^v),\ s.t.\ \ q_j \geq 0,\ \sum_{j=1}^N q_j = 1, \end{aligned} \tag{3.8}$$

where $\mathbb{H}(\hat{P}^v)$ is the entropy of the empirical distribution of the v-th view, that does not depend on the parameters q_j of the multi-view CMM.

It is quite straightforward to see that the optimized objective is *convex*, since it is the sum of the single view objectives which are convex functions (Section 3.1). The same iterative algorithm as for the single view CMM can be used to find the *global minimum* of (3.8) and it can be shown that the rule for updating the priors is given by:

$$q_j^{(t+1)} = \frac{q_j^{(t)}}{V} \sum_{v=1}^V \sum_{i=1}^N \frac{\hat{P}^v(\mathbf{x}_i^{(v)}) f_j^v(\mathbf{x}_i^{(v)})}{\sum_{j'=1}^N q_{j'}^{(t)} f_{j'}^v(\mathbf{x}_i^{(v)})}, \tag{3.9}$$

which is a generalization of the single view case (3.5). Obviously, the prior q_j associated with the j-th instance is a measure of *how likely this instance is to be an exemplar*, taking into account all views.

The multi-view CMM has some very important characteristics. To capture the diversity among the views, they are allowed to have distributions $f_j^v(\mathbf{x}^{(v)})$ coming from different exponential families, i.e. have different β^v values and Bregman divergences d_{φ_v}. For example, a Gaussian CMM can be used for one view and a Bernoulli CMM for

another. A key property of single view CMMs is convexity, which enables the identification of the global optimum q_j values, and we wish to preserve this property in the multi-view setting. As a result, summing the single view objectives to construct the multi-view objective is a natural choice. Moreover, the priors q_j are the same across all views, to allow the extraction of representative exemplars based on every view. Therefore, an instance whose corresponding prior has a high value, will more or less be a good exemplar for all views. Finally, for the q_j updates (3.9) only the pairwise distances $d_{\varphi_v}(\mathbf{x}_i^{(v)}, \mathbf{x}_j^{(v)})$ in each view are required, as $C_{\varphi_v}(\mathbf{x}_i^{(v)})$ cancels out. Hence, we can apply the multi-view CMM even when the actual data points are unknown.

Splitting the dataset into M disjoint clusters, $\{\mathcal{C}_k\}_{k=1}^M$, is done by selecting the instances with the M highest q_j values to serve as exemplars. To find the exemplars, we iteratively update the priors (3.9) until the M highest q_j values correspond to the same instances for a number of consecutive iterations. Moreover, we require that the order among the M highest q_j values remains the same during these iterations. Afterward, we assign each of the remaining $N - M$ instances to the cluster $\mathcal{C}_k$, associated with the k-th exemplar, that has the largest sum of posterior probabilities over all views (3.10). Notice that we denote by $\mathcal{X}^E = \left\{\boldsymbol{x}_k^E\right\}_{k=1}^M \subset \mathcal{X}$ the set containing the exemplars and refer to the prior probabilities and component distributions (in the v-th view) of the exemplars, as q_k^E and $f_k^{vE}(\mathbf{x}^{(v)})$, $k = 1, \ldots, M$, respectively.

$$\mathcal{C}_k = \left\{\boldsymbol{x}_k^E\right\} \cup \left\{ \boldsymbol{x}_i \left| \sum_{v=1}^{V} \frac{q_k^E f_k^{vE}(\mathbf{x}_i^{(v)})}{\sum_{j=1}^{N} q_j f_j^v(\mathbf{x}_i^{(v)})} > \sum_{v=1}^{V} \frac{q_l^E f_l^{vE}(\mathbf{x}_i^{(v)})}{\sum_{j=1}^{N} q_j f_j^v(\mathbf{x}_i^{(v)})}, \forall l \neq k, \boldsymbol{x}_i \notin \mathcal{X}^E \right. \right\} \quad (3.10)$$

To employ multi-view CMMs, appropriate values for the β^v constants must be chosen. Since a separate CMM is defined for each view, we can identify a reasonable range of β^v values by following the ideas of the single view case (Section 3.1), leading to the derivation of the following empirical β_0^v value:

$$\beta_0^v = N^2 \log N / \sum_{i,j=1}^{N} d_{\varphi_v}(\mathbf{x}_i^{(v)}, \mathbf{x}_j^{(v)}). \quad (3.11)$$

As for the complexity of multi-view CMMs, the update of the priors costs $O(N^2 V \tau)$ scalar operations, where τ is the number of iterations until convergence. If the pairwise distances $d_{\varphi_v}(\mathbf{x}_i^{(v)}, \mathbf{x}_j^{(v)})$ of the views are not given, their calculation usually costs an extra $O(N^2 V d)$, $d = \max\left\{d^{(1)}, d^{(2)} \ldots, d^{(V)}\right\}$, scalar operations.

3.3 Weighted Multi-view Convex Mixture Models

In this section, we present another CMM-based multi-view clustering approach, where weights are assigned to the views and adjusted through training [106].

3.3.1 Model Description

From the multi-view CMM objective (3.8), it can be observed that all views contribute equally to the sum, regardless of how "good" each view is for the problem at hand. Our target is to locate exemplars in the dataset by *allowing the views to participate with different weights* to the objective function, measuring how "informative" is the corresponding view, and by *learning those weights automatically*, i.e. as part of the learning process. Such an approach generalizes the previous one and could be helpful in cases where a view is irrelevant to the clustering task, or contains noise.

To accomplish our target we introduce a weighted combination of exemplar-based models. For the v-th view a CMM distribution $Q^v(\mathbf{x}^{(v)})$, of the same form as in (3.7), is defined and a positive weight π^v is associated with it. The views are combined by summing the corresponding weighted CMMs.

In more detail, suppose we are given a dataset with N instances and V views, $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^N$, where $\boldsymbol{x}_i = \left\{\mathbf{x}_i^{(v)}\right\}_{v=1}^V$, $\mathbf{x}_i^{(v)} \in \Re^{d^{(v)}}$. Our model, to which we will refer to as *weighted multi-view convex mixture model* (weighted multi-view CMM), is given by:

$$F\left(\boldsymbol{x} = \left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(V)}\right\}\right) = \sum_{v=1}^{V} \pi^v Q^v(\mathbf{x}^{(v)}) = \sum_{v=1}^{V} \pi^v \sum_{j=1}^{N} q_j f_j^v(\mathbf{x}^{(v)}),\ \mathbf{x}^{(v)} \in \Re^{d^{(v)}},$$

$$\text{where } f_j^v(\mathbf{x}^{(v)}) = C_{\varphi_v}(\mathbf{x}^{(v)}) e^{-\beta^v d_{\varphi_v}(\mathbf{x}^{(v)}, \mathbf{x}_j^{(v)})},\ \pi^v \geq 0,\ \sum_{v=1}^{V} \pi^v = 1,\ q_j \geq 0,\ \sum_{j=1}^{N} q_j = 1. \tag{3.12}$$

Note the imposed constraints on the weights π^v. Due to these restrictions, $F(\boldsymbol{x})$ has a probabilistic interpretation. Specifically, it is a mixture model, whose number of components is equal to the number of the views and each component is a CMM distribution $Q^v(\mathbf{x}^{(v)})$, corresponding to the v-th view. Hence, *the weights can also be seen as the prior probabilities of the views* under the mixture model.

The above formulation has some rather attractive characteristics. Similar to the multi-view CMM (Section 3.2), as the statistical properties of individual views may differ substantially, different views are allowed to have component distributions $f_j^v(\mathbf{x}^{(v)})$ coming from different exponential families, i.e. have different β^v values and Bregman divergences d_{φ_v}. Moreover, all views share the same priors q_j in order to interact and allow the extraction of representative exemplars based on every view. Since a CMM is used for each view, all instances will be considered as possible cluster centroids (i.e. exemplars) during training. Finally, a low π^v value indicates that the v-th view conveys little information regarding the partitioning of the dataset.

3.3.2 Model Training and Multi-view Clustering

Since $F(\boldsymbol{x})$ can be viewed as a mixture model, to partition the dataset $\mathcal{X}$, we must maximize the log-likelihood (3.13) w.r.t. the parameters $\{\pi^v\}_{v=1}^V$, $\{q_j\}_{j=1}^N$, s.t. the con-

straints $\pi^v \geq 0$, $\sum_{v=1}^{V} \pi^v = 1$, $q_j \geq 0$, $\sum_{j=1}^{N} q_j = 1$. It must be stressed that in contrast to multi-view CMMs, this optimization task is *not convex* due to the introduction of the weights π^v. However, we hope to compensate for the lost convexity by the ability to estimate different weights for the views.

$$L\left(\mathcal{X}; \{\pi^v\}_{v=1}^{V}, \{q_j\}_{j=1}^{N}\right) = \sum_{i=1}^{N} \log\left(\sum_{v=1}^{V} \pi^v Q^v(\mathbf{x}_i^{(v)})\right) = \sum_{i=1}^{N} \log\left(\sum_{v=1}^{V} \pi^v \sum_{j=1}^{N} q_j f_j^v(\mathbf{x}_i^{(v)})\right) \tag{3.13}$$

Local maxima of the log-likelihood can be found by applying the EM algorithm [12, 35]. This algorithm uses an initial guess for the parameters and iteratively adjusts them, such that the likelihood always increases, until a local optimum is reached. Our model has only prior probabilities that can be adjusted, hence initializing them uniformly and *avoiding multiple restarts for EM* is a natural choice (this approach is followed in the experiments), i.e. $\pi^{v(0)} = 1/V, q_j^{(0)} = 1/N$. Of course, if prior knowledge for the quality of the views exists, this can be directly incorporated into the optimization by initializing π^v accordingly. To briefly illustrate the steps of EM, define $\{\mathcal{X}, \mathcal{Z}\}$ to be the complete dataset, where $\mathcal{Z} = \{z_i\}_{i=1}^{N}$ contains the latent variables indicating the mixture component responsible for generating each instance, i.e. $z_i \in \{1, 2, \ldots, V\}$. The analytical derivation of the EM equations can be found in Appendix A.

E-step

In practice we are not given the complete dataset, but only the observations $\mathcal{X}$. Our state of knowledge for the latent variables is described through the posterior probabilities $P(z_i = v|\boldsymbol{x}_i)$ (3.14), which at iteration t are calculated $\forall i \in \{1, 2, \ldots, N\}, \forall v \in \{1, 2, \ldots, V\}$ as:

$$P^{(t)}(z_i = v|\boldsymbol{x}_i) = \frac{\pi^{v(t)} Q^{v(t)}(\mathbf{x}_i^{(v)})}{\sum_{v=1}^{V} \pi^{v(t)} Q^{v(t)}(\mathbf{x}_i^{(v)})} \,. \tag{3.14}$$

M-step

The posterior probabilities of the E-step are useful in estimating new values for the parameters during the M-step. By setting to zero the derivatives of the constrained complete dataset log-likelihood expectation (see equation (A.7) in Appendix A), under the posterior probabilities distribution, w.r.t. $\{\pi^v\}_{v=1}^{V}$, $\{q_j\}_{j=1}^{N}$, and a little manipulation, we get the updates for the parameters:

$$\pi^{v(t+1)} = \frac{1}{N} \sum_{i=1}^{N} P^{(t)}(z_i = v|\boldsymbol{x}_i), \tag{3.15}$$

$$q_j^{(t'+1)} = \frac{q_j^{(t')}}{N} \sum_{i=1}^{N} \sum_{v=1}^{V} P^{(t)}(z_i = v|\boldsymbol{x}_i) \frac{f_j^v(\mathbf{x}_i^{(v)})}{\sum_{j'=1}^{N} q_{j'}^{(t')} f_{j'}^v(\mathbf{x}_i^{(v)})} \,. \tag{3.16}$$

Some remarks on the optimization process, whose pseudo-code is illustrated in Algorithm 3.1, follow. First, the new estimation $q_j^{(t'+1)}$ depends on the previous value $q_j^{(t')}$, therefore a nested loop in the M-step of the EM algorithm is required to perform multiple updates on q_j for the same set of posterior probabilities values in order to get $q_j^{(t+1)}$. This loop finishes when the change on q_j values between two iterations is less than a small value ϵ' (line 21). Second, EM terminates when the likelihood between consecutive pairs of E and M steps changes less than a small value ϵ (line 24). Third, we must explicitly incorporate into the calculation of the posterior probabilities (3.14) the $C_{\varphi_v}(\mathbf{x}_i^{(v)})$ values (for the q_j estimations (3.16) the $C_{\varphi_v}(\mathbf{x}_i^{(v)})$ values still cancel out). Hence, the pairwise distances alone do not suffice to compute the updates and the dataset instances are required in the general case, contrary to the single view and multi-view CMMs, but for certain distributions $f_j^v(\mathbf{x}^{(v)})$ this is not necessary, as demonstrated in the experimental section for the Gaussian distribution. Fourth, the same empirical β_0^v values (3.11) as in Section 3.2 can be adopted to guide the search for appropriate β^v values. Finally, it is obvious that the view weights π^v are determined automatically during the M-step.

If we wish to split the dataset $\mathcal{X}$ into M disjoint clusters, $\{\mathcal{C}_k\}_{k=1}^M$, after EM termination, we must select the instances that will act as exemplars. For this purpose the instances with the M highest q_j values are chosen, denoted by the set $\mathcal{X}^E = \left\{\boldsymbol{x}_k^E\right\}_{k=1}^M \subset \mathcal{X}$. The remaining $N - M$ instances are assigned to the cluster $\mathcal{C}_k$, associated with the k-th exemplar, that has the largest posterior probability $P(\mathcal{C}_k|\boldsymbol{x}_i)$:

$$P(\mathcal{C}_k|\boldsymbol{x}_i) = \frac{q_k^E \sum_{v=1}^V \pi^v f_k^{vE}(\mathbf{x}_i^{(v)})}{\sum_{v=1}^V \pi^v \sum_{j=1}^N q_j f_j^v(\mathbf{x}_i^{(v)})}\,, \tag{3.17}$$

where we refer to the prior and the component distribution (of the v-th view CMM) corresponding to exemplar $\boldsymbol{x}_k^E$, as q_k^E and $f_k^{vE}(\mathbf{x}^{(v)})$ respectively (the proof of (3.17) can be found in Appendix A). Hence, the cluster assignments are given by:

$$\mathcal{C}_k = \left\{\boldsymbol{x}_k^E\right\} \cup \left\{\boldsymbol{x}_i \,\middle|\, P(\mathcal{C}_k|\boldsymbol{x}_i) > P(\mathcal{C}_l|\boldsymbol{x}_i), \forall l \neq k, \boldsymbol{x}_i \notin \mathcal{X}^E\right\}. \tag{3.18}$$

3.3.3 Additional Aspects

If the data points of each view are mapped from input space to a higher dimensional feature space (i.e. kernel space), through a nonlinear transformation $\phi^{(v)}$, our method can be readily applied to the mapped data $\left\{\phi^{(1)}(\mathbf{x}_i^{(1)}), \phi^{(2)}(\mathbf{x}_i^{(2)}), \ldots, \phi^{(V)}(\mathbf{x}_i^{(V)})\right\}, i = 1, 2, \ldots, N$ and thus perform multi-view clustering in feature space. By representing the instances in a new space, a clearer group structure may emerge. A kernel function $\mathcal{K}^{(v)}(\mathbf{x}_i^{(v)}, \mathbf{x}_j^{(v)})$ [41] can be employed to get the inner products in feature space without explicitly determining $\phi^{(v)}$, giving rise to the kernel matrix $K^{(v)} \in \Re^{N\times N}$, where $K_{ij}^{(v)} = \mathcal{K}^{(v)}(\mathbf{x}_i^{(v)}, \mathbf{x}_j^{(v)}) = \phi^{(v)}(\mathbf{x}_i^{(v)})^\top \phi^{(v)}(\mathbf{x}_j^{(v)})$. Hence, it is not necessary to define $\phi^{(v)}$, when the calculations of the underlying algorithm only involve inner products in feature space (for certain kernel functions the corresponding transformation is intractable). This is

Algorithm 3.1 EM for weighted multi-view CMMs.

Input: Multi-view dataset $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^N$, where $\boldsymbol{x}_i = \left\{\mathbf{x}_i^{(v)}\right\}_{v=1}^V$

Output: Model parameters estimation: $\{\pi^v\}_{v=1}^V$, $\{q_j\}_{j=1}^N$

// Initialize the parameters.

Set $\pi^{v(0)} = 1/V, \forall v = 1, \ldots, V$

Set $q_j^{(0)} = 1/N, \forall j = 1, \ldots, N$

Set $t = 0$

repeat

  // E-step.

  for $i = 1$ to N **do**

    for $v = 1$ to V **do**

      $P^{(t)}(z_i = v|\boldsymbol{x}_i) = \frac{\pi^{v(t)} Q^{v(t)}(\mathbf{x}_i^{(v)})}{\sum_{v=1}^V \pi^{v(t)} Q^{v(t)}(\mathbf{x}_i^{(v)})} = \frac{\pi^{v(t)} \sum_{j=1}^N q_j^{(t)} f_j^v(\mathbf{x}_i^{(v)})}{\sum_{v=1}^V \pi^{v(t)} \sum_{j=1}^N q_j^{(t)} f_j^v(\mathbf{x}_i^{(v)})}$

    end for

  end for

  // M-step.

  for all π^v, $v = 1$ to V **do** // Update the weights π^v.

    $\pi^{v(t+1)} = \frac{1}{N} \sum_{i=1}^N P^{(t)}(z_i = v|\boldsymbol{x}_i)$

  end for

  Set $t' = t$

  repeat // Update the priors q_j.

    for all q_j, $j = 1$ to N **do**

      $q_j^{(t'+1)} = \frac{q_j^{(t')}}{N} \sum_{i=1}^N \sum_{v=1}^V P^{(t)}(z_i = v|\boldsymbol{x}_i) \frac{f_j^v(\mathbf{x}_i^{(v)})}{\sum_{j'=1}^N q_{j'}^{(t')} f_{j'}^v(\mathbf{x}_i^{(v)})}$

    end for

    $t' = t' + 1$

  until $\sum_{j=1}^N \left| q_j^{(t')} - q_j^{(t'-1)} \right| < \epsilon'$

  $t = t + 1$

  Set $\left\{q_j^{(t)}\right\}_{j=1}^N = \left\{q_j^{(t')}\right\}_{j=1}^N$

until $\left| L\left(\mathcal{X}; \left\{\pi^{v(t)}\right\}_{v=1}^V, \left\{q_j^{(t)}\right\}_{j=1}^N\right) - L\left(\mathcal{X}; \left\{\pi^{v(t-1)}\right\}_{v=1}^V, \left\{q_j^{(t-1)}\right\}_{j=1}^N\right) \right| < \epsilon$

return $\{\pi^v\}_{v=1}^V = \left\{\pi^{v(t)}\right\}_{v=1}^V$, $\{q_j\}_{j=1}^N = \left\{q_j^{(t)}\right\}_{j=1}^N$

the case for our approach when considering Gaussian CMMs in the feature space of each view, as $d_{\varphi_v}(\phi^{(v)}(\mathbf{x}_i^{(v)}), \phi^{(v)}(\mathbf{x}_j^{(v)})) = \|\phi^{(v)}(\mathbf{x}_i^{(v)}) - \phi^{(v)}(\mathbf{x}_j^{(v)})\|^2 = K_{ii}^{(v)} + K_{jj}^{(v)} - 2K_{ij}^{(v)}$. Consequently, the required pairwise distances are expressed using only the entries of the kernel matrices. This also makes our method directly applicable for unsupervised *multiple kernel learning*[3] [110], in case we are given multiple kernels for the instances of a single view dataset and wish to find an appropriate combination of the kernels that clusters the data efficiently, i.e. our algorithm will treat each kernel as being a distinct view, will compute the pairwise distances as shown above and a combination of the kernels will be obtained through the weights π^v.

As for the complexity of the EM for our model, the calculation of the posteriors $P(z_i = v|\boldsymbol{x}_i)$ requires $O(N^2V)$ scalar operations, while the updates on the weights π^v and the priors q_j cost $O(NV)$ and $O(N^2V)$ scalar operations respectively. Assuming τ EM iterations are performed until convergence and τ' iterations in each nested loop of the M-step when estimating the priors q_j, the overall complexity is $O(N^2V\tau + NV\tau + N^2V\tau\tau') = O(N^2V\tau\tau')$. Finally, if we are not given the pairwise distances $d_{\varphi_v}(\mathbf{x}_i^{(v)}, \mathbf{x}_j^{(v)})$ of each view, their computation usually costs an extra $O(N^2Vd)$ scalar operations, where $d = \max\{d^{(1)}, d^{(2)}, \ldots, d^{(V)}\}$.

3.4 Empirical Evaluation

3.4.1 Experimental Setup

The performance of the multi-view CMM and the weighted multi-view CMM is studied on both synthetic and real data. The real datasets are a collection of academic web pages and a set of images on Internet pages, where multiple views occur naturally.

Our focus is to conduct a comparison between the two proposed methods, to investigate whether assigning different weights to views (weighted multi-view CMM) provides any gains over an unweighted combination of the views (multi-view CMM). Moreover, a single view CMM (Section 3.1) is applied to each of the individual views of the datasets, to examine if multiple views boost the clustering quality. Note that both our algorithms reduce to the single view CMM when only one view is present. Since the easiest way to partition data with multiple representations is to concatenate the views (e.g. by appending the vectors) and then apply a single view algorithm on this concatenation, the single view CMM is also tested using the concatenated view, in order to explore if our multi-view approaches lead to improved performance.

Gaussian CMMs are adopted for all cases and views, i.e. $d_{\varphi_v}(\mathbf{x}_i^{(v)}, \mathbf{x}_j^{(v)}) = \|\mathbf{x}_i^{(v)} - \mathbf{x}_j^{(v)}\|^2$. Remember that for the single view and the multi-view CMM only the pairwise distances are required to calculate the updates (equations (3.5) and (3.9)), as $C_{\varphi_v}(\mathbf{x}_i^{(v)})$ always cancels out, but this does not hold for the weighted multi-view CMM in general.

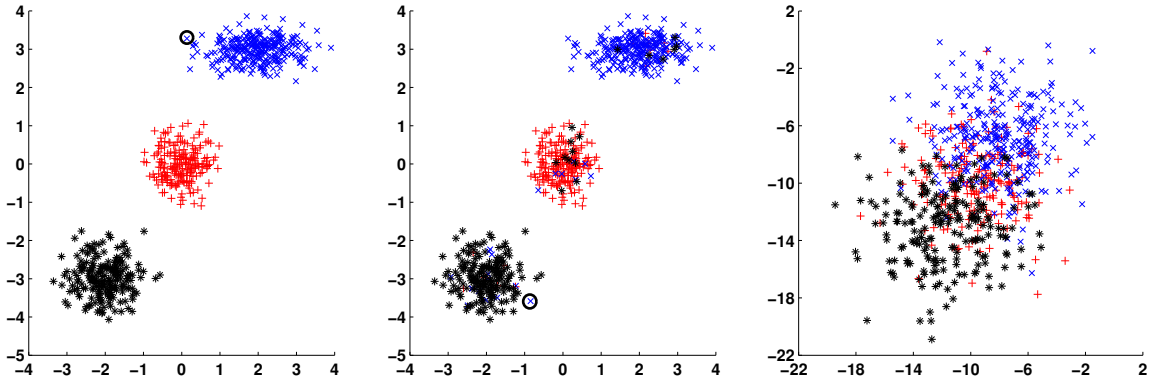
[3] For details on multiple kernel learning see Chapter 1, Section 1.3.

However, for Gaussian components, $C_{\varphi_v}(\mathbf{x}_i^{(v)}) = \left(\frac{\beta^v}{2\pi}\right)^{d^{(v)}/2}$, hence $C_{\varphi_v}(\mathbf{x}_i^{(v)})$ does not depend on the instance values $\mathbf{x}_i^{(v)}$. Therefore, the pairwise distances along with each view's dimensionality $d^{(v)}$, suffice to calculate the updates of the weighted multi-view CMM (see equations (3.14)-(3.16)), without needing the instance values. In our experiments we have removed $C_{\varphi_v}(\mathbf{x}_i^{(v)})$ from the update rule (3.14), as if it is canceling out. This is done as we wish to treat problems where only the pairwise distances are available for each view and not the instances themselves. In such cases, the dimensionality of the views is not known in order to compute $C_{\varphi_v}(\mathbf{x}_i^{(v)})$. Such problems are very common in practice and we would like to test weighted multi-view CMMs under this setting. Moreover, for the weighted multi-view CMM a single execution of the EM algorithm has been always performed using a uniform initialization ($\pi^{v(0)} = 1/V, q_j^{(0)} = 1/N$), since no prior information for the quality of the views exists in any of the datasets. The multi-view CMM is also executed once ($q_j^{(0)} = 1/N$), as the prior updates converge to the global optimum (Section 3.2).

Additionally, in each experiment the partition returned by all the aforementioned clustering methods is used to initialize an execution of the kernel *k*-means algorithm[4] [37, 90]. Such a run is conducted in order to determine whether there is room for improving CMMs results, or they are already close to a very good solution that cannot be further fine tuned. A linear kernel, in order to be consistent with the choice of $d_{\varphi_v}(\mathbf{x}_i^{(v)}, \mathbf{x}_j^{(v)}) = \|\mathbf{x}_i^{(v)} - \mathbf{x}_j^{(v)}\|^2$, is selected for each view and since kernel *k*-means is a single view method, a final kernel is built as a weighted sum of the individual view kernels. Those weights when fine tuning the weighted multi-view CMM are the final π^v values, while for the multi-view CMM they are set equal to $1/V$. For the single view cases no weight is used. Note that for the concatenated case the linear kernel is calculated on the appended view vectors.

To further explore the potential of our two methods, we compare them to a multi-view algorithm from the literature [13], namely correlational spectral clustering (CSC), which is built upon kernel canonical correlation analysis (KCCA) [51]. This approach simultaneously uses all views, which are all thought of as being of the same quality (i.e. no view weights are available), to find appropriate projection directions that maximize the correlation between the projected views and then applies *k*-means to the projections of the views to get a partitioning of the instances. For the experiments with CSC we use the algorithm implementation made available by the authors of [13] and adopt a very similar experimental protocol to [13]. Specifically, the number of projection axes is set equal to the cluster number, a linear kernel is selected (for the same reason as for kernel *k*-means), the KCCA regularization parameters are determined automatically using grid search, and *k*-means is restarted 30 times with random initializations and the run with the smallest *k*-means objective is kept. Note that the grid search steps grow exponentially with the number of views and together with the fact that KCCA requires solving a generalized eigenvalue problem (which is a timely procedure), make

[4]Kernel *k*-means is outlined in Chapter 1, Section 1.1.3.



(a) (b) (c)

Figure 3.1: Examples of the synthetic dataset: (a) the original dataset generated from three 2-d Gaussian distributions; (b) one of the "corrupted" views for $m = 50$ and zero translation. The circled point in (a) (blue class) is wrongly represented here as belonging to the black class (circled point); (c) one of the noisy views.

the application of CSC prohibitive for datasets with many views. Also note, that since it is not clear which view's projections to use to get the final clustering with *k*-means, we cluster each of the available views' projections and report results for the best and worst performing ones.

For each dataset the ground-truth class of every instance is available and the number of clusters is set equal to the true class number, unless stated otherwise. To assess the returned clusters quality the average entropy metric [10, 11, 34], which measures the impurity of the partitions w.r.t. the ground-truth classes, is used. Average entropy is given by (3.19), where N is the dataset size, M the number of clusters, C the number of classes, n_k^h the number of points in cluster k from class h and n_k the size of the k-th cluster. Lower average entropy values indicate that each cluster consists of instances belonging to the same class.

$$H = \sum_{k=1}^{M} \frac{n_k}{N} \left(-\sum_{h=1}^{C} \frac{n_k^h}{n_k} \log \frac{n_k^h}{n_k} \right) \tag{3.19}$$

It must be emphasized that in all tested methods the ground-truth labels have not been used during training. They are used only to compute the performance measures after training.

3.4.2 Synthetic Datasets

The muli-view and weighted multi-view CMMs are first tested on a dataset with 700 instances, generated from three two-dimensional Gaussian distributions (Figure 3.1(a)). Each of the distributions represents a distinct class and this serves as the ground-truth. From this (original) dataset seven artificial views were created. For each of the first five views, as an initial step, all original instances were equally translated and, then, m

of them were randomly selected and replaced by new ones. To replace each of the m instances, we randomly picked a class, different from the one that the instance belongs to in the original dataset, and generated a new point from the corresponding class distribution. Therefore, each view is "corrupted", as according to the ground-truth m points belong to an incorrect class. For the experiments we set $m = 50$ and an example is illustrated in Figure 3.1(b). For the remaining two views, a high amount of zero-mean Gaussian noise was added to the original instances (noise std $= 2.5$), making it hard to separate the classes (Figure 3.1(c)).

Independently clustering any of the five "corrupted" views will probably result in misclassifing all m misplaced instances. We wish to examine if the simultaneous consideration of multiple views helps to "fix" some of these errors. The noisy views contain little useful information for the problem at hand and we want to explore how that fact is reflected by the weights π^v of the weighted multi-view CMM. Note that the original dataset is correctly separated by a CMM, i.e. $H = 0$.

Four noise-free datasets were constructed, including $2, 3, 4$ and 5 "corrupted" views respectively. Also, four noisy datasets were created by adding the two noisy views to the noise-free datasets. Results for the noise-free and noisy datasets are reported in Table 3.1 and Table 3.3, respectively, for three clusters and $\beta^v = \beta_0^v$ (3.11).

From Table 3.1 we see that the multi-view methods (on the rest of this section (Section 3.4.2), when writing multi-view methods we refer to the CMM-based ones and not CSC) always outperform the best single view (apart from one case), indicating that multiple views contribute to the correction of the errors in the individual views. The concatenated view is always inferior or equal to at least one of the multi-view approaches. When no kernel *k*-means fine tuning is used, both multi-view approaches are ahead. Therefore, appending the view vectors is not a good strategy, something widely mentioned in the literature (e.g. [11, 34]).

Moreover, from Table 3.2 we observe that the weighted multi-view CMM roughly assigns the same weights to the views, something expected given that all views are of similar quality, hence it behaves like the multi-view CMM. This observation is in accordance with the results, where the two methods exhibit similar performance. Also, the multi-view schemes take advantage of every available view, since the entropy constantly drops as the number of views increases. Note that kernel *k*-means always degrades the performance of the multi-view methods. Finally, CSC[5] is systematically beaten by a large margin by all CMM-based methods and its performance barely increases as more views become available, highlighting the strength of CMMs.

When noise comes into play, the true potential of the weighted multi-view CMM becomes apparent, since it achieves by far the least entropy in all cases (Table 3.3). This happens because it *assigns very small weights to the noisy views* (as can be seen in Table 3.4), hence they are almost eliminated from the clustering process, while the

[5]We stress that in none of the experiments of this or the following sections we have applied kernel *k*-means to the partitions returned by CSC.

Table 3.1: Results on the noise-free artificial datasets with Gaussian CMM-based methods, in terms of entropy and three clusters ($m = 50$, $\beta^v = \beta_0^v$). The "Yes", "No" columns indicate whether kernel *k*-means fine tuning is applied or not. Results for the CSC approach are also reported.

Method	**2 views**		**3 views**		**4 views**		**5 views**	
	Kernel *k*-means		Kernel *k*-means		Kernel *k*-means		Kernel *k*-means	
	No	Yes	No	Yes	No	Yes	No	Yes
Worst single view CMM	0.300	0.300	0.572	0.300	0.572	0.303	0.572	0.303
Best single view CMM	0.299	**0.299**	0.299	0.299	0.299	0.299	0.299	0.299
Concatenated view CMM	0.322	0.320	0.265	**0.262**	0.147	**0.191**	0.130	**0.124**
Multi-view CMM	**0.289**	0.320	**0.133**	**0.262**	0.097	0.195	0.081	**0.124**
Weighted multi-view CMM	**0.289**	0.326	0.176	0.266	**0.086**	**0.191**	**0.060**	**0.124**
CSC - worst view	0.745		0.766		0.766		0.766	
CSC - best view	0.743		0.741		0.739		0.735	

Table 3.2: Indicative weights assigned to the views by the weighted multi-view CMM for the noise-free artificial datasets.

	2 views	**3 views**	**4 views**	**5 views**
View 1	0.502	0.336	0.251	0.201
View 2	0.498	0.333	0.249	0.199
View 3	-	0.331	0.248	0.198
View 4	-	-	0.252	0.201
View 5	-	-	-	0.201

noise-free views are equally treated. Therefore, this method works as if the noise does not exist. Indeed, note that its performance is relatively close to that of the noise-free setting. The advantages of automatically determining the view weights are now clearly exposed, as the weighted multi-view CMM exhibits robustness to noise and to non-informative views in general.

In contrast, the multi-view CMM splits are greatly affected by the noise, due to the equally weighted views, and are considerably inferior to the corresponding noise-free ones and to those of the best single view (which is a noise-free view). CSC is also affected by the presence of noise, as the entropy has increased compared to that reported in Table 3.1, and is largely outperformed by the weighted multi-view CMM, demonstrating the need for methods that distinguish noisy views. Once again, CSC is beaten by the multi-view CMM and the best single view. Additionally, the computational time concerns regarding CSC (see Section 3.4.1) became evident when handling more than four views, when it took several hours to find the clusters, while our two frameworks required a few minutes. Finally, similar conclusions as above can be drawn regarding the concatenated view and the application of kernel *k*-means.

To further investigate the behavior of the weighted multi-view CMM, we studied the sensitivity of the weights on the noise level present on the views. Analytically, we

Table 3.3: Results on the noisy artificial datasets with Gaussian CMM-based methods, in terms of entropy and three clusters ($m = 50$, $\beta^v = \beta_0^v$). The "Yes", "No" columns indicate whether kernel *k*-means fine tuning is applied or not. Results for the CSC approach are also reported.

Method	**2 views + noisy views**		**3 views + noisy views**		**4 views + noisy views**		**5 views + noisy views**	
	Kernel *k*-means		Kernel *k*-means		Kernel *k*-means		Kernel *k*-means	
	No	Yes	No	Yes	No	Yes	No	Yes
Worst single view CMM	0.935	0.943	0.935	0.943	0.935	0.943	0.935	0.943
Best single view CMM	0.299	0.299	0.299	0.299	0.299	0.299	0.299	0.299
Concatenated view CMM	0.960	0.748	0.477	0.631	0.472	0.496	0.377	0.402
Multi-view CMM	0.385	0.751	0.655	0.631	0.147	0.478	0.576	0.601
Weighted multi-view CMM	**0.256**	**0.281**	**0.220**	**0.242**	**0.127**	**0.206**	**0.116**	**0.157**
CSC - worst view	1.001		1.078		1.078		1.078	
CSC - best view	0.833		0.814		0.800		0.745	

Table 3.4: Indicative weights assigned to the views by the weighted multi-view CMM for the noisy artificial datasets.

	2 views + noisy views	**3 views + noisy views**	**4 views + noisy views**	**5 views + noisy views**
View 1	0.442	0.304	0.224	0.182
View 2	0.435	0.300	0.223	0.181
View 3	-	0.299	0.222	0.180
View 4	-	-	0.226	0.184
View 5	-	-	-	0.183
Noisy view 1	0.064	0.051	0.054	0.046
Noisy view 2	0.059	0.046	0.051	0.044

created several datasets for various amounts of noise that each consisted of two noise-free views (for all cases these are the ones used for the two-view experiment above) and two noisy views with the same amount of noise. To construct the noisy views, random zero-mean Gaussian noise with a different standard deviation for each noise level was added to the original dataset. In order to alleviate randomness in our experiments, for each noise level we created ten datasets and repeated the clustering. The average and standard deviation of the weight values, corresponding to the four views, over the ten runs are depicted in Figure 3.2. It can be seen that, as the noise increases, the differences between the weights of the noise-free and the noisy views become greater, which is something that we would naturally expect. Note that for noise std $= 0.3$ the weight values of the noisy views approach those of the noise-free ones as a low amount of noise is present, while for noise std $= 2.5$ they attain very small values.

Finally, a dataset which combines views with two different noise levels (two views for each noise level) and two noise-free views was constructed. The average and standard deviation of the weights over ten trials are shown on the right-most corner of Figure 3.2.

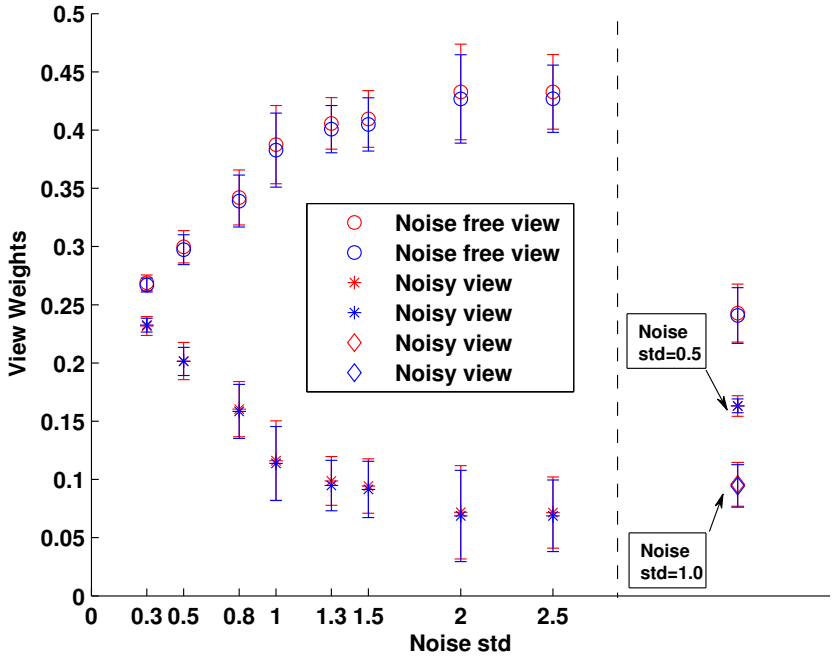


Figure 3.2: View weights (average and std over 10 trials) assigned by the weighted multi-view CMM to datasets consisting of two noise-free and two noisy views (both with the same amount of noise), for various noise levels. On the right, the weight averages and std for a dataset where views with different amounts of noise simultaneously exist, are shown.

We observe, that the less informative a view is, the smaller its weight. Also, views with the same amount of noise are assigned very similar weights. From Figure 3.2, we can conclude that the weighted multi-view CMM identifies noisy views and treats them according to their noise level.

3.4.3 WebKB Dataset

The WebKB dataset is a popular collection for testing multi-view algorithms [10, 11, 14, 34, 80], made up of web pages related to the computer science departments of various universities. Here the version described in [11] is used, consisting of six classes (course, department, faculty, project, staff and student) and two views. The views are the text of the pages and the anchor text of all inbound links. As all web pages do not have inbound links, such instances were removed from the dataset, resulting in 2076 instances with both views available.

Term frequency inverse document frequency (tfidf) vectors were constructed for each view and normalized to unit length, so that the squared Euclidean distances of the Gaussian CMMs reflect the cosine similarity, which is usually employed to document clustering. The number of clusters was always set to six. Experiments with the CMM-based methods were performed for $\beta^v = \beta_0^v$ (3.11). We also considered other β^v values for each tested method, by setting $\beta^v = \alpha\beta_0^v$ and repeating the clustering for several α

Table 3.5: WebKB results with Gaussian CMM-based methods for $\beta^v = \beta_0^v$ and $\beta^v = \alpha^* \beta_0^v$, in terms of entropy and six clusters. The "Yes", "No" columns indicate whether kernel *k*-means fine tuning is applied or not. Results for the CSC approach are also reported.

Method	$\boldsymbol{\beta^v = \beta_0^v}$		$\boldsymbol{\beta^v = \alpha^* \beta_0^v}$	
	Kernel *k*-means		Kernel *k*-means	
	No	Yes	No	Yes
Single view CMM - text	1.536	1.513	1.485 ($\alpha^* = 1.5$)	1.492 ($\alpha^* = 1.5$)
Single view CMM - anchor text	1.554	1.471	1.440 ($\alpha^* = 3.5$)	1.329 ($\alpha^* = 3.5$)
Concatenated view CMM	1.559	1.537	1.481 ($\alpha^* = 1.7$)	1.490 ($\alpha^* = 1.7$)
Multi-view CMM	1.498	1.450	1.396 ($\alpha^* = 1.5$)	1.316 ($\alpha^* = 1.5$)
Weighted multi-view CMM	1.431	1.427	**1.299** ($\alpha^* = 3.5$)	**1.307** ($\alpha^* = 3.5$)
CSC - text	1.411			
CSC - anchor text	1.309			

Table 3.6: Indicative weights assigned to the views by the weighted multi-view CMM ($\beta^v = \beta_0^v$) for the WebKB dataset.

Text view	0.126
Anchor text view	0.874

values. The value of α that yields the least entropy (denoted as α^*) was selected as the best solution and its results are reported here. This was done in order to show that results can be possibly improved by trying β^v values around β_0^v. Note that a common value α was used for all views in the multi-view algorithms. When applying kernel *k*-means, the β^v values already picked by the CMMs were retained. Obviously, for CSC no β^v parameter to fine tune exists.

From Table 3.5 it is apparent that the weighted multi-view CMM is only beaten by CSC and only when $\beta^v = \beta_0^v$. It is superior though when β^v is fine tuned, demonstrating that gains in performance are possible by searching around β_0^v (such gains are observed for all CMM-based methods). The multi-view CMM is constantly overcome by both its weighted counterpart and CSC. Comparing the two proposed approaches, the gap in performance mainly emanates from the different view weights. Table 3.6 contains the weights returned by the weighted multi-view CMM when $\beta^v = \beta_0^v$, where a higher value is given to the anchor text view. Note that the weighted multi-view CMM achieves its best entropy ($H = 1.299$) for the optimum β^v and no kernel *k*-means post-processing (fourth column). This is lower than the multi-view CMM best ($H = 1.316$), which is achieved for the optimum β^v with kernel *k*-means post-processing (fifth column). This indicates that for the WebKB dataset our weighted algorithm provides higher gains, without needing fine tuning of the returned clusters. The multi-view approaches are always ahead of the single views and the concatenated view, demonstrating once again the advantages of incorporating multiple views to the clustering task and the inefficiency of naive vector merging. The concatenated view in most cases is even worse than the

Table 3.7: IntAd results with Gaussian CMM-based methods for $\beta^v = \beta_0^v$, in terms of entropy and different number of clusters. The "Yes", "No" columns indicate whether kernel *k*-means fine tuning is applied or not. Results for the CSC approach are also reported.

Method	**2 clusters**		**4 clusters**		**6 clusters**	
	Kernel *k*-means		Kernel *k*-means		Kernel *k*-means	
	No	Yes	No	Yes	No	Yes
Worst single view CMM	0.517	0.497	0.513	0.474	0.478	0.460
Best single view CMM	**0.366**	0.382	**0.316**	0.314	0.366	0.344
Concatenated view CMM	0.496	0.496	0.489	0.489	0.472	0.474
Multi-view CMM	0.481	0.362	0.393	**0.284**	0.393	0.290
Weighted multi-view CMM	0.468	**0.349**	0.403	0.347	0.404	**0.267**
CSC - worst view	0.517		0.425		0.389	
CSC - best view	0.459		0.386		0.343	

single views. Finally, for $\beta^v = \beta_0^v$ kernel *k*-means improves the results, while for the best β^v it does so in half of the cases.

3.4.4 Internet Advertisements Dataset

The Internet advertisements dataset (IntAd) [42] contains images from various web pages that are characterized either as advertisements or non-advertisements (i.e. the ground-truth consists of two classes). The instances are described in terms of six views, which are the geometry of the images (width, height, aspect ratio), the phrases in the url of the pages containing the images (base url), the phrases of the images' url (image url), the phrases in the url of the pages the images are pointing at (target url), the anchor text and the text of the images' alt (alternative) html tags (alt text). All views have binary features, apart from the geometry view whose features are continuous. Details for the construction of the dataset can be found in [67]. Note that there are several missing views in this dataset. Specifically, the anchor text view is missing for 94% of the images and the geometry view for 30%, therefore we decided not to include those views in our empirical evaluation. After removing the instances that were missing any of the four remaining views, 2369 images were retained for the experiments.

Similarly to WebKB, we generated normalized tfidf vectors to reflect the cosine similarity and performed experiments both for $\beta^v = \beta_0^v$ (3.11) and $\beta^v = \alpha^* \beta_0^v$. Different cluster numbers were tried, specifically two, four and six.

The results in Tables 3.7-3.8[6] show that one of the two proposed methods achieves the least entropy in most cases (in eight out of 12). In the other cases, the best single view (two times for $\beta^v = \beta_0^v$ and one for $\beta^v = \alpha^* \beta_0^v$) and CSC (only once, for six clusters and $\beta^v = \beta_0^v$) are superior. Note that our two multi-view approaches are always ahead

[6]The values reported for CSC are the same in both tables, since for CSC no β^v parameter to fine tune exists.

Table 3.8: IntAd results with Gaussian CMM-based methods for $\beta^v = \alpha^* \beta_0^v$, in terms of entropy and different number of clusters. The "Yes", "No" columns indicate whether kernel *k*-means fine tuning is applied or not. Results for the CSC approach are also reported.

Method	**2 clusters**		**4 clusters**		**6 clusters**	
	Kernel *k*-means		Kernel *k*-means		Kernel *k*-means	
	No	Yes	No	Yes	No	Yes
Worst single view CMM	0.517 ($\alpha^* = 1$)	0.497 ($\alpha^* = 1$)	0.472 ($\alpha^* = 1$)	0.460 ($\alpha^* = 1$)	0.450 ($\alpha^* = 1.2$)	0.422 ($\alpha^* = 1.2$)
Best single view CMM	0.366 ($\alpha^* = 1$)	0.382 ($\alpha^* = 1$)	**0.316** ($\alpha^* = 1$)	0.314 ($\alpha^* = 1$)	0.353 ($\alpha^* = 3$)	0.318 ($\alpha^* = 3$)
Concatenated view CMM	0.462 ($\alpha^* = 2$)	0.456 ($\alpha^* = 2$)	0.391 ($\alpha^* = 0.5$)	0.356 ($\alpha^* = 0.5$)	0.402 ($\alpha^* = 0.5$)	0.366 ($\alpha^* = 0.5$)
Multi-view CMM	0.386 ($\alpha^* = 1.5$)	**0.288** ($\alpha^* = 1.5$)	0.362 ($\alpha^* = 1.2$)	0.324 ($\alpha^* = 1.2$)	0.339 ($\alpha^* = 1.2$)	0.283 ($\alpha^* = 1.2$)
Weighted multi-view CMM	**0.337** ($\alpha^* = 3.5$)	0.299 ($\alpha^* = 3.5$)	0.357 ($\alpha^* = 3$)	**0.295** ($\alpha^* = 3$)	**0.331** ($\alpha^* = 1.2$)	**0.271** ($\alpha^* = 1.2$)
CSC - worst view	0.517		0.425		0.389	
CSC - best view	0.459		0.386		0.343	

Table 3.9: Indicative weights assigned to the views by the weighted multi-view CMM ($\beta^v = \beta_0^v$) for the IntAd dataset.

	2 clusters	**4 clusters**	**6 clusters**
Image url view	0.047	0.047	0.047
Base url view	0.237	0.237	0.237
Target url view	0.355	0.355	0.355
Alt text view	0.361	0.361	0.361

of the worst single view. Therefore, for the IntAd dataset, if we test the views one by one, we sometimes get a better partitioning than simultaneously using all of them, particularly for $\beta^v = \beta_0^v$. The two aforementioned multiple view algorithms though, provide higher quality solutions more systematically, especially if the β^v values are fine tuned, since in Table 3.8 the best single view is ahead for only one setting. It is important to stress that CSC is inferior to both the weighted multi-view CMM and the multi-view CMM when $\beta^v = \alpha^* \beta_0^v$ and also for $\beta^v = \beta_0^v$ when kernel *k*-means is applied. Moreover, it is worse than the best single view CMM most of the times (in nine out of 12). This result shows that CMMs are a powerful clustering technique and support our decision to adopt them for multi-view learning.

Once again the concatenated view is always inferior to the multi-view schemes (only CSC performs worse in a few cases) and also to the best single view (sometimes even to the worst single view). The weighted multi-view CMM is superior to the multi-view CMM for eight out of 12 cases and is the best overall performer six times out of 12. The difference among the two methods is greater in Table 3.8, where the weighted multi-

view CMM is ahead for all but one case. As an indication of how our weighted algorithm handles this dataset, its view weights for the run with $\beta^v = \beta_0^v$ are given in Table 3.9. Note that the weight values are identical for the different cluster numbers, since they do not depend on this parameter. Finally, the application of kernel *k*-means seems to be beneficial for all cases.

3.5 Summary

In this chapter, we have presented two multi-view approaches that identify exemplars in the dataset by simultaneously exploiting all available representations (views) of the instances and are built upon the convex mixture models (CMMs) proposed in [71]. The first, called multi-view CMM, considers all views as being equally important (i.e. of equal quality) and is characterized by convexity, thus the global optimum solution can be uncovered, as well as by the ability to handle views with different statistical properties. The second, called weighted multi-view CMM, assigns different weights to the views which are automatically determined during training, providing robustness against noisy or low quality views. This method can be interpreted as a mixture model whose components are CMMs (one for each view) and, like the multi-view CMM, takes into account the different statistical properties of the views (convexity, however, is lost in this case). Both our algorithms are computationally efficient and involve simple iterative updates of the parameters during optimization, which for the weighted multi-view CMM are executed using the well known EM procedure.

The two multi-view frameworks have been tested on several diverse datasets and compared to the single view CMM [71] (applied to each individual view and the concatenated view), as well as to the correlational spectral clustering method (CSC) of [13]. In general, the results verify the superiority of our approaches, with the weighted multi-view CMM emerging as the best method. Its performance is constantly the best for the noisy versions of the synthetic datasets. When no noise is present in the synthetic data, it is only matched by the multi-view CMM. This is expected, as all views are approximately of the same quality and therefore the impact of weights is minimal. Also, experiments with varying amount of noise on the views, showed that the weights assigned to them are in direct association with their noise level. For the real datasets, our methods are ahead for most of the cases, especially for $\beta^v = \alpha^* \beta_0^v$, providing high quality solutions more systematically than the compared clustering schemes.

Overall, the experiments have shown that multiple views are beneficial in identifying good partitions, particularly if the views participate with different weights. Moreover, it has been demonstrated that the concatenation of the representations is not an effective multi-view strategy and that the success of fine tuning the solutions of the CMM-based methods with kernel *k*-means is dataset dependent.

Chapter 4

Kernel-based Weighted Multi-view Clustering

Multi-modal datasets are very common in practice due to the use of different measuring methods (e.g. infrared and visual cameras), or of different media, like text, video and audio. Each instance in these datasets has multiple representations, called *views*, from various feature spaces. Typical examples include web pages, represented by both text and hyperlinks, and images, where color and texture information can be utilized. The existence of such data has raised interest in the so called *multi-view learning*[1], which has been extensively studied under the semi-supervised classification setting [14, 17, 33, 84, 125]. This chapter focuses on *multi-view clustering* [10, 13, 50, 65, 77], where the absence of a ground-truth to guide the learning process makes the underlining task much harder. The main challenge that arises is to find a suitable way of simultaneously exploiting the, possibly, complementary information of all available views in order to derive a robust partitioning, considering the diversity (e.g. different statistical properties) and the disagreement (i.e. different views produce different partitionings) of the views.

Surprisingly, most multi-view methods rely equally on every view, something that may lead to performance degradation in the case of degenerate views (e.g. noisy or irrelevant views). Identifying and appropriately handling such views is difficult though. The approach presented in this chapter [107] tackles this problem from the kernel

[1] For details on multi-view learning see Chapter 1, Section 1.2.

perspective, i.e. data points are mapped to a nonlinear high-dimensional space through a kernel function [41]. Each view is represented by a kernel matrix and views are combined using a weighted sum of the kernel matrices, accompanied by an appropriate constraint on the weights. *The weights express the quality* (*importance in clustering*) *of the views* and determine their *degree of contribution* to the final solution accordingly. They are *learned automatically*, along with the inference of the cluster labels, through *closed-form expressions*, by minimizing the typical intra-cluster variance objective of k-means in the space induced by combining the individual kernels. Two iterative optimization strategies are developed, one based on kernel k-means [37, 90] and the other on spectral techniques [37].

Our strategy of mixing the kernels is inspired by *multiple kernel learning*[2] [62, 88, 110, 126, 130], where for a singly represented dataset a, usually linear, combination of basis kernels is sought along with the partitioning, to solve the kernel selection problem. In our case those kernels are derived from the views. Thus a connection between these popular machine learning problems emerges. There appears to be some dispute over the sparsity[3] of the combination weights, with some authors favoring high sparsity [88, 110, 126] and others a more uniform solution [62, 130]. We believe that a good choice lies somewhere between the two ends, such that an algorithm is flexible enough to allow the data to harness the kernel weights, without being too prone to either end. For this reason, the proposed methodology *incorporates a parameter controlling this flexibility that must be specified prior to execution*. Experiments on synthetic and real world datasets support the above claim and indicate that view weighting under our framework is successful in reflecting the underlying properties of the studied data. The main contributions of the presented approach can be summarized in:

1) The estimation of view weights, a subject generally overlooked in multi-view clustering.

2) The inclusion of a parameter that controls the sparsity of the weights.

3) The use of kernels to represent the views and the way they are combined, which connects multi-view clustering to multiple kernel learning.

The rest of this chapter is organized as follows. Section 4.1 presents the foundations of our multi-view method, which is detailed in Section 4.2. The experiments follow in Section 4.3, before the concluding remarks of Section 4.4.

4.1 Kernel-based Clustering

Two kernel-oriented methods for optimizing the intra-cluster variance are described in this section, which are both considered under our framework.

[2]For details on multiple kernel learning see Chapter 1, Section 1.3.

[3]Sparsity is defined relative to the number of kernels in the solution that carry significant weights.

4.1.1 Kernel *k*-means

Kernel *k*-means [37, 90] is a generalization of the standard *k*-means algorithm where the dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x}_i \in \Re^d$ is mapped from input space to a higher dimensional reproducing kernel Hilbert space $\mathcal{H}$, a.k.a. feature space, via a nonlinear transformation $\phi : \mathcal{X} \to \mathcal{H}$.

To partition dataset $\mathcal{X}$ into M disjoint clusters, $\{\mathcal{C}_k\}_{k=1}^M$, the intra-cluster variance in feature space (4.1) is minimized over clusters $\{\mathcal{C}_k\}_{k=1}^M$, where $\mathbf{m}_k$ is the k-th cluster center and δ_{ik} is an indicator variable with $\delta_{ik} = 1$ if $\mathbf{x}_i \in \mathcal{C}_k$ and $\delta_{ik} = 0$ otherwise.

$$\mathcal{E}_{\mathcal{H}} = \sum_{i=1}^{N} \sum_{k=1}^{M} \delta_{ik} \|\phi(\mathbf{x}_i) - \mathbf{m}_k\|^2, \ \mathbf{m}_k = \frac{\sum_{i=1}^N \delta_{ik} \phi(\mathbf{x}_i)}{\sum_{i=1}^N \delta_{ik}} \tag{4.1}$$

A kernel function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \Re$ [41] is employed to get the inner products in feature space without explicitly defining transformation ϕ (usually the corresponding transformation is intractable). Using the kernel function the kernel matrix $K \in \Re^{N \times N}$, $K_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$, can be computed, which is the most common way of representing a dataset in feature space. Note that, although the centers $\mathbf{m}_k$ cannot be analytically calculated, since ϕ is unknown, the squared Euclidean distances in (4.1) can be estimated based only on the kernel matrix entries:

$$\|\phi(\mathbf{x}_i) - \mathbf{m}_k\|^2 = K_{ii} - \frac{2 \sum_{j=1}^N \delta_{jk} K_{ij}}{\sum_{j=1}^N \delta_{jk}} + \frac{\sum_{j=1}^N \sum_{l=1}^N \delta_{jk} \delta_{lk} K_{jl}}{\sum_{j=1}^N \sum_{l=1}^N \delta_{jk} \delta_{lk}} , \tag{4.2}$$

which suffices to cluster the instances. More details on kernel *k*-means can be found in Chapter 1, Section 1.1.3.

4.1.2 Spectral Clustering

According to [37], the intra-cluster variance (4.1) can be equivalently posed as a trace difference:

$$\mathcal{E}_{\mathcal{H}} = tr(K) - tr(Y^\top K Y), \ \ Y \in \Re^{N \times M}, \ Y_{ik} = \frac{\delta_{ik}}{\sqrt{\sum_{j=1}^N \delta_{jk}}} . \tag{4.3}$$

The first term on the above equation is a constant, therefore the minimization of (4.3) is equivalent to the maximization of $tr(Y^\top K Y)$ w.r.t. the indicator matrix Y. Due to the discrete nature of Y this becomes a hard optimization problem, but if Y is relaxed to be an arbitrary orthonormal matrix (i.e. $Y^\top Y = I$), a standard result in linear algebra states that the optimal Y is composed of the top M eigenvectors of the kernel matrix K. Therefore, spectral methods (see Chapter 1, Section 1.1.2) which calculate the top eigenvectors of an appropriate matrix and then perform post-processing on these eigenvectors to recover a partitioning can substitute kernel *k*-means. A popular spectral technique is that of [82].

4.2 Multi-view Kernel *k*-means and Multi-view Spectral Clustering

Motivated by the absence of multi-view clustering methods that differentiate the contribution of the views according to the conveyed information, we present a simple and effective kernel-based scheme which embeds in the clustering process an automatic "ranking" of the views. This "ranking" should wipe out a completely uninformative view, but also allow a less informative one to contribute, with a smaller degree, to the clustering solution.

4.2.1 Model Description

Consider a dataset $\mathcal{X}$ with N instances and V views: $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$, where $\boldsymbol{x}_i = \left\{\mathbf{x}_i^{(v)}\right\}_{v=1}^{V}$ and $\mathbf{x}_i^{(v)} \in \Re^{d^{(v)}}$ are the view vectors for instance $\boldsymbol{x}_i$. As already discussed in Section 4.1, to apply kernel methods, the dataset is implicitly mapped to a feature space and is represented through a kernel matrix. Here it is assumed that V kernel matrices are available, $\left\{K^{(v)}\right\}_{v=1}^{V}$, to which (unknown) transformations $\left\{\phi^{(v)}\right\}_{v=1}^{V}$ and feature spaces $\left\{\mathcal{H}^{(v)}\right\}_{v=1}^{V}$ correspond. To take advantage of all views, we propose the following kernel combination, where w_v are the view weights and p is an exponent:

$$\widetilde{K} = \sum_{v=1}^{V} w_v^p K^{(v)},\ w_v \geq 0,\ \sum_{v=1}^{V} w_v = 1,\ p \geq 1. \tag{4.4}$$

It is easy to verify that the composite matrix $\widetilde{K}$ is a valid kernel matrix, i.e. a positive semidefinite matrix, to which a transformation $\widetilde{\phi}(\boldsymbol{x}_i) = \left[\sqrt{w_1^p}\phi^{(1)}(\mathbf{x}_i^{(1)})^\top, \ldots, \sqrt{w_V^p}\phi^{(V)}(\mathbf{x}_i^{(V)})^\top\right]^\top$ corresponds, i.e. $\widetilde{K}_{ij} = \widetilde{\phi}(\boldsymbol{x}_i)^\top\widetilde{\phi}(\boldsymbol{x}_j)$, that maps the instances to feature space $\widetilde{\mathcal{H}} = \mathcal{H}^{(1)} \times \ldots \times \mathcal{H}^{(V)}$. The weight values, w_v, of the combination (the w_v^p values to be precise) *represent the relevance* of each kernel (view) to the clustering task.

This technique of kernel mixing is widespread in multiple kernel learning, where usually the p-norm constraint is applied, i.e. $\widetilde{K} = \sum_{v=1}^{V} w_v K^{(v)}, w_v \geq 0, \sum_{v=1}^{V} w_v^p \leq 1, p \geq 1$. Different norms allow for different levels of sparsity on the weights, with the 1-norm [88, 110, 126] favoring very sparse weights and the ∞-norm [62] reducing to the unweighted case, i.e. $\widetilde{K} = \sum_{v=1}^{V} K^{(v)}$. Norms for $p > 1$ provide a tradeoff between these two extremes [62, 130]. We shall shortly discuss how the exponent p in the above kernel mixture (4.4) affects sparsity likewise. However, it must be clarified that in this chapter we do not focus, by any means, on kernel learning, but exploit kernels as a tool for representing and combining views in multi-view learning.

In order to partition the dataset into M disjoint clusters, $\{\mathcal{C}_k\}_{k=1}^{M}$, and simultaneously exploit all views by learning a suitable kernel $\widetilde{K}$ of the form (4.4), the intra-cluster variance in space $\widetilde{\mathcal{H}}$ (4.5) is minimized over the clusters and the weights, w.r.t. the con-

straints in (4.6). Note that we do not optimize w.r.t. p, which must be fixed a priori.

$$\mathcal{E}_{\widetilde{\mathcal{H}}} = \sum_{i=1}^{N}\sum_{k=1}^{M} \delta_{ik} \|\widetilde{\phi}(\boldsymbol{x}_i) - \widetilde{\mathbf{m}}_k\|^2,\ \widetilde{\mathbf{m}}_k = \frac{\sum_{i=1}^{N} \delta_{ik}\widetilde{\phi}(\boldsymbol{x}_i)}{\sum_{i=1}^{N}\delta_{ik}} \tag{4.5}$$

$$\min_{\{\mathcal{C}_k\}_{k=1}^{M},\{w_v\}_{v=1}^{V}} \mathcal{E}_{\widetilde{\mathcal{H}}},\ s.t.\ \ w_v \geq 0,\ \sum_{v=1}^{V} w_v = 1,\ p \geq 1 \tag{4.6}$$

Using (4.2) and (4.4) the objective is rewritten as:

$$\begin{aligned}\mathcal{E}_{\widetilde{\mathcal{H}}} &= \sum_{i=1}^{N}\sum_{k=1}^{M} \delta_{ik} \left(\widetilde{K}_{ii} - \frac{2\sum_{j=1}^{N}\delta_{jk}\widetilde{K}_{ij}}{\sum_{j=1}^{N}\delta_{jk}} + \frac{\sum_{j=1}^{N}\sum_{l=1}^{N}\delta_{jk}\delta_{lk}\widetilde{K}_{jl}}{\sum_{j=1}^{N}\sum_{l=1}^{N}\delta_{jk}\delta_{lk}} \right) \\ &= \sum_{v=1}^{V} w_v^p \sum_{i=1}^{N}\sum_{k=1}^{M} \delta_{ik} \left(K_{ii}^{(v)} - \frac{2\sum_{j=1}^{N}\delta_{jk}K_{ij}^{(v)}}{\sum_{j=1}^{N}\delta_{jk}} + \frac{\sum_{j=1}^{N}\sum_{l=1}^{N}\delta_{jk}\delta_{lk}K_{jl}^{(v)}}{\sum_{j=1}^{N}\sum_{l=1}^{N}\delta_{jk}\delta_{lk}} \right) \Rightarrow\end{aligned}$$

$$\mathcal{E}_{\widetilde{\mathcal{H}}} = \sum_{v=1}^{V} w_v^p \sum_{i=1}^{N}\sum_{k=1}^{M} \delta_{ik} \|\phi^{(v)}(\mathbf{x}_i^{(v)}) - \mathbf{m}_k^{(v)}\|^2,\ \mathbf{m}_k^{(v)} = \frac{\sum_{i=1}^{N}\delta_{ik}\phi^{(v)}(\mathbf{x}_i^{(v)})}{\sum_{i=1}^{N}\delta_{ik}}\ . \tag{4.7}$$

Under the spectral perspective, (4.5) can also be stated in terms of matrix traces, where Y is defined as in (4.3):

$$\mathcal{E}_{\widetilde{\mathcal{H}}} = tr(\widetilde{K}) - tr(Y^\top \widetilde{K} Y) = \sum_{v=1}^{V} w_v^p \left(tr(K^{(v)}) - tr(Y^\top K^{(v)} Y) \right). \tag{4.8}$$

From (4.7) and (4.8) it is obvious that the intra-cluster variance in feature space $\widetilde{\mathcal{H}}$ is the weighted sum of the intra-cluster variances of the individual views' feature spaces, $\mathcal{H}^{(v)}$, *under a common clustering*. Minimizing the view disagreement is the basic principle over which multi-view approaches are built [10].

4.2.2 Model Training

Two iterative algorithms that in each iteration alternate between updating the clusters and reestimating the weights are proposed. One follows the distance-based formulation of $\mathcal{E}_{\widetilde{\mathcal{H}}}$ (4.5) and the other the trace-based spectral formulation (4.8). They are called multi-view kernel *k*-means (MVKKM) and multi-view spectral clustering (MVSpec), respectively.

Updating the clusters for given weights - MVKKM algorithm

When the weights w_v are known, the cluster assignments that minimize the intra-cluster variance can be found in the same way as when only a single kernel is available. The composite kernel, $\widetilde{K} = \sum_{v=1}^{V} w_v^p K^{(v)}$, is first calculated and then kernel *k*-means is applied in space $\widetilde{\mathcal{H}}$. Note that kernel *k*-means requires an initial set of clusters as input. The partitioning returned by the previous MVKKM iteration is used for initializing kernel *k*-means for the current iteration.

Updating the clusters for given weights - MVSpec algorithm

Like MVKKM, the composite kernel is first calculated and then the relaxed version of (4.8) (i.e. Y is allowed to be an arbitrary orthonormal matrix) is considered to compute Y. The optimal solution is composed of the M largest eigenvectors of $\widetilde{K}$, according to the discussion in Section 4.1.2. Note that Y should not be discretized during the iterative process. Otherwise, the monotonic convergence of MVSpec cannot be guaranteed.

Updating the weights for given clusters - MVKKM algorithm

For ease of computation, the form of the objective described in (4.7) is considered together with the constraints from (4.6). It is easy to verify that the constrained objective is convex w.r.t. the weights when $p > 1$, hence their optimal values that minimize $\mathcal{E}_{\widetilde{\mathcal{H}}}$ for the current partitioning can be determined. After some manipulation the following closed-form solution emerges (the analytical proof is provided in Appendix B):

$$w_v = 1/\sum_{v'=1}^{V}\left(\frac{\mathcal{D}_v}{\mathcal{D}_{v'}}\right)^{\frac{1}{p-1}} \text{ if } p>1, \text{ where } \mathcal{D}_v = \sum_{i=1}^{N}\sum_{k=1}^{M}\delta_{ik}\|\phi^{(v)}(\mathbf{x}_i^{(v)}) - \mathbf{m}_k^{(v)}\|^2. \tag{4.9}$$

For $p = 1$ the optimization problem (4.6) becomes a linear program. Its solutions lie on the corners of the simplex in the positive orthant spanned by the constraints, which results in a completely sparse outcome:

$$w_v = \begin{cases} 1, & v = \text{argmin}_{v'}\,\mathcal{D}_{v'} \\ 0, & \text{otherwise} \end{cases} \quad \text{if } p = 1. \tag{4.10}$$

Updating the weights for given clusters - MVSpec algorithm

We follow an analogous procedure to that of MVKKM with the only difference being that the relaxed formulation of (4.8) is used instead of (4.7). All the above remarks regarding the convexity of the objective and the optimality of the weights carry over to MVSpec. Thus, if $p > 1$ the weights are updated through (4.9), while if $p = 1$ through (4.10), where now $\mathcal{D}_v = tr(K^{(v)}) - tr(Y^\top K^{(v)} Y)$.

Initialization and post-processing

In order to apply both algorithms, initial values for the view weights are required. A uniform weighting ($w_v = 1/V$) of the kernels can be used, which is a reasonable choice, unless prior knowledge regarding the quality of the views is available. Additionally, MVKKM requires an initial set of clusters. To locate a meaningful initial partitioning before executing MVKKM, which is very important in avoiding poor minima during the subsequent iterations, the global kernel *k*-means algorithm [105] (see Chapter 1, Section 1.1.4) is applied that yields near-optimal solutions in a deterministic-incremental fashion. Finally, in the MVSpec method, the eigenvectors are discretized after convergence using *k*-means, as in [82], to get the disjoint clusters.

4.2.3 Discussion

In this section, some aspects of the proposed methods are analyzed, starting with the effect of the p exponent. As can be seen from (4.9), the less the intra-cluster variance $\mathcal{D}_v$ of a view the larger its weight. For $p = 1$ a completely sparse solution emerges (4.10), regardless of the relative differences in $\mathcal{D}_v$ among the views. Hence, $p = 1$ may discard useful views and thus is effective when a single view is of good quality. For $p > 1$ it is easy to see (4.9) that the greater (smaller) the p value the less (more) sparse the weights w_v become, i.e. the relative differences in $\mathcal{D}_v$ among the views are suppressed (enhanced). Therefore, a very large p value is useful when kernels of similar quality are available. In practice, intermediate p values are a more reasonable choice, since the most common scenario is that views with complementary information and also degenerate ones exist for the same problem. The above remarks also hold for the w_v^p values, which are the actual coefficients used to combine the kernels (4.4). Hence, as p increases the w_v^p values become more uniform.

To demonstrate the above a bit more formally, the ratio between any two weights, $w_v/w_{v'}$, can be considered as an indicator for the sparsity of the solution. The more this ratio tends to 1 the less sparse the outcome. Assume a fixed clustering, i.e. a fixed $\mathcal{D}_v$ and $\mathcal{D}_{v'}$. From (4.9), $\frac{w_v}{w_{v'}} = \left(\frac{\mathcal{D}_{v'}}{\mathcal{D}_v}\right)^{\frac{1}{p-1}}$ and $\frac{w_v^p}{w_{v'}^p} = \left(\frac{\mathcal{D}_{v'}}{\mathcal{D}_v}\right)^{\frac{p}{p-1}}$, $p > 1$. As p increases, the exponents $1/(p-1)$ and $p/(p-1)$ decrease, therefore both ratios get closer to 1. Hence, the distribution of the w_v and w_v^p values becomes less sparse as p increases. Finally, note that $0 < p < 1$ is not permitted, as in this case the constrained optimized objectives (4.7), (4.8) become concave w.r.t. the weights, thus the updates, which take the same form as in (4.9), will increase $\mathcal{E}_{\widetilde{\mathcal{H}}}$.

Regarding the computational complexity, during each of the τ' iterations two main operations take place; the estimation of the view weights and the cluster updates. These operations require $O(N^2V)$ and $O(N^2\tau)$ scalar computations, respectively, for MVKKM (τ are the kernel *k*-means iterations). For MVSpec the corresponding cost is $O(N^2MV)$ and $O(N^2M)$ (top M eigenvectors of $\widetilde{K}$) respectively. For both methods an additional $O(N^2V)$ operations are necessary per iteration, to calculate the composite kernel. Thus, the overall cost for MVKKM is $O(N^2(V+\tau)\tau')$, while for MVSpec is $O(N^2MV\tau')$. Note that MVKKM additionally requires a cluster initialization step, while MVSpec an eigenvector discretization step.

It is known that kernel *k*-means monotonically decreases the intra-cluster variance. The update on the weights further reduces the objective value. Hence, the distance-based iterative scheme is guaranteed to monotonically converge to a local minimum of $\mathcal{E}_{\widetilde{\mathcal{H}}}$. Moreover, we anticipate this to be a good local mode, since the iterative process starts with a high quality set of clusters, due to the global kernel *k*-means initialization, which is refined after estimating new values for the weights. As previously mentioned, the spectral approach provides a matrix Y that is optimal for the current weights w.r.t. a *relaxed version* of the considered problem (4.8), where Y is allowed to be an arbitrary orthonormal matrix. The subsequent update on the weights further reduces

the objective, leading to a monotonic convergence to a local minimum as well. Note that a discrete partitioning is obtained only after the MVSpec method has converged. Therefore, it remains to be seen if the decision to relax Y and thus locate the optimal Y in each iteration is effective, compared to MVKKM which at each iteration operates with discrete cluster assignments.

We decided to apply the intra-cluster variance function for multi-view clustering as this is one of the most popular clustering criteria and is well posed for kernel-based learning. Moreover, it fits well to the task of automatically constructing a "ranking" of the views, through the kernel combination of (4.4), and it gives rise to two iterative schemes where the update of the weights and the corresponding partitioning are calculated very easily. Iterative frameworks are constantly gaining ground in multiple kernel learning [28, 88, 124, 126], and are proving to be quite efficient.

Finally, it is crucial for the application of both methods that views have comparable intra-cluster variances in feature space. Hence, views must be normalized, for example, as in the experiments, by dividing each view's kernel entries $K_{ij}^{(v)}$ by the average of the pairwise square distances of the view's instances in feature space:$\sum_{i=1}^{N}\sum_{j=1}^{N}(K_{ii}^{(v)} - 2K_{ij}^{(v)} + K_{jj}^{(v)})/N^2$.

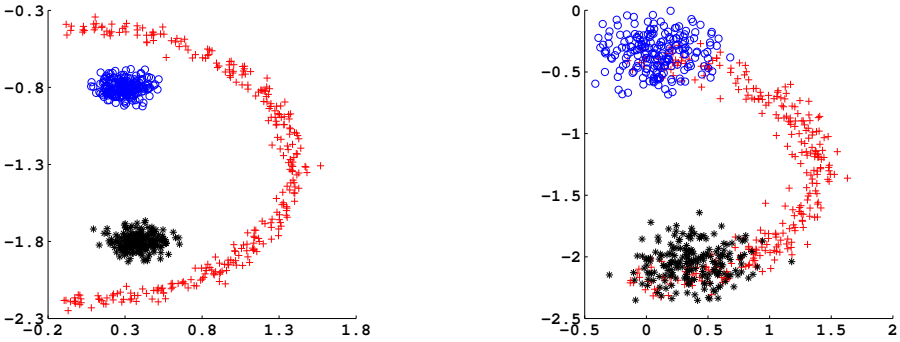
4.3 Empirical Evaluation

The performance of MVKKM and MVSpec[4] is studied on synthetic data as well as on a collection of images and a set of handwritten digits, where multiple views occur naturally. The aim of the experimental evaluation is twofold. First to investigate the p parameters's impact on the returned clusters and the kernel combination coefficients w_v^p, and second to inspect how effective view weighting under our framework is, compared to other multi-view algorithms.

To achieve these goals the two proposed algorithms are executed for various p values, $p > 1$. Moreover, two trivial kernel combinations, $p = 1$ and uniform, are considered. $p = 1$ corresponds to selecting the best kernel, through the weight update process, and splitting the dataset using the information of this kernel only, i.e. it is the best single view case. The uniform combination evenly considers all kernels to obtain a split of the data, i.e. we fix $w_v = 1/V$ in our algorithms and no weight updates are performed (the uniform combination does not depend on the p value). In addition, they are compared to correlational spectral clustering (CSC) [13] and the weighted multi-view convex mixture models (MVCMM) we proposed in Chapter 3 (Section 3.3) of this thesis.

CSC projects the views, which are all thought of as being of the same quality (i.e. no view weights are available), through kernel canonical correlation analysis (KCCA) [51] and then clusters these projections with *k*-means. As in [13], the number of projection axes is set equal to the number of clusters, the KCCA regularization parameters are

[4]Matlab code is available at: `http://www.cs.uoi.gr/~gtzortzi`.



(a) View 1 (b) View 2 - Noisy

Figure 4.1: The two synthetic views. Different symbols represent the three sought clusters.

determined using grid search and *k*-means is restarted 30 times with random initializations and the run with the smallest *k*-means objective is kept.

Remember that, in MVCMM each view is modeled by a convex mixture model (CMM) [71] and an automatically tuned weight is associated with each view. The only parameter that must be determined in advance is a β parameter which controls the sharpness of the components of the CMMs. To locate a good β value, we calculate a reference value β_0, according to the empirical rule presented in Chapter 3 (equation (3.11)), and search around it. In particular, β values in the range $[0.5, 1, 1.5, \ldots, 7]\beta_0$ are tried and the best MVCMM run is reported.

For all datasets the ground-truth labels are given and are only used to assess the quality of the returned solution with the NMI criterion (4.11)[5]. Higher NMI values indicate a better match between cluster labels and class labels.

$$NMI = \frac{2\sum_{k=1}^{M}\sum_{h=1}^{C}\frac{n_k^h}{N}\log\frac{n_k^h N}{\sum_{i=1}^{M} n_i^h \sum_{i=1}^{C} n_k^i}}{H_M + H_C} \tag{4.11}$$

The number of clusters is set equal to the true number of classes and linear kernels are employed for MVKKM, MVSpec and CSC to represent the views, unless stated otherwise. For MVCMM, Gaussian convex mixture models are used (see Chapter 3). Note that the global kernel *k*-means algorithm is utilized to locate initial clusters for MVKKM (Section 4.2.2), thus avoiding the need for multiple restarts. We do not apply a similar procedure to initialize the *k*-means step in CSC, since we adopt the experimental protocol of the CSC paper [13].

4.3.1 Synthetic Data

To outline the basic properties of the proposed algorithms, a three cluster toy example was created, consisting of two views where the second view is a noisy version of the

[5] N is the dataset size, M the number of clusters, C the number of classes, n_k^h the number of points in cluster k belonging to class h, and H_M, H_C the entropy of the clusters and the classes, respectively.

Table 4.1: NMI score and kernel coefficients distribution ($w_v^p / \sum_{v'=1}^{V} w_{v'}^p$) of MVKKM and MVSpec on the synthetic dataset, for several p values and for the uniform case.

	MVKKM			**MVSpec**		
	NMI	**Coefficients**		**NMI**	**Coefficients**	
		View 1	View 2		View 1	View 2
$p = 1$	1	1	0	0.681	1	0
$p = 1.3$	1	0.85	0.15	0.671	0.84	0.16
$p = 1.5$	1	0.77	0.23	0.663	0.74	0.26
$p = 2$	0.769	0.64	0.36	0.632	0.66	0.34
$p = 4$	0.749	0.58	0.42	0.593	0.62	0.38
$p = 6$	0.747	0.56	0.44	0.593	0.62	0.38
Unif.	0.701	0.5	0.5	0.552	0.5	0.5

first (Figure 4.1). Due to the nonlinearly separable nature of the dataset, an rbf kernel is adopted for each view and its parameter is determined through exhaustive search (here $\sigma = 0.2$ for both views).

From Table 4.1 it is evident that as p increases the coefficients w_v^p become more uniform and clustering degrades. This is anticipated since the first view contains all the necessary information to correctly split the data, while the second mixes the clusters. Thus, as the contribution of the second "noisy" view increases, it becomes less probable to recover the true assignments. For small p values, which admit sparser outcomes, the weighting is consistent with the noise level present on the views and MVKKM manages to correctly cluster the data points. Note that even the noisy view contains structural information, hence it is expected to receive nonzero weight even for small p ($p = 1.3, 1.5$). MVSpec, although its coefficients match those of MVKKM, achieves low NMI. We observed that spectral clustering on the first view alone fails to recover the clusters (we executed the popular normalized cut method of [82] for several σ values), giving similar results to MVSpec for $p = 1$ and explaining the deficit of MVSpec.

4.3.2 Multiple Features Dataset

Multiple features is a handwritten digits (0-9) database from the UCI repository [42] (Figure 4.2). The digits (200 per class) are represented by several attribute sets (i.e. views), namely Fourier coefficients, profile correlations, Karhunen-Love coefficients, pixel averages and Zernike moments (note that this is the order of the views in Figure 4.3). From the original dataset several four class subsets were created and the most representative ones are presented here. As attributes within the same view exhibit significantly different scales, all views' attributes were normalized to unit variance. Moreover, kernel entries were divided by the average pairwise square distance of the corresponding view, as discussed in Section 4.2.3. This preprocessing was also applied

Figure 4.2: Examples of handwritten digits contained in the Multiple features dataset.

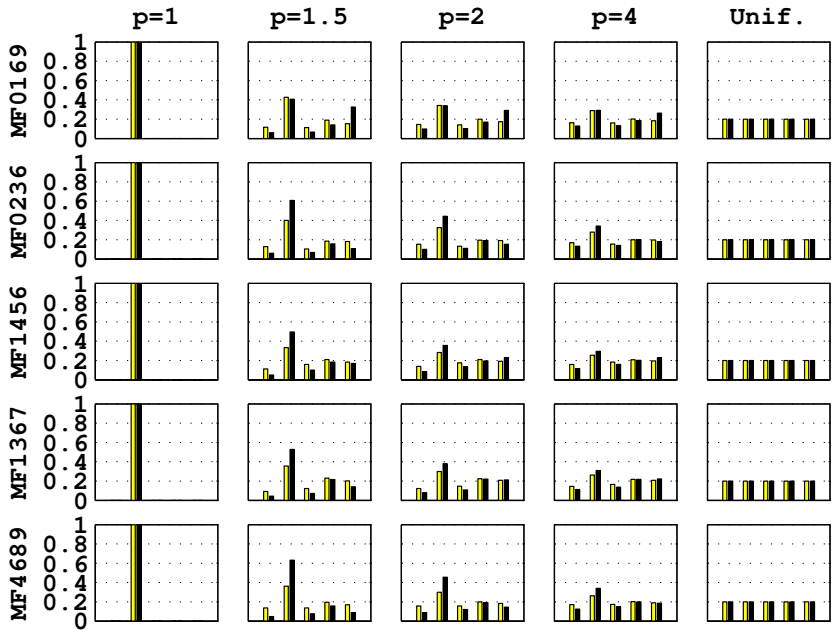
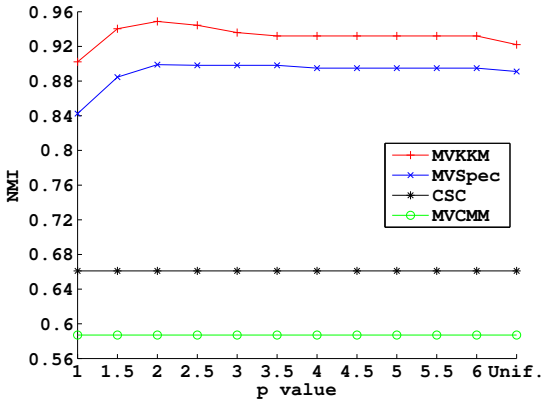


Figure 4.3: MVKKM (yellow) and MVSpec (black) kernel coefficients distribution ($w_v^p / \sum_{v'=1}^{V} w_{v'}^p$) on the Multiple features dataset, for several p values and for the uniform case.

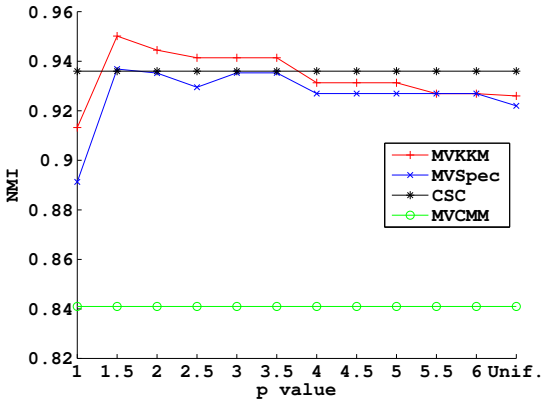
to CSC and MVCMM[6].

The comparison of the four tested algorithms is provided in Figure 4.4, where the subsets are named according to the included numerals. Note that CSC and MVCMM do not depend on p. MVKKM is superior to MVSpec for almost all p values, indicating that the distance-based formulation of the objective is more effective. MVCMM, despite assigning weights to the views, always yields the least NMI, thus highlighting the potential of our clustering technique. CSC is quite competitive, being slightly (except for MF0169 and MF4689) inferior to MVKKM and MVSpec for the best p. Moreover, the single view case ($p = 1$) proves to be the worst, while the uniform (Unif.) is close in accuracy to that for the best p. This fact together with i) the effectiveness of the (unweighted) CSC method for most subsets and ii) the minor, only, drop in NMI as p increases (for MVSpec even an increase is observed for MF1367 and MF4689), hence the kernel coefficients, w_v^p, distribution evolves towards uniformity (Figure 4.3), lead us

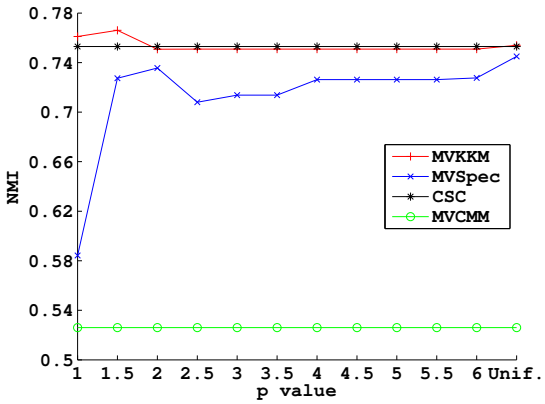
[6]MVCMM is not kernel-based, therefore the distances in the Gaussian components were instead normalized.



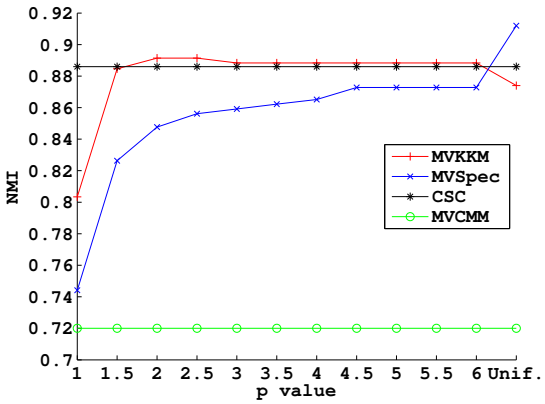
(a) MF0169



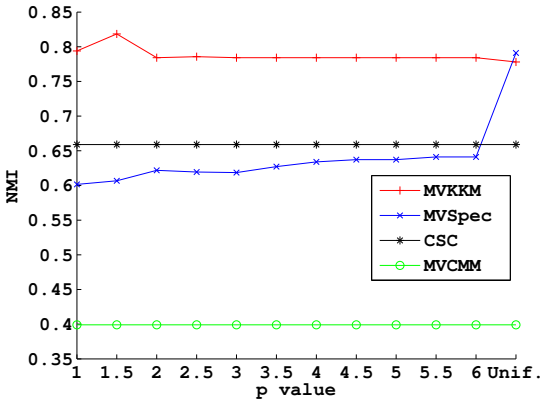
(b) MF0236



(c) MF1456



(d) MF1367



(e) MF4689

Figure 4.4: NMI score of the compared methods on the Multiple features dataset, for several p values and for the uniform case.

Table 4.2: Categories contained in the tested Corel subsets.

Subset	**Categories**			
Corel1	*owls*	*wildlife*	*trains*	*cargo ships*
Corel2	*buses*	*leopards*	*trains*	*cargo ships*
Corel3	*buses*	*leopards*	*cars*	*passenger ships*
Corel4	*owls*	*wildlife*	*hawks*	*roses*
Corel5	*eagles*	*elephants*	*trains*	*passenger ships*

Figure 4.5: Examples of images contained in the Corel collection.

to conclude that all views contribute significantly in the Multiple features dataset. Still though, a p value that admits some sparsity on the solution can enhance performance, particularly for MVKKM where $p = 1.5$ or $p = 2$ is always the best choice.

4.3.3 Corel Images Dataset

A part of the popular Corel collection consisting of 34 categories, each with 100 images, serves as our second real multi-modal paradigm (Figure 4.5). Images consist of a salient foreground object, but within each class there is great variance in terms of distance and angle of the object, color, lighting, and background composition, making this dataset difficult for unsupervised learning. Attribute vectors that represent the images in terms of seven views, three color-related views (color histogram, moment and coherence) and four texture-related views (coarseness and directionality of tamura texture, wavelet and mrsar texture) are available for this collection[7] (note that this is the order of the views in Figure 4.6). Many four class subsets were extracted and the most representative ones are included in the experiments (Table 4.2). The kernels were normalized as for the Multiple features dataset.

Results are depicted in Figure 4.7. MVKKM for $p = 2$ considerably outperforms the other three algorithms and its kernel coefficients, w_v^p, distribution (Figure 4.6) indicates that a nonuniform mixture is suited to this dataset, thus explaining the deficit of larger p values and CSC. Moreover, its advantage over MVSpec, for which the NMI increases as p increases and a uniform solution is preferable, is significant for all p values. The difference between the two clustering schemes can be explained from Figure 4.6, where a disagreement is observed regarding which view should acquire the highest weight (except for Corel1) and a more peaked coefficient distribution for MVSpec. It seems that MVSpec selects inappropriate views, indicating that the relaxed problem becomes detached from the actual objective (4.8) during the iterative process (it even yields worse results than CSC). It is worth noting that both MVKKM and MVSpec underperform

[7] http://www.cs.virginia.edu/~xj3a/research/CBIR/Download.htm

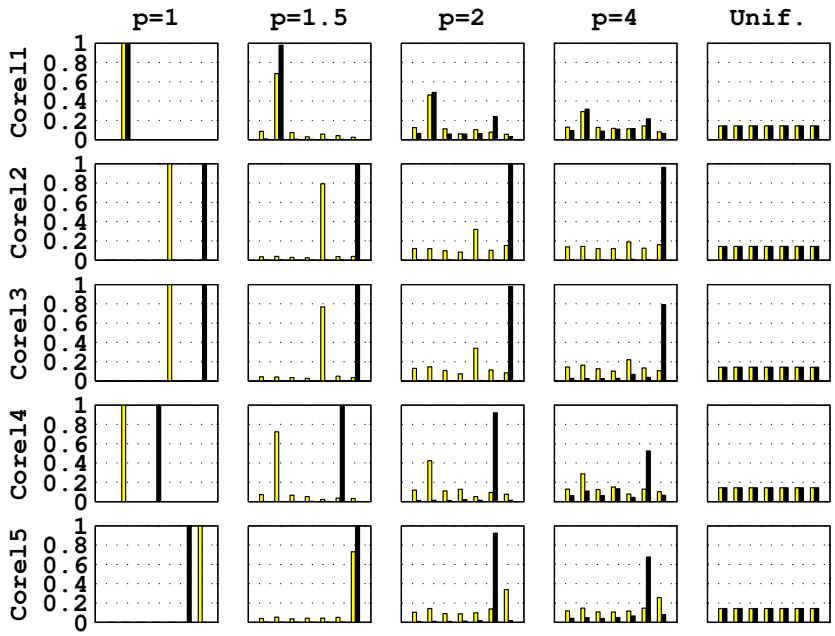


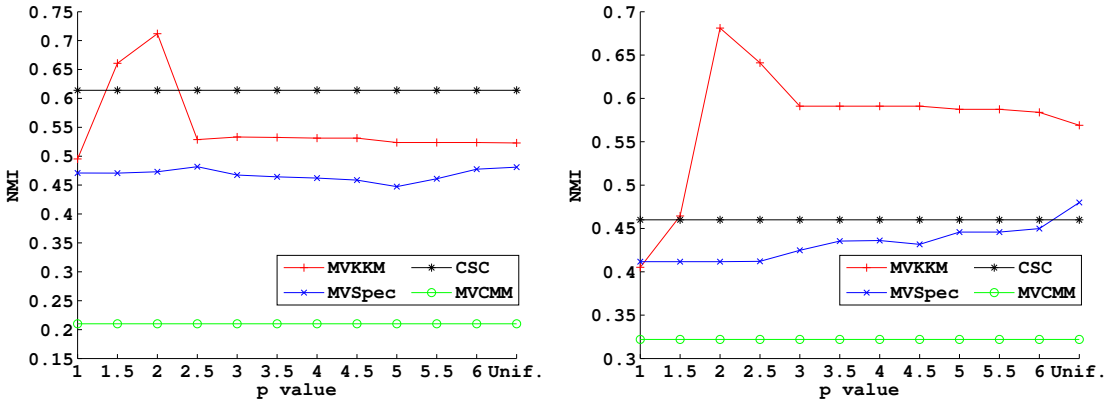
Figure 4.6: MVKKM (yellow) and MVSpec (black) kernel coefficients distribution ($w_v^p / \sum_{v'=1}^{V} w_{v'}^p$) on the Corel dataset, for several p values and for the uniform case.

for very small p ($p = 1$, $p = 1.5$), i.e. for very sparse combinations, thus exploiting information from all views is necessary for the tested real data. Finally note that MVCMM is not performing well on this or the previous dataset, despite automatically estimating view weights. This emanates from the very sparse solution recovered by the method, that assigns a zero weight to most views.

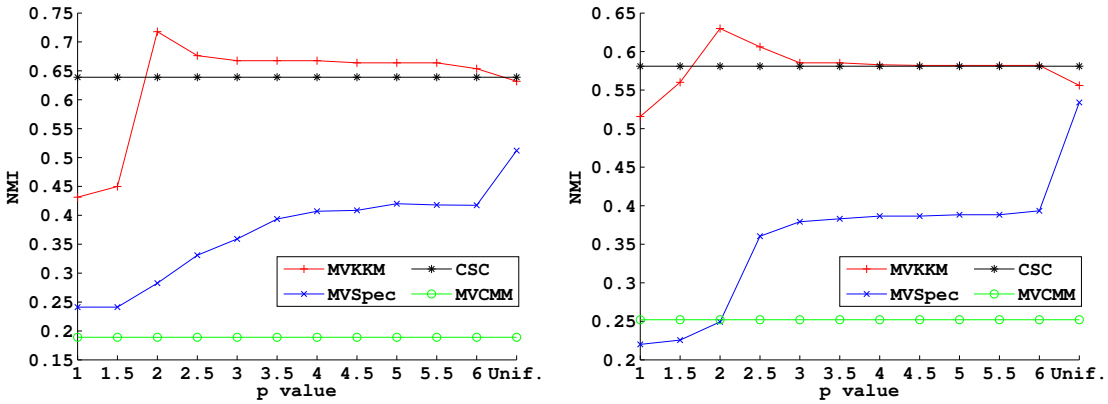
4.3.4 Discussion

The empirical evaluation has shown that the distance-based formulation of the objective provides better results than the spectral. There is dual reason for this behavior. First, MVSpec provides at each iteration a continuous solution Y which at the end is discretized to obtain the final partitioning. The continuous solution runs the risk of deviating from the original non-relaxed objective, especially in iterative algorithms, such as MVSpec, where the weights get updated based on the relaxed objective. On the contrary, MVKKM provides a discrete partition in every iteration, thus following "closely" the intra-cluster variance objective. Hence, the relaxation can lead to the selection of suboptimal views, whose influence is further enhanced for sparser solutions (i.e. for smaller p). This case arose for the Corel dataset (Figures 4.6-4.7) and explains why MVSpec usually attains its highest NMI for the uniform case.

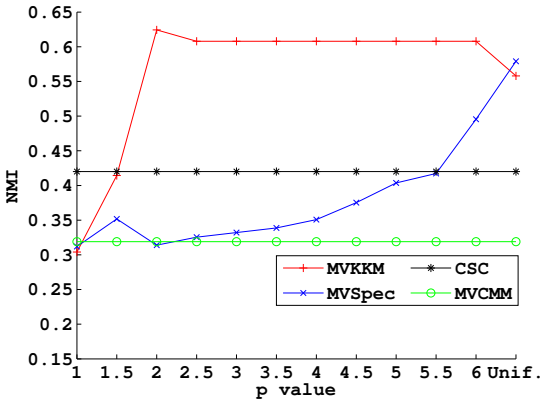
Second, for the initialization of MVKKM the global kernel k-means procedure was employed, which is deterministic and very effective [105]. As the experiments with the synthetic and Multiple features datasets indicate, a properly initialized kernel k-means can locate better clusters than spectral techniques, since MVKKM outperforms MVSpec



(a) Corel1 (b) Corel2



(c) Corel3 (d) Corel4



(e) Corel5

Figure 4.7: NMI score of the compared methods on the Corel dataset, for several p values and for the uniform case.

despite both techniques resulting in similar w_v^p values. The two previous observations also elucidate why CSC performs better than MVSpec for most p values on the real data.

Further ground for the above remarks was provided when we executed for the Corel subsets i) a run of kernel *k*-means using the MVSpec-derived composite kernel and ii) spectral analysis over the MVKKM-derived composite kernel. The results were always inferior to those reported for MVKKM in Section 4.3.3, demonstrating that the distance-based formulation infers both better cluster structures and view weights.

Furthermore, for MVKKM, which is always the best of the tested methods, selecting either the best view or equally all views proves to be inadequate, highlighting the importance of allowing the clustering algorithm to mix views more robustly and finding a balance between sparsity and uniformity. This is also reported in some of the existing multi-view and multiple kernel learning studies, such as [62, 70, 124, 131]. The appropriate p value is, of course, dataset dependent.

Finally, a word on the computational complexity of the proposed methods and specifically on the number of weight and cluster updates performed. For the Multiple features dataset, MVKKM and MVSpec need between 3-5 and 4-10 iterations to converge, respectively, while for the Corel dataset they need between 4-11 and 5-15 iterations respectively, depending on the p value and the dataset subset. It is evident that both algorithms quickly converge and, in general, the more the final weights deviate from their initial, uniform, values (as is the case for smaller p values, or the Corel dataset) the more iterations are necessary.

4.4 Summary

In this chapter, we have presented two multi-modal approaches that represent modalities through kernel matrices and optimize the intra-cluster variance function. A weighted combination of the kernels that reflects the views' relevance to the clustering task is automatically learned, using closed-form updates, along with the cluster assignments. This combination utilizes a p exponent to control the sparsity of the weights, which resembles the p-norm constraint applied in multiple kernel learning [62]. Both methods, particularly MVKKM, compare favorable to existing ones, underlying the strength of our framework and that view weighting can boost the quality of the partitioning, if the sparsity of the weights is appropriately moderated.

CHAPTER 5

RATIO-BASED MULTIPLE KERNEL CLUSTERING

The success of large margin techniques in supervised learning, particularly that of support vector machines (SVM) [19], has generated great interest in extending such techniques to the unsupervised setting, leading to the, so called, *maximum margin clustering*[1] (MMC) problem [121]. Given a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x}_i \in \Re^d$, MMC approaches attempt to find a labeling (clustering) $\mathbf{y} = [y_1, \ldots, y_N]^\top$, $y_i \in \{\pm 1\}$, of the instances, such that a subsequent training of a standard SVM [12, 19] would result in a margin that is maximal over all possible labellings. MMC is formulated as:

$$\min_{\mathbf{y}} \min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^N \xi_i, \tag{5.1}$$

$$s.t. \;\; -\ell \leq \sum_{i=1}^N y_i \leq \ell,\; \mathbf{y} \in \{\pm 1\}^N,\; y_i\left(\mathbf{w}^\top \phi(\mathbf{x}_i) + b\right) \geq 1 - \xi_i,\; \xi_i \geq 0,$$

where $\mathbf{w}$, b are the coefficients of the SVM hyperplane ($\|\mathbf{w}\|$ is the reciprocal of the margin), $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_N]^\top$ is the vector of slack variables capturing the misclassification error, $C > 0$ is the regularization constant and ϕ is a transformation that maps the instances to a higher dimensional feature space that is implicitly defined using the kernel trick ($\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$) [41]. Moreover, to prevent the trivially "optimal" solution of assigning all instances to the same cluster and thus obtaining an infinite margin ($\|\mathbf{w}\| = 0$), a cluster balance constraint ($-\ell \leq \sum_{i=1}^N y_i \leq \ell$) was introduced by Xu et al. [121], where $\ell \geq 0$ is a constant controlling the imbalance of the clusters.

[1] For details on maximum margin clustering see Chapter 1, Section 1.3.2.

The MMC problem is non-convex with integer parameters $\mathbf{y}$, making the optimization much trickier than that of the standard (convex) SVM where the labels $\mathbf{y}$ are known in advance. To solve (5.1), some approaches employ semidefinite programing (SDP) [110, 121, 122], others exploit the cutting plane method [115, 130] and others rely on alternating between the outer and the inner minimization [129].

It is well-known that the performance of kernel-based approaches [92], like MMC, heavily depends on the choice of the kernel. However, it is often unclear which is the best kernel for a particular task. *Multiple kernel learning*[2] (MKL) [49], which has been mainly studied under the SVM paradigm [19], attempts to simultaneously locate the hyperplane with the largest margin and also learn a suitable kernel. The kernel, $\widetilde{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_j) = \widetilde{\phi}(\mathbf{x}_i)^\top \widetilde{\phi}(\mathbf{x}_j)$, is usually parametrized by a vector $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_V]^\top$ of parameters. Most existing MKL approaches focus on supervised learning and several of them, in principle, derive from the following optimization (subject to some slight modifications) (e.g. [62, 63, 88, 124]):

$$\min_{\boldsymbol{\theta}, \mathbf{w}, b, \boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \xi_i, \tag{5.2}$$

$$s.t. \ \ \theta_v \geq 0, \ \ \|\boldsymbol{\theta}\|_p^p \leq 1, \ \ y_i \left(\mathbf{w}^\top \widetilde{\phi}(\mathbf{x}_i) + b \right) \geq 1 - \xi_i, \ \ \xi_i \geq 0,$$

Kernel parameters θ_v are limited to nonnegative values to ensure the learned kernel is positive semidefinite and the p-norm constraint is employed to avoid overfitting. Usually the kernel is parametrized as a linear combination of some given basis kernels and either the 1-norm that promotes sparsity [88, 96, 133], or a more general p-norm, $p \geq 1$, [62, 63, 124], is chosen. There also exist a few studies that consider nonlinear combinations of basis kernels [28, 48], or even general types of parametric kernels [45, 111]. The optimization problem in (5.2) is non-convex due to $\boldsymbol{\theta}$. Depending on the form of the kernel parametrization and the choice of p-norm, various optimization strategies have been proposed, several of which alternate between updating $\boldsymbol{\theta}$ and solving a standard SVM to obtain $\mathbf{w}$, b and $\boldsymbol{\xi}$. For example, semi-infinite linear programming [62, 96, 133], gradient-based methods [45, 48, 88, 111] and closed-form methods [63, 124].

Extending MKL to the clustering domain, and in particular to MMC problems, is an interesting research direction, however, existing work is rather limited. The methods of [110, 130] seek to find a linear mixture of the basis kernels along with the cluster assignments, such that the margin is maximized, in essence combining (5.1) and (5.2). Here, we follow a similar path, but propose [109] a novel objective that considers the *ratio between the margin* (a notion of cluster separability) *and the intra-cluster variance criterion* of kernel *k*-means [37, 90] (a notion of cluster coherence). Hence, both the separation and the compactness of the clusters are explicitly taken into account, which can possibly improve on the solutions returned by approaches utilizing either of the

[2] For details on multiple kernel learning see Chapter 1, Section 1.3.

two. Importantly, the margin has been shown to suffer from a major deficiency when applied to supervised MKL [45]. It can become arbitrarily large by a simple scaling of the kernel, thus it is inappropriate for assessing the quality of the learned kernel. The same can be demonstrated to hold for unsupervised MKL and we prove that *our ratio-based objective is invariant to kernel scaling*, thus overcoming this deficiency. Moreover, its global optimum solution is *invariant to the type of p-norm constraint on the kernel parameters $\boldsymbol{\theta}$* (when a linear combination of basis kernels is employed), making the selection of a suitable norm less crucial.

A simple gradient-based optimization procedure that alternates between updating the kernel parameters $\boldsymbol{\theta}$ and the cluster assignments $\mathbf{y}$ is devised, avoiding the invocation of complex optimizers, such as the SDP solvers [110] and the cutting plane method [130]. Experiments on several datasets, including two collections of handwritten numerals and two image collections, reveal the superiority of the proposed method over approaches that rely solely on the margin or the intra-cluster variance.

The rest of this chapter is organized as follows. Section 5.1 introduces our ratio-based formulation and presents its invariance properties and optimization details. Experiments follow in Section 5.2, while Section 5.3 provides a summary of the chapter.

5.1 The RMKC Algorithm

5.1.1 Problem Formulation

Consider a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x}_i \in \Re^d$, for which we want to simultaneously infer the cluster labels and also perform kernel learning under the large margin framework. While presenting our method we shall restrict ourselves on a linear combination of basis kernels, which is the most common technique of parametrizing kernels for MKL [63, 88, 124]. Later we will show that our model can accommodate more general parametric forms of kernels.

Assume that V basis kernels, $\mathcal{K}^{(v)} : \mathcal{X} \times \mathcal{X} \to \Re$, are available, each implicitly inducing a transformation $\phi^{(v)} : \mathcal{X} \to \mathcal{H}^{(v)}$ on the instances to a feature space $\mathcal{H}^{(v)}$ through $\mathcal{K}^{(v)}(\mathbf{x}_i, \mathbf{x}_j) = \phi^{(v)}(\mathbf{x}_i)^\top \phi^{(v)}(\mathbf{x}_j)$. A linear mixture of kernels gives rise to a composite kernel $\widetilde{\mathcal{K}}$:

$$\widetilde{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{v=1}^{V} \theta_v \mathcal{K}^{(v)}(\mathbf{x}_i, \mathbf{x}_j),\ \theta_v \geq 0, \tag{5.3}$$

that is parametrized by $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_V]^\top$. Since $\widetilde{\mathcal{K}}$ is a valid kernel it holds that $\widetilde{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_j) = \widetilde{\phi}(\mathbf{x}_i)^\top \widetilde{\phi}(\mathbf{x}_j)$, $\widetilde{\phi} : \mathcal{X} \to \widetilde{\mathcal{H}}$, and, actually, $\widetilde{\phi}(\mathbf{x}_i) = \left[\sqrt{\theta_1}\phi^{(1)}(\mathbf{x}_i)^\top, \ldots, \sqrt{\theta_V}\phi^{(V)}(\mathbf{x}_i)^\top\right]^\top$ due to the linear combination.

We propose a new formulation that does not depend only on the margin, like most existing MMC and MKL studies, but utilizes *the ratio between the margin and the intra-cluster variance* objective of kernel k-means [37, 90] in feature space $\widetilde{\mathcal{H}}$. Minimizing

such a ratio can lead to superior partitionings as *both compact and well-separated clusters are sought*. Moreover, as it will be proved, it makes our formulation invariant to kernel scaling, an important property when kernel learning is involved [45]. Denoting by $\mathbf{y} = [y_1, \ldots, y_N]^\top$, $y_i \in \{\pm 1\}$, the vector of the instances' cluster labels, we consider the following optimization problem:

$$\min_{\boldsymbol{\theta}, \mathbf{y}} \mathcal{J}(\boldsymbol{\theta}, \mathbf{y}),\ s.t.\ \theta_v \geq 0,\ \|\boldsymbol{\theta}\|_p^p = 1,\ -\ell \leq \sum_{i=1}^{N} y_i \leq \ell,\ \mathbf{y} \in \{\pm 1\}^N, \tag{5.4}$$

$$\mathcal{J}(\boldsymbol{\theta}, \mathbf{y}) = \min_{\mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2}\mathcal{E}(\boldsymbol{\theta}, \mathbf{y})\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i,\ s.t.\ y_i\left(\mathbf{w}^\top \widetilde{\phi}(\mathbf{x}_i) + b\right) \geq 1 - \xi_i,\ \xi_i \geq 0. \tag{5.5}$$

Here $\mathcal{E}(\boldsymbol{\theta}, \mathbf{y})$ is the kernel k-means criterion (5.6) describing the intra-cluster variance[3], where $\widetilde{\mathbf{m}}_k$ is the k-th cluster center and δ_{ik} is a cluster indicator variable with $\delta_{i1} = 1$ if $y_i = -1$ and $\delta_{i2} = 1$ if $y_i = 1$. Note that due to the SVM-like formulation we are limited to two-cluster solutions, i.e. $k \in \{1, 2\}$, which is the typical case for MMC methods.

$$\mathcal{E}(\boldsymbol{\theta}, \mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{2} \delta_{ik}\|\widetilde{\phi}(\mathbf{x}_i) - \widetilde{\mathbf{m}}_k\|^2, \tag{5.6}$$

$$\delta_{ik} = \begin{cases} 1, & y_i = 2k - 3 \\ 0, & \text{otherwise} \end{cases},\ \widetilde{\mathbf{m}}_k = \frac{\sum_{i=1}^{N} \delta_{ik}\widetilde{\phi}(\mathbf{x}_i)}{\sum_{i=1}^{N} \delta_{ik}}$$

The squared Euclidean distances in $\mathcal{E}(\boldsymbol{\theta}, \mathbf{y})$ can be posed solely in terms of the entries of the kernel matrix $\widetilde{K} \in \Re^{N \times N}$ corresponding to $\widetilde{\mathcal{K}}$ ($\widetilde{K}_{ij} = \widetilde{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_j)$), from which it follows that:

$$\mathcal{E}(\boldsymbol{\theta}, \mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{2} \delta_{ik}\left(\widetilde{K}_{ii} - \frac{2\sum_{j=1}^{N} \delta_{jk}\widetilde{K}_{ij}}{\sum_{j=1}^{N} \delta_{jk}} + \frac{\sum_{j=1}^{N}\sum_{l=1}^{N} \delta_{jk}\delta_{lk}\widetilde{K}_{jl}}{\sum_{j=1}^{N}\sum_{l=1}^{N} \delta_{jk}\delta_{lk}}\right). \tag{5.7}$$

Additionally, by using (5.3), the composite kernel matrix $\widetilde{K}$ can be written as the sum of the basis kernel matrices $K^{(v)} \in \Re^{N \times N}$, i.e. $\widetilde{K} = \sum_{v=1}^{V} \theta_v K^{(v)}$, leading to:

$$\mathcal{E}(\boldsymbol{\theta}, \mathbf{y}) = \frac{1}{N}\sum_{v=1}^{V}\theta_v \sum_{i=1}^{N}\sum_{k=1}^{2} \delta_{ik}\left(K_{ii}^{(v)} - \frac{2\sum_{j=1}^{N} \delta_{jk}K_{ij}^{(v)}}{\sum_{j=1}^{N} \delta_{jk}} + \frac{\sum_{j=1}^{N}\sum_{l=1}^{N} \delta_{jk}\delta_{lk}K_{jl}^{(v)}}{\sum_{j=1}^{N}\sum_{l=1}^{N} \delta_{jk}\delta_{lk}}\right). \tag{5.8}$$

For the above optimization problem (5.4), it is easy to verify that its objective function $\mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$ at a given $\{\boldsymbol{\theta},\ \mathbf{y}\}$ is defined as the optimal objective value of a problem (5.5) that closely resembles the standard SVM. The only difference is that the variance to margin ratio is employed in place of the margin. Similar to MMC methods [121, 129], a cluster balance constraint ($-\ell \leq \sum_{i=1}^{N} y_i \leq \ell$) must be imposed to prevent meaningless

[3]For simplicity, on the following, we shall also refer to the intra-cluster variance as the variance of the clusters.

solutions from arising. Finally, the composite kernel coefficients θ_v are required to be nonnegative so that $\widetilde{K}$ is a valid kernel and a p-norm constraint is introduced to avoid overfitting, as in (5.2).

Hence, the optimization in (5.4) searches for a pair of $\{\boldsymbol{\theta}, \mathbf{y}\}$ values that yields a small variance to margin ratio ($\mathcal{E}(\boldsymbol{\theta}, \mathbf{y})\|\mathbf{w}\|^2$) regularized by the misclassification error (captured by the slack variables $\boldsymbol{\xi}$). We shall call this approach Ratio-based Multiple Kernel Clustering, abbreviated as RMKC.

It should be clarified that the actual problem we are trying to solve is (s.t. the constraints in (5.4)-(5.5)):

$$\min_{\boldsymbol{\theta},\mathbf{y},\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\mathcal{E}(\boldsymbol{\theta}, \mathbf{y})\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i, \tag{5.9}$$

which is rather difficult to directly optimize, since it constitutes a non-convex problem with integer parameters $\mathbf{y}$. Reformulating it as in (5.4), analogously to Rakotomamonjy et al. [88], will enable us to devise an alternating optimization strategy, that benefits from differentiability w.r.t. $\boldsymbol{\theta}$ and does not demand the use of complex solvers.

5.1.2 Properties of RMKC

In this section, two properties of RMKC are presented, which highlight some important advantages of combining the margin with the variance of the clusters.

Suppose the composite kernel $\widetilde{\mathcal{K}}$ (5.3) is scaled by $\alpha > 0$, i.e. $\widetilde{\mathcal{K}}' = \alpha\widetilde{\mathcal{K}}$. Then the corresponding transformation becomes $\widetilde{\phi}' = \sqrt{\alpha}\widetilde{\phi}$. Moreover, as $\widetilde{\mathcal{K}}$ is a linear combination of basis kernels, its scaling can be equivalently posed as a scaling on its parameters, i.e. $\boldsymbol{\theta}' = \alpha\boldsymbol{\theta}$.

Proposition 5.1. (**Scale Invariance**) *If a kernel $\widetilde{\mathcal{K}}$ of the form defined in* (5.3) *is scaled by a scalar $\alpha > 0$, then $\mathcal{J}(\alpha\boldsymbol{\theta}, \mathbf{y}) = \mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$.*

Proof. From (5.7)-(5.8) it is evident that $\mathcal{E}(\alpha\boldsymbol{\theta}, \mathbf{y}) = \alpha\mathcal{E}(\boldsymbol{\theta}, \mathbf{y})$, hence:

$$\mathcal{J}(\alpha\boldsymbol{\theta}, \mathbf{y}) = \min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\alpha\mathcal{E}(\boldsymbol{\theta}, \mathbf{y})\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i,$$

$$s.t.\ \ y_i\left(\mathbf{w}^\top\left(\sqrt{\alpha}\widetilde{\phi}(\mathbf{x}_i)\right) + b\right) \geq 1 - \xi_i,\ \xi_i \geq 0.$$

Setting $\mathbf{w} = \mathbf{w}'/\sqrt{\alpha}$ and substituting in the above equation completes the proof, as (5.5) is recovered. ∎

Our quest for an objective that satisfies Proposition 5.1 was inspired by Gai et al. [45], where it was illustrated that relying solely on the margin is not sufficient to perform kernel learning in the supervised case. Analogously, if $\mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$ in (5.4) is

replaced with the more conventional margin-based objective:

$$\mathcal{J}'(\boldsymbol{\theta}, \mathbf{y}) = \min_{\mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i,\ s.t.\ \ y_i\left(\mathbf{w}^\top\widetilde{\phi}(\mathbf{x}_i) + b\right) \geq 1 - \xi_i,\ \xi_i \geq 0, \tag{5.10}$$

it can be shown that an arbitrarily small $\mathcal{J}'(\boldsymbol{\theta}, \mathbf{y})$ value can be achieved by scaling the composite kernel, thus constituting the margin criterion unsuitable for evaluating the true quality of the kernel while learning $\{\boldsymbol{\theta}, \mathbf{y}\}$.

To analyze this in more detail, let $\{\mathbf{w}^*, b^*, \boldsymbol{\xi}^*\}$ be the optimal solution of (5.10) for some fixed $\{\boldsymbol{\theta}, \mathbf{y}\}$. Assume the composite kernel is scaled by $\alpha > 1$ and for the hyperplane coefficients we select $\mathbf{w}' = \mathbf{w}^*/\sqrt{\alpha}$ and $b' = b^*$. Then the hyperplane of the scaled kernel equals $\mathbf{w}'^\top\left(\sqrt{\alpha}\widetilde{\phi}(\mathbf{x}_i)\right) + b' = \mathbf{w}^{*\top}\widetilde{\phi}(\mathbf{x}_i) + b^*$, hence it is identical to that of the case without scaling and therefore results in the same slack variables, i.e. $\boldsymbol{\xi}' = \boldsymbol{\xi}^*$. Thus $\frac{1}{2}\|\mathbf{w}'\|^2 + C\sum_{i=1}^{N}\xi_i' = \frac{1}{2}\frac{\|\mathbf{w}^*\|^2}{\alpha} + C\sum_{i=1}^{N}\xi_i^* < \frac{1}{2}\|\mathbf{w}^*\|^2 + C\sum_{i=1}^{N}\xi_i^* = \mathcal{J}'(\boldsymbol{\theta}, \mathbf{y})$, due to $\alpha > 1$. Obviously, the optimal solution of the scaled problem ($\mathcal{J}'(\alpha\boldsymbol{\theta}, \mathbf{y})$) satisfies $\mathcal{J}'(\alpha\boldsymbol{\theta}, \mathbf{y}) \leq \frac{1}{2}\|\mathbf{w}'\|^2 + C\sum_{i=1}^{N}\xi_i' < \mathcal{J}'(\boldsymbol{\theta}, \mathbf{y})$. It is now evident that by enlarging the composite kernel, the margin-based objective (5.10) can become arbitrarily small.

Note that in the linear combination case (5.3), where scaling the composite kernel is equivalent to scaling its parameters, the scaling issue can be handled through the p-norm constraint on $\boldsymbol{\theta}$. However, this is not possible for nonlinear mixtures of basis kernels. On the contrary, our ratio-based objective (5.5) is scale invariant for arbitrary forms of composite kernels (the proof is analogous to Proposition 5.1) and also allows for norm invariance.

Proposition 5.2. (**Norm Invariance**) *Consider a kernel $\widetilde{\mathcal{K}}$ of the form defined in* (5.3) *as well as a) the optimization problem described by* (5.4) *without the p-norm constraint on $\boldsymbol{\theta}$ (p1) and b) the same problem* (5.4)*, but with the slightly more general p-norm constraint $\|\boldsymbol{\theta}\|_p^p = c$, $c > 0$, in place of $\|\boldsymbol{\theta}\|_p^p = 1$ (p2). If $\{\boldsymbol{\theta}_a^*, \mathbf{y}_a^*\}$ is a global optimal solution of p1 then $\left\{\frac{c^{1/p}}{\|\boldsymbol{\theta}_a^*\|_p}\boldsymbol{\theta}_a^*, \mathbf{y}_a^*\right\}$ is a global optimal solution of p2. Also, if $\{\boldsymbol{\theta}_b^*, \mathbf{y}_b^*\}$ is a global optimal solution of p2 then $\{\boldsymbol{\theta}_b^*, \mathbf{y}_b^*\}$ is a global optimal solution of p1.*

Proof. From the scale invariance property and since $\{\boldsymbol{\theta}_a^*, \mathbf{y}_a^*\}$ is a global optimum of p1 we get $\mathcal{J}\left(\frac{c^{1/p}}{\|\boldsymbol{\theta}_a^*\|_p}\boldsymbol{\theta}_a^*, \mathbf{y}_a^*\right) = \mathcal{J}(\boldsymbol{\theta}_a^*, \mathbf{y}_a^*) \leq \mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$ for any $\{\boldsymbol{\theta}, \mathbf{y}\}$ satisfying the constraints of p1. Note that the admissible $\boldsymbol{\theta}$ values for problem p2 are a subset of those allowed in p1, hence the above inequality also holds for every $\{\boldsymbol{\theta}, \mathbf{y}\}$ adhering to the constraints of p2 (the constraints for $\mathbf{y}$ are identical in p1 and p2). Together with the fact that $\left\|\frac{c^{1/p}}{\|\boldsymbol{\theta}_a^*\|_p}\boldsymbol{\theta}_a^*\right\|_p^p = c$ the first part of the proof is completed.

For any $\{\boldsymbol{\theta}, \mathbf{y}\}$ complying to the constraints of p1 it holds that $\left\{\frac{c^{1/p}}{\|\boldsymbol{\theta}\|_p}\boldsymbol{\theta}, \mathbf{y}\right\}$ is admissible for p2, since $\left\|\frac{c^{1/p}}{\|\boldsymbol{\theta}\|_p}\boldsymbol{\theta}\right\|_p^p = c$. The scale invariance property and the global optimality of $\{\boldsymbol{\theta}_b^*, \mathbf{y}_b^*\}$ w.r.t. p2 yields $\mathcal{J}(\boldsymbol{\theta}_b^*, \mathbf{y}_b^*) \leq \mathcal{J}\left(\frac{c^{1/p}}{\|\boldsymbol{\theta}\|_p}\boldsymbol{\theta}, \mathbf{y}\right) = \mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$, thus completing the second part of the proof. ∎

Proposition 5.2 implies that the global optimal solution of the proposed formulation (5.4) is insensitive to the selected type of p-norm constraint, up to a scaling on the composite kernel parameters. The norm constraint can be even dropped from (5.4) without affecting its optimal solution. Of course, a solver that locates local optima of the ratio-based objective may produce different solutions when different p-norms are employed for the same problem, but at least the overall best will be the same, making the choice of the p-norm less crucial.

5.1.3 Optimizing the RMKC Objective

An iterative algorithm that alternates between updating the cluster labels $\mathbf{y}$ and reestimating the composite kernel coefficients $\boldsymbol{\theta}$, starting from some initial $\{\boldsymbol{\theta}, \mathbf{y}\}$ value, is presented and its main steps are summarized in Algorithm 5.1 and Algorithm 5.2.

Evaluating the Objective Function

To compute the value of the objective function $\mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$ for some fixed $\{\boldsymbol{\theta}, \mathbf{y}\}$, we need to solve the convex SVM-like optimization problem in (5.5). This can be facilitated by turning to its dual, which can be obtained by incorporating the constraints into the primal via Lagrange multipliers and setting the derivatives w.r.t. $\mathbf{w}$, b, and $\boldsymbol{\xi}$ to zero. After some manipulation the following dual emerges:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2\mathcal{E}(\boldsymbol{\theta}, \mathbf{y})} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \widetilde{K}_{ij}, \; s.t. \; 0 \leq \alpha_i \leq C, \; \sum_{i=1}^{N} \alpha_i y_i = 0. \tag{5.11}$$

Since the cluster variance $\mathcal{E}(\boldsymbol{\theta}, \mathbf{y})$ is a constant for given $\{\boldsymbol{\theta}, \mathbf{y}\}$, it can be included in the kernel matrix and, thus, (5.11) actually coincides with the dual of the standard SVM, with $\frac{1}{\mathcal{E}(\boldsymbol{\theta}, \mathbf{y})} \widetilde{K}$ as the kernel matrix. Hence, the optimal solution for (5.11), denoted by $\boldsymbol{\alpha}^*$, can be located using any of the existing SVM solvers (the optimal values for $\mathbf{w}$, b, and $\boldsymbol{\xi}$ in (5.5) are calculated based on the solution of the dual). Moreover, due to strong duality, the value of $\mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$ can be directly acquired from the dual:

$$\mathcal{J}(\boldsymbol{\theta}, \mathbf{y}) = \sum_{i=1}^{N} \alpha_i^* - \frac{1}{2\mathcal{E}(\boldsymbol{\theta}, \mathbf{y})} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i^* \alpha_j^* y_i y_j \widetilde{K}_{ij}. \tag{5.12}$$

It should be clarified that whenever $\mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$ is evaluated for a different pair of $\{\boldsymbol{\theta}, \mathbf{y}\}$ values, the optimal dual parameters $\boldsymbol{\alpha}^*$ must be reestimated, since they depend on $\{\boldsymbol{\theta}, \mathbf{y}\}$.

Updating the Kernel Parameters

Changing the composite kernel coefficients so that the ratio-based objective $\mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$ is reduced, while keeping the cluster labels $\mathbf{y}$ fixed, can be effectively performed by means

of gradient descent. Due to strong duality between (5.5) and (5.11) (see the preceding subsection), we can exploit (5.12) to compute the gradient of $\mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$ w.r.t. $\boldsymbol{\theta}$.

Proof for the differentiability of $\mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$ comes from Danskin's theorem [32], similar to [88, 111]. To apply this theorem to our problem, two conditions must be satisfied. First, the optimal solution $\boldsymbol{\alpha}^*$ of (5.11) must be unique. This can be ensured by demanding the composite kernel matrix $\widetilde{K}$ to be strictly positive definite for every admissible $\boldsymbol{\theta}$. Second, the objective function optimized in the dual (5.11) must be continuously differentiable w.r.t. $\boldsymbol{\theta}$, which can be ensured by demanding $\widetilde{K}$ to be continuously differentiable w.r.t. $\boldsymbol{\theta}$. As $\widetilde{K}$ is a linear mixture of basis kernel matrices $K^{(v)}$, both requirements are fulfilled as long as every $K^{(v)}$ is strictly positive definite. The theorem also states that $\mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$ can be differentiated as if $\boldsymbol{\alpha}^*$ does not depend on $\boldsymbol{\theta}$. Therefore, the derivatives can be obtained from (5.12) as:

$$\frac{\partial \mathcal{J}(\boldsymbol{\theta}, \mathbf{y})}{\partial \theta_v} = \frac{1}{2\mathcal{E}(\boldsymbol{\theta}, \mathbf{y})^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i^* \alpha_j^* y_i y_j \widetilde{K}_{ij} \frac{\partial \mathcal{E}(\boldsymbol{\theta}, \mathbf{y})}{\partial \theta_v} - \frac{1}{2\mathcal{E}(\boldsymbol{\theta}, \mathbf{y})} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i^* \alpha_j^* y_i y_j \frac{\partial \widetilde{K}_{ij}}{\partial \theta_v}, \tag{5.13}$$

where $\frac{\partial \widetilde{K}_{ij}}{\partial \theta_v} = K_{ij}^{(v)}$ and $\frac{\partial \mathcal{E}(\boldsymbol{\theta}, \mathbf{y})}{\partial \theta_v}$ follows directly from (5.8). Note that in order to calculate the derivatives, we must first obtain $\boldsymbol{\alpha}^*$ by solving (5.11) for the current $\{\boldsymbol{\theta}, \mathbf{y}\}$ values.

The procedure for updating $\boldsymbol{\theta}$ for given $\mathbf{y}$, begins by executing a standard gradient descent update on $\boldsymbol{\theta}$, using (5.13). Afterwards, $\boldsymbol{\theta}$ is projected back to its feasible set, so that the positivity and p-norm constraints (5.4) are enforced. In this chapter, we consider the values $p = 1, 2$ and execute the projections as shown in [40, 94]. Note that the gradient descent step size, η, is adjusted according to the Armijo rule, which may require additional optimizations of the dual.

Updating the Cluster Labels

Finding a new set of cluster assignments $\mathbf{y}'$ that will further decrease $\mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$ (keeping the kernel parameters $\boldsymbol{\theta}$ fixed) is not straightforward, since the underlying optimization is a non-convex integer problem. Some single kernel MMC approaches relax $\mathbf{y}$ on the continuous domain to ease the optimization (e.g. [110, 121]), however, in the end the relaxed solution should be mapped back to the discrete space. Here, on the contrary, our aim is to work directly on the discrete cluster labels without any relaxations.

We have developed a practical search framework, where an improved cluster labeling $\mathbf{y}'$ is obtained by moving instances between the two clusters. One possible direction would be to change the cluster label of a single instance only and then proceed with reestimating $\boldsymbol{\theta}$. However, we have empirically found that such a minor modification on $\mathbf{y}$ results in premature convergence as the algorithm overcommits to the initial assignments. A better strategy is to change the labels of multiple instances before reestimating $\boldsymbol{\theta}$. The strategy we follow is motivated by several graph partitioning heuristics that have been applied to clustering, prominently the Kernighan-Lin algorithm [59]: an

initial split of the graph is revamped by exchanging several nodes (specified in an incremental fashion) between partitions and selecting the best subset of these nodes. Based on this idea, we build a sequence of L candidate cluster label vectors, $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(L)}$, ($L$ is user-defined) and select the one generating the greatest improvement on $\mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$ in order to update $\mathbf{y}$. These L candidate label vectors are constructed incrementally (one after the other), such that compared to the previous candidate label vector, the next contains one more instance whose label has been changed (i.e. they differ in one element). Given $\mathbf{y}^{(l)}$, the $(l+1)$-th instance to change clusters is selected to be the one that is expected to produce the smallest objective value when added to the current l changes, thus constructing $\mathbf{y}^{(l+1)}$.

A meaningful approach for picking the $(l+1)$-th instance is to rank the contending instances based on the confidence about their labeling according to the current (after l cluster moves) separating hyperplane and select the one with the smallest $y_i(\mathbf{w}^\top \widetilde{\phi}(\mathbf{x}_i) + b)$ value. This way misclassified instances (if any exist) have a higher priority to change clusters, since $y_i(\mathbf{w}^\top \widetilde{\phi}(\mathbf{x}_i) + b) < 0$, followed by those falling inside the margin (if any exist), since $0 \leq y_i(\mathbf{w}^\top \widetilde{\phi}(\mathbf{x}_i) + b) < 1$, and finally those away from the margin, since $y_i(\mathbf{w}^\top \widetilde{\phi}(\mathbf{x}_i) + b) \geq 1$.

More specifically, let $\mathbf{y}^{(0)}$ to be the vector of the cluster labels before commencing the update process. Assume that $\mathbf{y}^{(l)}$ has already been generated, thus at this point l instances have already changed clusters w.r.t $\mathbf{y}^{(0)}$. As mentioned, the $(l+1)$-th instance is selected to be the one we are the less confident about its labeling according to the separating hyperplane. However, when the labels change so does the hyperplane. Therefore, we must solve the dual (5.11) for the current assignments $\mathbf{y}^{(l)}$ to obtain the corresponding optimal hyperplane parameters $\mathbf{w}^{(l)^*}$ and $b^{(l)^*}$. Then, the index of the $(l+1)$-th instance is given by:

$$i^* = \underset{i: y_i^{(l)} = y_i^{(0)}}{\operatorname{argmin}} \, y_i^{(l)} \left(\mathbf{w}^{(l)^{*\top}} \widetilde{\phi}(\mathbf{x}_i) + b^{(l)^*} \right), \tag{5.14}$$

and the $(l+1)$-th candidate label vector is defined as:

$$y_i^{(l+1)} = \begin{cases} y_i^{(l)}, & i \neq i^* \\ -y_i^{(l)}, & i = i^* \end{cases} . \tag{5.15}$$

From (5.14), it is obvious, that an instance $\mathbf{x}_i$ whose label has already changed is not considered again as a contender, since $y_i^{(l)} \neq y_i^{(0)}$, and the selected one flips its label (5.15). Moreover, observe that the label changes of all previous steps are retained when constructing $\mathbf{y}^{(l+1)}$, leading to an incremental reassignment of the instances. The above is repeated for $l = 0, 1, \ldots, L-1$.

The returned cluster assignments that are used to update $\mathbf{y}$ correspond to the cluster label vector $\mathbf{y}^{(l^*)}$ attaining the smallest objective value (i.e. $\mathbf{y}' = \mathbf{y}^{(l^*)}$):

$$l^* = \underset{0 \leq l \leq L}{\operatorname{argmin}} \, \mathcal{J}(\boldsymbol{\theta}, \mathbf{y}^{(l)}). \tag{5.16}$$

Algorithm 5.1 RMKC.

Input: Basis kernel matrices $\{K^{(v)}\}_{v=1}^{V}$, Initial composite kernel coefficients $\boldsymbol{\theta}^{(0)}$ and cluster assignments $\mathbf{y}^{(0)}$

Output: Final kernel coefficients $\boldsymbol{\theta}$ and cluster assignments $\mathbf{y}$

Set $t = 0$
Set parameters L, ℓ and C
Set $\widetilde{K}^{(0)} = \sum_{v=1}^{V} \theta_v^{(0)} K^{(v)}$
repeat
  Solve the dual (5.11) for $\widetilde{K}^{(t)}$ (i.e. $\boldsymbol{\theta}^{(t)}$) and $\mathbf{y}^{(t)}$ to obtain ${\boldsymbol{\alpha}^{(t)}}^*$
  for $v = 1$ to V **do** // Update $\boldsymbol{\theta}$.
    $\theta_v^{(t+1)} = \theta_v^{(t)} - \eta^{(t)} \left. \frac{\partial \mathcal{J}(\boldsymbol{\theta}, \mathbf{y})}{\partial \theta_v} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}, \mathbf{y}=\mathbf{y}^{(t)}, \boldsymbol{\alpha}^*={\boldsymbol{\alpha}^{(t)}}^*}$
  end for
  Project $\boldsymbol{\theta}^{(t+1)}$ to satisfy the constraints in (5.4)
  $\widetilde{K}^{(t+1)} = \sum_{v=1}^{V} \theta_v^{(t+1)} K^{(v)}$
  $\mathbf{y}^{(t+1)} =$ CLUSTER_UPD ($\widetilde{K}^{(t+1)}$, $\mathbf{y}^{(t)}$) // Update $\mathbf{y}$.
  $t = t + 1$
until converged
return $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)}, \mathbf{y} = \mathbf{y}^{(t)}$

Note that if none of the candidate label vectors $\mathbf{y}^{(l)}$ reduces the objective, then $l^* = 0$ from (5.16), and no label change is accepted. This ensures that the ratio-based objective never increases after updating $\mathbf{y}$.

The procedure for modifying $\mathbf{y}$, as described up to this point, selects L instances belonging to either of the two clusters and flips their label to construct the candidate label vectors. Some trial experiments indicated that a better approach is to restrict all L instances that change clusters to originate from the same (i.e. a single) cluster. For this reason, our final procedure is divided into two phases. In the first phase the candidate vectors are formed by moving L instances from the cluster associated with the $+1$ label to the cluster associated with the -1 label, while in the second phase the opposite movement direction is considered. The two phases are independent from each other, both starting from $\mathbf{y}^{(0)}$. Hence, one phase does not take into account the cluster changes of the other. At the end, the best of the $2L$ candidate vectors is selected to update the cluster labels. To implement the above idea, in (5.14) we must, additionally to $y_i^{(l)} = y_i^{(0)}$, require that $y_i^{(l)} = +1$ ($y_i^{(l)} = -1$) for the first (second) phase contending instances. Our complete, two phase, framework is shown in Algorithm 5.2.

An issue we have yet to touch on is how to impose the cluster balance constraint (5.4). Fortunately, this is rather straightforward under our framework, since we can define an upper bound on the number L of candidate label vectors in each phase and, therefore, on the number of instances allowed to change clusters, to guarantee that the constraint is never violated. For the first phase $L \leq (\ell + \sum_{i=1}^{N} y_i^{(0)})/2$, while for the second $L \leq (\ell - \sum_{i=1}^{N} y_i^{(0)})/2$. Note that $\sum_{i=1}^{N} y_i^{(0)}$ describes the initial imbalance before moving any instances (which, of course, satisfies the constraint) and $\ell \geq 0$ the

Algorithm 5.2 RMKC - cluster update.

Input: Current composite kernel matrix $\widetilde{K}$ and cluster assignments $\mathbf{y}$
Output: Updated cluster assignments $\mathbf{y}'$

function CLUSTER_UPD ($\widetilde{K}$, $\mathbf{y}$)
  // First phase.
  Set $\mathbf{y}^{(0)} = \mathbf{y}$
  for $l = 0$ to $L - 1$ **do**
    Solve the dual (5.11) for $\widetilde{K}$ and $\mathbf{y}^{(l)}$ to obtain $\mathbf{w}^{(l)^*}$ and $b^{(l)^*}$
    Calculate $\mathbf{y}^{(l+1)}$ (5.15) with the added constraint $y_i^{(l)} = +1$ in (5.14)
  end for
  // Second phase. This phase ignores the cluster moves of the first.
  Set $\mathbf{y}^{(L+1)} = \mathbf{y}$
  for $l = L + 1$ to $2L$ **do**
    Solve the dual (5.11) for $\widetilde{K}$ and $\mathbf{y}^{(l)}$ to obtain $\mathbf{w}^{(l)^*}$ and $b^{(l)^*}$
    Calculate $\mathbf{y}^{(l+1)}$ (5.15) with the added constraint $y_i^{(l)} = -1$ in (5.14)
  end for
  $l^* = \text{argmin}_{0 \leq l \leq 2L+1} \mathcal{J}(\boldsymbol{\theta}, \mathbf{y}^{(l)})$
  return $\mathbf{y}' = \mathbf{y}^{(l^*)}$
end function

maximum admissible imbalance. Also, notice that it is not necessary to set the same value for L on both phases. Hence, if the upper bound limits L to a tiny value on one phase, the other can still exploit a larger L.

5.1.4 Discussion

This section examines some additional aspects of the proposed RMKC method, starting with the convergence of the iterative algorithm used to optimize (5.4). In each iteration, the gradient descent update on $\boldsymbol{\theta}$ reduces the ratio-based objective value. Moreover, the subsequent update on $\mathbf{y}$ selects a candidate cluster label vector that further decreases the objective. Hence, the overall process is guaranteed to monotonically converge. The final solution, though, depends on the initial $\{\boldsymbol{\theta}, \mathbf{y}\}$ values, thus a local, and not the global, minimum of $\mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$ is located. Note that the solution also depends on the user-specified constants C, ℓ and L, as well as, on the selected p-norm for the composite kernel coefficients constraint.

An important advantage of RMKC is that it can be readily extended to learning general forms of parametric composite kernels $\widetilde{\mathcal{K}}$, such as a nonlinear mixture of basis kernels, without being restricted to just the linear combination case (5.3). The formulation itself remains unchanged (e.g. (5.4), (5.5), (5.6), (5.7), (5.11), (5.12)) and the iterative algorithm is applicable out of the box, if the gradient of the ratio-based objective can be computed. This is possible when the composite kernel matrix is strictly positive definite and continuously differentiable w.r.t. its parameters $\boldsymbol{\theta}$ (see Section 5.1.3). Of

course, $\frac{\partial \widetilde{K}_{ij}}{\partial \theta_v}$ and $\frac{\partial \mathcal{E}(\boldsymbol{\theta},\mathbf{y})}{\partial \theta_v}$ in (5.13) depend on the specific form of the composite kernel. Moreover, the scale invariance of our objective (i.e. scaling $\widetilde{\mathcal{K}}$ by a scalar $\alpha > 0$) also holds in the general case (the proof is analogous to that in Proposition 5.1), but the same is not true for the norm invariance. Note that scaling $\widetilde{\mathcal{K}}$ is no more equivalent to scaling the parameters $\boldsymbol{\theta}$. The ability to accommodate general kernel forms broadness the applicability of RMKC and constitutes an advantage over existing MKL approaches that are usually limited to a particular type of composite kernel.

5.2 Empirical Evaluation

To investigate the potential of combining the margin with the variance in the clustering objective and perform kernel learning, the presented RMKC framework is compared to: a) kernel *k*-means [37, 90], which serves as our baseline method, b) iterSVR [129], an iterative margin-based MMC approach that follows formulation (5.1), and c) the two iterative variance-based MKL approaches we proposed in Chapter 4 of this thesis that optimize (5.6), namely multi-view kernel *k*-means (MVKKM) and multi-view spectral clustering (MVSpec).

The evaluation is made on various diverse datasets from the UCI repository [42] (Ionosphere, Letter, Satellite, Multiple features and Optdigits), as well as on the COIL-20 image library of objects [81] and a subset of the Corel image collection[4]. Apart from Ionosphere, all other datasets contain instances of more than two categories. For this reason, we conduct experiments using pairs of the included categories. For Letter and Satellite we simply focus on the first two classes, i.e. A-B and C1(red soil)-C2(cotton crop), respectively, as in [129]. For the two databases of handwritten digits (i.e. Multiple features and Optdigits) we try several pairs of the contained numerals (0-9), while for the two image collections we consider pairs of the classes depicted in Figures 5.1-5.2. The tested pairs are shown in Tables 5.3-5.4. Since ground-truth information is available for every dataset, we employ the clustering accuracy metric to measure performance.

The datasets come in vectorial form, except the COIL-20 images for which we first extracted SIFT descriptors and then represented them using a bag of 1000 visual words, as in [56]. All UCI data were normalized to zero mean and unit variance on each attribute, while the COIL-20 and Corel instances were normalized to unit length. Multiple features and Corel are multi-view datasets, hence, for the same instance multiple sets of attributes are available. Each attribute set naturally defines a basis kernel and the linear kernel is employed here to represent each view. For the other, single view, datasets, we follow [110, 115] and construct 10 basis RBF kernels, where the kernel width σ varies from 10% to 100% of the range of distance between any two instances. Kernels are multiplicatively normalized [63].

Throughout the experiments, our algorithm is configured as follows: we fix the

[4] http://www.cs.virginia.edu/~xj3a/research/CBIR/Download.htm

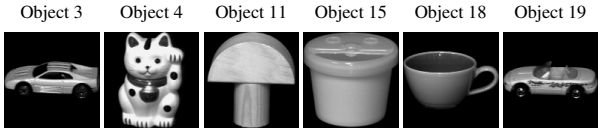


Figure 5.1: The COIL-20 objects considered in the experiments.

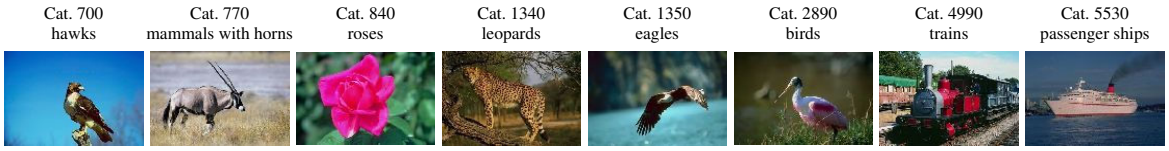


Figure 5.2: Indicative images of the Corel categories considered in the experiments.

number of candidate label vectors in each phase to $L = 30$, the cluster imbalance parameter to $\ell = 0.5N$ (for the Corel images only, $\ell = 0.2N$) and conduct a grid search on the set $\{10^{-2}, 10^{-1}, \ldots, 10^{2}\}$ to locate the best performing value for the C regularization constant in each dataset. The basis kernels are linearly combined (5.3) and their coefficients are uniformly initialized such that they adhere to the selected p-norm constraint, i.e. $\theta_v = \frac{1}{V^{1/p}}$. To initialize the cluster assignments $\mathbf{y}$, we extract several pairs of instances (usually $0.25N$ pairs) using a *k*-means++-like procedure [3], where the first instance is chosen randomly and the second is picked with a probability that is proportional to its distance from the first. For each such pair, the remaining $N - 2$ instances are assigned to the closest of the two instances in the pair, thus producing a partitioning of the data. The partitioning $\mathbf{y}$ with the minimum $\mathcal{J}(\boldsymbol{\theta}, \mathbf{y})$ value is used to initialize a run of RMKC. Since the procedure for choosing the initial $\mathbf{y}$ is nondeterministic, the RMKC performance is averaged over 30 runs for each tried set of parameters (L, ℓ, C, p-norm). Finally, the LIBSVM toolbox [22] is utilized for solving (5.11).

5.2.1 Norm Invariance in Practice

In Proposition 5.2, it was proved that the global optimal solution of our formulation (5.4) is invariant to the p-norm applied on the composite kernel coefficients $\boldsymbol{\theta}$, if $\widetilde{\mathcal{K}}$ is a linear mixture of basis kernels (5.3). However, the RMKC method locates local optima of the ratio-based objective. Hence, it is of particular interest to explore how these local optima vary for different choices of p-norm constraints.

To demonstrate this, RMKC is executed (according to the above configuration) for $p = 1, 2$ and also for the case where no norm constraint is imposed on $\boldsymbol{\theta}$ and the results are illustrated in Table 5.1. Note that for the last four datasets we report the average clustering accuracy (and its deviation) over all considered pairs of categories. Also, note that for Ionosphere, Letter and Satellite, where a single pair of classes is examined, deviations may still appear as RMKC is restarted 30 times. It can be observed that the solutions obtained across the different norms are very similar, therefore, in practice,

Table 5.1: RMKC clustering accuracy $(\%)$ (averaged over all pairs of categories considered in each dataset) for different p-norm constraints.

Dataset	No-norm	1-norm	2-norm
Ionosphere	71.51 ± 0.00	71.51 ± 0.00	71.51 ± 0.00
Letter	94.47 ± 0.00	94.47 ± 0.00	94.47 ± 0.00
Satellite	96.17 ± 0.50	96.19 ± 0.52	96.16 ± 0.51
COIL-20	98.75 ± 2.60	98.61 ± 2.65	98.43 ± 2.73
Corel	94.55 ± 1.62	94.64 ± 1.58	94.69 ± 1.62
Multiple features	99.58 ± 0.22	99.53 ± 0.37	99.59 ± 0.23
Optdigits	97.77 ± 2.45	97.65 ± 2.71	97.75 ± 2.50

the uncovered local optima are not significantly influenced by the choice of p-norm, although this cannot be theoretically guaranteed. On the following, we shall focus on the 1-norm, when presenting the results of our approach.

5.2.2 Comparative Results

We have conducted a comprehensive evaluation of RMKC, kernel *k*-means, iterSVR, MVKKM and MVSpec on all datasets. RMKC is set up as previously described. Kernel *k*-means is restarted 30 times, from randomly picked initial centers. For iterSVR we employ a similar setup to [129], i.e. the cluster imbalance parameter is fixed to $\ell = 0.03N$ for balanced and to $\ell = 0.3N$ for unbalanced datasets, while the initial cluster labels are obtained from the kernel *k*-means solution (iterSVR is, thus, repeated 30 times). For the C regularization constant, the same grid search as for RMKC is implemented. Finally, the sparsity controlling parameter p for MVKKM and MVSpec is selected by a grid search on the values $\{1, 1.5, \ldots, 5\}$.

Performance is measured in terms of average clustering accuracy (and its deviation) over the 30 restarts (MVKKM and MVSpec are deterministically initialized, as described in Chapter 4, Section 4.2.2, thus we have no restarts). Let us stress, that both kernel *k*-means and iterSVR are single kernel methods that do not implement kernel learning. For this reason, these algorithms are independently executed for each of the individual basis kernels in each data collection and the kernel attaining the highest accuracy is reported. Moreover, for iterSVR the average performance over all basis kernels is also shown. It is important to make clear that it is not possible to know a priori which is the best basis kernel for a given dataset.

In Table 5.2 we observe that iterSVR with the optimal basis kernel achieves the best accuracy, being closely matched by RMKC. Only for Ionosphere the difference is large, where, surprisingly, all three MKL approaches (RMKC, MVKKM and MVSpec) are even inferior to kernel *k*-means. However, this is a difficult dataset to cluster and all methods yield rather poor outcomes (accuracy does not exceed 75%).

Turning our attention to image clustering (Table 5.3), it is evident that our ratio-based objective constantly outperforms the other methods. For the COIL-20 objects,

Table 5.2: Clustering accuracy (%) of the compared methods on three popular UCI datasets.

Dataset	RMKC (1-norm)	MVKKM	MVSpec	Kernel *k*-means	IterSVR (best)	IterSVR (average)
Ionosphere	71.51 ± 0.00	71.23	70.66	73.22 ± 2.90	$\mathbf{74.83 \pm 1.65}$	71.83 ± 1.99
Letter (A-B)	94.47 ± 0.00	93.50	88.68	93.63 ± 0.00	$\mathbf{94.51 \pm 1.70}$	92.29 ± 1.97
Satellite (C1-C2)	96.19 ± 0.52	94.19	96.24	94.15 ± 0.03	$\mathbf{96.42 \pm 0.00}$	91.53 ± 5.58

Table 5.3: Clustering accuracy (%) of the compared methods on image clustering.

Dataset	RMKC (1-norm)	MVKKM	MVSpec	Kernel *k*-means	IterSVR (best)	IterSVR (average)
COIL-20						
3-19	$\mathbf{100.00 \pm 0.00}$	**100.00**	**100.00**	94.05 ± 10.27	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$
4-11	$\mathbf{100.00 \pm 0.00}$	77.78	**100.00**	96.30 ± 10.41	98.47 ± 8.37	98.34 ± 8.34
15-18	$\mathbf{100.00 \pm 0.00}$	90.28	95.83	97.57 ± 3.74	99.72 ± 0.35	99.21 ± 0.21
15-19	$\mathbf{94.44 \pm 10.59}$	68.06	86.11	86.57 ± 14.84	93.43 ± 14.30	91.86 ± 14.52
Corel						
700-4990	$\mathbf{97.62 \pm 0.65}$	95.00	95.00	85.98 ± 9.58	96.43 ± 0.25	83.19 ± 1.85
700-5530	92.60 ± 1.42	**94.00**	**94.00**	85.50 ± 0.00	88.63 ± 6.40	68.03 ± 3.49
770-840	$\mathbf{97.55 \pm 0.91}$	94.50	90.00	90.47 ± 0.37	94.20 ± 3.04	87.85 ± 0.58
770-1350	$\mathbf{94.03 \pm 1.72}$	93.50	92.00	88.72 ± 0.96	92.67 ± 1.27	84.10 ± 1.89
1340-1350	$\mathbf{95.50 \pm 0.00}$	95.00	95.00	91.00 ± 0.00	92.50 ± 0.00	83.71 ± 0.00
2890-4990	$\mathbf{90.57 \pm 4.79}$	87.00	86.00	85.00 ± 0.00	90.00 ± 0.00	73.04 ± 5.68

whose images are taken from different angles in a neutral background, hence are easy to distinguish, our approach manages to find the correct clusters for 3/4 of subsets and iterSVR appears to be its closest competitor. Clustering the Corel images is a more difficult task, due to variations in the composition of the depicted scene within each class. Here the differences of RMKC to iterSVR are more distinct and its closest competitor is MVKKM, which clearly displays the benefits of combining information from multiple views under MKL.

For the task of handwritten digits recognition (Table 5.4) the best performance is equally shared between RMKC and iterSVR across the two datasets. Note that for Multiple features, which, like Corel, is a multi-view dataset, RMKC is superior. MVKKM and MVSpec achieve the highest accuracy on a single case (Optdigits for the pair 1-7) and are superior to RMKC for only 3/12 of subsets.

Overall, the proposed RMKC algorithm obtains a higher clustering accuracy for the majority of the tested category pairs. The margin-based iterSVR approach seems to be close, or even better, for some cases, provided the optimal basis kernel is used (iterSVR(best)). However, in practice, the best kernel for a particular dataset is not a priori known. By looking at the Tables' last column, one can notice that iterSVR results degrade significantly if an inappropriate basis kernel is chosen. On the contrary, RMKC is able to automatically infer a meaningful kernel by combining the basis kernels.

Table 5.4: Clustering accuracy $(\%)$ of the compared methods on the task of handwritten digits recognition.

Dataset	RMKC (1-norm)	MVKKM	MVSpec	Kernel *k*-means	IterSVR (best)	IterSVR (average)
Mult. feat.						
1-7	99.62 ± 0.78	98.75	98.75	98.00 ± 0.00	$\mathbf{99.75 \pm 0.00}$	96.85 ± 0.00
2-7	$\mathbf{100.00 \pm 0.00}$	99.00	99.75	97.92 ± 0.24	99.75 ± 0.00	97.61 ± 1.73
2-3	$\mathbf{99.70 \pm 0.23}$	99.25	99.00	99.50 ± 0.00	99.50 ± 0.00	94.13 ± 7.16
3-8	99.28 ± 0.38	99.50	99.50	97.50 ± 0.00	$\mathbf{99.75 \pm 0.00}$	98.78 ± 0.04
5-6	$\mathbf{99.42 \pm 0.48}$	98.50	98.50	98.29 ± 0.09	98.75 ± 0.00	95.68 ± 2.37
6-8	$\mathbf{99.15 \pm 0.33}$	97.25	98.50	97.33 ± 0.16	99.00 ± 0.00	94.94 ± 6.47
Optdigits						
1-7	99.56 ± 1.41	**100.00**	**100.00**	89.38 ± 16.06	96.93 ± 9.83	94.26 ± 13.14
2-7	98.03 ± 1.31	96.35	92.42	95.03 ± 8.40	$\mathbf{99.32 \pm 0.16}$	98.88 ± 0.84
2-3	96.29 ± 5.44	90.56	88.89	89.92 ± 9.10	$\mathbf{96.50 \pm 0.82}$	95.59 ± 2.70
3-8	92.43 ± 8.00	94.12	93.28	92.56 ± 7.80	$\mathbf{96.20 \pm 0.16}$	95.01 ± 4.08
5-6	$\mathbf{99.72 \pm 0.00}$	99.45	99.45	99.57 ± 0.14	$\mathbf{99.72 \pm 0.00}$	99.33 ± 0.01
6-8	$\mathbf{99.89 \pm 0.14}$	99.15	98.87	99.32 ± 0.26	99.72 ± 0.00	99.45 ± 0.06

5.3 Summary

We have proposed a novel formulation that considers the ratio between the margin and the intra-cluster variance for multiple kernel learning in the unsupervised domain. Its objective is optimized by an iterative, gradient-based algorithm to obtain both the cluster assignments and the composite kernel parameters. Moreover, it is characterized by two important properties: it is invariant to scalings of the learned kernel and when basis kernels are linearly mixed it is also invariant (on its global optimum) to the type of p-norm constraint on the composite kernel parameters. Our framework compares favorably to approaches that rely either on the margin or the intra-cluster variance.

Chapter 6

Conclusions and Future Work

6.1 Concluding Remarks

The objective of this thesis was the development, implementation and evaluation of (unsupervised) clustering methodologies. During the elaboration of the thesis we mainly focused on three different axes: i) proximity-based clustering, ii) clustering of data available in multiple views and iii) learning the kernel along with the cluster assignments using multiple kernel learning (MKL) techniques.

Specifically, in Chapter 2, we presented an approach that tackles the initialization problem of the k-means algorithm [76] by altering its sum of the intra-cluster variances objective. Weights are assigned to the clusters in proportion to their variance and a weighted version of the k-means objective is optimized. The cluster weights predispose our method towards primarily minimizing those clusters that exhibit large intra-cluster variance. We have seen that by punishing large variance clusters, bad initializations can be readily overcome, to consistently uncover partitionings of high quality, irrespective of the initial choice of the cluster centers. Additionally, the clusters are balanced in terms of their variance. Moreover, we incorporated in our method an exponent that controls how strongly the weighted objective penalizes larger variance clusters relative to smaller variance clusters and developed a practical framework to automatically tune this exponent to the underlying dataset, to correctly uncover the intrinsic structures in the data. We also extended our approach to perform clustering in kernel space. The conducted experiments confirmed the robustness of the proposed method over bad initializations, as well as its efficacy, and demonstrated that k-means (or kernel k-means)

solutions can be significantly improved when initialized by the solution returned by our algorithm.

In Chapter 3 and Chapter 4, we focused on unsupervised multi-view learning and presented methods that assign weights to the views such that degenerate views are automatically identified and appropriately handled, a subject overlooked in the multi-view literature. Analytically, in Chapter 3, we employed convex mixture models (CMMs) [71] to represent the views and presented two multi-view approaches that identify exemplars (i.e. cluster representatives) in the dataset by simultaneously exploiting all available views. The first, considers all views as being equally important (i.e. it does not employ view weights) and is characterized by the ability to locate the global optimum solution, using simple iterative updates of the involved parameters, and the ability to handle views with different statistical properties. The second, associates a weight with each view that reflects the quality of the view and learns those weights during training. It can be interpreted as a mixture model whose components are CMMs (one for each view) and, like the first approach, takes into account the different statistical properties of the views (however, only local optima can be found in this case). The underlying optimization is carried out using the EM algorithm. The empirical evaluation revealed the superiority of our weighted framework and illustrated that exploiting multiple views can boost clustering performance, especially if the views participate with different weights. Moreover, it verified that concatenating the views is not an effective multi-view strategy.

In Chapter 4, we tackled the multi-view problem from a different perspective. We employed kernel matrices to represent the views and learned a weighted combination of the kernel matrices, that reflects the views' relevance to the clustering task, along with the cluster assignments. Our formulation utilizes a user specified exponent to control the sparsity of the weights, which resembles the p-norm constraint applied in MKL [62]. We have shown that a low value of the exponent results in retaining only the best view (sparse solution), which is useful if most views are of poor quality, while a large value leads towards a uniform solution, which is preferable when all views are of similar quality. Intermediate values provide a tradeoff between these ends. To learn the weights and partition the instances, two iterative algorithms were devised, one based on kernel k-means [37, 90] and another based on spectral techniques [37], where the view weights are updated using closed-form expressions. From the experiments, it was observed that the new methods, particularly the one based on kernel k-means, yield high quality partitionings and that view weighting enhances clustering performance, if the sparsity of the weights is appropriately moderated.

Finally, in Chapter 5 we proposed a novel MKL formulation that considers the ratio between the margin criterion of SVM [12, 19] and the intra-cluster variance criterion of kernel k-means [37, 90], to simultaneously infer an appropriate kernel (i.e. infer its parameters) and cluster the instances. Therefore, both the separation and the compactness of the clusters are utilized in the objective. We proved that the ratio-based objective is invariant to scalings of the learned kernel, hence it can correctly capture

its quality. The same does not hold when relying solely on the margin to perform MKL, which is the most common criterion employed by existing MKL approaches. Moreover, based on the scale invariance, we proved that when basis kernels are linearly mixed, our formulation is invariant (on its global optimum) to the type of norm constraint on the kernel parameters, making the selection of a suitable norm less crucial. Additionally, we have shown that different types of parametric kernels can be learned using our method. Numerical experiments demonstrated that our framework compares favorably to approaches that rely either on the margin or the intra-cluster variance.

6.2 Directions for Future Work

Next, we offer some insights on a number of open issues related to this thesis that could be studied in future work.

For the method presented in Chapter 2, it would be interesting to explore other possible ways of automatically determining the value of the exponent in the weighted objective, besides the one proposed here. For the CMM-based multi-view approaches (Chapter 3), a thorough empirical investigation of their application to kernel space clustering, following the ideas of Section 3.3.3, would provide further insight on the settings under which our multi-view methods prove advantageous. The weighted combination of the views' kernel matrices considered in Chapter 4, utilizes a user specified exponent to regulate the sparsity of the weights. Finding a way to automatically tune this exponent to the dataset would greatly enhance the applicability of our formulation. Moreover, exploiting other, nonlinear, combinations of the views is in our plans. Finally, our ratio-based MKL method (Chapter 5) is currently limited to two-cluster problems. Although it is possible to tackle multiple cluster problems by iteratively solving a sequence of two-cluster problems, an alternative strategy would be to extend our approach to directly handle multiple clusters, following the ideas in [130, 133].

Some additional and more general lines for future research are outlined on the following. Most of the existing multi-view methods, including those presented in this thesis, make an implicit assumption that all views are available for every instance in the dataset. However, in practice, incomplete views may occur, where the representations of certain instances are not available in those views. For example, on a collection of web pages that are represented in terms of their text and the anchor text of the inbound links, it is expected that the second view will be an incomplete view, since not all web pages have inbound links. Developing multi-view approaches which handle incomplete views and also assign weights to the views, would provide a more comprehensive solution to the multi-view problem. Moreover, the ideas of view weighting could be adapted to single view datasets in order to perform unsupervised feature (attribute) selection [52]/weighting [54]. Specifically, each feature can be treated as being a distinct view and the weights in the multi-view algorithm could be used to keep a subset of the features (feature selection) or assign different degrees of importance to

the features (feature weighting). Another intriguing direction for future work is to combine multi-view learning with MKL techniques in order to infer an appropriate kernel for each view. Various paths can be followed to solve this problem. For example, the kernel of each view can be learned independently from the other views. However, a more principled approach would be to consider all views while learning the kernels, to allow the views to interact. Moreover, the same or different types of parametric kernels can be employed for each view (e.g. a linear or nonlinear mixture of basis kernels) and views may be weighted or not. Finally, it would be interesting to employ the methods presented in this thesis to applications that involve data clustering, such as web data analysis, image segmentation, key frame extraction and bioinformatics.

BIBLIOGRAPHY

[1] M. E. Abbasnejad, D. Ramachandram, and M. Rajeswari. An unsupervised approach to learn the kernel functions: from global influence to local similarity. *Neural Computing and Applications*, 20(5):703–715, 2011.

[2] S. P. Abney. Bootstrapping. In *Association for Computational Linguistics (ACL)*, pages 360–367, 2002.

[3] D. Arthur and S. Vassilvitskii. *k*-means++: the advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1027–1035, 2007.

[4] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Int. Conf. on Machine Learning (ICML)*, 2004.

[5] A. M. Bagirov. Modified global *k*-means for minimum sum-of-squares clustering problems. *Pattern Recognition*, 41(10):3192–3199, 2008.

[6] A. M. Bagirov, J. Ugon, and D. Webb. Fast modified global *k*-means algorithm for incremental cluster construction. *Pattern Recognition*, 44(4):866–876, 2011.

[7] N. Baili and H. Frigui. Fuzzy clustering with multiple kernels in feature space. In *IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8, 2012.

[8] A. Banerjee and J. Ghosh. Frequency-sensitive competitive learning for scalable balanced clustering on high-dimensional hyperspheres. *IEEE Trans. on Neural Networks*, 15(3):702–719, 2004.

[9] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *J. of Machine Learning Research*, 6:1705–1749, 2005.

[10] S. Bickel and T. Scheffer. Multi-view clustering. In *Int. Conf. on Data Mining (ICDM)*, pages 19–26, 2004.

[11] S. Bickel and T. Scheffer. Estimation of mixture models using co-EM. In *European Conf. on Machine Learning (ECML)*, pages 35–46, 2005.

[12] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

[13] M. B. Blaschko and C. H. Lampert. Correlational spectral clustering. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[14] A. Blum and T. M. Mitchell. Combining labeled and unlabeled data with co-training. In *Conf. on Computational Learning Theory (COLT)*, pages 92–100, 1998.

[15] P. S. Bradley and U. M. Fayyad. Refining initial points for *K*-means clustering. In *Int. Conf. on Machine Learning (ICML)*, pages 91–99, 1998.

[16] U. Brefeld, T. Gärtner, T. Scheffer, and S. Wrobel. Efficient co-regularised least squares regression. In *Int. Conf. on Machine Learning (ICML)*, pages 137–144, 2006.

[17] U. Brefeld and T. Scheffer. Co-EM support vector learning. In *Int. Conf. on Machine Learning (ICML)*, pages 121–128, 2004.

[18] E. Bruno and S. Marchand-Maillet. Multiview clustering: a late fusion approach using latent models. In *ACM SIGIR Conf. (SIGIR)*, pages 736–737, 2009.

[19] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[20] X. Cai, F. Nie, and H. Huang. Multi-view *K*-means clustering on big data. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 2598–2604, 2013.

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

[22] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. on Intelligent Systems and Technology*, 2(3):1–27, 2011. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[23] O. Chapelle and A. Rakotomamonjy. Second order optimization of kernel parameters. In *NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels*, 2008.

[24] K. Chaudhuri, S. M. Kakade, K. Livescu, and K. Sridharan. Multi-view clustering via canonical correlation analysis. In *Int. Conf. on Machine Learning (ICML)*, pages 129–136, 2009.

[25] G. Cleuziou, M. Exbrayat, L. Martin, and J.-H. Sublemontier. CoFKM: A centralized method for multiple-view clustering. In *Int. Conf. on Data Mining (ICDM)*, pages 752–757, 2009.

[26] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110, 1999.

[27] C. Cortes, M. Mohri, and A. Rostamizadeh. L_2 regularization for learning kernels. In *Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 109–116, 2009.

[28] C. Cortes, M. Mohri, and A. Rostamizadeh. Learning non-linear combinations of kernels. In *Neural Information Processing Systems (NIPS)*, pages 396–404, 2009.

[29] C. Cortes, M. Mohri, and A. Rostamizadeh. Two-stage learning kernel algorithms. In *Int. Conf. on Machine Learning (ICML)*, pages 239–246, 2010.

[30] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola. On kernel-target alignment. In *Advances in Neural Information Processing Systems (NIPS)*, pages 367–373, 2001.

[31] I. Csiszár and P. C. Shields. Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory*, 1(4):417–528, 2004.

[32] J. M. Danskin. The theory of max-min, with applications. *SIAM J. on Applied Mathematics*, 14(4):641–664, 1966.

[33] S. Dasgupta, M. L. Littman, and D. A. McAllester. PAC generalization bounds for co-training. In *Neural Information Processing Systems (NIPS)*, pages 375–382, 2001.

[34] V. R. de Sa. Spectral clustering with two views. In *ICML Workshop on Learning with Multiple Views*, pages 20–27, 2005.

[35] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[36] F. Denis, A. Laurent, R. Gilleron, and M. Tommasi. Text classification and co-training from positive and unlabeled examples. In *ICML Workshop on the Continuum from Labeled to Unlabeled Data*, pages 80–87, 2003.

[37] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007.

[38] C. H. Q. Ding and X. He. K-means clustering via principal component analysis. In *Int. Conf. on Machine Learning (ICML)*, pages 225–232, 2004.

[39] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Int. Conf. on Data Mining (ICDM)*, pages 107–114, 2001.

[40] J. C. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l_1-ball for learning in high dimensions. In *Int. Conf. on Machine Learning (ICML)*, pages 272–279, 2008.

[41] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176–190, 2008.

[42] A. Frank and A. Asuncion. UCI machine learning repository, University of California, Irvine, School of Information and Computer Sciences, 2010. `http://archive.ics.uci.edu/ml`.

[43] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007. `http://www.psi.toronto.edu/affinitypropagation`.

[44] G. Fung, M. Dundar, J. Bi, and R. B. Rao. A fast iterative algorithm for fisher discriminant using heterogeneous kernels. In *Int. Conf. on Machine Learning (ICML)*, pages 40–47, 2004.

[45] K. Gai, G. Chen, and C. Zhang. Learning kernels with radiuses of minimum enclosing balls. In *Advances in Neural Information Processing Systems (NIPS)*, pages 649–657, 2010.

[46] J. Gao, J. Han, J. Liu, and C. Wang. Multi-view clustering via joint nonnegative matrix factorization. In *SIAM Int. Conf. on Data Mining (SDM)*, pages 252–260, 2013.

[47] R. Ghani. Combining labeled and unlabeled data for multiclass text categorization. In *Int. Conf. on Machine Learning (ICML)*, pages 187–194, 2002.

[48] M. Gönen and E. Alpaydin. Localized multiple kernel learning. In *Int. Conf. on Machine Learning (ICML)*, pages 352–359, 2008.

[49] M. Gönen and E. Alpaydin. Multiple kernel learning algorithms. *J. of Machine Learning Research*, 12:2211–2268, 2011.

[50] D. Greene and P. Cunningham. A matrix factorization approach for integrating multiple data views. In *European Conf. on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 423–438, 2009.

[51] D. R. Hardoon, S. Szedmák, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664, 2004.

[52] X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. In *Advances in Neural Information Processing Systems (NIPS)*, pages 507–514, 2005.

[53] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

[54] J. Z. Huang, M. K. Ng, H. Rong, and Z. Li. Automated variable weighting in k-means type clustering. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(5):657–668, 2005.

[55] C. Igel, T. Glasmachers, B. Mersch, N. Pfeifer, and P. Meinicke. Gradient-based optimization of kernel-target alignment for sequence kernels applied to bacterial gene start detection. *IEEE/ACM Trans. on Computational Biology and Bioinformatics*, 4(2):216–226, 2007.

[56] A. Kalogeratos and A. Likas. Dip-means: an incremental clustering method for estimating the number of clusters. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2402–2410, 2012.

[57] J. Kandola, J. Shawe-Taylor, and N. Cristianini. Optimizing kernel alignment over combinations of kernels. Technical Report 121, Department of Computer Science, Royal Holloway, University of London, UK, 2002.

[58] A. Keller. Fuzzy clustering with outliers. In *Int. Conf. of the North American Fuzzy Information Processing Society (NAFIPS)*, pages 143–147, 2000.

[59] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical J.*, 49(2):291–308, 1970.

[60] S.-J. Kim, A. Magnani, and S. P. Boyd. Optimal kernel selection in kernel fisher discriminant analysis. In *Int. Conf. on Machine Learning (ICML)*, pages 465–472, 2006.

[61] S. Kiritchenko and S. Matwin. Email classification with co-training. In *Centre for Advanced Studies Conf. (CASCON)*, pages 301–312, 2001.

[62] M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K.-R. Müller, and A. Zien. Efficient and accurate l_p-norm multiple kernel learning. In *Neural Information Processing Systems (NIPS)*, pages 997–1005, 2009.

[63] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien. L_p-norm multiple kernel learning. *J. of Machine Learning Research*, 12:953–997, 2011.

[64] M. Kloft, U. Rückert, and P. L. Bartlett. A unifying view of multiple kernel learning. In *European Conf. on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 66–81, 2010.

[65] A. Kumar and H. Daumé III. A co-training approach for multi-view spectral clustering. In *Int. Conf. on Machine Learning (ICML)*, pages 393–400, 2011.

[66] A. Kumar, P. Rai, and H. Daumé III. Co-regularized multi-view spectral clustering. In *Neural Information Processing Systems (NIPS)*, pages 1413–1421, 2011.

[67] N. Kushmerick. Learning to remove internet advertisements. In *Int. Conf. on Autonomous Agents*, pages 175–181, 1999.

[68] G. R. G. Lanckriet, N. Cristianini, P. L. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. In *Int. Conf. on Machine Learning (ICML)*, pages 323–330, 2002.

[69] G. R. G. Lanckriet, N. Cristianini, P. L. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *J. of Machine Learning Research*, 5:27–72, 2004.

[70] T. Lange and J. M. Buhmann. Fusion of similarity data in clustering. In *Neural Information Processing Systems (NIPS)*, pages 723–730, 2005.

[71] D. Lashkari and P. Golland. Convex clustering with exemplar-based models. In *Neural Information Processing Systems (NIPS)*, pages 825–832, 2007.

[72] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[73] J. Li and S. Sun. Nonlinear combination of multiple kernels for support vector machines. In *Int. Conf. on Pattern Recognition (ICPR)*, pages 2889–2892, 2010.

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

[75] Y.-Y. Lin, T.-L. Liu, and C.-S. Fuh. Dimensionality reduction for data in multiple feature representations. In *Neural Information Processing Systems (NIPS)*, pages 961–968, 2008.

[76] S. P. Lloyd. Least squares quantization in PCM. *IEEE Trans. on Information Theory*, 28(2):129–136, 1982.

[77] B. Long, P. S. Yu, and Z. M. Zhang. A general model for multiple view unsupervised learning. In *SIAM Int. Conf. on Data Mining (SDM)*, pages 822–833, 2008.

[78] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. of Computer Vision*, 60(2):91–110, 2004.

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

[80] I. Muslea, S. Minton, and C. A. Knoblock. Active + semi-supervised learning = robust multi-view learning. In *Int. Conf. on Machine Learning (ICML)*, pages 435–442, 2002.

[81] S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-20). Technical Report CUCS-005-96, Department of Computer Science, Columbia University, 1996.
http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php.

[82] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Neural Information Processing Systems (NIPS)*, pages 849–856, 2001.

[83] F. Nie, C. H. Q. Ding, D. Luo, and H. Huang. Improved minmax cut graph clustering with nonnegative relaxation. In *European Conf. on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 451–466, 2010.

[84] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Int. Conf. on Information and Knowledge Management (CIKM)*, pages 86–93, 2000.

[85] K. Nigam, A. McCallum, S. Thrun, and T. M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

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

[87] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. More efficiency in multiple kernel learning. In *Int. Conf. on Machine Learning (ICML)*, pages 775–782, 2007.

[88] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *J. of Machine Learning Research*, 9:2491–2521, 2008.

[89] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Int. Conf. on Machine Learning (ICML)*, pages 515–521, 1998.

[90] B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[91] M. Seeger. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, 2001.

[92] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[93] V. Sindhwani and D. S. Rosenberg. An RKHS for multi-view learning and manifold co-regularization. In *Int. Conf. on Machine Learning (ICML)*, pages 976–983, 2008.

[94] J. Songsiri. Projection onto an l_1-norm ball with application to identification of sparse autoregressive models. In *Asean Symposium on Automatic Control (ASAC)*, 2011.

[95] S. Sonnenburg, G. Rätsch, and C. Schäfer. A general and efficient multiple kernel learning algorithm. In *Neural Information Processing Systems (NIPS)*, 2005.

[96] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *J. of Machine Learning Research*, 7:1531–1565, 2006.

[97] T. Su and J. G. Dy. In search of deterministic methods for initializing k-means and gaussian mixture clustering. *Intelligent Data Analysis*, 11(4):319–338, 2007.

[98] S. Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7-8):2031–2038, 2013.

[99] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, Boston, MA, USA, 2005.

[100] W. Tang, Z. Lu, and I. S. Dhillon. Clustering with multiple graphs. In *Int. Conf. on Data Mining (ICDM)*, pages 1016–1021, 2009.

[101] M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *J. of Machine Learning Research*, 1:211–244, 2001.

[102] H. Tong, J. He, M. Li, C. Zhang, and W.-Y. Ma. Graph based multi-modality learning. In *ACM Multimedia (ACM MM)*, pages 862–871, 2005.

[103] G. Tzortzis and A. Likas. The global kernel k-means clustering algorithm. In *Int. Joint Conf. on Neural Networks (IJCNN)*, pages 1977–1984, 2008.

[104] G. Tzortzis and A. Likas. Convex mixture models for multi-view clustering. In *Int. Conf. on Artificial Neural Networks (ICANN)*, pages 205–214, 2009.

[105] G. Tzortzis and A. Likas. The global kernel k-means algorithm for clustering in feature space. *IEEE Trans. on Neural Networks*, 20(7):1181–1194, 2009.

[106] G. Tzortzis and A. Likas. Multiple view clustering using a weighted combination of exemplar-based mixture models. *IEEE Trans. on Neural Networks*, 21(12):1925–1938, 2010.

[107] G. Tzortzis and A. Likas. Kernel-based weighted multi-view clustering. In *Int. Conf. on Data Mining (ICDM)*, pages 675–684, 2012.

[108] G. Tzortzis and A. Likas. The minmax k-means clustering algorithm. *Pattern Recognition*, 47(7):2505–2516, 2014.

[109] G. Tzortzis and A. Likas. Ratio-based multiple kernel clustering. In *European Conf. on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, 2014 (to appear).

[110] H. Valizadegan and R. Jin. Generalized maximum margin clustering and unsupervised kernel learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1417–1424, 2006.

[111] M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *Int. Conf. on Machine Learning (ICML)*, pages 1065–1072, 2009.

[112] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *Int. Conf. on Computer Vision (ICCV)*, pages 1–8, 2007.

[113] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[114] C.-D. Wang, J.-H. Lai, and J.-Y. Zhu. A conscience on-line learning approach for kernel-based clustering. In *Int. Conf. on Data Mining (ICDM)*, pages 531–540, 2010.

[115] F. Wang, B. Zhao, and C. Zhang. Linear time maximum margin clustering. *IEEE Trans. on Neural Networks*, 21(2):319–332, 2010.

[116] J. Wang, H. Do, A. Woznica, and A. Kalousis. Metric learning with multiple kernels. In *Neural Information Processing Systems (NIPS)*, pages 1170–1178, 2011.

[117] J. Wang, J. Zhuang, and S. C. H. Hoi. Unsupervised multiple kernel learning. In *Asian Conf. on Machine Learning (ACML)*, pages 129–144, 2011.

[118] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *Int. Conf. on Computer Vision (ICCV)*, pages 975–982, 1999.

[119] M. Wu and B. Schölkopf. A local learning approach for clustering. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1529–1536, 2006.

[120] C. Xu, D. Tao, and C. Xu. A survey on multi-view learning. *Computing Research Repository*, abs/1304.5634, 2013.

[121] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1537–1544, 2004.

[122] L. Xu and D. Schuurmans. Unsupervised and semi-supervised multi-class support vector machines. In *AAAI Conf. on Artificial Intelligence (AAAI)*, pages 904–910, 2005.

[123] R. Xu and D. C. Wunsch II. Survey of clustering algorithms. *IEEE Trans. on Neural Networks*, 16(3):645–678, 2005.

[124] Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *Int. Conf. on Machine Learning (ICML)*, pages 1175–1182, 2010.

[125] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Association for Computational Linguistics (ACL)*, pages 189–196, 1995.

[126] H. Zeng and Y.-M. Cheung. Kernel learning for local learning based clustering. In *Int. Conf. on Artificial Neural Networks (ICANN)*, pages 10–19, 2009.

[127] H. Zeng and Y.-M. Cheung. Feature selection and kernel learning for local learning-based clustering. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(8):1532–1547, 2011.

[128] H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon. Spectral relaxation for K-means clustering. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1057–1064, 2001.

[129] K. Zhang, I. W. Tsang, and J. T. Kwok. Maximum margin clustering made practical. In *Int. Conf. on Machine Learning (ICML)*, pages 1119–1126, 2007.

[130] B. Zhao, J. T. Kwok, and C. Zhang. Multiple kernel clustering. In *SIAM Int. Conf. on Data Mining (SDM)*, pages 638–649, 2009.

[131] D. Zhou and C. J. C. Burges. Spectral clustering and transductive learning with multiple views. In *Int. Conf. on Machine Learning (ICML)*, pages 1159–1166, 2007.

[132] Z.-H. Zhou and M. Li. Semi-supervised regression with co-training. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 908–916, 2005.

[133] A. Zien and C. S. Ong. Multiclass multiple kernel learning. In *Int. Conf. on Machine Learning (ICML)*, pages 1191–1198, 2007.

Appendix A

Proofs Related to the Weighted Multi-view CMM Method (Chapter 3)

A.1 Proof of the EM Algorithm for the Weighted Multi-view CMM

For clarity we will restate here all mathematical quantities that are necessary for our proof and explicitly declare the parameters they depend upon. Given a dataset $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^N$, where $\boldsymbol{x}_i = \left\{\mathbf{x}_i^{(v)}\right\}_{v=1}^V$, $\mathbf{x}_i^{(v)} \in \Re^{d^{(v)}}$, the distribution of our mixture model is:

$$F\left(\boldsymbol{x}; \boldsymbol{\Theta}\right) = \sum_{v=1}^{V} \pi^v Q^v(\mathbf{x}^{(v)}; \{q_j\}_{j=1}^N),\ \mathbf{x}^{(v)} \in \Re^{d^{(v)}}, \tag{A.1}$$

$$\text{where } Q^v(\mathbf{x}^{(v)}; \{q_j\}_{j=1}^N) = \sum_{j=1}^{N} q_j f_j^v(\mathbf{x}^{(v)}),\ \boldsymbol{\Theta} = \left\{\{\pi^v\}_{v=1}^V, \{q_j\}_{j=1}^N\right\},$$

$$\pi^v \geq 0,\ \sum_{v=1}^{V} \pi^v = 1,\ q_j \geq 0,\ \sum_{j=1}^{N} q_j = 1.$$

Note that the exponential family distributions $f_j^v(\mathbf{x}^{(v)})$ are independent of the parameters $\boldsymbol{\Theta}$. Our target is to maximize the log-likelihood (A.2) of the dataset $\mathcal{X}$ under the mixture model distribution $F\left(\boldsymbol{x}; \boldsymbol{\Theta}\right)$, w.r.t. the parameters $\boldsymbol{\Theta}$.

$$L\left(\mathcal{X}; \boldsymbol{\Theta}\right) = \sum_{i=1}^{N} \log\left(\sum_{v=1}^{V} \pi^v Q^v(\mathbf{x}_i^{(v)}; \{q_j\}_{j=1}^N)\right) \tag{A.2}$$

Let $\{\mathcal{X}, \mathcal{Z}\}$ be the complete dataset, where $\mathcal{Z} = \{z_i\}_{i=1}^N$ contains the latent variables indicating the mixture component responsible for generating each instance, i.e. $z_i \in \{1, 2, \ldots, V\}$. It must be stressed that the z_i values are not known in practice. Now we will analytically prove each step of the EM process [12, 35].

From our model the next probabilities directly follow:

$$P(z_i = v; \boldsymbol{\Theta}) = \pi^v, \tag{A.3}$$

$$P(\boldsymbol{x}_i | z_i = v; \boldsymbol{\Theta}) = Q^v(\mathbf{x}_i^{(v)}; \{q_j\}_{j=1}^N). \tag{A.4}$$

E-step. This step uses the current parameter values $\boldsymbol{\Theta}^{(t)}$ to find the posteriors of the latent variables $P(z_i = v | \boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)})$, by applying the Bayes' theorem:

$$P(z_i = v | \boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)}) = \frac{P(z_i = v; \boldsymbol{\Theta}^{(t)}) P(\boldsymbol{x}_i | z_i = v; \boldsymbol{\Theta}^{(t)})}{P(\boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)})} = \frac{\pi^{v(t)} Q^v(\mathbf{x}_i^{(v)}; \{q_j^{(t)}\}_{j=1}^N)}{\sum_{v=1}^V \pi^{v(t)} Q^v(\mathbf{x}_i^{(v)}; \{q_j^{(t)}\}_{j=1}^N)} . \tag{A.5}$$

M-step. This step evaluates the expectation of the complete dataset log-likelihood under the current latent variables posterior distribution (A.6), for some general parameter value $\boldsymbol{\Theta}$.

$$\begin{aligned}
\mathcal{Q}(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(t)}) &= \sum_{i=1}^N < \log P(z_i, \boldsymbol{x}_i; \boldsymbol{\Theta}) >_{P(z_i | \boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)})} \\
&= \sum_{i=1}^N \sum_{v=1}^V P(z_i = v | \boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)}) \log \left(P(z_i = v; \boldsymbol{\Theta}) P(\boldsymbol{x}_i | z_i = v; \boldsymbol{\Theta}) \right) \\
&= \sum_{i=1}^N \sum_{v=1}^V P(z_i = v | \boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)}) \log \left(\pi^v Q^v(\mathbf{x}_i^{(v)}; \{q_j\}_{j=1}^N) \right) \\
&= \sum_{i=1}^N \sum_{v=1}^V P(z_i = v | \boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)}) \log \pi^v + \\
&\quad + \sum_{i=1}^N \sum_{v=1}^V P(z_i = v | \boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)}) \log \left(\sum_{j=1}^N q_j f_j^v(\mathbf{x}_i^{(v)}) \right)
\end{aligned} \tag{A.6}$$

Subsequently it maximizes this expectation to get a new estimate for the parameters, taking into account any imposed constraints. The constrained expectation is given by (A.7), where λ, μ are Lagrange multipliers (the nonnegativity constraints on π^v and q_j are not enforced into (A.7), since they will be satisfied by the obtained solution).

$$\mathcal{Q}_{con}(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(t)}) = \mathcal{Q}(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(t)}) + \lambda \left(\sum_{v=1}^V \pi^v - 1 \right) + \mu \left(\sum_{j=1}^N q_j - 1 \right) \tag{A.7}$$

We begin by optimizing (A.7) w.r.t. π^v, by setting the corresponding derivative equal

to zero:

$$\frac{\partial \mathcal{Q}_{con}(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(t)})}{\partial \pi^v} = 0 \;\Rightarrow\; \sum_{i=1}^{N} \frac{P(z_i = v|\boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)})}{\pi^v} + \lambda = 0$$

$$\Rightarrow\; -\lambda \pi^v = \sum_{i=1}^{N} P(z_i = v|\boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)}). \tag{A.8}$$

Summing over v and making use of the constraint $\sum_{v=1}^{V} \pi^v = 1$ we obtain:

$$-\lambda \underbrace{\sum_{v=1}^{V} \pi^v}_{=1} = \sum_{i=1}^{N} \underbrace{\sum_{v=1}^{V} P(z_i = v|\boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)})}_{=1} \;\Rightarrow\; \lambda = -N. \tag{A.9}$$

Using (A.9) to eliminate λ in (A.8) and rearranging, gives the new estimation:

$$\pi^{v(t+1)} = \frac{1}{N} \sum_{i=1}^{N} P(z_i = v|\boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)}).$$

The maximization of (A.7) w.r.t. q_j, by setting the corresponding derivative equal to zero, follows:

$$\frac{\partial \mathcal{Q}_{con}(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(t)})}{\partial q_j} = 0 \;\Rightarrow\; \sum_{i=1}^{N} \sum_{v=1}^{V} P(z_i = v|\boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)}) \frac{f_j^v(\mathbf{x}_i^{(v)})}{\sum_{j'=1}^{N} q_{j'} f_{j'}^v(\mathbf{x}_i^{(v)})} + \mu = 0. \tag{A.10}$$

By multiplying both sides of (A.10) with q_j and then summing over j, together with the constraint $\sum_{j=1}^{N} q_j = 1$, gives:

$$-\mu \underbrace{\sum_{j=1}^{N} q_j}_{=1} = \underbrace{\sum_{i=1}^{N} \sum_{v=1}^{V} P(z_i = v|\boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)}) \underbrace{\sum_{j=1}^{N} \frac{q_j f_j^v(\mathbf{x}_i^{(v)})}{\sum_{j'=1}^{N} q_{j'} f_{j'}^v(\mathbf{x}_i^{(v)})}}_{=1}}_{=1} \;\Rightarrow\; \mu = -N. \tag{A.11}$$

Using (A.11) to eliminate μ in (A.10) and rearranging, we get:

$$q_j = \frac{q_j}{N} \sum_{i=1}^{N} \sum_{v=1}^{V} P(z_i = v|\boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)}) \frac{f_j^v(\mathbf{x}_i^{(v)})}{\sum_{j'=1}^{N} q_{j'} f_{j'}^v(\mathbf{x}_i^{(v)})}\,. \tag{A.12}$$

Since we cannot solve analytically for q_j in (A.12), we must resort into iteratively performing updates on q_j during the M-step, before we proceed to the next EM iteration. That is the reason for writing t' for the new estimations in the following equation, instead of t.

$$q_j^{(t'+1)} = \frac{q_j^{(t')}}{N} \sum_{i=1}^{N} \sum_{v=1}^{V} P(z_i = v|\boldsymbol{x}_i; \boldsymbol{\Theta}^{(t)}) \frac{f_j^v(\mathbf{x}_i^{(v)})}{\sum_{j'=1}^{N} q_{j'}^{(t')} f_{j'}^v(\mathbf{x}_i^{(v)})}$$

∎

A.2 Proof of the Assignment Step for the Weighted Multi-view CMM

The weighted multi-view CMM represents all instances as possible exemplars, since each view's CMM has N components, centered at the corresponding instances. Viewing our model as a mixture model, for a given parameter value $\boldsymbol{\Theta}$ an instance $\boldsymbol{x}_i$ is softly assigned to the j-th component (cluster) with probability $P(c_i = j|\boldsymbol{x}_i; \boldsymbol{\Theta})$, where $c_i \in \{1, 2, \ldots, N\}$ indicates the CMM component responsible for generating $\boldsymbol{x}_i$. Apparently, the c_i values are unknown in practice. By applying Bayes' theorem we write:

$$P(c_i = j|\boldsymbol{x}_i; \boldsymbol{\Theta}) = \frac{P(c_i = j; \boldsymbol{\Theta})P(\boldsymbol{x}_i|c_i = j; \boldsymbol{\Theta})}{P(\boldsymbol{x}_i; \boldsymbol{\Theta})} = \frac{P(c_i = j; \boldsymbol{\Theta})P(\boldsymbol{x}_i|c_i = j; \boldsymbol{\Theta})}{\sum_{v=1}^{V} \pi^v Q^v(\mathbf{x}_i^{(v)}; \{q_j\}_{j=1}^{N})} . \quad \text{(A.13)}$$

For our model it holds that:

$$P(z_i = v|c_i = j; \boldsymbol{\Theta}) = P(z_i = v; \boldsymbol{\Theta}) = \pi^v, \quad \text{(A.14)}$$

$$P(c_i = j; \boldsymbol{\Theta}) = P(c_i = j|z_i = v; \boldsymbol{\Theta}) = q_j \ , \quad \text{(A.15)}$$

$$P(\boldsymbol{x}_i|z_i = v, c_i = j; \boldsymbol{\Theta}) = f_j^v(\mathbf{x}_i^{(v)}). \quad \text{(A.16)}$$

The second nominator term in (A.13) with the help of (A.14), (A.16) is estimated as:

$$P(\boldsymbol{x}_i|c_i = j; \boldsymbol{\Theta}) = \sum_{v=1}^{V} P(\boldsymbol{x}_i|z_i = v, c_i = j; \boldsymbol{\Theta})P(z_i = v|c_i = j; \boldsymbol{\Theta}) = \sum_{v=1}^{V} \pi^v f_j^v(\mathbf{x}_i^{(v)}). \quad \text{(A.17)}$$

Substituting (A.15) and (A.17) into (A.13) we obtain:

$$P(c_i = j|\boldsymbol{x}_i; \boldsymbol{\Theta}) = \frac{q_j \sum_{v=1}^{V} \pi^v f_j^v(\mathbf{x}_i^{(v)})}{\sum_{v=1}^{V} \pi^v Q^v(\mathbf{x}_i^{(v)}; \{q_j\}_{j=1}^{N})} . \quad \text{(A.18)}$$

Note that the above probability is used in equation (3.17) for those components whose corresponding instances are selected as the exemplars that represent each of the M clusters. ∎

Appendix B

Proofs Related to the MVKKM and MVSpec Methods (Chapter 4)

B.1 Proof of the Weight Update Formula for MVKKM

For convenience, let us rewrite the optimization problem for given clusters, using the form of the objective in (4.7):

$$\min_{\{w_v\}_{v=1}^V} \sum_{v=1}^{V} w_v^p \sum_{i=1}^{N}\sum_{k=1}^{M} \delta_{ik} \|\phi^{(v)}(\mathbf{x}_i^{(v)}) - \mathbf{m}_k^{(v)}\|^2,\ s.t.\ \ w_v \geq 0,\ \sum_{v=1}^{V} w_v = 1. \tag{B.1}$$

Consider the case for $p > 1$ and denote by $\mathcal{D}_v$ the intra-cluster variance of the v-th view feature space $\mathcal{H}^{(v)}$, i.e. $\mathcal{D}_v = \sum_{i=1}^{N}\sum_{k=1}^{M} \delta_{ik} \|\phi^{(v)}(\mathbf{x}_i^{(v)}) - \mathbf{m}_k^{(v)}\|^2$. By incorporating into the objective the sum-to-unity constraint, the Lagrangian becomes:

$$\mathcal{L} = \sum_{v=1}^{V} w_v^p \mathcal{D}_v + \lambda \left(\sum_{v=1}^{V} w_v - 1 \right). \tag{B.2}$$

Setting the derivative of the Lagrangian to zero yields

$$\frac{\partial \mathcal{L}}{\partial w_v} = 0 \ \Rightarrow\ p w_v^{(p-1)} \mathcal{D}_v + \lambda = 0 \ \Rightarrow\ w_v = \left(\frac{-\lambda}{p D_v} \right)^{\frac{1}{p-1}}. \tag{B.3}$$

By summing over all views, together with the constraint $\sum_{v=1}^{V} w_v = 1$, we get

$$\sum_{v'=1}^{V} \left(\frac{-\lambda}{p D_{v'}} \right)^{\frac{1}{p-1}} = 1 \ \Rightarrow\ (-\lambda)^{\frac{1}{p-1}} = 1 / \sum_{v'=1}^{V} \left(\frac{1}{p \mathcal{D}_{v'}} \right)^{\frac{1}{p-1}}. \tag{B.4}$$

Finally, substituting (B.4) into (B.3) completes the proof:

$$w_v = 1/\sum_{v'=1}^{V} \left(\frac{\mathcal{D}_v}{\mathcal{D}_{v'}}\right)^{\frac{1}{p-1}} \text{ if } p > 1. \tag{B.5}$$

Note that the nonnegativity of the weights was not enforced into (B.2), since it is verified by the solution (B.5), as $\mathcal{D}_v \geq 0$.

For the $p = 1$ case, it easy to see that for any weight values $w_{v'}$ obeying the constraints of (B.1) and corresponding $\mathcal{D}_{v'} \geq 0$, the following holds:

$$\mathcal{D}_{v^*} \leq \sum_{v'=1}^{V} w_{v'}\mathcal{D}_{v'}, \text{ where } v^* = \operatorname*{argmin}_{v'} \mathcal{D}_{v'}, \tag{B.6}$$

from which directly follows that (B.1) is minimized for

$$w_v = \begin{cases} 1, & v = \operatorname{argmin}_v \mathcal{D}_v \\ 0, & \text{otherwise} \end{cases} \text{ if } p = 1. \tag{B.7}$$

∎

B.2 Proof of the Weight Update Formula for MVSpec

Using the form of the objective in (4.8), the optimization problem for given clusters can be written as:

$$\min_{\{w_v\}_{v=1}^{V}} \sum_{v=1}^{V} w_v^p \left(tr(K^{(v)}) - tr(Y^\top K^{(v)} Y)\right), \; s.t. \; w_v \geq 0, \; \sum_{v=1}^{V} w_v = 1. \tag{B.8}$$

The similarity to the MVKKM optimization problem is evident, with the only difference being that $\mathcal{D}_v = tr(K^{(v)}) - tr(Y^\top K^{(v)} Y)$. Since $K^{(v)}$ is a positive semidefinite matrix and $Y^\top Y = I$, $Y \in \Re^{N \times M}$, by the Ky-Fan theorem [77] we have $\mathcal{D}_v \geq 0$. Therefore, the derivations are analogous to the MVKKM case. ∎

AUTHOR'S PUBLICATIONS

Journal Publications

J1. Grigorios Tzortzis and Aristidis Likas, The MinMax k-means Clustering Algorithm, *Pattern Recognition*, vol. 47, no. 7, pp. 2505-2516, 2014

J2. Grigorios Tzortzis and Aristidis Likas, Multiple View Clustering Using a Weighted Combination of Exemplar-based Mixture Models, *IEEE Trans. on Neural Networks*, vol. 21, no. 12, pp. 1925-1938, 2010

J3. Grigorios Tzortzis and Aristidis Likas, The Global Kernel k-means Algorithm for Clustering in Feature Space, *IEEE Trans. on Neural Networks*, vol. 20, no. 7, pp. 1181-1194, 2009

Conference Publications

C1. Grigorios Tzortzis and Aristidis Likas, Ratio-based Multiple Kernel Clustering, to appear in *European Conf. on Machine Learning and Principles and Practice of Knowledge Discovery in Databases* (*ECML/PKDD*), 2014

C2. Grigorios Tzortzis and Aristidis Likas, Kernel-based Weighted Multi-view Clustering, *Int. Conf. on Data Mining* (*ICDM*), pp. 675-684, 2012

C3. Grigorios Tzortzis and Aristidis Likas, Greedy Unsupervised Multiple Kernel Learning, *Hellenic Conf. on Artificial Intelligence* (*SETN*), pp. 73-80, 2012

C4. Grigorios Tzortzis and Aristidis Likas, Convex Mixture Models for Multi-view Clustering, *Int. Conf. on Artificial Neural Networks* (*ICANN*), vol. 2, pp. 205-214, 2009

C5. Grigorios Tzortzis and Aristidis Likas, The Global k-means Clustering Algorithm, *Int. Joint Conf. on Neural Networks* (*IJCNN*), pp. 1978-1985, 2008

C6. Grigorios Tzortzis and Aristidis Likas, Deep Belief Networks for Spam Filtering, *Int. Conf. on Tools with Artificial Intelligence* (*ICTAI*) vol. 2, pp. 306-309, 2007

Short Vita

Grigorios Tzortzis was born in Ioannina, Greece in 1984. He received the B.Sc. and M.Sc. degrees in Computer Science (grade "Excellent") from the University of Ioannina, in 2006 and 2008, respectively. Since 2008, he has been a Ph.D. candidate and a member of the Information Processing and Analysis research group (I.P.AN.) at the Computer Science and Engineering Department of the University of Ioannina. He has worked as an Instructor at the Department of Informatics Engineering of the Technological Educational Institute of Epirus, Greece and, also, on several research projects. He has received scholarships from the Bodossaki Foundation and the Greek State Scholarships Foundation (IKY). His research interests include machine learning, data mining and bioinformatics.